Web Information Systems
Analysis, Design, Development, and Implementation of
Business Sites, Collaboration Sites, Edutainment (e-Learning) Sites, and Infotainment (Information) Sites

Collection of Recent Papers

Bernhard Thalheim

Christian-Albrechts-University Kiel, Department of Computer Science, 24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de

A Web Information System (WIS) is an information system that can be accessed through the world-wide-web. On a high level of abstraction a WIS can be described by a storyboard [ST05b], which in an abstract way specifies who will be using the system, in which way and for which goals. In a nutshell, a storyboard consists of three parts:

– a story space, which itself consists of a hierarchy of labelled directed graphs called scenarios, one of which is the main scenario, whereas the others define the details of scenes, i.e. nodes in a higher scenario, and a plot that is specified by an assignment-free process, in which the basic actions correspond to the labels of edges in the scenarios,
– a set of actors, i.e. abstractions of user groups that are defined by roles, which determine obligations and rights, and user profiles, which determine user preferences,
– and a set of tasks that are associated with goals the users may have.

In addition, there are many constraints comprising static, dynamic and deontic constraints for pre- and postconditions, triggering and enabling events, rights and obligations of roles, preference rules for user types, and other dependencies on the plot. Details of storyboarding have been described in [ST05b]. An overview of our method for the design of WISs was presented in [ST05d].

While syntax and semantics of storyboarding has been well explored, its pragmatics apart from the use of metaphors [TD00] has not. Pragmatics is part of semiotics, which is concerned with the relationship between signs, semantic concepts and things of reality. This relationship may be pictured by the so-called semiotics triangle. Main branches of semiotics are syntactics, which is concerned with the syntax, i.e. the construction of the language, semantics, which is concerned with the interpretation of the words of the language, and pragmatics, which is concerned with the current use of utterances by the user and context of words for the user. Pragmatics permits the use of a variety of semantics depending on the user, the application and the technical environment. Most languages defined in Computer Science have a well-defined syntax; some of them possess a well-defined semantics; few of them use pragmatics through which the meaning might be different for different users.
Syntactics is often based on a constructive or generative approach: Given an alphabet and an set of constructors, the language is defined as the set of expressions that can be generated by the constructors. Constructions may be defined on the basis of grammatical rules.

Semantics of generative languages can be either defined by meta-linguistic semantics, e.g. used for defining the semantics of predicate logics, by procedural or referential semantics, e.g. operational semantics used for defining the semantics of programming languages, or by convention-based semantics used in linguistics. Semantics is often defined on the basis of a set of relational structures that correspond to the signature of the language.

Pragmatics has to be distinguished from pragmatism. Pragmatism means a practical approach to problems or affairs. According to Webster [Web91] pragmatism is the “balance between principles and practical usage”. Here we are concerned with pragmatics, which is based on the behaviour and demands of users, therefore depends on the understanding of users.

The six characteristics of WISs that were discussed in [ST05b] can be mapped to conceptual structures that are used for storyboard specification:

1. We start with the characteristics used for the strategic layer. Main specification elements used are intention and mission. They are mapped to metaphors, general goals, rhetorical figures, and patterns and grids of web pages discussed later.
2. The scenarios reflect the utilisation by actors, for which we envision a number of stories that correspond to real use. These scenarios may be captured through observation of reality. Story spaces and plots are recorded in various level of detail through the methods discussed in [ST05b]. The stories are reflected in the storyboard.
3. Content specification is the basis for the media types, i.e. data types and their functions, which will be introduced in part III. It combines data specification with user requirements and is reflected in the content portfolio.
4. Functionality is provided by the media types as required by the storyboard. Typical standard functions are navigation, retrieval (search), support functions, and feedback facilities.
5. Context is based on tasks, history, and environment. We use the specification of context for restructuring and functionality enhancement, which will form the basis of XSL transformations and the onion approach that is discussed in the last part of the book.
6. Presentation depends on the intention, the provider, the technical environment available and the users the WIS is targeting at. Presentation results in the layout and the playout of the WIS. Layout requires the development of multimedia presentations for each page. Playout additionally requires the development of functionality that supports visits of users depending on the story they are currently following to achieve their goals. Layout and playout integrate the chosen metaphors; they depend on chosen page patterns and grids as well as on quality requirements.

Conceptual structures and their association are depicted in Figure 1. We may separate the syntactics and pragmatics layers. Arrows are used for representing part-of- or
uses- or relates-associations. For instance, the story is based on the user and the functions. Information metaphors relate content to information.

![Fig. 1. The Web Utilization Space Based On the Characteristics of WIS](image)

We use the notions of information and content in a specific manner. Information as processed by humans, is carried by data that is perceived or noticed, selected and organized by its receiver, because of his subjective human interests, originating from his instincts, feelings, experience, intuition, common sense, values, beliefs, personal knowledge, or wisdom, simultaneously processed by his cognitive and mental processes, and seamlessly integrated in his recallable knowledge. Content is complex and ready-to-use data. Content management systems are information systems that support extraction, storage and delivery of complex data. Content may be enhanced by concepts that specify the semantic meaning of content objects and by topics that specify the pragmatic understanding of users.

Therefore, information is directed towards pragmatics, whereas content may be considered to highlight the syntactical dimension. If content is enhanced by concepts and topics, then users are able to capture the meaning and the utilisation of the data they receive. In order to ease perception we use metaphors. Metaphors may be separated into those that support perception of information and into those that support usage or functionality.

The information transfer from a user $A$ to a user $B$ depends on the users $A$ and $B$, their abilities to send and to receive the data, to observe the data, and to interpret the data. Let us formalise this process. Let $s_X$ denote the function user by a user $X$ for data extraction, transformation, and sending of data. Let $r_X$ denote the corresponding function for data receiving and transformation, and let $o_X$ denote the filtering or observation function. The data currently considered by $X$ is denoted by $D_X$. Finally, data filtered or observed must be interpreted by the user $X$ and integrated into the knowledge $K_X$ a user $X$ has. Let us denote by $i_X$ the binary function from data and knowledge to
knowledge. By default, we extend the function $i_X$ by the time $t_{i_X}$ of the execution of the function.

Thus, the data transfer and information reception (or briefly information transfer) is formally expressed it by

$$I_B = i_B(o_B(r_B(s_A(D_A))), K_B, t_{i_X}).$$

In addition, time of sending, receiving, observing, and interpreting can be taken into consideration. In this case we extend the above functions by a time argument. The function $s_X$ is executed at moment $t_{s_X}$, $r_X$ at $t_{r_X}$, and $o_X$ at $t_{o_X}$. We assume $t_{s_A} \leq t_{r_B} \leq t_{o_B} \leq t_{o_B}$ for the time of sending data from $A$ to $B$. The time of a computation $f$ or data consideration $D$ is denoted by $t_f$ or $t_D$, respectively. In this extended case the information transfer is formally expressed it by

$$I_B = i_B(o_B(r_B(s_A(D_A), t_{s_A}, t_{r_B}, t_{o_B}), K_B, t_{i_B}).$$

Fig. 2. Dimensions of understanding messages

The notion of information extends the dimensions of understanding of message displayed in Figure 2 to a web communication act that considers senders, receivers, their knowledge and experience. Figure 3 displays the multi-layering of communication, the influence of explicit knowledge and experience on the interpretation.

The communication act is specified by

- the communication message with the content or content chunk, the characterisation of the relationship between sender and receiver, the data that are transferred and may lead to information or misinformation, and the presentation,
- the sender, the explicit knowledge the sender may use, and the experience the sender has, and
- the receiver, the explicit knowledge the receiver may use, and the experience the receiver has.

The communication act is the basis for *sagas* that are the basic units of stories.

Users are reflected by actors that are abstractions of groups of users. Pragmatics and syntactics share data and functions. The functionality is provided through functions and their representations. The web utilisation space depends on the technical environment of the user. It is specified through the layout and the playout. Layout places content on the basis of a data representation and in dependence of the technical environment. Playout
Fig. 3. Dimensions of the communication act

is based on functionality and function representations, and depends on the technical environment.

We may base the pragmatics of storyboarding on two ingredients that are easy to develop:

**The utilisation portfolio of the WIS:** The utilisation portfolio consists of

- tasks users might wish to accomplish within the system,
- goals of user groups, i.e. actors, and
- actors that might use the web information system.

**The profiles of the users:** Profiles of users describe their abilities, skills, knowledge, and other dimensions. We map user profiles to preference rules for utilisation of systems.

**Table of Content**

1. Systematic development of internet sites: Extending approaches of conceptual modeling
   published in [DT03]
   related papers: [BZT03,DT01,DT04,FOST00,FT02]
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

2. The co-design approach to WIS development in e-business and e-learning applications
   published in [ST04c]
   related papers: [DT02,ST05a,LTV07,FJM+08,ST05c,tha04,tha03,ST04d,ST07a]
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

3. Structural media types in the development of data-intensive web information systems
   published in [ST04b]
   related papers: [KSTW04,TSR+04,Sri00,Fey03,FRR+04,ST04d,ST04e,FRuDMT07]
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008
4. Formalization of user preferences, obligations, and rights
   published in [STT06]
   related papers: [ST07c,?,STT07,ST07e,ST04a]
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

5. View Integration and Cooperation in Databases, Data Warehouses and Web Information Systems
   published in [MKDTZ05]
   related papers: [FRT05,MST05a]
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

6. Pragmatics of storyboarding for web information systems: Usage analysis
   published in [ST07d]
   related papers: [ST08b,FKT04a,FSB04,STZ04,FKT04b,FSBST04,ST06]
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

7. Context analysis: Towards pragmatics of information system design
   published in [ST08b]
   related papers: [KSTZ03,BZKST04]
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

8. Intention-Driven Screenography
   published in [MNST07a]
   related papers: [NT08,MNST07b]
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

9. Quality assurance in Web Information Systems development
   published in [STZ]
   related papers: []
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

10. Strategic modeling of web information systems
    published in [MST05c]
    related papers: [MST05b,LTV07,FKT04a]
    see also my website under miscellaneous or Verschiedenes: talks 2007/2008

11. Databases of Personal Identifiable Information
     published in [AFT08]
     related papers: [AFFT05,TAFAS08,AFT09]
     see also my website under miscellaneous or Verschiedenes: talks 2007/2008

12. Privacy enhanced information systems
     published in [AFFT05]
     related papers: [AFT08,TAFAS08,AFT09]
     see also my website under miscellaneous or Verschiedenes: talks 2007/2008, 2009

13. LIFE CASES: An Approach to Address Pragmatics in the Design of Web Information Systems
     published in [ST07b]
     related papers: [ST08d,TSM09,BT08]
     see also my website under miscellaneous or Verschiedenes: talks 2007/2008

14. Facets of Media Types
     published in [ST08c]
     related papers: [ST01,ST04b,MST05a]
     see also my website under miscellaneous or Verschiedenes: talks 2007/2008
15. Towards Semantic Wikis: Modelling Intensions, Topics, and Origin in Content Management Systems
   published in [FT09]
   related papers: []
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

   published in [BZKK+05]
   related papers: [SBZ04,JMR+03,JLG+04]
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

17. Logistics for learning objects
   published in [BZTT03b]
   related papers: [JMR+03,SBZ04]
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

18. Towards Generative Engineering of Content-Intensive Applications
   published in [BZ04b]
   related papers: [Bie08,BZ04a,BZTT03a,BST06,BZSBTT03,NT04,JLG+04,ST04c]
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

19. Adapting to Changing Times and Needs
   published in [BZ04b,JLG+04,NT04]
   related papers: [ST08f,ST04c]
   see also my website under miscellaneous or Verschiedenes: talks 2007/2008

20. Storyboarding Concepts for Edutainment WIS
    published in [ST08f]
    related papers: [BZ04a,BZTT03a,BZSBTT03,NT04,JLG+04,ST04c]
    see also my website under miscellaneous or Verschiedenes: talks 2007/2008

21. Pragmatics of Storyboarding - Web Information Systems Portfolios
    to be published in WebIST'2010
    related papers: [ST08f,ST06,ST07b,ST07d,ST08b,JLG+04]
    see also my website under miscellaneous or Verschiedenes: talks 2007/2008

22. The Enhanced Entity-Relationship Model
    published in [Tha09]
    related papers: [Tha00,Tha07a,ST08e,ST08a,Tha07b,DMT07,MDT04]
    see also my website under miscellaneous or Verschiedenes: talks 2007/2008

23. Generalisation and specialisation
    published in [Tha09]
    related papers: [Tha00,BT07]
    see also my website under miscellaneous or Verschiedenes: talks 2007/2008

24. Abstraction
    published in [Tha09]
    related papers: [Tha00]
    see also my website under miscellaneous or Verschiedenes: talks 2007/2008
References


[LTV07] L. Landwehr, B. Thalheim, and A. Vitzthum. Navigieren in wissensstrukturen -
benutzerprofile und neue retrievalstrukturen. In Tagung Kunst- und Medientechnologie,
Karlsruhe, pages 8–9, 2007.

University Kiel, Institute of Computer Science and Applied Mathematics, Kiel,
2004.

[MKDTZ05] H. Ma, K.-D. Schewe, B. Thalheim, and J. Zhao. View integration and cooperation
in databases, data warehouses and web information systems. Journal on Data

[MNST07a] T. Moritz, R. Noack, K.-D. Schewe, and B. Thalheim. Intention-driven screenogra-


[ST04b] K.-D. Schewe and B. Thalheim. Web Information Systems, chapter Structural me-
dia types in the development of data-intensive web information systems, pages
34–70. IDEA Group, 2004.

[ST04c] K.-D. Schewe and B. Thalheim. The co-design approach to wis development in
e-business and e-learning applications. In Workshop Fragmentation versus Integrat-
ion, co-located with Fifth International Conference on Web Information Systems

[ST04d] K.D. Schewe and B. Thalheim. Web information systems: Usage, content, and


Chapter V

Systematic Development of Internet Sites: Extending Approaches of Conceptual Modeling

Bernhard Thalheim
Brandenburg University of Technology at Cottbus, FRG

Antje Düsterhöft
University of Applied Science Wismar, FRG

ABSTRACT

Internet information services are developed everywhere. Such services include content generation and functionality support that have to be modeled in a consistent way. Here, we show how concepts of conceptual modeling can be used for systematic development of Internet sites. We give an introduction into our methodology of modeling Internet applications resulting in the Cottbus Internet site development language (SiteLang). The language has an operational semantics based on entity-relationship structuring and abstract state machines. It allows specification of entire Web sites, i.e., of structuring, behavior, information support, and of the interaction and story space. The methodology supports applications in different environments, e.g., Internet Web sites, WAP technology, or TV channels.
INTRODUCTION

Nowadays, “Internet” is one of the main buzzwords in journals and newspapers. However, to become a vital and fruitful source for information presentation, extraction, acquisition, and maintenance, concepts have to be developed that allow flexible and adaptable services. Fundamental design concepts are required but still missing.

Internet Information Systems

In the scope of this paper, we observe a number of general tendencies that were easing Web site development.

Tendency toward Large Sophisticated Web Sites

In the beginning, Web sites of more than 500 pages were rarely encountered. Meanwhile, Web sites are becoming larger and larger and tend to provide a total and integral service according to the entire scope. The maintenance and update problem forced Web developers to reuse techniques developed for conceptual modeling of database systems.

Transfer of Business Applications to the Web

Enterprises discovered that their business can be partially transferred to the Web. The support of business processes in the Web is, however, completely different. Large and closed B2B sites are now developed and employed by almost all larger companies with exchange of products.

From Fancy Graphics Heaps toward Skillful XML Suites

In the past, Internet site development was largely understood as development of graphical site presentation. More than three-score good books are found on sites of booksellers. Meanwhile, people understood the necessity of sophisticated script languages. The advent of XML led to unification and simplification of Web site programming.

Database Middleware

Because the maintenance of consistency of Web sites can be based on database paradigms, database support has been intensively included in Web sites. Currently, information-intensive Web sites are entirely based on database support (sometimes also using incomplete systems such as mySQL).

Thus, we observe that Internet sites become similar to large information systems with a specific scope on distributivity of subsites and sophisticated interaction support.
Specific Demands of Large Internet Sites

Internet information services have to cope with a broad variety of problems beside information presentation, information generation, and graphics or multimedia design.

*Services need to be personalized in order to be acceptable for any user.* Because users are treating services in a different form, flexibility of information presentation, of query and answer dialogues, and of interpretation of user’s intentions become crucial. User abilities in asking a query are very different. User semantics differ. Users are misinterpreting results obtained through summarization and contraction, for instance, in multidimensional databases. Summarized data often carry a different semantics (Agrawal, Gupta, & Sarawagi, 1997). Users are used to a certain cultural environment.

*Personalization of information,* information content, and information representation is playing the major success factor of information services that have to cope with restricted display spaces such as videotext, WAP or phone displays, and Internet services distributed over TV nets based on set-top box technology. Usually, Internet services are developed for Internet display only and do not cope with restrictions such as channel capacity. In our Internet projects, we aimed early on multidisplay via Internet, TV, videotext, and phone display. The specific requirements led us to search for languages enabling us in flexible and user-specific displays of infons (information units).

Data may vary over time. This approach is partially supported by temporal databases (Etzion, Jajodia, & Sripada, 1998; Snodgrass, 2000). Data may have a temporal dimension based on a time algebra or a time validity for defining, querying, or modifying, may have been based on user-defined time, and may be computed under restrictions of transaction time. Temporal logics used in service must be based on branching time, because we need to take into consideration different schedules of user groups, a variety of user-group specific breakpoints. Branching time needs to be handled dynamically by generating service pages on demand (Caumanns, 2000) based on the current state of the service.

*Users act in a specific context* (Feyer et al., 2001). Their context depends on and is limited by technical environment such as available equipment, channel and server capacity currently available, and understanding of their tasks. Context is based on the tasks currently under consideration. Users are entering sites with certain occasions and intentions in their mind. Well-developed Internet sites use this knowledge for modeling scenarios of usage (Schewe & Thalheim, 2000). These scenarios can be integrated into stories and composed to the story space of an Internet service. Services may also vary according to availability of server functionality, communication channel, and
Systematic Development of Internet Sites   83

according to user’s preferences. This variation led to the ACE approach of Lewerenz (2000), where the adaption of services to heterogeneous and dynamic environments has been investigated. This approach is based on component technology (Rieckmann, 2001) and allows the derivation of the XML content whenever the page content can be specified on the basis of parametric components.

These problems lead to a misunderstanding of services, unacceptable presentations, and counterintensional use of services. Therefore, we need a way to represent these services in a consistent and concise way.

Challenges of Site Development

The Cottbusnet information site team has developed and participated in development of a number of Internet sites together with companies such as the Lausitzer Informations- und Medienzentrum GmbH: city or town information sites (currently 18; see, for instance www.cottbus.de), region service sites (five German regions), community sites for associations (2), one learning site, B2B, B2C, A2C, and B2A sites, respectively. The methodology presented in Schewe & Thalheim (2000) has been developed in parallel to the projects mentioned above. During development of Internet sites, we faced a number of difficult challenges that required a full and powerful theory of the Internet site:

• Full flexibility: Internet services often require full flexibility of a site’s scenarios. The set of scenarios necessary to be supported looks graphically similar to a complete graph (clique). It is possible in such cases to use any menu point and to jump to any other dialogue step. For instance, visitors of information sites do not want to read the same information several times. This problem is even more severe for learning sites (see for example, Caumanns, 2000; http://DaMiT.dfki.de). In this case, any student wants to see the information needed at the given moment and the amount of repetition has to be thoughtfully restricted.

• Support of tracing: Site visitors do not want to insert their data twice or read the same information. Once they identified themselves or are identifiable, they want to keep their identity during the entire site visit and have the site adaptable to their actual visit. In one of our industrial projects, for instance, we faced this requirement as the killing requirement: Based on set-top-box technology, a user can access their Internet suite via TV, depending on their previous and their actual visits. Because TV representation is limited, content display is rather limited. At the same time, interaction facilities are limited.

• Push-up content just-in-time: Users want to get the content depending on their actual specific information requirements and demand. Tools
(Caumanns, 2000) allow content systems for learning sites to be built with the ability to automatically adapt the content depending on the learning situation, depending on the learning stage and on the specific user profile. In order to support this, sophisticated views should be materialized similar to OLAP (Agrawal, Gupta, & Sarawagi, 1997; Lewerenz, Schewe, & Thalheim, 1999) approaches. These views are extended to media objects (Feyer et al., 2000) by functionality attachment and are based on the container shipment.

- **No transactional semantics**: Despite the massive parallel access and execution in Web site service, the approach to consistency enforcement applied especially to DBMS cannot be used for providing operational semantics. Users access Web sites in parallel. They complete their tasks or partially complete their task or leave the page whenever they like, expecting sometimes also partial recording. This variety has to be explicitly modeled or supported by robust systems. In both cases, modeling becomes a nightmare.

**Focus of This Paper**

This paper gives an introduction into our methodology of modeling Internet applications resulting in the *Cottbus Internet site development language* (SiteLang). The language has an operational semantics based on entity-relationship structuring and abstract state machines. It allows specification of entire Web sites, i.e., of structuring (structure, static integrity constraints), behavior (processes and dynamic semantics), information support (views, units, and containers) and of the interaction and story space (scenes, media objects, and dialogue steps). The methodology supports applications in different environments, e.g., Internet Web sites, WAP technology, or TV channels. SiteLang has been developed with the cooperation of the Cottbus team (see acknowledgement) and is based on components developed by the team members. The theoretical background of this work generalizes the approaches of Schewe (2000) and Thalheim (2000).

**BACKGROUND**

**Intensive Research in a Broad Variety of Groups**

Recent conference series such as CMWWW, Hypertext, SAC, WebDB, WebNet, or WWWW have shown a lot of research in this area. Almost all conferences, such as ER or HICSS, devote a special section to Web development. Programming an agent to search for references on Web site development...
brought us more than 1400 references. Almost in any university, a Web development group exists. There are some surveys that summarize the achievements in the Web site development business (Fraternali, 1999; Schewe & Thalheim, 2000). A rough classification of the approaches developed so far could highlight the achievements as follows [references named are only exemplary; for more complete references, see the survey mentioned above or Thalheim (2000b)]:

- Development of specific languages such as IIDM, JML, WCML, WebML, and WSDM, e.g., in Bonifati et al., 2000; Ceri, Fraternali, & Paraboschi, 1999; Gaedke & Turowski, 1999; Göschka & Schranz, 2001.
- Extending hypertext methodologies, e.g., HDM and RMM, see, for instance, Baresi, Garzotto, & Paolini, 2000; Garzotto, Paolini, & Schwabe, 1993; Isakowitz, Kamis, & Koufaris, 1998; Nanard, Nanard, & Kahn, 1998; Rossi, Schwabe, & Lyardet, 2000; Rossi, Garrido, & Schwabe, 2000; Schranz, 1998; Güell, Schwabe, & Vilain, 2000).
- Development of entire workbenches such as Araneus (Mecca, Merialdo, & Atzeni, 1999; Thalheim, 2000b).
- One of the major drawbacks of conceptual modeling research in the Internet area is its unlikeness. Paper refers mostly to own groups that do not consider the experience gained in other groups. There is no generalization of the approaches. This lack is caused by the large variety of applications and is similar to the situation in conceptual modeling, where it is commonly agreed that no common language will overtake all other approaches despite UML. UML, however, does not have any approach beyond plans to incorporate meaning or semantics (Schewe, 2000).

**Generalizable Approaches**

There are some approaches that might be adopted to the development of Internet sites.

*Workflow and Pattern*

Workflow approaches try currently to cover the entire behavior of systems beginning with modeling of implementable processes to conceptual modeling of behavior toward representation of results obtained in the analysis phase of systems development. Nevertheless, this research has led to a good understanding of processes. Workflows can be constructed on the basis of basic processes by application of basic control constructors such as sequence, parallel split, exclusive choice, synchronization, and simple merge, by applica-
tion of advanced branching and synchronization control constructors such as multiple choice, multiple merge discriminator, n-out-of-m join, and synchronizing join, and by application of structural control and state-based control constructors.

*Wegner’s Interaction Machines*

Despite the presence of an active research community that studies conceptual models for ISs, this term still lacks precise formal underpinnings. We use the interaction machine approach (Goldin, Srinivasa, & Thalheim, 2000) in order to reason formally on information services. An interaction machine (Wegner & Goldin, 1999) can be understood as Turing machine with multiuser dynamic oracles (MIM) or with single-user dynamic oracles (SIM).

As shown in Börger (2002), they can be entirely based on abstract state machines.

```plaintext
if condition
    then state = nextstate(state, database, input)
    database = update(state, database, input)
    output = input o output(state, database, input)
```

The output and the input can be structured by channels. Thus, different users can receive only an output directed to them. The input of a user different from the user is invisible to other users. Usually, the output is the concatenation of the input and the new output. We can use another function for sophisticated output functions. This model is convenient for reasoning abstractly on interactive systems properties such as reliability, consistency, and liveness (Thalheim, 2000b).

Wegner’s interaction machines approach is based on four assumptions:

- User should observe machine behavior instead of having a complete understanding of the machines’ I/O behaviors
- Modeling of systems cannot be entirely inductive. Instead, coinduction needs to be used, at least for the earlier design phases.
- Users Compete for Resources, thus, modeling includes an explicit specification of cooperation and competition.
- Interactive behavior cannot be entirely modeled on the basis of I/O behavior but rather in term of interaction streams.

*State Chart Diagrams*

The state chart approach has been proved to be useful for state-oriented modeling of database applications and more generally for use-case diagrams.
Despite its usage in UML, it has a precise semantics that can be easily integrated into ER modeling approaches. Furthermore, this approach can be extended toward modeling user interfaces (Horrocks, 1999).

**User Interaction Modeling**

Interaction involves several partners (grouped according to characteristics; group representatives are called “actors”), manifests itself in diverse activities and creates an interplay between these activities. In order to develop usable Web sites, the story of the application has to be neatly supported (Schewe & Thalheim, 2000; Thalheim, 2000b).

Interaction modeling includes modeling of environments, tasks, and actors beside modeling of interaction flow, interaction content, and interaction form (Lewerenz, 2000).

### SITE SPECIFICATION

The aim of the specification of large, information-intensive Web sites is to develop Web sites that are generatable on the basis of the specifications, which are error-prone and simple to change. Site specification might be rather complex. For this reason, we prefer the following modeling concepts:

- **Stories** are collected into the story space. The story space describes the set of all possible intended interactions of actors with the system.
- **Scenarios** are used to describe a specific utilization of the Web site by an actor at a given moment of time. We might have a large variety of scenarios. For this reason, scenarios need to be integrated into the story space.
- **Scenes** are macrosteps within a scenario. They involve a number of actors in various roles, rights, and occasions. They are supported by media objects.
- **Dialogue steps** describe specific actions of enabled actors that act in specific context. Dialogue steps are supporting different scenes. Any scene has a dialogue expression expressing the dialogue workflow.
- **Media objects** describe the “adornment” of a scene similar to the movie or drama production.

Each of the concepts has a specific representation formalism defined in the description language SiteLang and will be illustrated in the next parts.

### Story and Scenario Specification

Interaction description can be based on notions developed in linguistics,
in movie business, or for the description of graphical animations. We use the aids, concepts, and experiences for acquiring an adequate conceptual design. Let us consider the concepts in detail.

The story space can be modeled by many-dimensional (multilayered) graphs (Schewe & Thalheim, 2000):

\[
\text{Story-space(Site)} = ( \{ \text{scene} \}, E, \lambda, \kappa )
\]

\[
\text{Transition } E \subseteq \{ \text{scenes} \} \times \{ \text{scenes} \}
\]

\[
\lambda : \{ \text{scene} \} \rightarrow \text{sceneDescription}
\]

\[
\kappa : E \rightarrow \text{events} \times \text{ifCond} \times \text{postCond}
\]

The function \( \lambda \) is described in detail in Section 3.1.1. Each scene is marked by its identifier. It is associated with a media object, with a set of involved actors, the context of the scene, the applicable representation styles, and a dialogue step expression for the specification of the scene.

The function \( \kappa \) is used for the adornment of transitions by the events that allow movement from one scene to another by the interaction of actors. A transition may be blocked until a certain post condition is valid for the acceptance of the completion of the corresponding scene. A transition may be used if a condition is valid for the states of the system. Thus, we define

\[
\text{TransitionDescription } \subseteq (\text{events } \cup \text{activities}) \times \text{actors} \times \text{preConds} \times \text{postConds} \times \\
\text{priorities} \times \text{frequencies} \times \text{repetition rates}
\]

Whenever a scenario is running, a certain state of the systems may be associated with the scenario. This state might be changed during the scenario, e.g., on the basis of input information provided by the actor or by temporary information used for recording actions of the actor.

A *scenario* is a run through a system by a cooperating set of actors. A scenario may be cyclic or completely sequential. A scenario is composed of scenes, which are discussed below. We model the order of scenes on scenarios by *event sequences*. Each scene belongs to a general activity. Each scenario has its history which can be accumulated and used for the representation of the *history* of the part of the scenario visited so far. This history is used for the *escort information* (Feyer, Schewe, & Thalheim, 1998) attached to the media objects of the scenes of the scenario.
We distinguish between the abstract scenario and the run-time scenario. The latter extends the specification of abstract scenarios by instantiation of parameters according to the concrete run-time history, according to the users acting currently in the scenario, by the environment (channel, technical equipment) currently used.

The abstract scenario specification consists of the following:

- Dialogue scenes representing actions to be performed for a specific task and dialogue steps for representation of an episode
- Actors with information needs and interaction requirements
- Media object satisfying interaction requirements and information needs
- Description of the intention of the dialogue scenes and steps consistent with the goals according to the mission and outcome of the step and with the occasions depending on the aim
- Context of the dialogue scenes and steps consistent with tasks and the history, with particular attention paid to the environment according to the occasion

The following scenario is used in one of our community service Web sites for membership application:

Membership Application extends scenario Purchase with Approval

Intention: Actor wants to apply for membership

Roles: Actor as applicant

Systems: Electronic Commerce Web Server, Certification Authority, ...

Scenario

Dialogue Scene: Making Membership Application Available

Goal: Membership Application Available

Outcome: Membership Application is available

Actions: Actor invokes the membership application page

....

Dialogue Scene: Approving Payment

Occasion: Membership requires payment first

Goal: Association budget balance increased
Outcome: Certification authority has approved membership fee payment

Information need: Association account, bank details

Actions: Membership data are processed

- Web server receives payment information
- Certification authority consulted
- Certification authority approves payment
- Web server notifies notifies actor

Context: safe connection, existing application

... 

The abstract scenarios are integrated into the story space. An example of the story space of city information systems is the following brief description of the www.cottbus.de story space:

Entry { Business { Portrait1 | Technology Region | Real Estate

| Environment | Fairs, Exhibitions, Events1 | Projects | Traffic1
| Culture & Tourism { Portrait | Tourist Service Center
| Calendar of Events | Eating & Dining
| Spare time, Relaxation, Entertainment | Sport | Traffic2}

| Inhabitants { Political life | Town management | Guide for inhabitants
| Education | Social life and Associations | Traffic }

For instance, we identified more than 30 scenarios for city information sites:

- Tourist content, e.g., inhabitants permanently living in the town, inhabitants temporarily living in the town, short-term tourist, businessman on short-term trip, vacationist planning a trip for more than five days, teenager with uncertain wishes, senior with profile of interest, visitor on weekend trip, visitor of highlights and festivals
- Official content, e.g., inhabitant on search through directory of public authority, inhabitant on search for emergency cases, inhabitant orienting
In order to handle this variety, we used the trick to interview the personnel currently providing the service outside the Web area. Due to this large variety of scenarios, we integrate scenarios and stories to *story trees or graphs* for navigation, usage pattern, etc. We use approaches of case-based reasoning and decision theory. A composition of scenarios in information sites is displayed in Figure 1. We represent scenes by frameboxes and transitions with corresponding events by oval boxes.

**Scenes**

Scenes are specified on the basis of the following specification frame:

\[
\text{Scene} = ( \text{Scene-ID}, \text{DialogueStepExpression}, \text{MediaObject}, \\
\text{Actors}, \text{ActorID}, \text{Right}, \text{Tasks}, \text{Assigned Roles}, \\
\text{Representation (styles, defaults, emphasis, ...)}, \\
\text{Context (equipment, channel, particular)})
\]

The scene identification is the conceptual identification of the scene. It is extended by a run-time identification in order to distinguish scenes, to record the communication channel, and to abide by specific communication restric-
tions. The *scene* of an abstract scenario is associated with a consistent *expression of dialogue steps* considered in detail in Section 3.1.2.

A scene is supported by *media objects*. Media objects can be parameterized. Typical parameters are the representation style, the actor frame, and the context frame. Therefore, we distinguish between media objects and *run-time media objects* which are sent to the actor in a concrete run time. Furthermore, *involved actors* are specified in dependence on their profiles, tasks assigned to them, their access and manipulation rights, and their roles to be taken during visiting the scene. This specification is based on Altus (2000) and is similar to profiles of actors in information systems. It is our aim to specify generic scenes. Thus, we add the *representation styles* that can be applied to the media object of the scene. Representation depends on the equipment of the actor. In the city site projects, we have experience with different representation styles: Internet display with high-speed channel, Internet display with medium-speed display (default style), videotext, and WAP display. For instance, for videotext, any graphical information is cut or replaced by textual information. Finally, the *context of access* is specified. Access determines the display facilities. Channels can be of high or low speed. The particular of using a scene by an actor depends on the scenario history.

**Dialogue Step Specification**

Dialogue steps are the elementary units of Internet sites. They are used to represent the basic state transformations triggered by actors. The distinction between dialogue scenes and dialogue steps is not strict but depends on the application. We use media objects for scenes. Dialogue steps use media objects or parts of them. In some applications, we might use scenes consisting of one dialogue step. In this case, we can integrate the dialogue step specification directly into the scene specification.

*Dialogue steps* can be represented by tables, e.g., the dialogue steps of an actor A in a community service:

<table>
<thead>
<tr>
<th>Step name</th>
<th>On event</th>
<th>If cond</th>
<th>Then with unit</th>
<th>Using processes</th>
<th>Let manipulation</th>
<th>By actor</th>
<th>Accept on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registration</td>
<td>ClickReg(A)</td>
<td>¬Member(A) ∧ permission(A)</td>
<td>Member-ship_choices</td>
<td>Collect_Member_Data(A)</td>
<td>Insert(Member, A)</td>
<td>A</td>
<td>Paymnet(A) = ok</td>
</tr>
<tr>
<td>Withdrawal</td>
<td>ClickWDraw(A)</td>
<td>Member(A)</td>
<td>Member-Data(A)</td>
<td>Collect_reason(A)</td>
<td>Delete(Member,A) ∧ send_Message (A, Withdraw)</td>
<td>A</td>
<td>Commit (A, Withdraw)</td>
</tr>
</tbody>
</table>
Based on the properties of the actions, we conclude, for instance, that after withdrawal, a previous member cannot participate in the discussions in the community. The table frame we used is called task property frame in Paech (1998). A task property frame is defined by a task name, reasons for task involvement, an aim, a postcondition (enabled next activities), the information from the database, the information for the database, the resources (actor, resources, partner), and a starting situation (precondition, activity, priority, frequency, repetition rate).

We chose the frames above instead of the ECA frame which is used for rule triggering systems because of the necessity to be as flexible as possible in the specification. The integration of postconditions into the preconditions of the next dialogue steps is possible whenever the dialogue step has only one output or the postconditions are identical for all incoming dialogue steps. Also, the trigger framework is too weak and contradictory (Thalheim, 2000).

**Dialogue scenes** are represented by frameboxes. We represent dialogue steps by ellipses. The transitions among dialogue steps are represented by curves. Thus, we obtain the graphical representations displayed in Figure 3. The additional information attached to dialogue steps can be omitted whenever we want to have a schema surveyable.

We adopt the graphical notation developed for state charts, e.g., the default start step of a scene is denoted by a solid circle, the end state by a solid circle surrounded by an empty circle, the history entry into a scene is denoted by an “H” surrounded by an empty circle. Furthermore, we can adopt

**Figure 2: Representation of dialogue steps within a dialogue scene.**
refinement and clustering, concurrency, delays and time-outs, transient states, event priorities, and parameterized states.

The preconditions can be specified on the basis integrity constraint pattern (Thalheim, 2000): condition, localization (local restrictions, navigation, etc.), validity condition, exception condition, enforcement pattern, and conditional operation (for example, synchronization conditions). Further, the precondition can be extended by additional context restrictions. The postcondition is used for description of acceptance conditions. If those are invalid, the entire dialogue step is not accepted.

Search is a general functionality of a site. The search can be extended to booking. This frame is supported by the scene demonstrating the search and booking of a hotel. The entry dialogue step enables a number of choices, such as search via categories (depending on the categories specified for events, restaurants, hotels) or properties (date, time, location, etc.), search via the map and locations, search via direct targeted input, or search via full display. On the map, several locations are noted. The user can select those for targeted input. There are also points of interest that might be used for targeted search. If the actor however chooses the full display option, then the clarification step allows restriction of the results of the search step by step. At each step, the actor can leave the scene. Thus, each step might be the end step. By default, we assume this behavior. Defaults are not displayed. The presentation of search results depends on the scene, the restrictions the actor made for the scope of the search. For instance, in the hotel search scene, we used the following presentation structure:

\[
\text{hotelObject} = (\text{identification(name, hotelObject's link, area (in raw map)), rooms (# per category), prices (range per category), equipment (icon-based))}
\]

\[
\text{hotelObject's link} = \text{additional (address, contact info, picture(s), video, map (with different travel details), travel info, location (of closest places of interest, distance info) + reservation connection)}
\]

Site Implementation

The Onion Approach

Layering of applications is one of the general approaches used in computer science. We observe that Web site functionality and content may be layered as well. On the outer layer, the presentation facilities may be introduced. Typical
functions are style and context functions. Containers are used to ship the information, and the functionality is provided by the Web engine. Their functionality is described in Feyer, Schewe, & Thalheim (1998). Containers contain information units that in general are views enriched by functions for operating on the views. Views may provide information to the dialogue or may be used for updating the database. Further, views may be materialized. If they are materialized, then the view handler provides an automatic refreshment support. Thus, we can use the onion system architecture displayed in Figure 3.

*The DBS Functionality Behind*

In Feyer, Schewe, & Thalheim (1998), the codesign approach has been developed. This approach has been generalized to media objects. Media objects consist of abstract containers, supplied DBMS processes, and database manipulations requests. Basic media objects (Schewe, 2000) are characterized by syntactic expressions, have a semantical meaning, and are used within a certain pragmatical framework.

During run time, the media object is extended by specific escort information (Feyer, Schewe, & Thalheim, 1998). This escort information is represented for user support. It allows the user to see the history of steps performed before being in the current state. Escort information is further generated from the story space. In this case, a user is informed on alternative paths that could...
be used to reach the given scene and which might be used for backtracking from the current scene.

For the generation of media objects and their composition on the basis of information units, we extend the classical SQL frame:

```
SELECT Structure FROM DB relations WHERE Condition
GROUP BY Substructure HAVING Condition
```

to the frame

```
GENERATE MAPPING: Vars -> Structure
FROM Views WHERE Selection condition
REPRESENT USING Style guide & Abstraction
BROWSING DEFINITION Condition & Navigation
```

In the case of the cultural events, we obtain:

Media object (CulturalEventSelectionPanel) =

Structure of container

unit todayEvents =

```
generate EventsOf (:Today)
<[EventID]Title,Location,Time,
link(newPage(EventID)>
```

from EventGeneral, Scheduled, At

where ....

represent using paragraphStyle

browsing definition longList

navigationByScrolling

unit tomorrowEvent ....

unit thisWeekEvent ....
Views, and therefore, the media object may have hidden parameters (for instance, EventID) that are not visible to the actor. They can be parameterized by variables (for instance, :Today). For media objects, we reuse ideas developed for OLAP technology (Thalheim, 2000): Views on ER schemata (abstraction on schemata (aggregation, scoping,...), versions, variations of generation functions, display with canonical functionality (drill-down, roll-up, rotate, pivot, push, pull, dimension, aggregation), using generic evaluation functions and models, implicit incorporation of hierarchies and implicit incorporation of time, space,....

Scenes are dynamically adapted to the environment, to the specific profile of the actor, and are extended by the escort information. The specific representation is selected among the possible representations of the media object. The context of the media object also depends on the scenario history at run time. Thus, we extend the context of the container by the run-time history depending on the scenario chosen by the actor acting on the site. In order to handle the dynamic changes and the history, we record the control information in the scene control component. Finally, the specific actor might obtain during the scenario an identity or other specific information such as products selected for later purchase. This information is stored in the component for scene actors. For instance, given an Internet channel with medium capacity, the entry page of a city’s information sites may be composed by a star graphics, extended by additional information such as town maps, imprint, and contact information. For example, in this case, the www.cottbus.de site has the following entry scene specification:

\[
\text{Add (Add (Add (ComposeGen ( genFrame, ComposeStar(t1, (t2, t3, t4), ov)), NavQuick (n1, n2, n3), style2 ), Text (entry), style1), NavQuick (townMaps) + imprint + cont1, derivStruct(site), style2))}
\]
The corresponding media object which is the central object of the entry page uses the following queries and navigation elements:

\[ t_1 = \text{emblem(pyramid)} \ldots t_3 = \text{RollNav( emblem(tourist), tourPage, repres1)} \]

\[ n_1 = \text{Sub( Query( calend(today), order1), repres2)} \]

\[ \text{entry} = \text{“Known as the city ... <a>Energie Cottbus</a> ...”} \]

\[ \text{cont1} = \text{ComposeSub( email, guestsBook, seq1)} \]

The event database is used by different parts of the Web site. Some of the events are accessed frequently. Thus, we use materialization of components of sites. Pages are as follows:

- **Static**, i.e., they have been developed once and stay for a longer period
- **Pseudo-static**, i.e., generated and transformed to static pages whenever an update of the corresponding data is made
- **Fully dynamic**, i.e., are generated during processing the request of an actor

**The XML-Perl Functionality Behind**

The language Perl neatly support the onion approach, because we can use Perl for conservative expansion of views to units, of units to containers, and of containers to presentation objects. Further, the database connectivity is entirely supported. This onion generation mechanism is also used for the expansion of XML specifications to page objects.

**CONCLUSION**

In this paper, we presented the Web site specification language Site Lang. This language is based on a layering of Web site specification into the story space, dialogue scenes, and dialogue steps. An aim during specification is to entirely include all main scenarios in the application into the scenes and the story space. A scenario is, in this case, a run through the scenes and the story space. There might be more utilization scenarios in the interaction space. But, we are interested in a complete inclusion of the main scenarios. If the system is supporting more than the scenarios of the interaction space, then we can restrict the system. Our aim is, however, to develop a language that allows us to specify Web sites. The main advantage of our approach is its well-foundedness. Based on the extended ER model, on the abstraction layer approach, and on the
abstract state approach, we obtain an operational semantics that can be used for derivation and proof of properties. At the same time, we can use abstract state machines for generation of executable prototypes of the Web site.

The specification of a Web site can also be made manually. However, it is definitely preferable to develop, to extend, and to maintain a site via automated tools. Such tools allow generation of the site content, the page presentation, the navigation structure, the indexing support, etc. The next step in our team is the development of such tools.

We are concentrated on information-intensive and large Web sites. Small Web sites may not need any specification languages. Web sites that are not information-intensive can be developed in a similar fashion. The SiteLang supports Web site specification for e-commerce sites, for information sites, for community sites, and for learning sites. Identity sites are usually smaller if they do not include specifics of e-commerce or community sites. Also, entertainment sites are not within the scope of this paper, because quality criteria to be applied to such sites (Schewe & Thalheim, 2000) are not oriented toward content but rather toward presentation and multimedia objects.

ACKNOWLEDGEMENT

The research presented and summarized in this paper is based on the work of the Internet services Cottbusnet team at Cottbus Tech. We are very thankful to Ph.D. and Master students working and contributing in these projects, and thankful to our Master students handling the implementations and developments within these projects, and furthermore, to our commercial partners supporting and running the implementations. The research represented here is based on a tight collaboration with K.-D. Schewe (Schewe & Thalheim, 2000).

BIBLIOGRAPHY


The Co-Design Approach to WIS Development in E-Business and E-Learning Applications

Klaus-Dieter Schewe\(^1\), Bernhard Thalheim\(^2\)

\(^1\) Massey University, Department of Information Systems & Information Science Research Centre
Private Bag 11 222, Palmerston North, New Zealand
k.d.schewe@massey.ac.nz

\(^2\) Christian Albrechts University Kiel
Department of Computer Science and Applied Mathematics
Olshausenstr. 40, D-24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de

Abstract. We argue that the generic co-design approach to the development of web information systems is powerful enough to cover diverse applications. We illustrate the validity of this claim by comparing the key features of the approach with needs arising in e-business and e-learning applications. While this does not exclude that there still exist application-specific features that need to be handled separately, it underlines that it is advisable to consider the most generic approach first. In addition, it is recommendable trying to generalise specific features that arise in e-business and e-learning to the whole area of web information systems.

1 Introduction

Data-intensive information systems are a well-researched field that has found a lot of applications. Over the last decade there has been a shift of interest toward web information systems (WISs), i.e. information systems that are made available via the world-wise web. This area also has found a lot of applications in information services [10], electronic learning [5, 4, 3, 11, 12, 19], electronic business [14, 15], and others.

In these application fields, in particular in Electronic Business (EB) and Electronic Learning (EL) we now find groups of researchers claiming the field to be not just a field of application but a genuine research area in its own right. Of course, both areas have to link to business or education, respectively, but the question is, whether there are enough research problems that would lead to substantively different results than those achieved for WISs, i.e. on a more generic level.

In this position paper we argue that it is better to treat them still as application areas for WISs and not as separate fields in their own right.
In order to present evidence for this claim we will briefly sketch the key concepts of the generic co-design approach to the development of web information systems [16], i.e. storyboarding [8] and media types [9]. Both components have been deeply analysed with respect to their expressive power in [18, 16, 17] and [13, 16], respectively. Both approaches have also been tailored to applications in EB [15] and EL [19].

We will demonstrate that key features such as story spaces, actors, media types and adaptivity each have a specialisation in EB and EL. That is, the generic co-design is very well reflected in the application area, and most of the specific needs arising from the application will already be captured by the approach. While this does not exclude that there still exist application-specific features that need to be handled separately, it underlines that it is advisable to consider the most generic approach first.

There are lots of other generic approaches to the development of WIS such as ARANEUS [1, 2], WebML [6, 7] and OOHDM [20]. We do not want to discuss the individual merits of these approaches, their advantages and disadvantages (see [13, 17, 16] for some comparison). We are confident that our argument will also hold largely for these other approaches.

2 Story Spaces

On a high level of abstraction we may think of a WIS as a set of abstract locations, which abstract from actual pages. A user navigates between these locations, and on this navigation path s/he executes a number of actions. We regard a location together with local actions, i.e. actions that do not change the location, as a unit called scene.

Then a WIS can be described by a edge-labelled directed multi-graph, in which the vertices represent the scenes, and the edges represent transitions between scenes. Each such transition may be labelled by an action executed by the user. If such a label is missing, the transition is due to a simple navigation link. The whole multi-graph is then called the story space. A story is a path in the story space. It tells what a user of a particular type might do with the system.

The combination of different stories to a subgraph of the story space can be used to describe a “typical” use of the WIS for a particular task. Therefore, we call such a subgraph a scenario. Usually storyboarding starts with modelling scenarios instead of stories, coupled by the integration of scenarios to the story space. At a finer level of details we may add a triggering event, a precondition and a postcondition to each action, i.e.
we specify exactly, under which conditions an action can be executed and which effects it will have.

Looking at scenarios or the whole story space from a different angle, we may concentrate on the flow of actions:

- For the purpose of storyboarding actions can be treated as being atomic, i.e. we are not yet interested in how an underlying database might be updated. Then each action also belongs to a uniquely determined scene.
- Actions have pre- and postconditions, so we can use annotations to express conditions that must hold before or after an action is executed.
- Actions can be executed sequentially or parallel, and we must allow (demonic) choice between actions.
- Actions can be iterated.
- By adding an action \texttt{skip} we can then also express optionality and iteration with at least one execution.

These possibilities to combine actions lead to operators of an algebra, which we will call a \textit{story algebra}. Thus, we can describe a story space by an element of a suitable story algebra. We should, however, note already that story algebras have to be defined as being many-sorted in order to capture the association of actions with scenes.

In the area of EL the story space corresponds to the curriculum, which can be modelled by an \textit{outline graph} that describes the navigation of learners through an e-learning system [3]. Formally, an \textit{outline graph} is a finite, directed graph $G = (V, E)$, i.e. $V$ and $E$ are finite sets with $E \subseteq V \times V$. The vertices, i.e. the elements of $V$, are called \textit{learning units}, and the edges, i.e. the elements of $E$ are \textit{links} between these units. Thus, scenes in story spaces correspond to learning units in EL systems.

Take for example a course dealing with e-learning systems. Then we might have learning units such as Introduction, Learner Profiling, Course Outlining, Data Management, Adaptivity, Style Definition, and Implementation.

Actions on a learning unit may depend on the successful completion of the learning unit by the learner. According to our view this is part of the action specification, and should be left for further refinement of the outline.

From a more conceptual point of view we model learner interaction with an e-learning system according to two primitives: transition between learning units and using the functionality offered at a given learning unit. This allows for a two-step modelling procedure to be applied. First at
a coarse-grained level learning units and navigation links are modelled. Then in a refinement step the actual activities of the learners are added. This procedure allows for a good separation of concern with respect to personalisation and localisation.

In the area of EB, the notions of story space and scenario have been adopted directly [14]. Describing the application story for e-business systems permits concentration on the business aspects. In particular, the accentuation of the communication aspects can be more easily approached on such a level of abstraction. Thus, it satisfies the primary criterium for conceptual modelling of not being implementation-biased. Furthermore, the approach is both grounded in solid mathematical theory such as Kleene algebras and deontic logic [16], and in the sophisticated pragmatics of a successful business branch – movie production – which may be considered a godfather for the approach.

3 Actors

A second key feature of storyboarding concerns the description of the actors, i.e. groups of users with the same profile. Users can be classified according to their roles, intentions and behaviour. We use the term actor for such a group of users. The role of an actor indicates a particular purpose of the system. As such it is usually associated with obligations and rights, which lead to deontic integrity constraints. Roles are also connected with the tasks, but tasks will be handled separately. The intention of an actor can be modelled by goals, i.e. postconditions to the story space. Modelling the behaviour of an actor leads to user profiles, which can be modelled by giving values for various properties that characterize a user. Furthermore, each profile leads to rules that can again be expressed by constraints on the story space.

In addition, each actor has an information portfolio, which specifies the information needs as well as the information entered into the system. We do not model the information portfolio as part of an actor, but instead of this we will model the information “consumed” and “produced” with each more detailed specification of a scene. However, we associate information consumption and production with each scene of the story space. Assuming that there is a database for the data content of the WIS with database schema $S$, information consumption on a scene $s$ definitely accounts for a view $V_s$ over $S$. That is, we have another schema $S_V$ and a computable transformation from databases over $S$ to databases over $S_V$. 
Such a transformation is usually expressed by a query $q_V$. Such views will form the basis of the theory of media types.

The presence of roles indicates a particular purpose of the system. For instance, in a web-based conference system we may have roles for the programme committee chair(s), the programme committee members, and for authors. On the other hand, in an on-line loan system we may not wish to distinguish roles, as all actors will only appear in the one role of a customer.

A role is defined by the set of actions that an actor with this role may execute. Thus, we first associate with each scene in the story space a set of role names, i.e. whenever an actor comes across a particular scene, s/he will have to have one of these roles. Furthermore, a role is usually associated with obligations and rights, i.e. which actions have to be executed or which scenes are disclosed.

An obligation specifies what an actor in a particular role has to do. A right specifies what an actor in a particular role is permitted to do. Both obligations and rights together lead to complex deontic integrity constraints. The intention of an actor can be expressed by goals, which can be modelled by postconditions to the story space.

Modelling the behaviour of an actor leads to user profiles. We may ask which properties characterize a user and provide values for each of these properties. Each combination of such values defines a profile, but usually the behaviour for some of these profiles is the same. Furthermore, each profile leads to rules that can again be expressed by constraints on the story space.

The dimensions used in user profiles depend on the application. Formally, in order to describe such user profiles, we start with a finite set $\Delta$ of user dimensions. For each dimension $\delta \in \Delta$ we assume to be given a scale $sc(\delta)$. Formally, a scale is a totally ordered set. If $\Delta = \{\delta_1, \ldots, \delta_n\}$ is a set of user dimensions, then the set of user profiles is $gr(\Delta) = sc(\delta_1) \times \ldots \times sc(\delta_n)$. A user type is a convex region $U \subseteq gr(\Delta)$.

The analogue of the actor in EB and EL applications are the customer and the learner, respectively. In particular, user types become customer types or learner types, respectively. The only decisive feature in both application areas is the decision on the relevant user dimensions and the associated scales. Formally, however, there is no need to introduce a particular new theory.
4 Media Types

The central concept for modelling the content and the functionality of a WIS is the media type. We may assume that we have an underlying database. On the basis of a given database schema we may introduce interaction types as views that are extended by operations. This permits to apply completely different design criteria for the database schema and the interaction schema. One major facility used in interaction types is the possibility to create a navigation structure via URLs and links. Media types arise from tailoring the information types in such a way that different presentation options will be enabled.

A media type is an interaction type $M$ together with an cohesion order $\preceq_M$ (or a set of promimity values) and a set of hierarchical versions $H(M)$.

Cohesion introduces a controlled form of information loss. Formally, we define a partial order $\leq$ on content data types, which extends sub-typing. If $\text{cont}(M)$ is the content data type of a interaction type $M$ and $\sup(\text{cont}(M))$ is the set of all content expressions $\text{exp}$ with $\text{cont}(M) \leq \text{exp}$, then a preorder $\preceq_M$ on $\sup(\text{cont}(M))$ extending the order $\leq$ on content expressions is called an cohesion preorder.

Small elements in $\sup(\text{cont}(M))$ with respect to $\preceq_M$ define information to be kept together, if possible. Clearly, $\text{cont}(M)$ is minimal with respect to $\preceq_M$. This enables a controlled form of information decomposition. If we want to decompose an interaction type or if we are forced to decompose according to user requirements or technical restrictions, then we may choose a minimal elements $t_1 \in \sup(\text{cont}(M))$ with respect to $\preceq_M$ such that it satifies the representation requirements. Note that if we only provide a preorder, not an order, then there may be more than one such $t_1$.

Taking just $t_1$ instead of $\text{cont}(M)$ means that some information is lost, but this only refers to the first data transfer. When transferring $t_1$, we must include a link to a possible successor containing detailed information. In order to determine such a successor we can continue looking at all content types $t' \in \sup(\text{cont}(M))$ with $t_1 \not\preceq_M t'$. These are just those containing the complimentary information that was lost. Again we can choose a least type $t_2$ among these $t'$ with respect to $\preceq_M$ that requires not more than the available capacity. $t_2$ would be the desired successor.

Proceeding this way the whole communication is broken down into a sequence of suitable units $t_1, t_2, \ldots, t_n$ that together contain the information provided by the interaction type. Of course, the cohesion pre-order
suggests that the relevance of the information decreases, while progressing with this sequence. The user may decide at any time that the level of detail provided by the sequence \( t_1, \ldots, t_i \) is already sufficient for his/her needs. An alternative to cohesion preorders is to use proximity values, which we do not discuss here.

Another possibility to tailor the information content of interaction types is to consider dimension hierarchies as in OLAP systems. Flattening of dimensions results in information growth, its converse in information loss. Such a hierarchy is already implicitly defined by the component or link structures, respectively.

For an interaction type \( M \) let \( \bar{H}(M) \) be the set of all interaction types arising from \( M \) by applying a sequence of flat-operations or their converses to interaction types or underlying database types. A set of hierarchical versions of \( M \) is a finite subset \( H(M) \) of \( \bar{H}(M) \) with \( M \in H(M) \). Each cohesion order \( \preceq_M \) on \( M \) induces an cohesion order \( \preceq_{M'} \) on each element \( M' \in H(M) \).

Media types are used directly in EL applications [12] and in EB applications [13].

5 Adaptivity

In order to avoid an unnecessary replication of WISs with the undesired increase in maintenance costs we prefer to have a single server-side specification of the system, which adapts itself to various needs of the clients. Basically, we see two dimensions of this adaptivity problem.

The first one concerns the distinction between the adaptivity of content and process. Processes determine which functionality becomes available to users and which not. Process adaptivity thus results in discarding access to part of the system and to reduce the number of available actions. Content adaptivity on the other hand refers to tailoring the data content that is presented to a user. This may result in reducing the content, splitting or merging it, or aggregate it to a more compact representation.

The second dimension of adaptivity concerns the parameters that determine the adaptation. Here we distinguish between adaptivity to users, locations, channels and devices. With respect to the users we consider preferences of users and requirements resulting from profiling characteristics of users. With respect to locations we consider the actual location of a user. With respect to channel we consider the communication channel used by a user, e.g. fixed line, telephone, broadband, mobile, TV cable-
Content adaptivity can be approached on the level of media types. In fact, that is exactly the purpose of cohesion. We already described the adaption procedure. Obviously, there is no need for a particular theory in the areas of EL or EB.

Process adaptivity to can be obtained on the level of the story space. For this we may exploit the fact that a story space can be described by an expression in a many-sorted Kleene algebra with tests (MKATs) [?,16]. The general approach is as follows. We express the story space or parts of it by some process expression \( p \). We then formulate a problem by using equations or conditional equations in this MKAT. Furthermore, we obtain equations, which represent application knowledge. This application knowledge arises from events, postconditions and knowledge about the use of the WIS for a particular purpose. We then apply all equations to solve the particular problem at hand.

The equations that define the application knowledge formulate preferences of a user. Furthermore, we formalised the intention of a user by some propositional formula \( \psi \). Then the problem is to find a minimal expression \( p' \) in the MKAT such that \( p\psi = p'\psi \) holds.

The equational reasoning with MKATs used for process adaptivity is independent from the application. The application area only determines the equations and the goal formulae. In EL these correspond to learning style and learning goals [11]. In EB no particular notions were introduced.

6 Conclusion

In this position paper we claimed that the generic co-design approach to the development of web information systems is powerful enough to cover diverse applications, in particular in EB and EL, and therefore it would not be justified to treat EB and EL as fields in their own right. We demonstrated the validity of this claim by comparing the key features of the approach with corresponding features in these two application areas.

From this we may conclude that a generic theory for WISs is needed, and that the development of such a theory is the preferable scientific approach. This does not exclude that there still exist application-specific features that need to be handled separately. In addition, it is recommendable trying to generalise specific features that arise in applications to the whole area of WISs.
The Co-Design Approach to WIS Development

References


Chapter II

Structural Media Types in the Development of Data-Intensive Web Information Systems

Klaus-Dieter Schewe, Massey University, New Zealand

Bernhard Thalheim, Brandenburgian Technical University, Germany

ABSTRACT

In this chapter, a conceptual modeling approach to the design of web information systems (WIS) will be outlined. The notion of media type is central to this approach. Basically, a media type is defined by a view on an underlying database schema, which allows us to transform the data content of a database into a collection of media objects that represent the data content presented at the web interface. The view is extended by operations and an adaptivity mechanism, which permits the splitting of media objects into several smaller units in order to adapt the WIS to different user preferences, technical environments and communication channels. The information entering the design of media types is extracted from a previous story boarding phase. In consecutive phases, media types have to be extended by style patterns as the next step toward implementation.
INTRODUCTION

In this chapter, we address the conceptual modeling of web information systems following the abstraction layer model that was already presented in another chapter of this book (Kaschek et al., 2003). We concentrate only on the structural aspects, i.e., operations will not be discussed. Thus, the central task will be the specification of the data content that is to be made available on the Web. The goal is to provide conceptual means for describing the content in a way that it can be tailored to different users, different end-devices and different communication channels without designing multiple systems.

The chapter will guide the reader through a three-layer model of describing such data. Describing the structure of the data as it is presented on the Web will lead to defining the structure of media types. However, these structures will be full of redundancies and, thus, hard to maintain as such. Therefore, the data has to be restructured in order to define a suitable database schema, which defines the second layer. As database design follows different objectives, the content specification should lead to views, i.e., transformations, which turn the content of a database into the content of a media type. The third layer is made up by data types, i.e., immutable sets of values that can be used in the description of the other two layers.

Thus, a media type will basically be defined by a view on an underlying database schema, which allows us to transform the data content of a database into a collection of media objects that represents the data content presented at the web interface. Then, we extend media types in a way that they become adaptive to users, devices and channels. The adaptivity of a media type permits the automatic splitting of media objects into several smaller units, allowing a user to retrieve information, in a step-by-step fashion, with the most important information presented first. The reader will see two different ways to specify which data should preferably be kept together, and how this will impact on the splitting of media objects.

The result of conceptual modeling will be a media schema, i.e., a collection of media types, which adequately represents the data content of the story board. In this chapter, we will formalize this idea of conceptual modeling of web information systems by the use of media types. In the remainder of this chapter, we will first look at related work on conceptual modeling of web information systems. This will provide the reader with the necessary framework of the theory of media types. In a second step, we will illustrate in detail, but quite informally, the central ideas underlying media types. This is to convince the reader about the naturalness of the approach. The third and fourth steps are
devoted to introducing, formally, data types, database schemata, and media
types on top of such schemata. The adaptivity extension will be introduced last,
before we summarise the chapter, put again the media types into the context of
the abstraction layer model and discuss follow-on activities, and draw some
conclusions. Throughout the chapter, we will use a simple bottleshop example.

**RELATED WORK**

There are a few major groups working on conceptual modeling of web
information systems. One of them is the group around Paolo Atzeni (University
of Rome), who defined the ARANEUS framework (Atzeni et al., 1998). This
work emphasises that conceptual modeling of web information systems should
approach a problem triplet consisting of content, navigation and presentation.
This leads to modeling databases, hypertext structures and page layout.

However, the ARANEUS framework does not explicitly separate be-
tween the content at the database layer and the web information system layer,
i.e., the aspect of dealing with views. The navigation is not treated as an
integrated part of such views, but more as an “add-on” to the content
specification. Besides navigation, no further operations that could cover the
functionality of the system are handled. There is quite a fast drop from
conceptual modeling to the presentation, instead, of a more sophisticated work
on the presentation and implementation aspects. The conceptual modeling
approach is not integrated in an overall methodology, e.g., the aspect of story
boarding is completely neglected. Adaptivity, with respect to users, used
technology, and channels is not incorporated into the conceptual model.

Of course, the work in the group continues and addresses most of these
issues. Also, other authors refer to the ARANEUS framework. The work in
Baresi et al. (2000) addresses the integrated design of hypermedia and
operations, thus, picking up the functionality aspect but remaining on a very
informal level. Similarly, the work in Bonifati et al. (2000) presents a web
modeling language WebML and starts to discuss personalisation of web
information systems and adaptivity, but, again, is very informal.

Another group working on integrated conceptual modeling approaches is
the one around Gustavo Rossi (Argentina) and Daniel Schwabe (Brazil), who
developed the OOHDM framework (Schwabe et al., 1996), which is quite
similar to the ARANEUS approach. A major difference is that the Roman
group has its origins in the area of databases, whereas the Latin American group
originally worked in the area of hypertext (Schwabe & Rossi, 1998).
The OOHDM framework (see also Rossi et al., 2000) emphasizes an object layer, hypermedia components, i.e., links (discussed in more detail in Rossi et al., 1999)) and an interface layer. This is, more or less, the same idea as in the work of the Roman group, except that Rossi and Schwabe explicitly refer to an object oriented approach.

A third group working on integrated conceptual modeling approaches to web information systems is the group around Stefano Ceri and Piero Fraternali (Polytecnico di Milano) (see Ceri et al., 2002 and Fraternali, 1999). The work emphasizes a multi-level architecture for the data-driven generation of web sites, thus, taking the view aspect into account. The work addresses the personalisation of web sites by providing user-dependent site views, thus, being aware of the problem of adaptivity. The work emphasises structures, derivation and composition, i.e., views, navigation and presentation, thus, addressing the same problem triplet as the ARANEUS framework. Remaining differences to our work on media types are the fast drop from the conceptual level to the presentation, the treatment of navigation as an “add-on” and not an integrated part of the views, and the missing emphasis on higher-level methods such as story boarding. Besides that, the theory of media types is formally more elaborate.

Our own work started with the Cottbusnet project, addressing the design and development of a regional information service (see Thalheim, 1997) for a detailed description of the project, its approach, and the achieved results). As a large project arising from practice, all problems had to be addressed at the same time. This even included the demand for adaptivity to different end-devices. The research challenge was to formalise the concepts and to bring them into the form of an integrated design and development methodology for web information systems.

This resulted in a methodology oriented at abstraction layers and the co-design of structure, operations and interfaces (see Schewe & Thalheim 2000). Central to the methodology is the story boarding (Feyer et al., 1998; Feyer & Thalheim, 1999). The work in Düsterhöft and Thalheim (2001) contains an explicit language SiteLang for the purpose of story boarding. The work in Schewe et al. (2002) applies story boarding to electronic banking. For the conceptual level, the methodology provides the theory of media types (Feyer et al., 1998; Feyer et al., 2000) — more elaborate work on this subject is contained in Schewe and Thalheim (2000). The theory of media types addresses the objectives described in the previous subsection. It will be described in detail in the remainder of this chapter.
Finally, there is uncountable work on the *eXtensible Markup Language* (XML). The Roman group has investigated how ARANEUS could be supported by XML (Mecca et al., 1999), and we did the same with the theory of media types (Kirchberg et al., 2003). However, as Lobin emphasises in Lobin (2000), XML should be considered as a “bridge” between the areas of databases and the Web; and, as such, is neither completely part of databases nor of web documents. Therefore, it is debatable whether XML should be treated as a new data model or just kept for modeling views.

**CONCEPTUAL MODELING OF CONTENT AND ADAPTIVITY**

The content aspect concerns the question: Which information should be provided? This is tightly coupled with the problem of designing an adequate database. However, the organisation of data that is presented to the user via the pages in a web information system differs significantly from the organisation of data in the database. We conclude that modeling the content of a web information system has to be addressed on at least two levels: a logical level leading to databases, and a conceptual level leading to the content of pages. Both levels have to be linked together.

Consider an arbitrary web page. Ignore all the fancy graphics, colours, etc., and concentrate on the data content. For instance, consider a page used in a bottleshop site, showing the label on a wine bottle together with a description of the wine. You can describe the content saying that it consists of the picture of the label, the name of the wine, its year, the description of the winery, the used grapes, information about colour, bouquet, acidity level, residue sugar, and many other details. You can write down this description in a formalised way, e.g.:

```
```

The components of this complex value need not only be strings, tuples, numbers, etc. They can be also complex values, such as tuples, sets, or URLs representing links to other pages.
The URL associated with the arbitrary web page can be considered as an abstract identifier associated with this complex value. Thus, an adequate abstract description would be a pair \((u, v)\), where \(u\) is a URL and \(v\) is a complex value, as shown above. We decide to call such a pair a media object. The term ‘object’ is used because pairs constructed out of an abstract identifier and a complex value are called ‘objects’ in the context of object oriented databases. Here, they refer to ‘media,’ as their collection describes the content of the whole web information system as some kind of electronic media.

**Content Types of Media Types**

The example suggests that several raw media objects share the same structure. Continuing the bottleshop example, there may be lots of different wines but, in all cases, the structure of the complex value would almost look the same. Classification abstraction means to describe the common structure of these values. Formally, we can introduce data types, such as \(STRING\) for character strings, \(NAT\) for natural numbers, \(DATE\) for dates, etc. We may even have composed types, such as \(ADDRESS\). In general, we may think of a data type as providing us with a set of values. Thus, the wines in the bottleshop examples could be described by the following expression:

\[
(name : STRING, picture : PIC, year : NAT, winery : (name : STRING, region : STRING, country : STRING, address : ADDRESS), composition : \{ (grape : STRING, percentage : NAT) \}, \ldots)
\]

We have taken the freedom to assume that we may use a data type \(PIC\) for pictures. We may also assume to be given a type \(URL\) for all the possible URLs. This data type also uses the notation \((\ldots)\) for tuple types, i.e., all possible values are tuples, and \(\{\ldots\}\) for set types, i.e., all possible values are sets.

We call an expression as the one above a content type. Content types are a relevant part of media types. In our example, we would define a media type with the name WINE and specify that the expression above is its content type. In general, every data type can become the content type of a media type. Thus, if \(M\) denotes a media type, the media objects of type \(M\) are pairs \((u, v)\), consisting of a value \(u\) of type \(URL\) and a value \(v\) of the content type of \(M\). We call \(u\) the URL of the media object and \(v\) its content value.

For example, if we define the content type of a media type WINE as above, a media object of type WINE could be:

taking a “virtual” URL and the complex value that we started with.

A little subtlety comes in here, which makes the definition still a bit more complicated. When a value of type URL appears inside the content value \( v \), this may be a URL somewhere outside the Web Information System that we want to develop. However, in the case where the URL is an internal one, it will be the URL of a media object, say of type \( M’ \). For instance, for a wine, i.e., a media object of type \( \text{WINE} \), the description of the winery may involve a history component. As this history can be a long structured text on its own, we may separate it from the wine and place it on a separate page that is reachable by a navigation link. However, we would always get a link to (the URL of) a media object of type \( \text{WINERY\_HISTORY} \).

Therefore, we extend content types in such a way that, instead of the type URL, we may also use links \( \ell : M’ \), with a unique link name \( \ell \) and a name of a media type \( M’ \). In our \( \text{WINE} \) example, the content type would change to

\[
(\text{name} : \text{STRING}, \text{picture} : \text{PIC}, \text{year} : \text{NAT}, \text{winery} : (\text{name} : \text{STRING}, \text{region} : \text{STRING}, \text{country} : \text{STRING}, \text{address} : \text{ADDRESS}, \text{history} : \text{WINERY\_HISTORY}), \text{composition} : \{(\text{grape} : \text{STRING}, \text{percentage} : \text{NAT})\}, … )
\]

We used a bit of meta-level syntax here: small capitals are used for the names of media types, italic letters for data types, and normal letters for link names and other labels.

In order to obtain a media object, we would have to replace the links \( \ell : M’ \) by the datatype URL first. However, the link \( \ell : M’ \) would force us to use only values of type URL that are URLs of the media type \( M’ \).

In summary, the content of a web page may be described as a complex value. Combining a complex value with the URL for the web page forms a media object. Several media objects may share the same structure; in other words, the complex values share the same structure. A content type is the generic expression capturing the structure of these complex values.
Database Types

Media objects support an individual user and only provide a section of the data of the system. The question arises how the data can be described globally. In fact, we would need to combine all the content types. For instance, in the bottleshop example we will not only have the content types of media types \texttt{WINE} and \texttt{WINERY\_HISTORY}. There may also be a \texttt{PRICE\_LIST} or an \texttt{OFFER}. Some of the data that we use in media objects of type \texttt{WINE} may also appear in media objects of type \texttt{OFFER}, e.g., the information about the winery or the grapes. Similarly, we could have added the price of a wine (for a bottle or a case) to the media object of type \texttt{WINE}.

This indicates that it is not the best idea to directly use the content types of the media types for global data storage because this may lead to redundancy. Instead, we reorganise the global data content and set up a database. Designing the schema for such a database underlies completely different quality criteria. For instance, for databases we would like to avoid redundancies as much as possible. We would also have to pay much attention to providing fast and concurrent access to the data.

Therefore, we use a separate layer defined by database types. Thus, we obtain a description of the static components on at least two layers: the global or database layer, and the local or media type layer. More than that, we even use a three layer approach consisting of a layer of data types, a layer of database types, and a layer of raw media types.

The Data Type Layer

The first layer is defined by data types, which define sets of possible values for using them in content types and database types. We have already seen examples of such data types, which are either base types such as \texttt{STRING}, \texttt{NAT}, \texttt{URL}, etc., or complex types, composed of base types and other complex types such as tuple or set types. The example content type that we have seen already used such tuple and set types. Tuple types (also called record types) can be written using the notation \((a_1: t_1, \ldots, a_n: t_n)\) with arbitrarily chosen labels \(a_1 \ldots, a_n\) and data type names, \(t_1, \ldots, t_n\). Set types can be written using the notation \(\{t\}\). In the cases of tuple and set types, we call \(t_1, \ldots, t_n\) (or \(t\), respectively) the component types of the complex type.

To repeat, the content type of raw media type \texttt{WINE} was a tuple type

\[
\text{name} : \text{STRING}, \text{picture} : \text{PIC}, \text{year} : \text{NAT}, \text{winery} : (\text{name} : \text{STRING}, \text{region} : \text{STRING}, \text{country} : \text{STRING}, \text{address} : \text{ADDRESS}, \text{history}
\]
with the labels $a_1 = \text{name}$, $a_2 = \text{picture}$, $a_3 = \text{year}$, $a_4 = \text{winery}$, $a_5 = \text{composition}$, etc., and the component types $t_1 = \text{STRING}$, $t_2 = \text{PIC}$, $t_3 = \text{NAT}$, $t_4 = (\text{name: ...})$, $t_5 = \{(\ldots)\}$, etc. The component type $t_4$ is, again, a tuple type; the component type $t_5$ is a set type, which has a tuple type as its component type, etc. We see that, with composed data types, we can have any depth of nesting.

We call the collection of all possible data types a \textit{type system}. There are lots of choices for type systems. Which one is used in a particular application depends on the needs of the application and on the systems that are available. For our purposes here, the theory of media types, we may take the viewpoint that any type system will work as long as one of its base types is the data type \textit{URL}.

### The Database Type Layer

The second layer is defined by \textit{database types} over the type system. These database types are, more or less, defined in the same way as the content types. For instance, we may define a database type as:

```
(name : \text{STRING}, year : \text{CARD}, composition : \{(grape : \text{STRING}, percentage : \text{CARD})\}, colour : \text{STRING}, bouquet : \text{STRING}, acidity : \text{STRING}, sugar : \text{DECIMAL})
```

In particular, they may already include links. However, relational database systems would not be able to support such links. In general, the format of database types depends on the \textit{data model} that has been chosen for the application.

### The Media Type Layer

The third layer is defined by the \textit{media types} which we have already discussed above. Remaining questions are: how these media types differ from the database types and how they are connected to the database types.

We already stated that, when we look at the content types of media types only, there is no formal difference between such content types and database types. The major difference concerns their purpose and usage. The database types are used to represent the global data content of the WIS. The collection of all the database types — this is what we call a \textit{database schema} — is organised in such a way that the desirable quality criteria for databases, such
as fast access, redundancy freeness, etc., can be met, whereas the content types of media types are organised in a way that the information needs of the users who navigate through the WIS are met. Redundancy among content types is not only unavoidable, it is even intended.

Furthermore, links may already exist between database types, but this need not be the case. For the media types, however, the links are an important means to represent the navigation structure of the WIS. Thus, links between media types are unavoidable and intended.

Finally, media types are not independent from the database types. The information represented by the content types is already present in the database schema. Therefore, we need transformations from the database schema to the content types of the raw media types. Such transformations are called views.

Therefore, we have to add a description of the transformation from the database schema to the content type. Such a transformation is called a query. In fact, a view is given by an input database schema, an output database schema — in our case the content type — and a query, which transforms a database over the input schema into a database over the output schema. In general, a query could be any such (computable) transformation. In most cases, however, the query languages that come with the used data model determine which queries can be used. In doing so, they limit the expressiveness of the queries.

Adaptivity

Adaptivity, in general, deals with the ability of a system to adapt itself to external needs. We distinguish between three different lines of adaptivity.

• Adaptivity to the user deals with needs arising from different users. Users may prefer to receive information in a dense or sparse form. In the former case, a larger portion of information would be transmitted to the user, whereas, in the latter case, the information would be delivered step by step. Furthermore, it should be possible to provide different levels of detail and let the user switch between these levels.

For instance, taking up again the example of wines, a user may prefer to see the information about a wine together with detailed information about the grapes or the winery. Another user may prefer to see a list of several wines at a time, each coupled with only rough information about name, year, grapes.

• Adaptivity to the technical environment copes with technical restrictions of end-devices. For instance, if mobile end-devices with small screens or TV-based end-devices are to be supported, it would, nevertheless, be desirable to have only one conceptual description, and to tailor the media
objects to the specific needs of the technical environment only when these needs arise.

• Adaptivity to the communication channel deals with adaptation to needs arising from various communication channels. As users do not want to wait too long for the information to be transferred, a restricted channel capacity should imply a step-by-step delivery of the information.

Structural media types deal with adaptivity by extending the definition of media types by cohesion, which allows a controlled form of information loss. On the level of content types, we determine which data should preferably be kept together. In all cases where user preferences or technical constraints force the information to be split, the split will be made in such a way that data that is to be kept together will be kept together. The lost information will become available in a follow-on step.

DATABASE TYPES AND DATA TYPES

In this section, we start describing media types in a more formal way. We concentrate first on the data type and the database layer, that are used to define media types. Recall that the data type layer introduces data types, i.e., sets of possible values, whereas the database layer introduces database types, which define the structure of the possible databases.

This section can be treated as a short introduction to conceptual data modeling. In particular, if the central notions of this section, such as database schema, are known, it is possible to skip this section and proceed directly with the introduction of media types in the next section.

Our presentation in this section will introduce a data model that is quite close to the Entity-Relationship model. It lies somewhere between the basic Entity-Relationship model and the Higher-order Entity-Relationship model (Thalheim, 2000). Recall from the introduction above that structural media types could take any other data model. The choice of data model for this section is mainly due to the fact that its structural units leave only a small gap between the data model and the media types.

Data Types

Data types are used to classify values. In general, types are used for classification. Thus, the name “data type” suggests that we classify data. However, in our context, the term “data” is used for several quite different
purposes. We should distinguish between the data treated by data types, the
data treated by database types, and the data treated by media types. Therefore,
in the context of data types, we talk of values; in the context of databases, we will
talk of objects; and in the context of media types, we will talk about media objects.

Thus, a data type gives some notation for a set of values. To be precise,
data types should also provide the operations that can be applied to these
values. For the moment, these operations are not important for us, so we will
ignore them. A collection of data types that is defined by some syntax is called
a type system. If we use abstract syntax, we can define a type system by
writing:

\[ t = \mathbb{B} \mid (a_1 : t_1, \ldots, a_n : t_n) \mid \{t\} \mid [t] \mid (a_1 : t_1) \uplus \cdots \uplus (a_n : t_n) \]

This description of syntax needs some further explanation. We use the
symbol \( b \) to represent an arbitrary collection of base types, such as the
following types:

- \( \text{CARD} \) for non-negative integers, \( \text{INT} \) for integers,
- \( \text{CHAR} \) for characters, \( \text{STRING} \) for character strings,
- \( \text{DATE} \) for date values, \( \text{BOOL} \) for truth values,
- \( \text{URL} \) for URL-addresses, \( \text{MAIL} \) for e-mail-addresses, \( \text{OK} \) for a single
value \( \text{ok} \), etc.

Thus, the definition of the type system above states that a type \( t \) is either
a base type or has one of the following four forms:

- We can have \( t = (a_1 : t_1, \ldots, a_n : t_n) \) with arbitrary, pairwise different labels
  \( a_1, \ldots, a_n \) and types \( t_1, \ldots, t_n \). Such a type is called a tuple type. The types
  \( t_1, \ldots, t_n \) used to define the tuple type are called the component types
  of the tuple type. The intention of a tuple type is to provide a set of tuple
values, each of which has the form \((a_1 : v_1, \ldots, a_n : v_n)\), using the labels
  \( a_1, \ldots, a_n \) and values \( v_1 \) of type \( t_1 \), \( v_2 \) of type \( t_2 \), etc.
- We can have \( t = \{t\} \) with another type \( t' \). Such a type is called a set type.
The intention of a set type is to provide a set of values, each of which is itself a finite set \( \{v_1, \ldots, v_n\} \), with values \( v_1, \ldots, v_n \) of type \( t' \).
- Similarly, we can have \( t = [t] \) with another type \( t' \). Such a type is called a list type.
The intention of a list type is to provide a set of values, each of which is a finite, ordered list \([v_1, \ldots, v_n]\), with values \( v_1, \ldots, v_n \) of type \( t' \).
- Finally, we can have \( t = (a_1 : t_1) \uplus \cdots \uplus (a_n : t_n) \) with arbitrary, pairwise
different labels \( a_1, \ldots, a_n \) and types \( t_1, \ldots, t_n \). Such a type is called a union
type. The types $t_1, \ldots, t_n$ used to define the union type are called the component types of the union type. The intention of a union type is to provide a set of values, each of which has the form $(a_i; v_i)$, using one of the labels $a_1, \ldots, a_n$ and values $v_1$ of type $t_1$, $v_2$ of type $t_2$, etc.

We say that $(\ldots)$, ${\cdot}$ and $[\cdot]$ and $\cup \ldots$ are constructors for records, sets, lists and unions.

**Example 1.** Define a data type NAME by:

\[
\text{NAME} = (\text{first_names} : [\text{STRING}], \text{middle_initial} : (y : \text{CHAR}) \cup (n : \text{OK}), \text{titles} : \{\text{STRING}\}, \text{family_name} : \text{STRING}) .
\]

This is a tuple type with four components according to the following intention:
- The first component gives a list of character strings for the first names.
- The second component is either a single character for a middle initial or \text{ok}, which indicates that there is no middle initial.
- The third component gives a set of character strings for titles.
- The last component is a single character string for the family name.

Thus, possible values of type NAME would be:

\[
(\text{first_names} : [\text{“George”, “Francis”, “Leopold”}], \text{middle_initial} : (n : \text{ok}), \text{titles} : \{\text{“Professor”, “Sir”}\}, \text{family_name} : \text{“Stocker”})
\]

and

\[
(\text{first_names} : [\text{“Harry”}], \text{middle_initial} : (y : \text{‘F’}), \text{titles} : \{\}, \text{family_name} : \text{“Rugger”})
\]

Let us now define the semantics of data types, i.e. we will associate with each data type $t$ a set $\text{dom}(t)$ of values, called values of type $t$. In the definition of data types, and in Example 1, we already gave a glimpse of what the definition of semantics will look like.

We follow the inductive definition of data types to define the set $\text{dom}(t)$ of values of type $t$ as follows:
For a base type \( t \), we obtain
- \( \text{dom}(\text{CARD}) = \mathbb{N} = \{0, 1, 2, \ldots\} \) is the set of natural numbers.
- \( \text{dom}(\text{INT}) = \mathbb{Z} = \{0, 1, -1, 2, -2, \ldots\} \) is the set of integers.
- \( \text{dom}(\text{CHAR}) = \{A, a, B, b, \ldots\} \) is a fixed set of characters.
- \( \text{dom}(\text{BOOL}) = \{T, F\} \) is the set of Boolean truth values.
- \( \text{dom}(\text{OK}) = \{\text{ok}\} \).
- \( \text{dom}(\text{DATE}) = \{\text{dd-mm-yyyy}\} \) is the set of all date values.
- \( \text{dom}(\text{STRING}) = \text{dom}(\text{CHAR})^* = \{a_1, \ldots, a_n| n \in \mathbb{N}, a_i \in \text{dom}(\text{CHAR})\} \) is the set of all character strings over \{A, a, B, b, \ldots\}.
- \( \text{dom}(\text{MAIL}) \) is the set of all syntactically valid mail addresses.
- \( \text{dom}(\text{URL}) \) is the set of all syntactically valid URLs.

If further base types are defined, we associate with them a fixed set of values in the same way.

- \( \text{dom}(a_1 : t_1, \ldots, a_n : t_n) = \{(a_1 : v_1, \ldots, a_n : v_n)| v_i \in \text{dom}(t_i) \text{ for all } i = 1, \ldots, n\} \) is the set of \( n \)-tuples with component values in \( \text{dom}(t_i) \) for \( i = 1, \ldots, n \).
- \( \text{dom}(\{t\}) = \{A \in \mathcal{P}(\text{dom}(t))| |A| < \infty\} \) is the set of all finite subsets of \( \text{dom}(t) \). Thus, we have \( \text{dom}(t) = \{v_1, \ldots, v_k| k \in \mathbb{N} \text{ and } v_i \in \text{dom}(t) \text{ for all } i = 1, \ldots, k\} \).
- Similarly, \( \text{dom}([t]) = \{[v_1, \ldots, v_k]| k \in \mathbb{N} \text{ and } v_i \in \text{dom}(t) \text{ for all } i = 1, \ldots, k\} \) is the set of all finite lists with elements in \( \text{dom}(t) \).
- \( \text{dom}(a_1 : t_1 \uplus \cdots \uplus a_n : t_n) = \{(a_1 : v_1)| v_1 \in \text{dom}(t_1)\} \uplus \cdots \uplus \{(a_n : v_n)| v_n \in \text{dom}(t_n)\} \) is the disjoint union of the sets \( \text{dom}(t_i) \) \((i = 1, \ldots, n)\), i.e. elements in \( \text{dom}(t_i) \) are labeled with \( a_i \) before the union is built.

Example 2. Let us continue Example 1 and consider again the data type:

\[
\text{NAME} = (\text{first_names} : [\text{STRING}], \text{middle_initial} : (y : \text{CHAR}) \uplus (n : \text{OK}), \\
\text{titles} : \{\text{STRING}\}, \text{family_name} : \text{STRING}) .
\]

As the outermost constructor is the tuple type constructor, each value in \( \text{dom}(\text{NAME}) \) is a tuple of the form:

\[
(\text{first_names} : v_1, \text{middle_initial} : v_2, \text{titles} : v_3, \text{family_name} : v_4).
\]

Here, \( v_1 \) must be a value of type \([\text{STRING}]\), i.e. is a finite list of strings: \( v_1 = [v_1, v_2, \ldots, v_k] \) with all \( v_i \in \text{dom}(\text{STRING}) \). \( v_2 \) must be a value of type \((y : \text{CHAR}) \uplus (n : \text{OK})\), so it is either \((y : v_2)\) with a value \( v_2 \in \text{dom}(\text{CHAR}) \) or
(n : ok). $v_3$ is a value of type \{\texttt{STRING}\}, i.e., it is a finite set of strings: $v_3 = \{v_{31}, v_{32}, \ldots, v_{3r}\}$ with all $v_{3i} \in \text{dom(STRING)}$. Finally, $v_4$ is a character string.

**Database Types**

We now proceed with the second layer, which deals with database types and database schemata. Look again at the bottleshop example that we used so far. A basic unit of information in this example is given by the wines. So, we may describe a wine by its name, its year, the used grapes, information about colour, bouquet, acidity level, and residue sugar. Assume that we do not have to provide further details. We may want to keep the description of the winery separate, but it is not intended to provide any details about grapes. So, we create attribute names such as name, year, composition, colour, bouquet, acidity and sugar. These attributes are sufficient to describe wines, provided that, for each of these attributes, we declare a data type, which will describe the possible values for this attribute.

Thus, a first definition would be the following:

A *database type of level* \(0\) has a name \(E\) and is described by a finite set \(\text{attr}(E) = \{a_1, \ldots, a_m\}\) of attributes. Each attribute \(a_i\) is associated with a data type \(\text{type}(a_i)\).

We will extend this definition below by adding key attributes.

**Example 3.** Taking the example of wines from above, we define a database type with the name \texttt{WINE} and the attribute set

\[\text{attr(WINE)} = \{\text{name, year, composition, colour, bouquet, acidity, sugar}\}.\]

Associated data types could be \text{type(name)} = \texttt{STRING}, \text{type(year)} = \texttt{CARD}, \text{type(colour)} = \texttt{STRING}, \text{type(bouquet)} = \texttt{STRING}, \text{type(acidity)} = \texttt{STRING}, \text{type(sugar)} = \texttt{DECIMAL}, and \text{type(composition)} = \{(grape : \texttt{STRING}, percentage : \texttt{CARD})\}. Thus, we would describe an object of type \texttt{WINE} by a combination of values for all these types or, equivalently, by a single value of the following tuple type:

\[(\text{name : STRING, year : CARD, composition : \{ (grape : STRING, percentage : CARD) \}, colour : STRING, bouquet : STRING, acidity : STRING, sugar : DECIMAL})\]

using the attribute names of the database type as labels.
Similarly, we can define a database type \textit{WINERY} with attributes name, founded, owners, address, and maybe more. We omit the details of such a type.

Besides database types of level 0, we may also define database types on higher levels. For instance, in the bottleshop example, we may relate wines with wineries that produce them. In this case, the entity types \textit{WINE} and \textit{WINERY} would become components of a new type \textit{PRODUCER}. Such a type is a \textit{database type of level k}, with \( k-1 \) being the maximum of the levels of the component types. Of course, we may extend the higher-level database type by adding attributes such as start\_date, end\_date, and maybe others.

A database type of level \( k \) has a name \( E \) and consists of:

- a set \( \text{comp}(E) = \{ r_1 : E_1, \ldots, r_n : E_n \} \) of components with pairwise different role names \( r_i \) and names \( E_i \) of database types,
- a set \( \text{attr}(E) = \{ a_1, \ldots, a_m \} \) of attributes, each associated with a data type \( \text{type}(a_j) \), and
- a key \( \text{id}(E) \subseteq \text{comp}(E) \cup \text{attr}(E) \),

such that the database types \( E_i \in S \) are all on levels lower than \( k \), with at least one database type of level exactly \( k-1 \).

Note that the role names are only needed to allow the same database type to appear more than once as a component of another database type. Obviously, in this case, the occurrences have to be distinguished by using different role names.

\textbf{Example 4.} Let us complete the description of a relationship type \textit{PRODUCER}. We define this type by the set of components \( \text{comp}(\text{PRODUCER}) = \{ \text{of} : \text{WINE}, \text{by} : \text{WINERY} \} \) and the set of attributes \( \text{attr}(\text{PRODUCER}) = \{ \text{start\_date}, \text{end\_date} \} \). The associated types can be \( \text{type} (\text{start\_date}) = \text{DATE} \) and \( \text{type} (\text{end\_date}) = (f : \text{DATE}) \cup (\text{nf} : \text{OK}) \).

In Example 3, we have seen that we can associate a single data type with a database type of level 0 by turning the attribute names into labels of a tuple type. The question is how this can be generalised to database types on higher levels. The easiest solution would be to treat the attributes in the same way as database types of level 0, and to use the role names also as labels, using the data types associated with the lower level database types as components. For instance, for the database type \textit{PRODUCER} in Example 4, we would obtain the data type:
(of : \( t_{\text{WINE}} \) by : \( t_{\text{WINERY}} \), start_date : \( DATE \), end_date : (f : \( DATE \) \( \cup \) (nf : \( OK \)) ),

with the component type \( t_{\text{WINE}} \) defined in Example 3, and a component type \( t_{\text{WINERY}} \) defined elsewhere. Values of this type would be called objects of type PRODUCER.

Though this approach is formally correct, it has certain disadvantages, as the resulting data types will be quite big and, in each value, we would have to repeat values that describe on lower levels. Instead of this, we will exploit the keys, i.e., combinations of components and attributes that uniquely identify objects.

In the following, we often write \( E = (\{ r_1 : E_1, \ldots, r_n : E_n \}, \{ a_1, \ldots, a_m \}, id(E)) \) to denote a database type. The first component in this triple is the component set \( \text{comp}(E) \). The second component is the attribute set \( \text{attr}(E) \). The third component is the key \( \text{id}(E) \).

We use this notation to associate two data types with each database type \( E \). These data types are called the associated data type of \( E \) and the associated key type of \( E \). These types are defined as follows:

- The associated data type of \( E \), denoted as \( \text{type}(E) \), is \( (r_1 : \text{key-type}(E_1), \ldots, r_n : \text{key-type}(E_n), a_1 : \text{type}(a_1), \ldots, a_m : \text{type}(a_m)) \).
- The associated key type of \( E \), denoted as \( \text{key-type}(E) \), is defined analogously with the difference that only those \( r_i \) and \( a_j \) are considered that occur in \( \text{id}(E) \).

In particular, if we have a database type \( E \) of level 0, then \( \text{comp}(E) \) is the empty set. This implies that there will be no labels \( r_i \) in \( \text{type}(E) \) nor in \( \text{key-type}(E) \).

**Example 5.** In Example 3, we have seen a database type of level 0. The following definition extends this database type by a key:

\[
\text{WINE} = (\{ \}, \{ \text{name, year, composition, colour, bouquet, acidity, sugar} \}, \{ \text{name, year} \}) .
\]

We keep the associated data types:

\[
\begin{align*}
\text{type(name)} &= \text{STRING} , \\
\text{type(year)} &= \text{CARD} , \\
\text{type(composition)} &= \{ (\text{grape} : \text{STRING}, \text{percentage} : \text{CARD}) \} ,
\end{align*}
\]
type(colour) = STRING,
type(bouquet) = STRING,
type(acidity) = STRING,
type(sugar) = DECIMAL.

So, as we have already seen in Example 3, the associated data type type(WINE) would be the data type:


The associated key type key-type(WINE) would be the data type:

(name : STRING, year : CARD).

In Example 4, we discussed a database type PRODUCER of level 1. We extend this definition by a key, which leads to:

PRODUCER =({of : WINE, by : WINERY}, {start_date, end_date},
{of : WINE, by : WINERY, start_date}).

We keep the same associated data types:

\[
\begin{align*}
type(\text{start\_date}) & = DATE, \\
type(\text{end\_date}) & = (f : DATE) \uplus (nf : OK).
\end{align*}
\]

Then, the associated data type and the associated key type of the database type PRODUCER are:

(of : (name : STRING, year : CARD), by : \ldots, start\_date : DATE, end\_date : (f : DATE) \uplus (nf : OK)),

and

(of : (name : STRING, year : CARD), by : \ldots, start\_date : DATE),

respectively. In both data types, the dots have to be replaced by the associated key type key-type(WINERY). As we omitted the definition of the database type
WINERY, we have to omit the replacement of these dots by a real data type, too.

**Cluster Types**

Before we introduce database schemata and databases, we still have to discuss one further extension. This will be the introduction of so-called cluster types. In order to get started, look again at Example 5, in which we presented a database type PRODUCER of level 1. The component database types of PRODUCER were the types WINE and WINERY. Now, assume that our bottleshop also offers beers, and these beers are produced by breweries. So, we would add two other database types of level 0: BEER and BREWERY. We can now extend the of-component in PRODUCER in such a way that it refers to WINE or BEER. In the same way, we could change the by-component so that it refers to WINERY or BREWERY. In order to do so, replace the of-component in PRODUCER by a new type BEVERAGE, and the by-component by a new type COMPANY, while defining BEVERAGE by the “disjoint union” of WINE and BEER, and COMPANY by the “disjoint union” of WINERY and BREWERY. Such disjoint unions are defined by cluster types.

A cluster type of level \( k \), written \( E = (id_1 : E_1) \odot \ldots \odot (id_n : E_n) \), has a name \( E \) and consists of a non-empty sequence of components \( E_1, \ldots, E_n \), which can be database types or cluster types, with pairwise different component identifiers \( id_i \), such that the level \( k \) is the maximum of the levels of the \( E_i \).

**Example 6.** Formalising our motivating example from above we define the two cluster types BEVERAGE = (w : WINE) \odot (b : BEER) and COMPANY = (w : WINERY) \odot (b : BREWERY) of level 0.

Now we can extend the definition of associated data types and associated key types to cluster types. According to the motivation above, it should not be surprising to see that we now use union types. Therefore, let \( E = (id_1 : E_1) \odot \ldots \odot (id_n : E_n) \) be a cluster type. The associated data type \( \text{type}(E) \) and the associated key type \( \text{key-type}(E) \) are defined as follows:

\[
\text{type}(E) = \text{key-type}(E) = (id_1 : \text{key-type}(E_1)) \uplus \ldots \uplus (id_n : \text{key-type}(E_n)).
\]

**Database Schemata and Databases**

Let us now conclude the presentation of the global database layer by defining database schemata. A database schema is simply a collection of
database and cluster types. Of course, if $E_i$ is a component of a database or cluster type $E$, and $E$ is defined in the schema, then $E_i$ must also be defined in the schema. Formally, we can define a database schema as follows:

A database schema $S$ is a set of database types and cluster types satisfying the following two conditions:

1. If $E \in S$ is a database type, then, for all components $r_i : E \in \text{comp}(E)$, we must also have $E_i \in S$.
2. If $E = E_1 \oplus \ldots \oplus E_k$ is a cluster type in $S$, then, for all components $E_i (i = 1, \ldots, k)$, we must also have $E_i \in S$.

We define the semantics of database schemata by the collection of possible databases by the database schema. Thus, let $S$ be a database schema. For each database or cluster type $E \in S$, we have defined an associated data type type($E$) and an associated key type key-type($E$). As these two are indeed data types, they define fixed sets of values, which we call the set of objects of type $E$ and the set of keys of type $E$, respectively:

$$\text{Obj}(E) = \text{dom}(\text{type}(E))$$

$$\text{Key}(E) = \text{dom}(\text{key-type}(E))$$

As the key id($E$) in a database type $E$ is a subset of $\text{comp}(E) \cup \text{attr}(E)$, each object of type $E$ can be projected to a key of type $E$. Let $O\{\text{key-type}(E)\} \in \text{Key}(E)$ denote the projection of the object $O \in \text{Obj}(E)$ to the value of its key. For a cluster type $E$ each object $O$ of type $E$, we have $O\{\text{key-type}(E)\} = O$. We use the sets of objects and keys for the database and cluster types $E \in S$ to define a database over $S$ as follows:

A database $db$ over $S$ assigns to each database or cluster type $E \in S$ a finite set $db(E) \subseteq \text{Obj}(E)$ of objects of type $E$ such that the following conditions are satisfied:

1. Key values are unique, i.e., there cannot be two different $O_1, O_2 \in db(E)$ with $O_1\{\text{key-type}(E)\} \neq O_2\{\text{key-type}(E)\}$.
2. Component values exist in the database, i.e., for each $O \in db(E)$ and each $r : E' \in \text{comp}(E)$, there must exist some $O' \in db(E')$, such that $r : O'\{\text{key-type}(E')\}$ is part of $O$.
3. Clusters are disjoint unions, i.e., for a cluster $E = (id_1 : E_1) \oplus \ldots \oplus (id_n : E_n)$, we obtain $db(E) = \{(id_i : O_i\{\text{key-type}(E_i)\} | O_i \in db(E_i) \text{ and } i \in \{1, \ldots, n\}\}$. 

Copyright © 2004, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
Finally, let us briefly introduce a graphical representation for database schemata, which we call a (database) schema diagram:

- We use rectangles to represent database types on level 0 — in this case, we always have $\text{comp}(E) = \emptyset$ — and place the name of the type inside the rectangle.
- We use diamonds to represent database types on higher levels — in this case, we always have $\text{comp}(E) \neq \emptyset$ — and place the name of the type inside the diamond.
- We use arrows from a type to all its component types, and attach the role names to these arrows.
- We use $\oplus$ to represent cluster types, and arrows to the components of the cluster type.
- We attach attributes directly to the rectangles or diamonds.
- We underline the attributes in the key, and mark the arrows corresponding to components in the key with a dot.

If schema diagrams tend to become large, we usually omit the attribute names. Sometimes we even drop the names of the cluster types.

**Example 7.** Let us consider an example database schema for a bottleshop application illustrated in Figure 1. A bottleshop is to deliver wines, beers, juices and other soft drinks, and hard alcoholics drinks such as Cognac, Grappa, Scotch and Irish Whiskey. The sales are to be supported by a web information system. The Web Information System shall provide a catalogue containing the offered products, as well as occasional special offers. Additional information about wines, grapes, vineyards (also for Cognac, Grappa, Calvados and Whiskey) shall be provided, as well. The system must offer information about the shop itself, historical and actual information about certain grapes, wine producers, etc. The system should also provide special information about best sellers, new products, etc. Comments from (hopefully pleased) clients shall be made available on the Web.

**QUERIES AND STRUCTURAL MEDIA TYPES**

In this section, we address the layer of media types. The intention behind the media types is to provide a local view on the data that are stored in the underlying database. The emphasis of this layer is to specify data as they are
to appear on web pages, however, without any consideration of how the data should be presented. Thus, the media types are used to structure data according to a usage perspective rather than a storage perspective.

Formally, a structural media type is defined by a *content type* and a *defining query*. There is no big, formal difference between content types of structural media types and database types, except that content types should support links, whereas the existence of links in database types depends on the data model. The major difference is given by the fact that structural media types need a defining query which links the media type with an underlying database schema. If $Q$ is the defining query of a structural media type $M$, it may be executed on databases $db$ over the underlying database schema $S$. The query will result in the set of media objects of type $M$. Of course, if databases are updated, the database $db$ and consequently also the set of media objects changes.

In the following, we will first briefly discuss queries. It is beyond the scope of this chapter to go into details of query languages, but we will emphasize the need and difficulty of creating links (Schewe, 2001). In fact, each data model requires its own query languages, and the theory of media types can be based on any data model. In a second step, we will then give a formal definition of structural media types and media objects.
Queries

Let us consider queries. In general, a query is a transformation from one database schema into another one. In most cases, the target schema of such a transformation is significantly simpler than the source schema.

For instance, consider the database schema from the bottleshop example which we developed in the previous section. Suppose we are given a database \( db \) over this schema, i.e. we have information about wines, producers, vineyards, etc. Suppose we want to have a list of New Zealand wines from 1998 on, with at least 30% Cabernet Sauvignon, together with some information about the vineyard and the region. In this case, we define a target database schema with exactly one database type \( \text{NZ\_WINE} \). This database type—which is, of course, a database type of level 0—can be defined as follows:

\[
\text{NZ\_WINE} = (\{\}, \{\text{name, year, composition, vineyard, owners, since, history, region, climate}\}, \{\text{name, year}\}).
\]

The data types of the attributes can be defined as follows:

\[
\begin{align*}
type(\text{name}) &= \text{STRING} \\
type(\text{year}) &= \text{CARD} \\
type(\text{composition}) &= \{(\text{grape: STRING, percentage: CARD})\} \\
type(\text{vineyard}) &= \text{STRING} \\
type(\text{owners}) &= \{\text{STRING}\} \\
type(\text{since}) &= \text{DATE} \\
type(\text{history}) &= \text{STRING} \\
type(\text{region}) &= \text{STRING} \\
type(\text{climate}) &= \text{STRING}
\end{align*}
\]

Formally, a query can be defined as follows:

A query \( Q \) consists of a source database schema \( \text{src}(Q) \), a target database schema \( \text{trgt}(Q) \), and a map \( q(Q) \), which takes as its input all databases over \( \text{src}(Q) \) and produces databases over \( \text{trgt}(Q) \).

How would we process such a query? Of course, we can only sketch an idea of query processing. Actually, techniques used in query processing differ significantly from this simple outline here, but we are only interested in getting an idea of the database transformation, not of its technical realisation. In our
example query looking for New Zealand wines, we could take a database \( db \) over the source schema, i.e., the database schema from the previous section. The set \( db(WINE) \) contains finitely many values of type \( type(WINE) \), which is a tuple type with labels name, year, composition, colour, bouquet, acidity and sugar. First, we use a projection, i.e., we simply “forget” the last four components. Each tuple will be reduced to a tuple of a tuple type containing only the labels name, year and composition.

Looking into the details of the component labeled by ‘composition,’ we know that we must have a set of pairs, the first component of which is labeled by grape, the second one by percentage. We only keep those tuples, where this set contains a pair (“Cabernet Sauvignon”, \( x \)) with a value \( x \geq 30 \). This is called a selection. Similarly, we only keep tuples where the value \( y \) of the year component satisfies \( y \geq 1998 \). In doing this, we would already obtain the wines from 1998 on with at least 30% Cabernet Sauvignon.

However, we want to have New Zealand wines only, and for these we demand information about the vineyard and the region. Therefore, we have to look at the cluster \( BEVERAGE_H \) and the database type \( PRODUCER \). We only keep those tuples where the values for name and wine appear as a pair labeled by \( b \) in the component labeled by \( of \) in objects in \( db(PRODUCER) \). However, as we want to get vineyards and not distilleries, we only consider those objects in \( db(PRODUCER) \) with a \( by \)-component labeled by \( v \). Using this component, we obtain objects of type \( VINEYARD \). We rename the label name in these objects to vineyard, in order to avoid mixing up the name of wines with vineyard names, and we forget the size and grapes components. Adding the remaining components to the tuples gives new tuples with components name, year, composition, vineyard, owners, since, history, and \( in \).

The latter one can be used for another join with \( db(REGION) \). Again, in this way we obtain the name component of the region, which we rename to region, the climate component, the wines component, which we forget, and the specialities component, which we forget, as well. The \( in \)-component finally leads to \( db(COUNTRY) \). We only keep tuples where the join gives a value “NZ” for the code component.

This rough sketch of query processing shows how a database over the source schema can be transformed into a database over the target schema. The query may have produced a tuple such as the following:

\[
(\text{name} : \text{“Otago Red”}, \text{year} : 2000, \text{composition} : \{(\text{grape} : \text{“Cabernet Sauvignon”}, \text{percentage} : 65), (\text{grape} : \text{“Malbec”}, \text{percentage} : 35)\}, \text{vineyard} : \text{“Martha’s Vineyard”}, \text{owners} : \{\text{“Rudi}}
\]
Müller”, “Martha Thurgau”}, since: 1967, history: “Once upon a
time …”, region: “Otago”, climate: “The climate in Otago …”).

Now, recall that a media object is not only a complex value, but it is a pair
\((u, v)\), with a value \(u\) of type \(URL\) and a complex value \(v\). Queries, as we
discussed them above, would only result in a set of objects of the database
types in the target database schema. Thus, instead of obtaining the value above,
we would also like to obtain a URL, or at least a unique identifier, which could
later be replaced by an actual URL. As such URLs are not stored in the
database, they have to be created by processing the query.

This implies requiring that the query language should allow the creation of
URLs and links. We call this property \textit{create facility} (see, e.g., Abiteboul et
al., 2000, Chapter 6). In terms of our sketch of query processing, this simply
means that, at any stage, we may introduce URLs by transforming a set \(\{v_1,
\ldots, v_m\}\) or list \([v_1, \ldots, v_m]\) of values into a set \(\{(u_1, v_1), \ldots, (u_m, v_m)\}\) or list \([\{u_1,
v_1\}, \ldots, \{u_m, v_m\}]\) of pairs, with new values \(u_i\) of type \(URL\), respectively, or
simply by transforming a value \(v\) of any type into a pair \((u, v)\) with a value \(u\) of
type \(URL\).

Note that once we process a query on a database \(db\), we may add the
result to the database. Processing further queries may then also include the
newly created URLs into their results. In this way, navigation structures can be
set up.

\section*{Structural Media Types}

We are now ready to formally round up this section and define structural
media types. We first need an exact definition of a content type. Recall that a
content type is an extended data type, in which the place of a base type may
be occupied by a pair \(\ell : M’\), with some label \(\ell\) and a name \(M’\) of a media type
—in fact, this means to choose any name.

Thus, using abstract syntax again (as we did for data types on page \textbf{10}),
we can define the \textit{system of content types} by writing:

\[
ct = b \mid (a_1 : ct_1, \ldots, a_n : ct_n) \mid \{ ct \} \mid [ ct ] \mid (a_1 : ct_1) \cup \cdots \cup (a_n : ct_n) \mid \ell : N
\]

The explanation we gave on data types is still valid. However, the
extension by pairs \(\ell : M’\) is of pure syntactical nature. We cannot associate a
set of values with a content type. Instead, for a content type \(cont(M)\), we define
a representing datatype $t_M$, which results from $\text{cont}(M)$ by replacing all $\ell : M'$ by the base type $URL$. Of course, $t_M$ is a real data type, and this means that $\text{dom}(t_M)$ is defined.

Let $S$ be a database schema. A structural media type over $S$ has a name $M$ and consists of:
- a content data type $\text{cont}(M)$, in which the place of a base type may be occupied by a pair $\ell : M'$, and
- a defining query $q_M$ with source schema $S$ and target schema $\{(\text{url} : URL, \text{value} : t_M)\}$, where $t_M$ is the representing datatype of $M$.

**Example 8.** Let us define some structural media types over the database schema from the previous section. We will concentrate on the content type and the representing data type. However, we will only sketch the defining query, as we have not yet introduced a concrete query language.

Of course, the type $\text{type}(\text{NZ\_WINE})$ that we used in the target schema in the previous subsection is a content type defined as follows:

$$(\text{name} : \text{STRING}, \text{year} : \text{CARD}, \text{composition} : \{ (\text{grape} : \text{STRING}, \text{percentage} : \text{CARD}) \}, \text{vineyard} : \text{STRING}, \text{owners} : \{ \text{STRING} \}, \text{since} : \text{CARD}, \text{history} : \text{STRING}, \text{region} : \text{STRING}, \text{climate} : \text{STRING}).$$

However, it does not contain any links $\ell : M'$, which implies that it is identical with its representing data type $t_{\text{NZ\_WINE}}$. Adding a formal description of a defining query $q_{\text{NZ\_WINE}}$ as sketched in the previous subsection turns $\text{NZ\_WINE}$ into a structural media type.

Now assume that we want to represent information about New Zealand wines by a slightly changed structural media type $\text{NZ\_WINE}^*$ with the following content type:

$$(\text{name} : \text{STRING}, \text{year} : \text{CARD}, \text{composition} : \{ (\text{grape} : \text{STRING}, \text{percentage} : \text{CARD}) \}, \text{vineyard} : \text{STRING}, \text{owners} : \{ \text{STRING} \}, \text{since} : \text{CARD}, \text{history} : \text{STRING}, \text{region} : \text{REGION*}).$$

In this case, the content type contains the link region : REGION*. Consequently, the representing data type $t_{\text{NZ\_WINE}^*}$ is the following:

Of course, we also would have to define a structural media type with the name REGION\* with a content type chosen to be (name : STRING, climate : STRING), which, again, is identical to the representing data type t_REGION\*. The defining query $q_{REGION\*}$ for REGION\* would select all regions in New Zealand, reduce them to their name and climate components, and create URLs for all the resulting objects.

Then the defining query $q_{NZ\_WINE\*}$ for the structural media type NZ\_WINE\* would be processed in the same way as sketched in the previous subsection. However, components region and climate would be replaced by the URL that has been created while processing the query $q_{REGION\*}$. Furthermore, the operation create_urls would be applied to the result. We omit formal details on how to write the queries $q_{REGION\*}$ and $q_{NZ\_WINE\*}$.

Note that, with the introduction of links between structural media types, the defining queries are no longer independent from each other. In Example 8, the processing of the query $q_{NZ\_WINE\*}$ depends on the result of the query $q_{REGION\*}$. As long as there are no cyclic dependencies, the query processing can be done in a particular order. However, in the case of cycles, the process requires calculating a fixed point. Queries that demand fixed point calculation are one of the most advanced topics in the field of databases.

**Content Schemata**

Finally, let us look at the analog of a database schema on the level of structural media types. For this, assume that we have fixed a database schema $S$. As structural media types abstract from content, a collection of structural media types over $S$ will be called a content schema. Analogously to the definition of database schemata, where we had to ensure that components of database types are defined in the database schema, we now have to ensure that the structural media types that occur in links are defined in the content schema. Therefore, we define a content schema as follows:

*A content schema $C$ over a database schema $S$ is a finite set of structural media types over $S$, such that for each $M \in C$ and each link $\ell : M' \text{ occurring in the content type } \text{cont}(M)$, we also have $M' \in C$.*

Copyright © 2004, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.
Finally, assume that we are given a database schema $S$ and a content schema $C$ over $S$. We may now define a site $s$ over $C$ analogously to a database over $S$.

For this, let $db$ be a database over $S$. We may evaluate the defining query $q_M$ for each structural media type $M \in C$. These result in sets $s(M)$ of pairs $(u, v)$, such that $u$ is a value of type $URL$ and $v$ is a value of the representing data type $t_M$ of $M$. We call $s(M)$ the set of media objects of type $M$ in the site $s$. Of course, we must ensure that for each URL $u'$ that appears inside the complex value $v$ at a place occupied by $\ell : M'$ in the content type $cont(M)$ of $M$, we have a media object $(u', v') \in s(M')$.

The family of all the sets $s(M)$ with $M \in C$ defines the site $s$ over $C$ determined by the database $db$ over $S$.

**ADAPTIVITY**

In this section, we extend media types by adding adaptivity to users, technical environment, and channels. In all three cases, the idea is to split the information provided by a structural media object in such a way that parts that are preferred to be kept together — we use the term cohesion for this — will be kept together, if this is possible. This can be seen as a controlled form of information loss.

**The Idea of Adaptivity**

As indicated above, cohesion intends to declare “parts” of a content type $cont(M)$ to belong closer together than others. We will approach this in two different, but similar, ways. In both cases, we consider all possible content types that result from $cont(M)$ by losing information. If $ct$ is such a content type with reduced information, we write $cont(M) \leq ct$. Thus, we first have to define this relation $\leq$, which, in fact, is a partial order.

The idea of using the partial order is by choosing one such possible content type $ct$, which is as close to the original $cont(M)$ as possible. Instead of creating a media object of type $M$ using the content type $cont(M)$, we reduce the content to the one described by the content type $ct$, and present this to the user. If further information is needed, the user may request it by following a link added to $ct$. There should be a complementary content type $ct'$ covering the lost information. This complementary content type can then be treated again in the same way.
For instance, take again the example of a structural media type \texttt{NZ\_WINE}. Its content type was defined as follows:

\[
\text{(name : STRING, year : CARD, composition : \{ (grape : STRING, percentage : CARD) \}, vineyard : STRING, owners : \{ STRING \}, since : CARD, history : STRING, region : STRING, climate : STRING).}
\]

Losing information would mean to drop the year, or the composition, or the percentages of the grapes, or the history, etc. Assume that we want to keep the information on name, year and composition together with the highest priority, followed by the information on the vineyard, its owners and history, and leaving the lowest priority for the information on the region and its climate. Then we could first choose the following content type \(ct_1\):

\[
\text{(name : STRING, year : CARD, composition : \{ (grape : STRING, percentage : CARD) \}, further : NZ\_WINE_2).}
\]

Here, the link further : \texttt{NZ\_WINE}_2 provides a link to a dynamically constructed structural media type named \texttt{NZ\_WINE}_2. The content type of this new structural media type should contain the lost information. Thus, it could be the following content type \(ct_2\):

\[
\text{(name : STRING, year : CARD, vineyard : STRING, owners : \{ STRING \}, since : CARD, history : STRING, further : NZ\_WINE_3).}
\]

Here, we find the information about the vineyard, its owners, and its history. We also repeat the name and year of the wine. Furthermore, the link further : \texttt{NZ\_WINE}_3 provides a link to another dynamically constructed structural media type named \texttt{NZ\_WINE}_3. The content type of this new structural media type should contain further information, because the information on the region is still lost. Thus, it could be the following content type \(ct_3\):

\[
\text{(name : STRING, year : CARD, region : STRING, climate : STRING).}
\]

The alternative is to provide a priori such a split of the content type. In our example, we would define the split by the three content types \(ct_1\), \(ct_2\) and \(ct_3\). These three content types define an antichain with respect to the partial order \(\leq\), i.e., each two of them are incomparable in the sense that \(ct_i \not\leq ct_j\) holds for all \(i \neq j\).
For both ideas, we first have to define the partial order $\leq$ on content types. This is done as follows:

- For any content type $ct$, we have $ct \leq ct$.
- For any content type $ct$, we have $ct \leq OK$.
- For content types of the form $(a_1 : ct_1, \ldots, a_m : ct_m)$, we have:

$$(a_1 : ct_1, \ldots, a_m : ct_m) \leq (a_{\sigma(1)} : ct'_{\sigma(1)}, \ldots, a_{\sigma(n)} : ct'_{\sigma(n)})$$

with injective $\sigma : \{1, \ldots, n\} \rightarrow \{1, \ldots, m\}$ and $ct_{\sigma(i)} \leq ct'_{\sigma(i)}$.

- For content types of the form $[ct]$, we have $[ct] \leq [ct']$ iff $ct \leq ct'$ holds.
- For content types of the form $(a_1 : ct_1) \sqcup \cdots \sqcup (a_m : ct_m)$, we have $(a_1 : ct_1) \sqcup \cdots \sqcup (a_m : ct_m) \leq (a_1 : ct'_{1}) \sqcup \cdots \sqcup (a_m : ct'_{m})$ iff $ct_1 \leq ct'_1$ holds for all $i = 1, \ldots, m$.
- For content types of the form $\{ct\}$, we have $\{ct\} \leq \{ct'\}$ iff $ct \leq ct'$ holds.

We use the notation $sup(cont(M))$ to denote the set of all content types $ct$ with $cont(M) \leq ct$. If, in the example above, we remove the links ‘further : NZ_WINE2’ and ‘further : NZ_WINE3’ in $ct_1$ and $ct_2$, respectively — these were added only as a link to follow-on information — then we obtain $cont(M) \leq ct_i$ for all $i = 1, \ldots, 3$.

### Cohesion Preorder

Let us now go into details of our first idea (Feyer et al., 2000). The partial order $\leq$ also defines a partial order on $sup(cont(M))$. However, the order is not total, i.e., there can be content types $ct_1, ct_2 \in sup(cont(M))$ with neither $ct_1 \leq ct_2$ nor $ct_2 \leq ct_1$. Thus, in case of making a choice among the elements in $sup(cont(M))$, both $ct_1$ and $ct_2$ have equal rights. Making a choice in favour of one of them, say $ct_1$, means to state that the information represented by $ct_1$ is more important than the one represented by $ct_2$.

Therefore, the first idea can be realised by extending $\leq$ to a total order. Nevertheless, we may state explicitly that we do not want to prefer $ct_1$ over $ct_2$ nor vice versa, even in case they were comparable with respect to $\leq$. Therefore, we only require to obtain an extension by a pre-order as defined now:

**A cohesion pre-order on a structural media type $M$ is a total pre-order $\leq_M$ on $sup(cont(M))$ extending the order $\leq$, i.e., whenever $ct_1 \leq ct_2$ holds, we also have $ct_1 \leq_M ct_2$.**
The major idea behind the cohesion pre-order is that, in order to adapt to different users, channels or environments, smaller content types with respect to $\preceq$ are preferred over larger content types. In all three cases, we assume that the amount of data to be transmitted at a time and presented to the user is limited by some bound. If the presentation of a structural media object of type $M$ exceeds this bound, the content type will be split according to the following procedure:

- First, we determine the maximum amount of data that should be transmitted.
- Then, we determine the least element $c_{t_1}$ with respect to $\preceq_M$ that requires not more than the available capacity. As $\preceq_M$ is only a preorder, there may be more than one such $t_1$, in which case, one of these content types is chosen randomly.
- Taking $t_1$ instead of $cont(M)$ means that some information is lost. Therefore, we include a link to a possible successor. The link name and the name of the successor structural media type will be randomly chosen.
- In order to determine such a successor, all content types $c't \in sup(cont(M))$ with $c_{t_1} \prec_M c't$ are considered. We choose a least content type $c_{t_2}$ among these $c't$ with respect to $\preceq_M$, such that $c_{t_2}$ does not require more than the available capacity.

Continuing this way the whole communication using the structural media type $M$ is broken down into a sequence of suitable units $c_{t_1}, c_{t_2}, \ldots, c_{t_n}$ that, together, contain the information provided by the structural media type.

**Example 9.** Let us consider again the content type of the structural media type $NZ\_WINE$ used earlier. However, as $sup(cont(NZ\_WINE))$ will be large, we reorganise the content type and outline the splitting procedure without looking into details of the content type. Thus, assume that we have the following content type:

($\text{wine-info}: (\text{name} : \text{STRING}, \text{year} : \text{CARD}, \text{composition} : \{ (\text{grape} : \text{STRING}, \text{percentage} : \text{CARD}) \}), \text{vineyard} : (\text{name} : \text{STRING}, \text{owners} : \{ \text{STRING} \}, \text{since} : \text{CARD}, \text{history} : \text{STRING}), \text{region} : (\text{name} : \text{STRING}, \text{climate} : \text{STRING})$).

In order to shorten our presentation, we consider only
and ignore the inner structure, which is indicated by the dots. Ignoring the inner structure, we could define a cohesion pre-order by:

\[(\text{wine-info} : \ldots, \text{vineyard} : \ldots, \text{region} : \ldots) \preceq (\text{wine-info} : \ldots, \text{vineyard} : \ldots) \preceq (\text{wine-info} : \ldots, \text{region} : \ldots) \preceq (\text{vineyard} : \ldots, \text{region} : \ldots) \preceq (\text{wine-info} : \ldots) \preceq (\text{vineyard} : \ldots) \preceq (\text{region} : \ldots)\]

Assume that only the complete content type exceeds the computed maximum capacity. Then, the first content type to be chosen would be:

\[(\text{wine-info} : \ldots, \text{vineyard} : \ldots),\]

which will be extended to:

\[(\text{wine-info} : \ldots, \text{vineyard} : \ldots, \text{next} : \text{NZ}_WINE_2).\]

This leaves the following content types:

\[(\text{wine-info} : \ldots, \text{vineyard} : \ldots, \text{region} : \ldots) \preceq (\text{wine-info} : \ldots, \text{region} : \ldots) \preceq (\text{vineyard} : \ldots, \text{region} : \ldots) \preceq (\text{region} : \ldots).\]

Thus, the second content type to be chosen will be:

\[(\text{wine-info} : \ldots, \text{region} : \ldots).\]

This will become the content type of the dynamically generated structural media type \(\text{NZ}_WINE_2\). The splitting process stops here, as further processing would not lead to more information.
Proximity Values

Finally, let us consider the alternative approach (Feyer et al., 1998). In this approach, the content types that will be chosen instead of \( \text{cont}(M) \) are determined a priori. We only determine whether they will be transmitted one by one, or whether some of them will be recombined. Thus, we choose a maximal antichain \( ct_1, \ldots, ct_n \) in \( \text{sup} (\text{cont}(M)) \) with respect to \( \leq \). This antichain already represents a possible split of information. In addition, we define a symmetric \((n \times n)\)-matrix \( \{ p_{ij} \}_{1 \leq i,j \leq n} \) of proximity values with \( 0 \leq p_{ij} \leq 1 \). The intention is that the higher the proximity value, the more do we wish to keep the components together.

Splitting is processed analogously to the case a using a cohesion pre-order:

- For each \( X \subseteq \{ 1, \ldots, n \} \), we determine its weight, i.e., \( w(X) = \sum_{i,j \in X \times X} p_{ij} \).
- For each \( X \subseteq \{ 1, \ldots, n \} \), we determine its greatest common subtype \( \text{gcs}(X) \), i.e., the greatest element \( ct_1 \in \text{sup}(\text{cont}(M)) \) with \( t_1 \leq ct_i \) for all \( i \in X \).
- Then, we choose the \( X \) with largest weight, such that the \( \text{gcs}(X) \) does not require more than the available capacity.

**Example 10.** We take the same media type as in Example 9 and the antichain

\[
ct_1 = (\text{wine-info : ...}) \quad ct_2 = (\text{vineyard : ...}) \quad ct_3 = (\text{region : ...}).
\]

Let the proximity values be chosen as \( p_{1,2} = 0.8, p_{1,3} = 0.5 \) and \( p_{2,3} = 0.1 \). Then, we obtain the following weights and greatest common subtypes:

<table>
<thead>
<tr>
<th>( X )</th>
<th>( w(X) )</th>
<th>( \text{gcs}(X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ 1 }</td>
<td>0</td>
<td>(wine-info : ...)</td>
</tr>
<tr>
<td>{ 2 }</td>
<td>0</td>
<td>(vineyard : ...)</td>
</tr>
<tr>
<td>{ 3 }</td>
<td>0</td>
<td>(region : ...)</td>
</tr>
<tr>
<td>{ 1, 2 }</td>
<td>0.8</td>
<td>(wine-info : ..., vineyard : ...)</td>
</tr>
<tr>
<td>{ 1, 3 }</td>
<td>0.5</td>
<td>(wine-info : ..., region : ...)</td>
</tr>
<tr>
<td>{ 2, 3 }</td>
<td>0.1</td>
<td>(vineyard : ..., region: ...)</td>
</tr>
<tr>
<td>{ 1, 2, 3 }</td>
<td>1.4</td>
<td>(wine-info : ..., vineyard : ..., region: ...)</td>
</tr>
</tbody>
</table>

The result will be the same sequence of content types as in Example 9.
We discussed adaptivity to users, environment and channels. This was done in the form of allowing structural media types to be extended by a controlled form of information loss, coupled with the notion of cohesion.

A **cohesion extension** of structural media type $M$ is given either by a cohesion pre-order $\preceq$ on $\text{sup}(\text{cont}(M))$ or by a pair consisting of maximal antichain $c_{t_1}, \ldots, c_{t_n}$ in $\text{sup}(\text{cont}(M))$ with respect to $\preceq$ and a symmetric $(n \times n)$-matrix $\{ p_{ij} \}_{1 \leq i, j \leq n}$ of proximity values with $0 \leq p_{ij} \leq 1$.

A **media type** is a structural media type $M$ together with a cohesion extension. A media schema is a content schema, in which all structural media types are media types.

Given a media schema, then, it is basically a content schema over an underlying database schema. Thus, any database determines sets of media objects. The cohesion extension further determines variants of the media objects that are dynamically constructed when the need arises. It leads to a step-by-step delivery of a media object.

**CONCLUSION**

We presented a conceptual model for data-intensive web information systems, which is centered around the central notion of *media type*. Roughly speaking, a media type is defined as an extended view on an underlying database schema, and includes operations and adaptivity features.

Conceptual modeling with structural media types is embedded in an integrated methodology based on an abstraction layer model for web information systems. Prior to this activity, we have an activity of story boarding, which models the WIS from a usage perspective (Kaschek et al., 2003). The scenes in the story board are used as the data source for the media types. Further on, the media types do not yet specify anything on their web presentation. For this, media types have to be associated with style options. Using these style options and suites of XML representations leads to the implementation. The complete methodology has been applied in more than 30 large projects. The report (Thalheim, 1997) describes the first of these projects; for others, the work and publication rights have been transferred to professional companies.

We have seen that several other groups have also developed conceptual modeling approaches for web information systems. The theory of media types
is one of these approaches. As work progresses, the ideas produced by the different groups are now converging, though the theory of media types is still the most advanced model with respect to formal foundations. Furthermore, it is still more elaborate with respect to adaptivity, the scope of the overall methodology, in particular, with respect to an integration with story boarding, the work on implementation and presentation issues, and applications in practical projects.

Taking the convergence of ideas, the currently existing advantages of the theory of media types in comparison to other approaches will disappear. For instance, other approaches will take up the work on adaptivity. Conversely, the theory of media types will benefit from the work of others and become even more elaborate. However, the convergence trend concerns the concepts, not concrete languages. In particular, the theory of media types will be likely to preserve its connections to theory.

The role of XML will become even more important. On one side, XML may pick up ideas that will enable a better support for media types or similar conceptual modeling approaches. On the other side, XML will always remain a concrete language and may distract from the important issue of conceptual modeling. If XML is treated as a data model, most of the hardest database problems still have to be solved in this context. Therefore, we think it is better not to fix the attention only on XML. Using XML for representing the views is uncritical, but it is unlikely that it will be able to replace completely the theory of media types. As this theory can be based on any data model, it is much more generic than any concrete language.

REFERENCES


---

Copyright © 2004, Idea Group Inc. Copying or distributing in print or electronic forms without written permission of Idea Group Inc. is prohibited.


Formalisation of User Preferences, Obligations and Rights

Klaus-Dieter Schewe¹, Bernhard Thalheim², Alexei Tretiakov¹

¹ Massey University, Department of Information Systems & Information Science Research Centre
Private Bag 11 222, Palmerston North, New Zealand
k.d.schewe | a.tretiakov@massey.ac.nz

² Christian Albrechts University Kiel
Department of Computer Science and Applied Mathematics
Olshausenstr. 40, D-24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de

Abstract

The aim of this paper is to formalise user preferences, obligations and rights in the context of web information systems (WISs), and to indicate how this formalisation can be used to reason about WIS specifications. This problem is approached on two levels of abstraction. On a high level a WIS is represented by a storyboard, which itself can be represented by an algebraic expression in a Kleene algebra with tests (KAT). Preferences can be formalised using the equational theory of KATs, which enable sophisticated propositional reasoning that can be applied to WIS personalisation. Obligations and rights give rise to a propositional deontic logic. On a lower level of abstraction detailed state specifications are added using extended views. This amounts to replacing KATs by higher-order dynamic logic, using a higher-order deontic logic, and the formalisation of proof obligations.

Introduction

Web information systems (WISs) are an important class of data-intensive systems with a lot of applications in e-business (Schewe et al., 2005a), e-learning (Schewe et al., 2005b), information services (Feyer et al., 1998), and many more. Each WIS provides a dialogue between the system and the users, the latter ones being very often unknown to the designers. Each user uses the system to perform a certain task, some of which may only be realised by a joint effort of several users.

One problem we are confronted with in WISs is to be aware of user preferences and goals such that the system can be set up in way that permits an automatic customisation to these preferences and goals. In order to do so we have to formalise the user preferences, which of course depend on a classification of users into user profiles or user types.
Furthermore, not all users are eligible to perform or contribute to the same tasks. This requires another classification of users according to their roles, and each role determines the rights and obligations of users in this role.

The aim of this paper is to formalise user preferences, obligations and rights and to indicate how this formalisation can be used to reason about the WIS specification. For this purpose we adopt the co-design approach from (Schewe and Thalheim, 2005) to WIS design. On a high-level of abstraction the major modelling activity is storyboarding (Kaschek et al., 2004; Schewe and Thalheim, 2005; Thalheim and Düsterhöft, 2001), which consists of defining the story space, the actors, and the tasks. The system is considered as a set of abstract locations called scenes, between which users navigate. On their paths (called stories) through these scenes the users execute actions. This leads first to modelling story spaces by directed labelled graphs that are further refined by story algebras (Schewe and Thalheim, 2004; Thalheim and Düsterhöft, 2001). Actor modelling comprises user profiling, role modelling plus modelling information portfolios. However, information portfolios are treated as part of supporting information for scenes (Schewe and Thalheim, 2005). We will present the gist of storyboarding in the next section.

In particular, we will formalise obligations and rights as part of user roles (Schewe and Thalheim, 2005) using a propositional variant of deontic logic (Eiter and Subrahmanian, 1999). This permits expressing deontic constraints already on a high level of abstraction. Furthermore, preferences will be associated with user profiles or user types (Schewe and Thalheim, 2005).

In (Schewe and Thalheim, 2004) it was actually shown that story algebras carry the structure of Kleene algebras with tests (KATs, (Kozen, 1997)), thus term rewriting can be applied to reason about story spaces, in particular, as KATs subsumes propositional Hoare logic (Kozen, 1999). In (Schewe and Thalheim, 2004) this was applied to personalisation with respect to user preferences, which can be formalised by equations, and user goals, which can be formalised by postconditions. However, it was left open where the preferences were to be taken from. According to this model several dimensions are used to characterise user properties and for each dimension a usually ordered set of values is provided. A user profile is determined by values for each of the dimensions, whereas a user type combines several value combinations. While this is an accurate approach in accordance with more application oriented approaches, e.g. learner modelling for e-learning systems, it bears the risk of having to define too many user types. Preferably, it would be better, if equations that express preferences would be made explicitly depending on values of some dimensions. This avoids identifying the user types and may be used to determine evolving user-specific story spaces. However, it is very unlikely that user types are compositional, so that we must expect non-monotonicity in personalisation using such an approach. We will demonstrate this approach to personalisation in another section.

Obligations and rights that are associated with roles have been formalised in (Schewe and Thalheim, 2005) using a propositional variant of deontic logic (Eiter and Subrahmanian, 1999). This permits expressing deontic constraints already on a high level of abstraction. However, all these high-level propositional approaches to formalising preferences, goals, obligations and rights with their reasoning facilities provide only a first and rather coarse handling of the problem. In order to fully capture the problem we have to look at a finer
level of detail taking the data content and the functionality that is available at each scene into account. For this the method in (Schewe and Thalheim, 2005) provides media types that are extended views on some underlying database schema. The data model for this schema is of minor importance as long as the query language used for the views is expressive enough. For instance, it is well possible to assume that the database schema is modelled using HERM (Thalheim, 2000), while the query language constructs rational trees (Schewe and Thalheim, 2005).

One of the extensions to the views captures explicitly adaptivity to users, channels and end-devices by using cohesion pre-orders or proximity values, which both enable a controlled split of information. For our purposes here, however, the extension by operations is a greater importance. As briefly indicated in (Schewe, 2004) the media types permit the refinement of propositional conditions in the story algebra by formulae in higher-order logic, while the operations refine the actions in the story space. These operations can be modelled as abstract programs, so that the whole story space will be refined by expressions of higher-order dynamic logic (Harel et al., 2000). This can be used to set up proof obligations for personalisation, static and dynamic consistency and story space correctness (Schewe, 2004).

With respect to obligations and rights, the deontic constraints associated with the storyboard can be refined as well using the full power of deontic logic for agents (Eiter and Subrahmanian, 1999; Elgesem, 1997; Nute, 1997). Similar use of deontic logic has been made in (Broersen et al., 2002; Dignum et al., 1996; Wieringa and Meyer, 1993; Wieringa and Meyer, 1991). In particular, we obtain a level of granularity that will allow us to model the cooperation of actors including messaging and task delegation. The non-monotonicity requests to take defeasibility into account as well (Antoniou et al., 2001). In the last section before the conclusion we will present this refined approach to the problem of formalising preferences, obligations and rights.

**Storyboarding**

A *storyboard* describes the ways users may choose to interact with the system. It consists of three parts: a model of the story space, a model of actors, and a model of tasks. The *story space* describes the paths users follow while navigating through the WIS. These paths are the *stories*. Actors are groups of users with the same behaviour, which leads to modelling *user profiles* and *roles*. *Tasks* link the actions of the story space with the actors, but we will not discuss tasks in this paper.

**Story Spaces**

On a high level of abstraction we may think of a WIS as a set of abstract locations, which abstract from actual pages. A user navigates between these locations, and on this navigation path s/he executes a number of actions. We regard a location together with local actions, i.e. actions that do not change the location, as a unit called scene.

Then a WIS can be described by a edge-labelled directed graph, in which the vertices represent the scenes, and the edges represent transitions between scenes. Each such transition may be labelled by an action executed by the user. If such a label is missing, the transition is due to a simple navigation link. The whole graph is then called the
Fig. 1. A Story Space for On-Line Loan Application.
**story space.** Scenes may be atomic or complex. In the latter case the scene itself represents another set of abstract locations, between which users can navigate. This gives rise to a multi-layer model.

**Example 1.** Figure 1 shows the graph of a story space for on-line loan applications. We numbered the scenes and the actions, so referring to this example we only talk of scene $s_i$ and action $\alpha_j$. In addition, scenes and actions are named, and the names have been chosen in a way to be understandable without deeper explanation.

The scene $s_0 = \text{start}$ is the start scene, i.e. a user using the WIS will start here. Only one action $\alpha_1 = \text{enter \_ load \_ system}$ can be chosen in this scene. This action leads to a transition to scene $s_1 = \text{type \_ of \_ loan}$, in which general information about loans will be provided. A user has several actions to choose from in this scene. The actions $\alpha_7 = \text{select \_ personal \_ loan}$ and $\alpha_{13} = \text{select \_ mortgage}$ lead to the scene $s_2 = \text{applicant \_ details}$, the starting scene for the core of a loan application. Alternatively, a user may choose the action $\alpha_2 = \text{look \_ at \_ loans \_ at \_ a \_ glance}$ requesting an overview on all available loans with a short description for each, which will be done in scene $s_{10} = \text{loan \_ overview}$. From here a simple transition without action, indicated by the label skip leads back to the scene $s_1$.

Similarly, actions $\alpha_3 = \text{request \_ personal \_ loan \_ details}$ and $\alpha_5 = \text{request \_ mortgage \_ details}$ to scenes $s_{11} = \text{personal \_ loan}$ and $s_{13} = \text{mortgage}$, respectively, in which details about a particular personal loan or mortgage will become available. In both cases a user can navigate back to scene $s_1$, or request samples for the personal loan or mortgage by choosing the action $\alpha_{12} = \text{look \_ at \_ personal \_ loan \_ samples}$ in scene $s_{11}$ or $\alpha_6 = \text{look \_ at \_ mortgage \_ samples}$ in scene $s_{13}$, respectively, which will lead to scene $s_{12} = \text{personal \_ loan \_ samples}$ or $s_{14} = \text{mortgage \_ samples}$, respectively. The last action $\alpha_{31} = \text{specify \_ purpose}$ in scene $s_1$ takes a user to scene $s_{15} = \text{loan \_ purpose}$, in which advice on suitable loans for a specified purpose can be sought.

The part of the story space starting from scene $s_2$ deals with gathering applicants' details, details of the selected mortgage or personal loan or mortgage such as the amount requested, terms and conditions and securities, details concerning the incoming of the applicants, and finally a confirmation of the application. It can be understood in the same way, as the names of scenes and actions are self-explanatory - using "pl" and "m" as shortcuts for personal loan and mortgage, respectively.

At a finer level of detail we would like to extend the story space and indicate, under which conditions an action or scene transition can or must be executed and what will be the effect of an action. In doing so we associate a **precondition** with each action to specify exactly, under which conditions an action can be executed. Such preconditions can be easily expressed on the basis of propositional logic. That is, we take a set \{\varphi_1,\ldots,\varphi_n\} of atomic propositions and complete this to a propositional logic defining the set $\Phi$ of **propositional formulae** in the usual way, i.e.

- Each atomic proposition $\varphi_i$ is a proposition in $\Phi$.
- If $\varphi,\psi \in \Phi$, then also $\neg \varphi, \varphi \land \psi, \varphi \lor \psi$ and $\varphi \rightarrow \psi$ are propositions in $\Phi$. 
Thus, the *precondition* of an action \( \alpha \) is a proposition \( \text{Pre}(\alpha) \in \Phi \). A user can only choose the action \( \alpha \) and execute it, if the precondition \( \text{Pre}(\alpha) \) is satisfied in the current state of the WIS.

In a similar way we associate a *postcondition* \( \text{Post}(\alpha) \in \Phi \) with each action \( \alpha \) to specify exactly, which effects the action will have. That is, if the action \( \alpha \) is selected and executed, the resulting state of the WIS will satisfy the proposition \( \text{Post}(\alpha) \).

**Example 2.** Take another look at the story space represented by Figure 1, in particular at the action \( \alpha_3 = \text{select mortgage} \). In this case we may want that a user can only execute this action, if s/he has received the necessary information about available mortgages, i.e. mortgages must be known. This can be simply formalised by requesting \( \text{Pre(select mortgage)} = \text{mortgages known} \) using an atomic proposition. The interpretation of this atom should be that a user knows about mortgages.

Analogously, we may set \( \text{Post(select mortgage)} = \text{mortgage selected} \), which states that after executing \( \alpha_3 \) a mortgage has been selected.

**Plots**

The introduction of pre- and postconditions of actions opens a new perspective to the story space. We may look at the *plot*, which give a detailed, yet still high-level specification of functionality. In addition to pre- and postconditions, actions can be executed sequentially or in parallel, they can be iterated, and if several actions are available, users can choose between them. These extensions should also become part of the plot. These possibilities to combine actions lead to operators of an algebra, which we will call a *story algebra*. Thus, we can describe a story space by an element of a suitable story algebra.

Let us take now a closer look at the storyboarding language SiteLang (Thalheim and Düsterhöft, 2001), which defines a story algebra that captures the details of plots. For this we take the set \( A \) of actions and the set \( S \) of scenes as the basis for defining the set of *processes* \( P = P(A,S) \) determined by \( A \) and \( S \).

The set of *processes* \( P = P(A,S) \) determined by the set of actions \( A \) and the set of scenes \( S \) and the extension of the scene assignment \( \sigma \) to a partial mapping \( \sigma : P \to S \) are inductively defined as follows:

- Each action \( \alpha \in A \) is also a process.
- *skip* (or 1) is a process with no effect.
- *fail* (or 0) is a process that is not executable.
- If \( p_1 \) and \( p_2 \) are processes, then also the *sequence* \( p_1 \cdot p_2 \) is a process.
- If \( p_1 \) and \( p_2 \) are processes, then also the *choice* \( p_1 + p_2 \) is a process.
- If \( p \) is a process, then also the *iteration* \( p^* \) is a process.
If \( p \) is a process and \( \varphi \) is a boolean condition, then the \textit{guarded process} \( p\varphi \) and the \textit{post-guarded process} \( p\varphi \) are processes.

We would like to define also the \textit{parallel process} \( p_1 \parallel p_2 \). However, as we are mainly interested in the stories, we can consider \( p_1 \parallel p_2 \) as a shortcut for \( p_1p_2 + p_2p_1 \), i.e. we just state that the order of the component processes does not matter.

A plot may contain boolean conditions. The rationale behind this is that such a condition can be identified with a test that checks the condition. In particular, we write \( \varphi \lor \psi \) for the disjunction of conditions \( \varphi \) and \( \psi \), \( \varphi \land \psi \) for their conjunction, and \( \neg \varphi \) for a negated condition. This unusual notation is in accordance with the notation for Kleene algebras with tests, which we will exploit in the next section. In particular, the overloaded use of \( + \) for disjunction and choice, \( \cdot \) for conjunction and sequence, 1 for true and \textit{skip}, and 0 for false and \textit{fail} does not lead to conflicts.

**Example 3.** A plot expression for the detailed loan application in Figure 1 can be described as follows:

\[
\begin{align*}
\alpha_1 (\varphi_0 (\alpha_2 \varphi_{13} +1) \alpha_5 + \varphi_1 \alpha_5 (\alpha_5 +1) \varphi_3 + \varphi_2 \alpha_4 (\alpha_6 +1) \varphi_4) & \cdot \varphi_8 (\alpha_4 \varphi_6 + \alpha_6 \varphi_7) \\
\end{align*}
\]

The scenes and actions have already been named in Figure 1. Furthermore, the plot involves the Boolean conditions

\[
\begin{align*}
\varphi_0 &= \text{information\_loan\_types\_needed} & \varphi_1 &= \text{information\_personal\_loans\_needed} \\
\varphi_2 &= \text{information\_mortgages\_needed} & \varphi_3 &= \text{personal\_loans\_known} \\
\varphi_4 &= \text{mortgages\_known} & \varphi_5 &= \text{available\_loans\_known} \\
\varphi_6 &= \text{personal\_loan\_selected} & \varphi_7 &= \text{mortgage\_selected} \\
\varphi_8 &= \text{personal\_loan\_application\_completed} & \varphi_9 &= \text{mortgage\_application\_completed} \\
\varphi_{10} &= \text{applied\_for\_personal\_loan} & \varphi_{11} &= \text{applied\_for\_mortgage} \\
\varphi_{12} &= \text{payment\_options\_clear} & \varphi_{13} &= \text{loans\_recommended}
\end{align*}
\]

As in the previous examples the names chosen for the propositional atoms should be self-explanatory, e.g. \( \varphi_0 = \text{information\_loan\_types\_needed} \) expresses that the user needs information about the available types of loans, while \( \varphi_{12} = \text{payment\_options\_clear} \) indicates that the different payment options for mortgages are understood by the user.

**User Roles: Deontic Logic for Web Information Systems**

Users can be classified according to their roles, goals and behaviour. We use the term \textit{actor} for such a group of users. The \textit{role} of an actor indicates a particular purpose of the
system. As such it is usually associated with obligations and rights, which lead to deontic integrity constraints.

Roles are used to classify actors according to their rights and obligations. For instance, in a web-based conference system we may have roles for the programme committee chair(s), the programme committee members, and for authors. In an on-line loan system we may distinguish between actors in the role of customers and those in the role of bank clerks. In most systems we may expect one default role that is taken on by any actors, whereas other roles require some form of identification or at least an active switch to this role.

Let us now look briefly at a more sophisticated way to express rights and obligations that depend on other actions. An obligation specifies what an actor in a particular role has to do. A right specifies what an actor in a particular role is permitted to do. Both obligations and rights together lead to complex deontic integrity constraints. We use the following logical language \( L \) for this purpose:

- All propositional atoms are also atoms of \( L \).
- If \( \alpha \) is an action on scene \( s \) and \( r \) is a role associated with \( s \), then \( O\text{do}(r,\alpha) \), \( P\text{do}(r,\alpha) \) and \( F\text{do}(r,\alpha) \) are atoms of \( L \).
- For \( \varphi, \psi \in L \) we also have \( \neg \varphi \), \( \varphi \land \psi \), \( \varphi \lor \psi \), \( \varphi \rightarrow \psi \) and \( \varphi \leftrightarrow \psi \) are also formulae in \( L \).

Here we use the more familiar notation for negation, conjunction and disjunction than in the previous subsection. For the time being both approaches do not interfere with each other.

The interpretation is standard. In particular, \( O\text{do}(r,\alpha) \) means that an actor with role \( r \) is obliged to perform action \( \alpha \), \( P\text{do}(r,\alpha) \) means that an actor with role \( r \) is permitted to perform action \( \alpha \), and \( F\text{do}(r,\alpha) \) means that an actor with role \( r \) is forbidden to perform action \( \alpha \).

**Example 4.** The on-line loan example plot from Example 3 only contains one role customer, thus all actions and scenes in this example are associated with these role. Nevertheless, we may express some deontic constraints using the logical language defined above.

A customer has the obligation to leave his/her details (action \( \alpha_9 \)), once a personal loan or a mortgage has been selected (condition \( \varphi_6 \) or \( \varphi_7 \)). Furthermore, if a mortgage is selected (condition \( \varphi_7 \)), the customer is obliged to describe securities (action \( \alpha_{16} \)) and to enter obligations (action \( \alpha_{17} \)), i.e. we obtain the deontic constraints \( \varphi_6 \lor \varphi_7 \rightarrow O\text{do}(\text{customer},\alpha_8) \) and \( \varphi_7 \rightarrow O\text{do}(\text{customer},\alpha_{16}) \land O\text{do}(\text{customer},\alpha_{17}) \). Furthermore, a customer is allowed to look at mortgage samples (action \( \alpha_6 \)), which is simply expressed by \( P\text{do}(\text{customer},\alpha_6) \).
We may of course extend the on-line loan system further by adding roles bank clerk and mortgage advisor and further deontic constraints.

**User Profiles and Types**

Modelling the behaviour of an actor leads to *user profiles*, which can be modelled by giving values for various properties that characterize a user. Furthermore, each profile leads to preference rules that can again be expressed by constraints on the story space.

While roles classify actors according to their rights and obligations, *user profiles* and *user types* classify actors according to their behaviour. That is, roles indicate a pro-active approach to WIS modelling, whereas user profiling is reactive.

The general approach is to start with characterising properties of users and to provide values for each of these properties. Each combination of such values defines a user profile. However, the behaviour for some of these profiles is usually the same. So we combine user profiles to user types.

Furthermore, each user profile or type leads to rules, in particular preference rules. According to the characterising aspects of WISs we may distinguish between preferences concerning the content, the functionality or the presentation. In the context of storyboarding such preferences can be expressed by constraints on the story space.

We will now discuss in more detail these user profiles and types. We start with a finite set $\Delta$ of *dimensions* capturing the properties of users. For each dimension $\delta \in \Delta$ we assume to be given a domain $\text{dom}(\delta)$. Some of these domains may be totally ordered, i.e. there is a total order $\leq$ on $\text{dom}(\delta)$ usually, we drop the index. A totally ordered domain is also called a *scale* of the dimension.

As there are usually many dimensions, each of which has a domain with at least two elements, this gives us many ways to combine these values to obtain user profiles. In order to be able to manage the combinatorial explosion, we introduce user types.

If $\Delta = \{\delta_1, \ldots, \delta_n\}$ is a set of dimensions, then the set of *user profiles* (or the *user-grid*) over $\Delta$ is $\text{gr}(\Delta) = \text{dom}(\delta_1) \times \ldots \times \text{dom}(\delta_n)$. A user type over $\Delta$ is a subset $U \subseteq \text{gr}(\Delta)$.

This way of defining a user type as any subset of the user-grid leaves a lot of freedom to define a set $\{U_1, \ldots, U_k\}$ of user-types for a particular story space. We must of course assure that this set of user types is *complete* in the sense that all user profiles are covered. That is, for each user profile $(v_{i_1}, \ldots, v_{i_k}) \in \text{gr}(\Delta)$, i.e. $v_i \in \text{dom}(\delta_i)$, there must exist at least one $j \in \{1, \ldots, k\}$ with $(v_{i_1}, \ldots, v_{i_k}) \in U_j$.

One might expect that user profiles that contribute to the same user type, do not differ too much in the values for the dimensions. For this consider cubes in the user-grid.

If $\Delta = \{\delta_1, \ldots, \delta_n\}$ is a set of dimensions, then a subset $U \subseteq \text{gr}(\Delta)$ is a cube iff it has the form $U = D_1, \ldots, D_n$, such that for all $i = 1, \ldots, n$, for which $\text{dom}(\delta_i)$ is totally ordered...
scale, whenever \( v_{i1}, v_{i2} \in D_i \) with \( v_{i1} \leq \delta v_{i2} \) holds, then also \( v_i \in D_i \) for \( v_{i1} \leq \delta v_i \leq \delta v_{i2} \).

Thus, one common way of defining user types is to consider cubes in the user-grid. We call these user types *kiviat*, because they can be easily represented by Kiviat graphs. We expect that most user types will be kiviat.

The major purpose of introducing user types is to provide ways to customise the WIS to its users, i.e. to personalise the system. In principle there are two ways of doing it. The first one would just introduce user-type-specific plots, whereas the second one would try to generate them out of preference rules and general rules about the story space. Therefore, it is a good idea to formulate preference rules. Even if user-specific plots are not derived but specified by WIS designers, the rules can be useful to quality check the customised plots.

In view of the algebraic constructors, that were used to specify plots, we consider preference rules in connection with these constructors. This leads to the following definition.

A *preference rule* associated with a user type \( U \) is expressed through one of the following equations:

- An equation of the form \( \varphi(\alpha + \beta) = \varphi\alpha \) expresses that a user conditionally prefers action \( \alpha \) over action \( \beta \), where the condition is expressed by \( \varphi \). The special case \( \varphi = \text{true} \) expresses an unconditional preference.

- An equation \( \begin{array}{l} p(\begin{array}{l} p_1 + p_2 \end{array}) = pp_1 \end{array} \) expresses the conditional preference of the process \( p_1 \) over \( p_2 \) after the process \( p \). The special case \( p = 1 \) expresses an unconditional preference.

- An equation \( \begin{array}{l} p_1p_2 + p_2p_1 = p_1p_2 \end{array} \) expresses a preference of order, i.e. if the processes \( p_1 \) and \( p_2 \) can be executed in arbitrary order, it is preferred to execute first \( p_1 \).

- An equation \( p^* = pp^* \) expresses that in case of an iteration it will be executed at least once.

**Story Space Customisation**

Let us now see how we can use preference rules to customise plots. For this we exploit the fact that our story algebra carries the structure of a Kleene algebra with tests (KAT) as shown in (Schewe and Thalheim, 2005).

**Kleene Algebras with Tests**

As the set of processes \( \mathcal{P} \) carries the structure of a Kleene algebra, the following conditions hold:

- \( + \) and \( \cdot \) are associative, i.e. for all \( p, q, r \) we must have \( p + (q + r) = (p + q) + r \) and \( p(qr) = (pq)r \);
• + is commutative and idempotent with 0 as neutral element, i.e. for all \( p, q \) we must have \( p + q = q + p \), \( p + p = p \) and \( p + 0 = p \);

• 1 is a neutral element for \( \cdot \), i.e. for all \( p \) we must have \( p1 = 1p = p \);

• for all \( p \) we have \( p0 = 0p = 0 \);

• \( \cdot \) is distributive over +, i.e. for all \( p, q, r \) we must have \( p(q + r) = pq + pr \) and \((p + q)r = pr + qr \);

• \( p^* q \) is the least solution \( x \) of \( q + px \leq x \) and \( qp^* \) is the least solution of \( q + xp \leq x \), using the partial order \( x \leq y \equiv x + y = y \).

In addition, the Boolean conditions involved in the processes, i.e. the "tests", form a Boolean algebra. In order to exploit these equations for story space customisation we further take the preference equations into consideration, so we can start from equations in the following form:

preferences: An equation of the form \( \varphi(\alpha + \beta) = \varphi\alpha \) expresses that a user conditionally prefers action \( \alpha \) over action \( \beta \), where the condition is expressed by \( \varphi \). The special case \( \varphi = 1 \) expresses an unconditional preference.

precondition: An equation of the form \( \overline{\alpha} \varphi = 0 \) (or equivalently \( \alpha = \varphi\alpha \)) expresses that the condition \( \varphi \) is a precondition for the action \( \alpha \).

postcondition: An equation of the form \( \alpha \overline{\varphi} = 0 \) (or equivalently \( \alpha = \alpha\varphi \)) expresses that the condition \( \varphi \) is a postcondition for the action \( \alpha \).

invariance: An equation of the form \( \alpha \varphi = \varphi\alpha \), which is equivalent to \( \overline{\varphi}\alpha = \alpha\overline{\varphi} \) and to \( \varphi\alpha\overline{\varphi} + \overline{\varphi}\alpha\varphi = 0 \), expresses that the condition \( \varphi \) (and so its negation \( \varphi \)) is invariant under the action \( \alpha \).

exclusion: An equation of the form \( \varphi\psi = 0 \) expresses that the conditions \( \varphi \) and \( \psi \) exclude each other.

parallelism: An equation of the form \( \alpha\beta = \beta\alpha \) expresses that the order of the actions \( \alpha \) and \( \beta \) is irrelevant, hence they behave as if they were executed in parallel.

Then we can formalise the following optimisation task, in which the minimality request refers to the order \( \leq \) on processes, i.e. elements of the KAT, that was defined above:

Given a process \( p \in K \) that represents a plot of a story space, and a set \( \Sigma \) of equations on \( K \) that represents (among other constraints) user preferences and a postcondition \( \psi \), we look for a minimal process \( p' \) such that \( p\varphi = p'\varphi \) holds.

That is, the resulting process \( p' \) is a personalisation of \( p \) according to the user intention formalised by \( \psi \). We illustrate this kind of propositional reasoning with KATs by the following example.
Example 5. Let us continue Example 3 and look at a user who is going to apply for a home loan. This can be expressed by the goal $\varphi_{10}$. Then we express application knowledge by the equations $\varphi_{10}\varphi_{11} = 0$ (a user either applies for a home loan or a mortgage, not for both), $\varphi_{10}\varphi_{3} = 0$ (a user applying for a home loan does not complete a mortgage application) and $\varphi_{10}\varphi_{7} = 0$ (a user either selects a home loan or a mortgage, but not both).

Then we can simplify $p\varphi_{10}$ with the expression $p$ from Example 3 step by step. First we get $(\varphi_{10} + \varphi_{11})\varphi_{10} = \varphi_{10}$, which can then be used for

$$
(\varphi_{6}\varphi_{5}(\varphi_{5} + 1)\alpha_{5}\alpha_{10}\alpha_{11}\alpha_{12}\varphi_{10} + \varphi_{1}\varphi_{6}\varphi_{5}\alpha_{14}\alpha_{15}\alpha_{16}\alpha_{17}(\varphi_{12}\alpha_{13}\alpha_{19})^{*}\varphi_{12}\alpha_{18}\varphi_{10})\varphi_{10} = \\
\varphi_{6}\varphi_{5}(\varphi_{5} + 1)\alpha_{5}\alpha_{10}\alpha_{11}\alpha_{12}\varphi_{10} + \varphi_{1}\varphi_{6}\varphi_{5}\alpha_{14}\alpha_{15}\alpha_{16}\alpha_{17}(\varphi_{12}\alpha_{13}\alpha_{19})^{*}\varphi_{12}\alpha_{18}\varphi_{10}
$$

Then finally we get

$$
(\alpha_{5}\varphi_{5} + \alpha_{11}\varphi_{7})\varphi_{6}\varphi_{5}(\varphi_{5} + 1)\alpha_{5}\alpha_{10}\alpha_{11}\alpha_{12}\varphi_{10} = \\
\alpha_{5}\varphi_{5}\varphi_{6}\varphi_{5}(\varphi_{5} + 1)\alpha_{5}\alpha_{10}\alpha_{11}\alpha_{12}\varphi_{10} \\
+ \alpha_{11}\varphi_{7}\varphi_{6}\varphi_{5}(\varphi_{5} + 1)\alpha_{5}\alpha_{10}\alpha_{11}\alpha_{12}\varphi_{10} = \\
\alpha_{5}\varphi_{5}\varphi_{6}\varphi_{5}(\varphi_{5} + 1)\alpha_{5}\alpha_{10}\alpha_{11}\alpha_{12}\varphi_{10}
$$

This means that the story space can be simplified to

$$
\alpha_{5}\left(\varphi_{6}(\alpha_{2}\varphi_{13} + 1)\alpha_{2} + \varphi_{4}\alpha_{3}(\varphi_{5} + 1)\varphi_{3} + \varphi_{2}\alpha_{4}(\varphi_{6} + 1)\varphi_{4}\right)^{*}\varphi_{3}
$$

$$
\alpha_{5}\varphi_{6}\varphi_{5}(\varphi_{5} + 1)\alpha_{5}\alpha_{10}\alpha_{11}\alpha_{12}\varphi_{10}
$$

This simply means that for a user who is looking for a home loan application the part of the story space that deals with mortgage application will be cut out.

Decidability and Complexity

Let us now look at the personalisation problem in the light of the theory of KATs. Our formalisation in the previous section has led to an optimisation problem: we look for a minimal $p' \leq p$ with $\Sigma \models p'\psi = p\psi$. However, in the theory of KATs so far only completeness and the decidability of decision problems have been investigated. In other words, we only know about the decidability and complexity of problems of the form $\Sigma \models p = q$.

However, if the expression $p$ that describes the story space is *-free, there will be only finitely many expressions $p'$ with $p' \leq p$. In this case we can rephrase our problem by the following decision problem:
Given process $p, p' \in K$ such that $p$ represents a story space and $p' \leq p$ holds, and a set $\Sigma$ of equations on $K$ that represents (among other constraints) user preferences and a postcondition $\psi \in B$, then decide, whether $\Sigma \models p\psi = p'\psi$ holds.

If $p$ involves the $*$-operator, say we have a sub-expression $q^*$, it may be the case that replacing $q^*$ by $\left(1 + q + q^2 + \cdots + q^n\right)$ leads to some $p'$, for which $\Sigma \models p\psi = p'\psi$ holds.

In this case adopt the following pragmatic strategy:

1. Check whether for some $n$ replacing $q$ by $\left(1 + q + q^2 + \cdots + q^n\right)$ leads to some $p'$ with $\Sigma \models p\psi = p'\psi$.

2. If such an $n$ exists, replace $p$ by $p'$, thus remove the occurrence of the $*$-operator.

3. If such an $n$ does not exist, leave the $*$-operator in $p$, and consider only those expressions $p' \leq p$, for which $q^*$ is replaced by $\left(q'\right)^*$ with $q' \leq q$.

While this strategy may miss out to find the optimal solution, it guarantees that we are able to reduce personalisation to a decision problem.

For the equational theory of KATs we know from (Kozen and Smith, 1996) that it is decidable, but PSPACE-complete. That is, we can decide in polynomial space, whether $p = q$ can be derived from the axioms of KATs. However, the decision problem we are dealing with is of the form $\Sigma \models p = q$ with a set of equations $\Sigma$, i.e. in the Horn theory of KATs.

From (Kozen, 2002) we already know that such problems are undecidable in general. Even if we reduce ourselves to Kleene algebras instead of KATs, we already get $\Sigma_0$-completeness, i.e. the problem is in general recursively enumerable hard.

However, the problem we deal with is not the general decision problem for the Horn theory, as in our case $\Sigma$ only contains equations that have a particular form. From (Kozen and Smith, 1996) we know that we can reduce equations of the form $r = 0$ - these include equations obtained from pre- and postconditions, exclusion conditions, and invariance equations - to the equational theory. Instead of showing $r = 0 \rightarrow p\psi = p'\psi$ we can equivalently show $p\psi + upu = p'\psi + upu$, where $u$ is a "universal" process. That is $u = (\alpha_1 + \cdots + \alpha_n)^*$, where the $\alpha_i$ are all the atomic actions that are not tests - it is easy to see (and crucial for the proof) that $p \leq u$ holds for all processes $p$. So we can remove all equations of the form $r = 0$ from $\Sigma$ and enrich the equation that is to be derived instead.

This leaves us with only two kinds of equations: those arising from conditional preferences and those arising from parallelism. Recall that the former ones have the form $\phi(\alpha + \beta) = \phi\alpha$. Here we apply a little trick and introduce exclusive postcondition $\psi_\alpha$ and $\psi_\beta$ for $\alpha$ and $\beta$, respectively. Thus, we have the equations $\psi_\alpha\psi_\beta = 0$, $\varphi_\alpha\alpha = 0$ and $\varphi_\beta\beta = 0$. Then, we can replace the preference equation by $\phi(\alpha + \beta)\psi_\beta = 0$. Hence,
we get again equations of the form $r = 0$, which can be used to reduce the problem to a decision problem in the equational theory of KATs, which is PSPACE-complete.

**Database Issues: The Power of Media Types**

Each user of a WIS has information needs that have to be satisfied by the system. These information needs are usually not known in advance. Part of the needed information may depend on other parts, on decisions made while navigating through the WIS, and even on the information provided by the user him/herself. That is, the information portfolio of a user depends on the path through the WIS, i.e. in our terminology on the story.

**Media Types**

Therefore, assuming that there is a database for the data content of the WIS with database schema $S$, the information need on a scene $s$ definitely accounts for a view $V_s$ over $S$. That is, we have another schema $S_v$ and a computable transformation from databases over $S$ to databases over $S_v$. Such a transformation is usually expressed by a query $q_v$.

This leads us to media types (Schewe and Thalheim, 2005). In principle, it is not important what kind of database we have as long as there exists a sufficiently powerful query mechanism that permits to define views. So assume to be given a database schema $S$, which itself is based on some underlying type system such as (using abstract syntax)

$$t = b \{ (a_1 : t_1, \ldots, a_n : t_n) | \{ t \} \}.$$

Here $b$ represents an arbitrary collection of base types, e.g., $BOOL$ for boolean values $T$ and $F$, $I$ for a single value 1, $TEXT$ for text, $PIC$ for images, $MPIC$ for video data, $CARD$ and $INT$ for numbers, $DATE$ for dates, $ID$ for object identifiers, $URL$ for URL-addresses, $MAIL$ for e-mail-addresses, etc. The constructors $(\cdot)$ and $(\cdot)$ are used for records and finite sets.

For each type $R \in S$ we have a representing type $t_R$. In a $S$-database $db$ each type $R \in S$ gives rise to a finite set $db(R)$ consisting of pairs $(i,v)$ with $i$ of type $ID$ and $v$ of type $t_R$. Using this we may set up a powerful query language adapting a logic programming approach as in (Abiteboul and Kanellakis, 1989).

Thus, a query will be expressed as a set of rules (precisely: a sequence of such sets). Evaluating the rule body on a given database will result in bindings of variables, and the corresponding bindings for the head together with the creation of new identifiers for unbound variables results in an extension to the database. These extensions are iterated until a fixed point is reached.

In order to formalise this, assume to be given countable sets of variables $V$ for each type $t$. These sets are to be pairwise disjoint. Variables and constants of type $t$ are terms of that type. In addition, each $R \in S$ is a term of type $\{ \text{id} : ID, \text{value} : t_R \}$, and for each variable $i$ of type $ID$ (and $URL$, respectively) there is a term $\hat{i}$ of some type $t(i)$. If
\( \tau_1, \ldots, \tau_k \) are terms of type \( t \), then \( \{ \tau_1, \ldots, \tau_k \} \) is a term of type \( \{t\} \), and if \( \tau_1, \ldots, \tau_k \) are terms of type \( t_1, \ldots, t_k \), respectively, then \( (a_i : \tau_i, \ldots, a_k : \tau_k) \) is a term of type \( (a_i : t_i, \ldots, a_k : t_k) \).

If \( \tau_1, \tau_2 \) are terms of type \( \{t\} \) and \( t \), respectively, then \( \tau_1(\tau_2) \) is a positive literal (also called a fact) and \( \neg \tau_1(\tau_2) \) is a negative literal. If \( \tau_1, \tau_2 \) are terms of the same type \( t \), then \( \tau_1 = \tau_2 \) is a positive literal and \( \tau_1 \neq \tau_2 \) is a negative literal. A ground fact is a fact without variables.

A rule is an expression of the form \( L_0 \leftarrow L_1, \ldots, L_k \) with a fact \( L_0 \) (called the head of the rule) and literals \( L_1, \ldots, L_k \) (called the body of the rule), such that each variable in \( L_0 \) not appearing in the rule's body is of type ID or URL, respectively. A logic program is a sequence \( P_1; \ldots; P_i \), in which each \( P_i \) is a set of rules.

Finally, a query \( Q \) on \( S \) is defined by a type \( t_Q \) and a logic program \( P_Q \) such that a variable \( \text{ans} \) of type \( \{\text{url} : \text{URL}, \text{value} : t_Q\} \) is used in \( P_Q \). A boolean query can be described as a query \( Q \) with type \( 1 \). Alternatively, as we are not interested in creating a URL for the answer, we can simplify the approach above and consider a logic program, in which a variable \( \text{ans} \) of type \( \{\text{BOOL} = 1\} \) appears.

A logic program \( P_1; \ldots; P_i \) is evaluated sequentially. Each set of rules is evaluated by computing an in inflationary fixed point as in inflationary DATALOG. That is, start with the set of ground facts given by the \( S \)-database \( db \). Whenever variables in the body of a rule can be bound in a way that all resulting ground literals are satisfied, then the head fact is used to add a new ground fact. Whenever variables in the head cannot be bound in a way that they match an existing ground fact, the variables of type ID and URL will be bound to new identifiers or URLs, respectively.

Example 6. Consider a query \( Q \) with type \( t_Q \) defined as

\[
\begin{align*}
\text{type} & : \text{STRING}, \\
\text{conditions} & : \text{STRING}, \\
\text{interest} & : \text{STRING}, \\
\text{amount} & : \text{CARD}, \\
\text{disagio} & : \text{RATIONAL}, \\
\text{interest\_rate} & : \text{RATIONAL}, \\
\text{object} & : \text{STRING}, \\
\text{securities} & : \{(\text{value} : \text{RATIONAL}, \text{object} : \text{STRING}), \\
\text{type} : \text{STRING}\}, \\
\text{customers} & : \{(\text{income} : \{(\text{type} : \text{STRING}, \\
\text{amount} : \text{CARD}, \\
\text{frequency} : \text{STRING})\}) \\
\text{obligations} & : \{(\text{type} : \text{STRING}, \text{amount} : \text{CARD}, \\
\text{frequency} : \text{STRING})\}\}
\end{align*}
\]

Assume the database schema \( S \) contains (among others) the types \( \text{Loan\_Type} \), \( \text{Customer} \), \( \text{Mortgage} \), \( \text{Owes\_Mortgage} \), \( \text{Security} \), \( \text{Income} \), and \( \text{Obligation} \) with the following representing types:
$t_{\text{Loan\_Type}} = (\text{type : STRING, conditions : STRING, interest : STRING})$

$t_{\text{Customer}} = (\text{customer\_no : CARD, name : STRING, address : STRING, date\_of\_birth : DATE})$

$t_{\text{Mortgage}} = (\text{type : ID, mortgage\_no : CARD, amount : CARD, disaio : RATIONAL, interest\_rate : RATIONAL, begin : DATE, end : DATE, object : STRING})$

$t_{\text{Owes\_Mortgage}} = (\text{who : ID, what : ID, begin : DATE, end : DATE})$

$t_{\text{Security}} = (\text{whose : ID, for : ID, value : CARD, object : STRING, type : STRING})$

$t_{\text{Income}} = \{\text{who : ID, type : STRING, amount : RATIONAL, frequency : RATIONAL, account : CARD}\}$

$t_{\text{Obligation}} = (\text{who : ID, type : STRING, amount : RATIONAL, frequency : RATIONAL, account : CARD})$

Then the following logic program $P_Q$ will produce an anonymised set of mortgages:

$M_1(t, c, i, a, d, p, o, S, C, i_m) \leftarrow \text{Mortgage}(i_m, (i, n, a, d, p, b, e, o)), \text{Loan\_Type}(i, (t, c, i));$

$\hat{S}(v, o, t') \leftarrow \text{Security}(i, (i, i_m, v, o, t')), M_1(t, c, i, a, d, p, o, S, C, i_m) .$

$\hat{C}(I, O, i_c) \leftarrow \text{Customer}(i, (n', n, a, db)), \text{Owes\_Mortgage}(i_n, (i, i_m, b, e)), M_1(t, c, i, a, d, p, o, S, C, i_m);$

$\hat{I}(t', a', f) \leftarrow \hat{C}(I, O, i_c), M_1(t, c, i, a, d, p, o, S, C, i_m),$

$\text{Income}(i_m, (i, t', a', f, ac)).$

$\hat{O}(t', a', f) \leftarrow \hat{C}(I, O, i_c), M_1(t, c, i, a, d, p, o, S, C, i_m),$

$\text{Obligation}(i, (i, t', a', f, ac));$

$M_2(t, c, i, a, d, p, o, \hat{S}, D) \leftarrow M_1(t, c, i, a, d, p, o, S, C, i_m);$

$\hat{D}(I, D) \leftarrow \hat{C}(I, O, i_c), M_2(t, c, i, a, d, p, o, \hat{S}, D),$

$M_1(t, c, i, a, d, p, o, S, C, i_m);$

$\text{Ans}(u, (t, c, i, a, d, p, o, \hat{S}, \hat{D})) \leftarrow M2(t, c, i, a, d, p, o, \hat{S}, D).$

This logic program consists of a sequence of six sets of rules. In a nutshell, the fixed-point computed by the first set will contain the additional relation $M_1$, which collects information about mortgages including conditions and interest (from $\text{Loan\_Type}$) and new identifiers $S$ and $C$ for the sets of securities and customers, respectively. The fixed-
A point computed by the next set of rules contains details for each of these sets. Furthermore, for the customers it creates new identifiers for sets of incomes and sets of obligations, the details of which are computed by the third set of rules. The last three sets of rules replace the identifiers for the sets of securities and customers, respectively, by the corresponding sets.

An interaction type has a name $M$ and consists of a content data type $\operatorname{cont}(M)$ with the extension that the place of a base type may be occupied by a pair $l : M'$ with a label $l$ and the name $M'$ of an interaction type, a defining query $q_M$ with type $t_M$, and a set of operations. Here $t_M$ is the type arising from $\operatorname{cont}(M)$ by substitution of URL for all pairs $l : M'$.

Finite sets $C$ of interaction types define content schemata. Then an $S$-database $db$ and the defining queries determine finite sets $\db(M)$ of pairs $(u,v)$ with URLs $u$ and values $v$ of type $t_M$ for each $M \in C$. We use the notion pre-site for the extension of $db$ to $C$. The pair $(u,v)$ will be called an interaction object in the pre-site $db$. A boolean query on $S \cup C$ defines a condition $\phi$.

A media type extends an interaction type $M$ together in two ways: It adds adaptivity and hierarchies, which can be formalised by a cohesion pre-order $\preceq_M$ (or a set of proximity values) and a set of hierarchical versions $H(M)$, respectively. For our purposes here we concentrate on the operations that already come with the interaction types, for the extensions see (Schewe and Thalheim, 2005).

In general, an operation on a type $R$ consists of a signature and a body. The signature consists of an operation name $O$, a set of input-parameter/type pairs $i : t_i$ and a set of output parameter/type pairs $o : t'_o$. The body is an abstract program using the following constructs:

- 1 and 0 are abstract programs meaning skip and fail, respectively.
- An assignment $x := \text{exp}$ with a variable $x$ and an expression of the same type as $x$ is an abstract program. The possible expressions are defined by the type system. In addition, we permit expressions $\{P\}$ with a logic program $P$, assuming that $P$ contains a variable $\text{ans}$. The expression $\{P\}$ is interpreted as the result of the logic program bound to $\text{ans}$.
- If $P$, $P_1$ and $P_2$ are abstract programs, the same holds for the iteration $P^*$, the choice $P_1 \cdot P_2$ and the sequence $P_1 \cdot P_2 = P_1 P_2$.
- If $P$ is an abstract program and $\phi$ is a condition, then the guarded program $\phi P$ and the postguarded program $P \phi$ are also abstract programs.
• If $x$ is a variable and $P$ is an abstract program, then the selection $@x \cdot P$ is also an abstract program.

There are a few subtleties regarding variable scoping and restrictions to assignments that have to be taken into account for operations on interaction types (Schewe and Thalheim, 2005), but for our purposes here it is not relevant to discuss them.

With respect to the connection to the story space, the propositional conditions $\varphi$ now have to be refined to conditions on $S \cup C$, while each action $\alpha$ on a scene $s$ is refined by operations associated with the media type that supports $s$.

**Correctness, Consistency and Personalisation**

With the introduction of media types to support scenes we can no longer rely on simple equational reasoning using KATs. Therefore we introduce a higher-order dynamic, where the order comes from the intrinsic use of the set constructor and the logic programs in queries. In fact, instead of using logic programs with a semantics defined by inflationary fixed-points, we could use directly higher-order logic enriched with a fixed-point operator.

As a consequence, we may consider a logic program $P$ as a representative of a higher-order logical formula, say $\varphi_P$. If $\{P\}$ is used as the right-hand side of an assignment, then it will correspond to a term $\text{Ians}.\varphi_P$ denoting the unique $\text{ans}$ satisfying formula $\varphi_P$.

That is, all conditions turn out to be formulae of a logic $L$, which happens to be a higher-order logic with an inflationary fixed-point operator. From the point of view of expressiveness the fixed-point operator is already subsumed by the order, but for convenience we do not emphasise this aspect here.

Furthermore, by adding terms of the form $\text{I}x.\varphi$ with a formula $\varphi$ and a variable $x$ all assignments in operations are just "normal" assignments, where the left-hand side is a variable and the right-hand side is a term of $L$.

We now extend $L$ to a dynamic logic by adding formulae of the form $[p] \varphi$ with an abstract program $p$ and a formula $\varphi$ of $L$. Informally, $[p] \varphi$ means that after the successful execution of $p$ the formula $\varphi$ necessarily holds (Harel et al., 2000). In addition, we use the shortcut $\langle p \rangle \varphi \equiv \neg [p] \neg \varphi$, so $\langle p \rangle \varphi$ means that after the successful execution of $p$ it is possible that the formula $\varphi$ holds.

Using our recursive definition of abstract programs the following rules apply to $[p] \varphi$ for a complex abstract program $p$:

$[1] \varphi \equiv \varphi$

$[0] \varphi \equiv 0$

$[x := t] \psi \equiv \psi \{x / t\}$ (substitute all free occurrences of $x$ in $\psi$ by $t$)
\[ [p_1, p_2] \psi = [p_1][p_2] \psi \]
\[ [p_1 + p_2] \psi = [p_1] \psi \land [p_2] \psi \]
\[ [p^+] \psi \equiv \text{the weakest solution } \varphi \text{ of } \varphi \leftrightarrow \psi \land [p] \varphi \]
\[ [\varphi \psi] \equiv \varphi \rightarrow [p] \psi \]
\[ [p \varphi] \psi \equiv [p](\varphi \rightarrow \psi) \]
\[ [\exists x \cdot p] \psi \equiv \forall x.[p] \psi \]

The equivalence for the iteration operator refers to the implication order, i.e. if \( \varphi \models \psi \) holds, then \( \psi \) is called weaker than \( \varphi \). Further rules looking also at the structure of \( \psi \) are given in (Harel et al., 2000).

With these preparations we can rethink the reasoning about the story space. In the sequel we will discuss three applications of dynamic logic:

- We take a look at proof obligations for the operations that result from the specification of the story space.
- We take a look at proof obligations for the operations that arise from static and dynamic integrity constraints on the underlying database schema.
- We reconsider WIS personalisation in the light of dynamic logic as opposed to KATs.

**Story Space Proof Obligations.** Let \( p \) denote the KAT expression that represents the complete story space. If all conditions in \( p \) are replaced by conditions on the pre-site and all actions are replaced by the abstract programs defining the realising operations, we obtain an abstract program, which by abuse of notation shall still be denoted \( p \).

As a WIS has a general purpose, this can be formalised by some post-condition \( \psi \). Thus, \([p] \psi\) describes the weakest condition, under which the purpose of the system can be achieved. If \( \varphi \) characterises a precondition that should be sufficient for the achievability of the WIS purpose, then we obtain \( \varphi \rightarrow [p] \psi \) as a general story space proof obligation. In most cases we should expect \( \varphi \equiv 1 \).

Similarly, we may concentrate on fragments \( p' \) of the story space expression of the form \( \varphi p \psi \), which corresponds to a Hoare triplet \( \{ \varphi \} \ p \{ \psi \} \) (Hoare, 1969) and thus gives rise to a special story space proof obligation \( \varphi \rightarrow [p'] \psi \).

**Consistency Proof Obligations.** A static constraint on the underlying database schema \( S \) is a condition \( \zeta \), in which the free variables are among the \( R \in S \). Such constraints give rise to the request that whenever an operation is started in a database satisfying \( \zeta \), then the database reached after successfully completing the execution of the operation, must necessarily satisfy \( \zeta \), too.
That is, for all operations \( p \) that are defined on a pre-site and all static constraints \( \zeta \) we obtain a static consistency proof obligation \( \zeta \rightarrow [p]\zeta \).

A dynamic constraint on the underlying database schema \( S = \{R_1, \ldots, R_n\} \) is a condition \( \zeta' \), in which the free variables are among in \( S \cup S' \) with \( S' = \{R'_1, \ldots, R'_n\} \) and each \( R'_i \) having the same type as \( R_i \). The additional variables \( R'_i \) are used to distinguish between \( S \)-databases \( db \), on which an operation \( p \) is started, and \( S' \)-databases \( db' \) resulting after \( p \) has been successfully executed.

Obviously, a dynamic constraint \( \xi \) has to be interpreted on a pair \( \{db, db'\} \) of databases. Following a standard approach to dynamic consistency (Schewe et al., 1992) we associate with \( \xi \) an abstract program

\[
p(\xi) = \@ \{R'_1, \ldots, R'_n\} \bullet \xi R_i := R'_i \ldots R_n := R'_n.
\]

Then dynamic consistency of an operation \( p \) with respect to \( \xi \) means that \( p \) must "specialise" \( p(\xi) \), i.e. we require that \( [p(\xi)]\psi \rightarrow [p]\psi \) for all conditions \( \psi \) on \( S \).

Fortunately, this proof obligation can be rephrased using a renaming \( p' \) of \( p(\xi) \) given by

\[
p' = \@ \{R'_1, \ldots, R'_n\} \bullet \xi \{R_1 / R_1', \ldots, R_n / R_n'\} R_i := R'_i \ldots R_n := R'_n.
\]

Then the dynamic consistency proof obligation for \( p \) with respect to \( \xi \) becomes

\[
([p']p)\{R_i = R_i'^* \land \cdots \land R_n = R_n'^*\}\{R_i^*/R_i', \ldots, R_n^*/R_n\}.
\]

Personalisation Proof Obligations. The general approach to personalisation that was outlined in the previous section is still the same, i.e. we can assume a set \( \Sigma \) containing general constraints on \( S \cup C \) and specific constraints that refer to preferences of a particular user type. Examples of such preferences are the following:

An equation \( p_1 + p_2 = p_1 \) expresses an unconditional preference of operation \( p_1 \) over \( p_2 \).

- An equation \( \varphi(p_1 + p_2) = \varphi p_1 \) expresses a conditional preference of operation \( p_1 \) over \( p_2 \) in case the condition \( \varphi \) is satisfied.

- Similarly, an equation \( p(p_1 + p_2) = pp_1 \) expresses another conditional preference of operation \( p_1 \) over \( p_2 \) after the operation \( p \).

- An equation \( pp_1 p_2 = p_1 p_2 \) expresses a preference of order.

- An equation \( p^* = pp^* \) expresses that in case of an iteration of operation \( p \) it will be executed at least once.
Furthermore, personalisation assumes a postcondition $\chi$ that expresses the goals of the particular user. Then personalisation of story space $p$ aims at a simpler story space $p'$ such that $\left[p\right]\chi \leftrightarrow \left[p'\right]\chi$ holds.

**Conclusion**

In this paper we formalised user preferences, obligations and rights and indicated how this formalisation can be used to reason about the WIS specification. The work was done in the context of the Co-Design approach to WIS development (Schewe and Thalheim, 2005), but it seems to be not too difficult to adapt it to other methods such as ARANEUS (Atzeni et al., 1998), OOHDM (Schwabe and Rossi, 1998), WebML (Ceri et al., 2003), HERA (Houben et al., 2003) or WSDM (De Troyer and Leune, 1998).

However, the formalisation requires that the method contains models of users, user roles, actions and logical conditions that bring these models together. These models are a central component of Co-Design, but they are only partially available in other methods. That is, the coupling of the work presented in this paper with the other mentioned WIS development methods requires that these methods first take up directly or in a modified form the concept of storyboarding from the Co-Design approach.

We demonstrated that capturing preferences, obligations and rights can be achieved on the basis of various logics. For the personalisation of WISs according to preferences and goals we adopted propositional reasoning with Kleene algebras with tests, which at the end amounts to a term rewriting approach. From this a new challenge arises that may allow us to dispense with explicitly modelling user types. In fact, the main purpose of user types is to associate preference rules with them, but these preferences may depend only on some of the characteristics of a user type. Thus, it may be advantageous to use conditional preference rules instead, and conditional term rewriting may turn out to be the key to personalisation on a propositional storyboard level. This path of research will be explored in the future.

We also demonstrated that obligations and rights can be formalised in (propositional) deontic logic, but we were not yet able to take this approach beyond the first step of just capturing deontic constraints. For our future research we plan to investigate more deeply the application of deontic logic for reasoning purposes on WIS specifications.

The major challenge in dealing with preferences, obligations and rights, however, is the coupling of the approach with the detailed conceptual model of WISs that is given by media types, which are extended views on some underlying database schema. In logical terms this basically means that we leave the safe grounds of propositional reasoning, as actions are no longer atomic but refined by abstract programs, while propositional conditions are refined by conditions that can be evaluated on underlying databases. In this paper we did not go further than indicating the use of higher-order dynamic logic to obtain proof obligations. In the same way we should expect proof obligations in higher-order deontic logic.

Thus, in dealing with preferences, obligations and rights we shift the focus of WISs from just conceptual modelling of particular data-intensive application systems to knowledge-intensive systems that require reasoning techniques in complex logics to be explored.
References


View Integration and Cooperation in Databases, Data Warehouses and Web Information Systems

Hui Ma¹, Klaus-Dieter Schewe¹, Bernhard Thalheim², Jane Zhao¹

¹ Massey University, Department of Information Systems &
Information Science Research Centre
Private Bag 11 222, Palmerston North, New Zealand
[h.maj.k.d.schewe[j.zhao]@massey.ac.nz

² Christian Albrechts University Kiel
Department of Computer Science and Applied Mathematics
Olshausenstr. 40, D-24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de

Abstract. View integration aims at replacing a set of existing views by
a single new one in such a way that with respect to information capacity
the new view dominates or is equivalent to the old ones. Therefore, in this
article we first investigate a theory of schema equivalence and dominance
for the higher-order Entity-Relationship model (HERM) based on the
notion of computable queries. We then develop formal transformation
rules for schema integration that are embedded in a pragmatic method
telling how they should be applied for integration.

We then apply the approach to views, which occur as the basic con-
stituents for user interfaces as formalised by the notion of dialogue type.
In two follow-on steps we apply the rule-based view integration tech-
nique to data warehouses and web information systems. In the case of
data warehouses the fundamental idea is the separation of input from op-
erational databases and output to on-line analytical processing (OLAP)
systems. Both the extraction of data from the operational databases and
the definition of the data-marts for OLAP can be formulated by views.

In the case of web information systems, views form the core of media
types, which provide abstract means for describing content, functionality,
context and adaptivity to user preferences and intentions, end-devices,
and channel limitations. In this case the queries defining the views must
be highly expressive, as they must involve the creation of abstract iden-
tifiers, complex values and links. We extend the transformation rules to
cope with these requirements.

View cooperation provides an alternative to view integration in which
the integrated view is only virtual. That is the constituting views are
kept and exchange functions are designed to provide the same function-
ality as if the views were integrated.

Keywords. view integration, schema equivalence, data warehouses, web
information systems, view cooperation

* The work in this paper was supported by FRST/NERF grant MAUX0025 “DIMO –
Distributed Multi-Level Object Bases” and MU/ABRF grants 57413 “Foundations
of Conceptual Modelling” and 57501 “Distributed Data Warehouses”.

1 Introduction

Database schema integration is an old issue that has attracted a lot of research [12-14,16,30]. The starting point for schema integration is a set of schemata over some data models. Usually the focus is on two schemata over the same data model. If the underlying data models differ, then we may assume some preprocessing transforming both schemata – or one of them, if this is sufficient – into an almost equivalent schema over a data model with higher expressiveness. Then schema integration aims at replacing the given schemata by a single new one in such a way that the new schema dominates or is equivalent to the old ones.

A view on some database schema consists of another schema called the target schema, and a defining query, which maps instances of the source schema to instances of the target schema. If we integrate the target schemata of views we talk of view integration [3, 29,30]. In this case we obtain embeddings for each target schema into the new schema. If these embeddings are coupled with the defining queries for the given views we obtain a new defining query, i.e. we obtain not only an integrated target schema, but an integrated view.

So the first thing we need is a theory of schema equivalence and dominance. Roughly, we may say that two schemata are equivalent, if they have the same information capacity, i.e. we may store the same information in both schemata. A schema dominates another one, if it has a larger information capacity. Various formal definitions for equivalence and dominance have been given [9, 19,30] and compared in [15,16]. Here we extend this work and introduce novel definitions for equivalence and dominance based on computable queries [5, 34], i.e. we base the transformation functions on the most general notion of query, but we discard total arbitrariness, as computable queries must respect isomorphisms. We compare the new notions with the ones defined in [9, 19,30]. We base this comparison on the higher-order Entity-Relationship model (HERM) [30].

On this basis we then reconsider schema integration following the framework in [15, 16]. In a nutshell, we first “clean” the given schemata by removing name conflicts, synonyms and homonyms, then we add inter-schema constraints, which leads to a single constrained schema, which is just the union of the given ones. To this schema we then apply formal equivalence transformation or augmentation rules, which will finally take us to an integrated schema that is either equivalent to the union of the given ones or dominates this.

One group of rules addresses the restructuring of the complex attributes, entity types, relationship types and clusters. Another group of rules considers the shifting of attributes over hierarchies and paths. A third group of rules deals with selected integrity constraints such as keys, functional, inclusion and join dependencies, cardinality constraints and path constraints. A fourth group of rules is devoted to aggregation, decomposition, specialisation and generalisation.

The transformation rules can be used as well for view cooperation [15, 28,30], which provides an alternative to view integration in which the integrated view is only virtual. That is the constituting views are kept and exchange functions are designed to provide the same functionality as if the views were integrated.
Views are important for user interfaces. In [21] an integrated concept of *dialogue type* was developed, which combines views with operations defined on them. As conceptual interface specifications are a viable source during the systems development process, the problem of view integration comes up naturally. With respect to the transformation rules, we now need an extension that deals with the operations.

Data Warehouses are data-intensive systems that are used for analytical tasks in businesses such as analysing sales/profits statistics, cost/benefit relation statistics, customer preferences statistics, etc. The term used for these tasks is “on-line analytical processing” (OLAP) [33] in order to distinguish them from operational data-intensive systems, for which the term “on-line transaction processing” (OLTP) has become common. The idea of a data warehouse [11] is to extract data from operational databases and to store them separately. Thus, a first problem in data warehouse design is to integrate views from various source databases. This point of view of data warehouse design as a view integration problem has been strongly promoted in [12, 32, 36]. In fact, there is another related problem in here, the support of OLAP applications by materialised views, which in their totality reflect the data warehouse. In this sense data warehouses also involve a view design problem.

In [17] dialogue types have been applied to data warehouses and OLAP systems. The principle result states that OLAP functionality corresponds to operations on views over the data warehouse. So we may apply our extended transformation rules to this problem. In fact, the main idea of data warehouses implies a separation of input from operational databases and output to views that contain the data for particular OLAP tasks. This implies a three-layer architecture for data warehouses and OLAP systems [38], which is the basis of a formal design approach for such systems [26, 37].

In the field of web information systems (WISs) the importance of views is commonly accepted [2, 4, 24, 25]. In [7, 6] the notion of *media type* has been introduced. A media type is an extended view over some underlying database. However, different to the approaches in [2, 4] the defining query already constructs the navigation structure. Furthermore, media types are prepared for dynamic WISs [18] by adding operations, and for adaptivity by adding cohesion preorders. As a consequence, transformation rules for the purpose of media type integration have to extend the view integration rules in a way that cohesion is covered as well. In this article we will develop these extensions.

**Outline.** We start in Section 2 with a look at published work on the topic of this article. In Section 3 we describe the basics of HERM emphasising our approach to schema equivalence and dominance. Section 4 is devoted to the process of schema and view integration. We describe first the process in general followed by a presentation of the transformation rules for this process. We also show how these rules can be used for view cooperation instead of integration. Section 5 applies the integration framework to data warehouses and OLAP systems. We emphasise the particular case of view integration and indicate additional
rules dealing with the adaptation of dialogue operations. In Section 6 we briefly present media types, i.e. other extended views that are used for web information systems. We particularly focus on the aspect of adaptivity for which we discuss cohesion. This is followed by an extension to the transformation rules dealing with the implications of view integration to cohesion. We conclude with a short summary and discussion of further research directions.

2 Related Work

The work on view integration in [13, 14, 29] is based on the Entity-Relationship model. Larson et al. [14] consider containment, equivalence and overlap relations between attributes and types that are defined by looking at “real world objects”. Equivalence between types gives rise to their integration, containment defines a hierarchy with one supertype and one subtype, and overlapping gives rise to new common supertype. The work by Spaccapietra and Parent [29] considers also relationships, paths and disjointness relations between types. The work by Koh et al. [13] provides additional restructuring rules for the addition or removal of attributes, generalisation and specialisation, and the introduction of surrogate attributes for types.

The work by Biskup and Convent in [3] is based on the relational data model with functional, inclusion and exclusion dependencies. The method is based on the definition of integration conditions, which can be equality, containment, disjointness or selection conditions. Transformations are applied aiming at the elimination of disturbing integration conditions. In the same way as our work it is based on a solid theory. On the other hand, it has never been applied to large systems in practice. The approach by Sciore et al. in [28] investigates conversion functions on the basis of contexts added to values. These contexts provide properties to enable the semantic comparability of values.

The work by Lehmann and Schewe [15, 16] assumes that the given schemata are defined on the basis of the higher-order Entity-Relationship model (HERM) [30] which is known to provide enough expressiveness such that schemata existing in practice can be easily represented in HERM. The work relies on the notions of equivalence and dominance as defined for HERM in [30].

In [15, 16] these notions of equivalence and dominance are also compared with those defined by Hull [9] and Qian [19]. Basically, the four different notions of schema dominance introduced by Hull differ by the way the transformation functions are defined. In the simplest case (calculus dominance) they correspond to calculus queries, whereas in the most general case (absolute dominance) there are no restrictions at all. In fact, taking computable queries will remove the arbitrariness from absolute dominance that has been criticised in [19] while taking into account the most general form of queries [5, 34].

The design of data warehouses and OLAP applications has been intensively studied. Widom [36] and Theodoratos [32] emphasise that data warehouse design is mainly a view integration problem. Whereas Lewerenz et al. [17], Thomson
[33], and Zhao and Schewe [26,37,38] also take the OLAP functionality into account.

The design and development of WISs has attracted a lot of attention. The ARANEUS framework in [2] emphasises that conceptual modelling of web information systems should approach a problem triplet consisting of content, navigation and presentation. This leads to modelling databases, hypertext structures and page layout. OOHDM [27] is quite similar to ARANEUS, but its origins are not in the area of databases but in hypertext and it explicitly refers to an object oriented approach. OOHDM emphasises an object layer, hypermedia components, i.e. links, and an interface layer. The work in [8] also starts from hypertext design. The work introduces “authoring in the large”, i.e., the conceptual modelling of information elements and navigation, and uses this to categorise different types of links. Another similar approach is WebML [4], which emphasises a multi-level architecture for the data-driven generation of WIS, thus takes the view aspect into account. Furthermore, it emphasises structures, derivation and composition, i.e. views, navigation and presentation, thus addresses the same problem triplet as the ARANEUS framework.

Our own work on WIS design combines the usage-oriented storyboarding methodology [22,23,31] and the content- and functionality-oriented theory of media types [6,20,24], which are embedded in an integrated co-design methodology based on an abstraction layer model [25].

3 Schemata, Equivalence and Dominance

As we will base our presentation on the higher-order Entity-Relationship model (HERM) [30], we start with a brief review of the model as far as it is important for our purposes here. In particular, we focus on algebraic and logical query languages for HERM. These will be needed to address the important issue of defining schema dominance and equivalence in a way that the expressiveness is sufficient for the integration of extended views as needed for data warehouses and web information systems. The basic case dealing only with plain views over HERM schemata, i.e. ignoring the extensions by operations, adaptivity, etc. was already handled in [15,16].

3.1 HERM

The major extensions of the HERM compared with the flat ER model concern nested attribute structures, higher-order relationship types and clusters, a sophisticated language of integrity constraints, operations, dialogues and their embedding in development methods. Here we only review some of the structural aspects.

In the following let \( \mathcal{A} \) denote some set of simple attributes. Each simple attribute \( A \in \mathcal{A} \) is associated with a base domain \( \text{dom}(A) \), which is some fixed countable set of values. The values themselves are of no particular interest.
In HERM it is permitted to define nested attributes. For this take a set \( \mathcal{L} \) of labels with the only restriction that labels must be different from simple attributes, i.e. \( \mathcal{L} \cap \mathcal{A} = \emptyset \).

**Definition 3.1.** A nested attribute is either a simple attribute, the null attribute \( \bot \), a tuple attribute \( X(A_1, \ldots, A_n) \) with pairwise different nested attributes \( A_i \) and a label \( X \in \mathcal{L} \) or a set attribute \( X(A) \) with a nested attribute \( A \) and a label \( X \in \mathcal{L} \). Let \( \mathcal{NA} \) denote the set of all nested attributes.

We extend \( \text{dom} \) to nested attributes in the standard way, i.e. a tuple attribute will be associated with a tuple type, a set attribute with a set type, and the null attribute with \( \text{dom}(\bot) = \{\} \), where \( \{\} \) is the trivial domain with only one value.

In principle we could also permit other constructors than tuple and set constructors, e.g. constructors \( (\cdot) \) and \( [\cdot] \) for multisets and lists. This would, however, only complicate our presentation here without leading to additional insights. We therefore disregard these other constructors.

On nested attributes we have a partial order \( \geq \) defined as follows.

**Definition 3.2.** \( \geq \) is the smallest partial order on \( \mathcal{NA} \) with

- \( A \geq \bot \) for all \( A \in \mathcal{NA} \),
- \( X(A) \geq X(A') \iff A \geq A' \) and
- \( X(A_1, \ldots, A_n) \geq X(A'_1, \ldots, A'_m) \iff \bigwedge_{1 \leq i \leq m} A_i \geq A'_i \).

A generalised subset of a set \( F \subseteq \mathcal{NA} \) of nested attributes is a set \( G \subseteq \mathcal{NA} \) of nested attributes such that for each \( A' \in G \) there is some \( A \in F \) with \( A \geq A' \).

It is easy to see that \( X \geq X' \) gives rise to a canonical projection \( \pi_X : \text{dom}(X) \to \text{dom}(X') \).

Let us now define the entity and relationship types and clusters in HERM using the following compact definition.

**Definition 3.3.** A level-\( k \)-type \( R \) consists of a set \( \text{comp}(R) = \{r_1 : R_1, \ldots, r_n : R_n\} \) of labelled components with pairwise different labels \( r_i \), a set \( \text{attr}(R) = \{A_1, \ldots, A_m\} \) of nested attributes and a key \( \text{key}(R) \). Each component \( R_i \) is a type or cluster of a level at most \( k \) - 1, but at least one of the \( R_i \) must be level-(\( k - 1 \))-type or -cluster.

For the key we have \( \text{key}(R) = \text{comp}'(R) \cup \text{attr}'(R) \) with \( \text{comp}'(R) \subseteq \text{comp}(R) \) and a generalised subset \( \text{attr}'(R) \) of the set of attributes.

A level-\( k \)-cluster is \( C = R_1 \oplus \cdots \oplus R_n \) with pairwise different components \( R_i \), each of which is a type of a level at most \( k \). At least one of the \( R_i \) must be level-\( k \)-type.

The labels \( r_i \) used in components are called roles. Roles can be omitted in case the components are pairwise different. A level-0-type \( E \) — here the definition implies \( \text{comp}(E) = \emptyset \) — is usually called an entity type, a level-\( k \)-type \( R \) with \( k > 0 \) is called a relationship type.
A HERM schema is a finite set $S$ of entity types, relationship types and clusters together with a set $\Sigma$ of integrity constraints defined on $S$. We write $(S, \Sigma)$ for a schema, or simply $S$, if $\Sigma = \emptyset$.

Note that the notion of level has only been introduced to exclude cycles in schemata. Besides this it has no other meaning. Conversely, if there are no cycles in a schema, then there is a straightforward way to assign levels to the types and clusters in the schema such that the conditions in Definition 3.3 are satisfied.

![HERM diagram for loan application](image)

**Fig. 1.** HERM diagram for loan application

**Example 3.1.** The following type definitions define a HERM schema for a loan application as it might be used by some bank:

- **Loan_Type** = ($\emptyset$, { type, conditions, interest }, { type } )
- **Customer** = ($\emptyset$, { customer_no, name, address, date_of_birth },
  { customer_no } )
- **Personal_Loan** = ({ type : Loan_Type }, { loan_no, amount, interest_rate, begin, end, terms_of_payment }, { loan_no } )
- **Mortgage** = ({ type : Loan_Type }, { mortgage_no, amount, disagio, interest_rate, begin, end, object }, { mortgage_no } )
- **Loan** = **Home_Loan** $\oplus$ **Mortgage**
- **Account** = ({ ln : Loan }, { account_no, balance }, { account_no } )
- **Account_Record** = ({ a : Account }, { record_no, type, amount, date }, { a : Account, record_no } )
- **Owes** = ({ who : Customer, what : Loan }, { begin, end },
  { who : Customer, what : Loan, begin } )
Security = (\{ whose : CUSTOMER, for : MORTGAGE \}, \{ value, object, type \}, \{ whose : CUSTOMER, for : MORTGAGE, object \})

Income = (\{ who : CUSTOMER \}, \{ type, amount, frequency, account \},
\{ who : CUSTOMER, account \})

Obligation = (\{ who : CUSTOMER \}, \{ type, amount, frequency, account \},
\{ who : CUSTOMER, account \})

For this it is easy to see that the types CUSTOMER and Loan_Type are on level 0, because they do not have components. Level-1-types are INCOME, OBLIGATION, MORTGAGE and PERSONAL_LOAN, because all their components are on level 0. Consequently, the cluster Loan is a level-1-cluster. The types OWES, SECURITY and ACCOUNT are then level-2-types, because all components are on level 1 or below, and finally, ACCOUNT_RECORD is a level-3-type.

Figure 1 provides a graphical representation of the schema in Example 3.1. We call this a HERM diagram. According to the common convention in Entity-Relationship diagrams we represented types on level 0 by rectangles and types on higher levels by diamonds. Clusters are represented by \( \oplus \). We use directed edges from a relationship type to each of its components, and from clusters to their components, and undirected edges between types and their attributes. Roles names are attached to the directed edges. Keys are marked by underlining attributes and marking the edges that correspond to components in the key by some dot.

As each HERM schema can be represented by such a HERM diagram, i.e. by a graph, we can apply all graph-theoretic notions. In particular, when we talk of paths in a HERM schema, we mean a path in the underlying undirected graph that results from ignoring the orientation of edges in the HERM diagram.

In order to define the semantics of HERM schemata we concentrate on identifier-semantics, also known as pointer-semantics. For this assume a countable set \( ID \) of identifiers with \( ID \cap D = \emptyset \) for all domains \( D \) used for simple attributes. Furthermore, we associate with each type \( R \in S \) a representing attribute

\[ X_R = R(X_{r_1}, \ldots, X_{r_n}, A_1, \ldots, A_m) \]

with new simple attributes \( X_{r_i} \) for each role \( r_i \) and \( \text{dom}(X_{r_i}) = ID \), as well as a key attribute

\[ K_R = R(X_{r_1}, \ldots, X_{r_n}, A'_{1}, \ldots, A'_{m}) \]

for \( \text{key}(R) = \{ r_{i_1} : R_{i_1}, \ldots, r_{i_k} : R_{i_k}, A'_{i_1}, \ldots, A'_{i_k} \} \). Obviously, we have \( X_R \geq K_R \).

**Definition 3.4.** An instance of a HERM schema \((S, \Sigma)\) is a family \( \{ db(R) \}_{R \in S} \) of finite sets. For each type \( R \) the set \( db(R) \) consists of pairs \((i, v)\) with \( i \in ID \) and \( v \in \text{dom}(X_R) \) subject to the following conditions:

- Identifiers are globally unique, i.e. whenever \((i, v_1) \in db(R_1)\) and \((i, v_2) \in db(R_2)\) hold, we must have \( R_1 = R_2 \) and \( v_1 = v_2 \).
Key values are locally unique, i.e. whenever \((i_1, v_1), (i_2, v_2) \in db(R)\) hold with \(\pi_{X_{R_i}}(v_1) = \pi_{X_{R_i}}(v_2)\), then we must have \(i_1 = i_2\).

Roles are always defined, i.e. whenever \((i, v) \in db(R)\) and \(\pi_{X_{R_i}} R(X_{r_{i,j}})(v) = i'\) for \(r_{i,j} : R_{i,j} \in \text{comp}(R)\), then \((i', v') \in db(R_{i,j})\) for some \(v' \in \text{dom}(X_{R_{i,j}})\).

The integrity constraints in \(\Sigma\) are satisfied.

For each cluster \(C = R_1 \oplus \cdots \oplus R_k\) the set \(db(C)\) is the disjoint union of the sets \(db(R_i)\) \((i = 1, \ldots, k)\).

We write \(\text{inst}(S, \Sigma)\) for the set of all instances over \((S, \Sigma)\).

### 3.2 Query Languages for HERM

As for the relational data model, basic queries against a HERM schema can be formulated both in an algebraic and a logical way. We will extend both the simple HERM algebra and the HERM calculus in a way that we can express more queries, but let us start with first-order queries.

**Definition 3.5.** The HERM algebra \(\mathcal{H}\) provides the operations \(\sigma_\varphi\) (selection) with a selection formula \(\varphi\), \(\pi_{A_1,\ldots,A_m}\) (projection) with a generalised subset \(\{A_1, \ldots, A_m\}\), \(\rho_f\) (renaming) with a renaming function \(f\), \(\bowtie_G\) (join) with a common generalised subset \(G\), \(\cup\) (union), \(\setminus\) (difference), \(\nu_{X_{A_1},\ldots,A_n}\) with attributes \(A_1, \ldots, A_n\) (nest), and \(\mu_A\) (unnest) with a set attribute \(A\).

As the details of these operations are not much different from the relational data model, we omit the details and refer to [30].

However, in order to make the HERM algebra operational for our purposes here, we need a little extension:

- We permit new type names \(R\) to be added to \(S\), and use assignments \(R := \exp\) with a HERM algebra expression \(\exp\). Then applying such a query to an instance of \((S, \Sigma)\) results in an instance over \((S \cup \{R\}, \Sigma)\). The type or cluster definition for \(R\) is implicitly determined by the expression \(\exp\).

- In order to satisfy the global uniqueness of identifiers, such an assignment involves the non-deterministic creation of identifiers in \(ID\) for the pairs \((i, v)\) in the added \(db(R)\). Such identifier creation has been investigated thoroughly in [35].

- While sequences of assignments only extend the schema, we also allow to drop types or clusters.

In summary, a HERM algebra program \(P\) has the form \(C_1; \ldots; C_r \setminus S'\), where each \(C_i\) is an assignment, say \(R_i := \exp_i\) that extends the schema by \(R_i\) and the instance by \(db(R_i)\), while \(S'\) is a subschema of \(S \cup \{R_1, \ldots, R_r\}\). Thus, \(P\) defines a mapping \(q(P)\) taking instances over \((S, \Sigma)\) to instances over \((S', \Sigma')\), though the set \(\Sigma'\) of constraints is left implicit.
The algebra may be further extended with \textsc{while}, in which case we talk of the \textit{extended \textsc{herm} algebra} $\mathcal{H}_{\text{ext}}$. In this case we add constructs of the form

\textbf{while} change \textbf{do} $C_1$; $\ldots$; $C_r$ \textbf{end}.

We obtain a logical perspective from the following simple observation. Each type $R \in \mathcal{S}$ – or more precisely, its representing attribute $X_R$ – defines a variable in a higher-order logic. The order depends on the depth of nesting of the attributes in $\text{attr}(R)$. For instance, if the set constructor is not used, we obtain a first-order variable. In fact, we obtain a \textit{type} for each such variable, where types correspond to nested attributes. Thus, a schema defines a signature, and each instance defines a finite structure for this signature.

There are a few subtleties to be aware of. First, for clusters we have to allow particular union variables to cope with the requirement to have disjoint unions. Second, we have to define first the logic and then permit only those structures that are in fact models for the theory defined by the restrictions in Definition 3.4.

Now use further variables and constants of any available type and define atoms as follows:

- A \textit{predicative atom} has the form $X(t_1, \ldots, t_n)$ with a higher-order variable $X$ and terms, i.e. variables or constants, $t_1, \ldots, t_n$ such that the types of the $t_i$ and the one of $X$ match properly.
- An \textit{equational atom} has the form $t_1 = t_2$ with terms of the same type or of different types, where one is a cluster and the other one is one of its components.

Finally, use the usual connectives $\land$, $\lor$, $\to$, $\exists$ and $\forall$ to define \textit{\textsc{herm} logic}. Then the concept of free and bound variables is defined as usual. We write $\text{fr}(\varphi)$ for the set of free variables of a formula $\varphi$.

\textbf{Definition 3.6.} A \textit{\textsc{herm} calculus query} has the form $X(x_0, \ldots, x_n) : \varphi$ with a formula $\varphi$ of \textsc{herm} logic such that $\text{fr}(\varphi) \subseteq \mathcal{S} \cup \{x_1, \ldots, x_n\}$ and $\{x_1, \ldots, x_n\} \subseteq \text{fr}(\varphi)$.

Obviously, we may interpret a formula $\varphi$ provided we are given a value assignment $\sigma(x_i)$ for all the variables $x_1, \ldots, x_n$ and an instance over $(\mathcal{S}, \Sigma)$. If according to this interpretation the formula $\varphi$ is interpreted as true, we obtain a tuple $(\sigma(x_0), \ldots, \sigma(x_n))$ with a new identifier $\sigma(x_0) \in \text{ID}$. The result is the set of all such tuples, and will be bound to variable $X$.

It can be shown that such \textsc{herm} calculus queries express exactly the same as \textsc{herm} algebra queries with non-deterministic identifier creation and assignments to new type variables. Using sequences and a fixed-point construction yields the same expressiveness as the extended \textsc{herm} algebra. We use the term \textit{extended \textsc{herm} calculus} for this approach to query languages.

Let us finally extend the discussion on queries to \textit{computable queries} in the spirit of [5]. Computable queries are defined on a semantic rather than a syntactic level. A query language that can express all computable queries is called
complete, but this has to be verified. It is known that the calculus and algebra queries defined so far (even in their extended form) are not complete.

Roughly speaking, a computable query is a query that commutes with isomorphisms. So we have to define a concept of isomorphism for HERM instances. For this consider bijections \( \psi_A : \text{dom}(A) \rightarrow \text{dom}(A) \) for each simple attribute \( A \in \mathcal{A} \) and \( \psi_{\text{ID}} : \text{ID} \rightarrow \text{ID} \). We call this family \( \Psi = \{ \psi_A \}_{A \in \mathcal{A} \cup \{ \text{ID} \}} \) a t-isomorphism. Obviously, we may extend \( \Psi \) to all nested attributes defining

\[
X_{(A_1, \ldots, A_n)}(v_1, \ldots, v_n) = (\psi_{A_1}(v_1), \ldots, \psi_{A_n}(v_n))
\]

and

\[
X_{\{A\}}(\{v_1, \ldots, v_n\}) = \{\psi_A(v_1), \ldots, \psi_A(v_n)\}.
\]

In particular, a t-isomorphism \( \Psi \) induces a mapping \( \Psi_S : \text{inst}(\mathcal{S}, \Sigma) \rightarrow \text{inst}(\mathcal{S}, \Sigma) \).

**Definition 3.7.** Two instances \( db_1 \) and \( db_2 \) over \((\mathcal{S}, \Sigma)\) are called *isomorphic* (notation: \( db_1 \simeq db_2 \)) iff there exists a t-isomorphism \( \Psi \) with \( \Psi_S(db_1) = db_2 \).

A computable query on a database schema \((\mathcal{S}, \Sigma)\) with output schema \((\mathcal{S}', \Sigma')\) is a partial recursive function \( q : \text{inst}(\mathcal{S}, \Sigma) \rightarrow \text{inst}(\mathcal{S}', \Sigma') \) mapping isomorphic databases to isomorphic databases, i.e.,

\[
\text{if } db_1 \simeq db_2 \text{ and } q(db_1) \downarrow \Rightarrow q(db_2) \downarrow \text{ and } q(db_1) \simeq q(db_2),
\]

where \( q(db_1) \downarrow \) means that the partial recursive function \( q \) is defined on \( db_1 \).

### 3.3 Schema Dominance and Equivalence

As already stated schema and view integration requires precise notions for schema dominance and equivalence, which we will introduce now.

**Definition 3.8.** A HERM schema \((\mathcal{S}', \Sigma')\) *dominates* another HERM schema \((\mathcal{S}, \Sigma)\) by means of the language \( \mathcal{L} \) (notation: \( \mathcal{S}, \Sigma \sqsubseteq_\mathcal{L} (\mathcal{S}', \Sigma') \)) iff there are mappings \( f : \text{inst}(\mathcal{S}, \Sigma) \rightarrow \text{inst}(\mathcal{S}', \Sigma') \) and \( g : \text{inst}(\mathcal{S}', \Sigma') \rightarrow \text{inst}(\mathcal{S}, \Sigma) \) both expressed in \( \mathcal{L} \) such that the composition \( g \circ f \) is the identity.

If we have \( \mathcal{S}, \Sigma \sqsubseteq_\mathcal{L} (\mathcal{S}', \Sigma') \) as well as \( \mathcal{S}', \Sigma' \sqsubseteq_\mathcal{L} (\mathcal{S}, \Sigma) \), we say that the two schemata are equivalent with respect to \( \mathcal{L} \) (notation: \( \mathcal{S}, \Sigma \equiv_\mathcal{L} (\mathcal{S}', \Sigma') \)).

According to our discussion of HERM query languages in the previous subsection we obtain three different notions of dominance and equivalence. \( \sqsubseteq_\mathcal{H} \) and \( \equiv_\mathcal{H} \) refer to the use of the HERM algebra or equivalently the HERM calculus as the language, in which the transformations \( f \) and \( g \) are to be expressed. Analogously, \( \sqsubseteq_\mathcal{H_{ext}} \) and \( \equiv_\mathcal{H_{ext}} \) refer to the use of the extended HERM algebra or the extended HERM calculus. Finally, \( \sqsubseteq_{\text{comp}} \) and \( \equiv_{\text{comp}} \) refer to the use of computable queries. According to these choices we talk of HERM dominance, extended HERM dominance and computable dominance, respectively.
For completeness, let us mention a fourth alternative. If we do not impose any restrictions on the language $\mathcal{L}$, i.e. $f$ is any injective and $g$ any surjective mapping, then we obtain absolute dominance $\sqsubseteq_{abs}$ and absolute equivalence $\equiv_{abs}$. In the following sections we will always refer to $\sqsubseteq_{comp}$ and $\equiv_{comp}$ and therefore drop the index and simply write $\sqsubseteq$ for dominance and $\equiv$ for equivalence.

In the literature several other notions of schema dominance and equivalence have been introduced. Hull in [9] introduces four different notions on the basis of the relational data model. In the case of calculus dominance $\sqsubseteq_{calc}$ the mappings $f$ and $g$ must be defined by safe relational calculus or equivalently the relational algebra. In the case of generic dominance $\sqsubseteq_{gen}$ the mappings $f$ and $g$ must be generic in the sense that they commute with permutations of domain values fixing only a finite set $Z$ of values. In the case of internal dominance $\sqsubseteq_{int}$ the mappings $f$ and $g$ may only introduce a finite set of new values. Finally, Hull also defines absolute dominance.

All these notions can be generalised to HERM schemata. For instance, the work in [10] uses the same absolute dominance relation as [9] without referring explicitly to the relational model. In [9] it has been shown that calculus dominance implies generic dominance, which itself implies internal dominance, which implies absolute dominance. All these implications are strict. In [15] it has been shown that calculus dominance implies HERM dominance $\sqsubseteq_\text{H}$, which implies generic dominance. Of course, both in internal dominance and extended HERM dominance we have to deal with computable queries, so both imply computable dominance.

ADT dominance as defined by Qian [19] is based on order-sorted signatures and algebras. A schema transformation, i.e. our $f$, must be defined as a signature interpretation. It has been shown in [19] that calculus dominance implies ADT dominance, which implies absolute dominance. These implications are strict. Fur-
therrmore, ADT dominance is incomparable with the other notions of dominance as defined by Hull. In [15] it has been shown that ADT dominance and HERM dominance are also incomparable. However, the transformations defined by Qian define computable queries, so ADT dominance implies computable dominance.

Figure 2 illustrates the relationship between the various notions of schema dominance. We base our work on computable dominance, because absolute dominance is too general. It does not really preserve the semantics of the original schemata. On the other hand, extended HERM dominance, internal dominance and ADT dominance are pairwise incomparable, but all three notions make perfect sense in that the examples that can be expressed in them (but maybe not in the other formalisms) are reasonable. Therefore, computable dominance has been chosen, because it covers all reasonable notions of dominance without running into the problems arising from absolute dominance. However, computable dominance still leaves a lot of latitude for schema integration, and in most practical cases this latitude will not be exploited, i.e. a weaker notion of dominance would be sufficient.

4 Schema and View Integration and Cooperation in Databases

In this section we address first schema integration in databases following [16]. Without loss of generality we assume that we are given only two HERM schemata \( (S_1, \Sigma_1) \) and \( (S_2, \Sigma_2) \). Before starting the integration process we assume that these schemata have been “cleaned”, i.e. we assume that all name clashes have been removed by renaming homonymous attributes. Ideally, we may assume that names used in both schemata are different.

4.1 Schema and View Integration Process

We first describe a pragmatic method for schema integration, and then explain how this method applies to view integration. The details are then filled by transformation rules described in the following subsections.

1. The first step is the homogenisation of the schemata. This includes the restructuring of the schemata turning attributes into entity types, entity types into relationship types and vice versa. Furthermore, we add attributes and shift attributes along hierarchies and paths. All these individual paces correspond to the application of transformation rules. The result of the homogenisation step are schemata \( (S'_1, \Sigma'_1) \) and \( (S'_2, \Sigma'_2) \).

2. The second step consists in adding inter-schema integrity constraints that describe the semantic relationships between the two schemata. Formally, we obtain another set of constraints \( \Sigma_0 \), and thus the result of this step is a single HERM schema \( (S'_1 \cup S'_2, \Sigma'_1 \cup \Sigma'_2 \cup \Sigma_0) \).

3. The third step is only a preparation of the following steps. Due to the expected large size of the schemata, these are divided into modules, each of
which describes a subschema. Corresponding modules are identified in order to approach the integration of modules first. If schemata are of moderate size, this step can be omitted.

4. Step four considers the integration of types on level 0, 1, etc., i.e. we start with entity types and level-0-clusters, then proceed with relationship types and clusters on level 1, then relationship types and clusters on level 2, etc. For each level we integrate corresponding types or clusters with respect to equality, containment, overlap and disjointness conditions. Note that this step is similar to the work done in [13, 14, 29].

5. The fifth step deals with the integration of paths using path inclusion dependencies.

6. Finally, we consider remaining integrity constraints such as (path) functional dependencies and join dependencies.

Let us now see how this method deals with view integration. First recall that a view is nothing but a stored query.

**Definition 4.1.** A view $V$ on a HERM schema $(S, \Sigma)$ consists of a schema $S_V$ and a query $q_V$ with a query mapping $\text{inst}(S, \Sigma) \rightarrow \text{inst}(S_V)$.

**Example 4.1.** Let $S_1$ be the HERM schema from Example 3.1. Let us define a view $V_1 = \text{Customer}_{\text{Credibility}}$ on this schema, in which the target schema $S_{V_1}$ consists of a single entity-type

$$\text{Customer}_{\text{Cr}} = (\emptyset, \{\text{customer\_no}, \text{name}, \text{date\_of\_birth},$$

$$\quad \text{inc\_oblig}\{(\text{type}, \text{amount}, \text{frequency})\},$$

$$\quad \text{loans}\{(\text{type}, \text{amount}, \text{account\_balance})\}\},$$

$$\quad \{\text{customer\_no}\})$$

What we want to obtain is the set of all customers with their customer number, name and date of birth, plus their set of incomes and obligations (each described by their type, amount and frequency of payment), plus their set of loans (each described by their type, amount and corresponding account balance).

The defining query for this view is a bit tricky, as it has to involve the construction of two set-valued attributes. Following [25] we could employ a generalised nest-operation or use an IQI-like logic program, which would look as follows:

$$C(i_c, c, n, b, O, L) \leftarrow \text{Customer}(i_c, (c, n, a, b));$$

$$\hat{L}(\text{“mortgage”}, a, c) \leftarrow \text{Mortgage}(i_m, (i_{it}, m, a, d, p, b, c, o)),$$

$$\quad \text{Loan}(i_{it}, (\text{mortgage : } i_m)),$$

$$\quad \text{Account}(i_a, (i_t, n, c),$$

$$\quad \text{Owes}(i_o, (i_{i_c}, i_{i_t}, b', e'),$$

$$\quad C(i_c, cc, nc, bc, O, L).$$
\[ \hat{L} (\text{“personal”}, a, c) \leftarrow \text{PERSONAL} \text{LOAN}(i_{pl}, (itt, l, a, p, b, c, t)), \\
\text{LOAN}(i_t, (\text{personal} : i_{pl})), \\
\text{ACCOUNT}(i_a, (i_i, n, c)), \\
\text{OWES}(i_o, (i_c, i_t, b', e')), \\
C(i_c, cc, nc, be, O, L), \]
\[ \hat{O} (\text{“income”}, a, f) \leftarrow C(i_c, cc, nc, be, O, L), \\
\text{INCOME}(i_i, (i_c, c, a, t, f)). \]
\[ \hat{O} (\text{“obligation”}, a, f) \leftarrow C(i_c, cc, nc, be, O, L), \\
\text{OBLIGATION}(i_o, (i_c, c, a, t, f)). \]
\[ \text{CUSTOMER}_{\text{Cr}}(i, (c, n, b, \hat{O}, \hat{L})) \leftarrow C(i_c, c, n, b, O, L). \]

In a nutshell – without going into formal details – this logic program is a sequence of clause sets, separated by a semicolon. These clause sets are evaluated sequentially. Each clause set itself is evaluated by computing an inflationary fixed point, i.e. for each clause we bind the variables on the right hand side using the given instance of the database, and add the corresponding value for the left hand side to the database. The interesting point is that the variables that only occur on the left hand side will be bound to new identifiers, and for each such variable \( X \) we obtain a new predicate \( \hat{X} \) so that we can compute a set corresponding to each identifier.

In our special case here, we first construct a set \( C \) of customers with their identifier \( i_c \), customer number \( c \), name \( n \), date of birth \( b \) and identifiers \( O \) and \( L \) representing the set of obligations and loans, respectively. In the second clause set we construct the corresponding sets \( \hat{O} \) and \( \hat{L} \) of obligations and loans, respectively, for each customer in \( C \). In the final clause set we replace the identifiers \( O \) and \( L \) by the corresponding sets \( \hat{O} \) and \( \hat{L} \), respectively, and create a new identifier \( i \) for each element of the result.

So the view integration problem starts with two views \( V_1 \) and \( V_2 \) on the same HERM schema \( (S, \Sigma) \), and should result in a new integrated view \( V \) such that \( S_V \) results from integration of the schemata \( S_{V_1} \) and \( S_{V_2} \), and for each instance \( db \) over \( (S, \Sigma) \) the two query results \( q_{V_1}(db) \) and \( q_{V_2}(db) \) together are equivalent to \( q_V(db) \).

Now, if the schemata \( S_{V_1} \) and \( S_{V_2} \) are “cleaned”, we may combine the queries \( q_{V_1} \) and \( q_{V_2} \) into one yielding a query mapping \( \text{inst}(S, \Sigma) \rightarrow \text{inst}(S_{V_1} \cup S_{V_2}) \) defined by the query \( q_{V_1} \cup q_{V_2} \). If we simply integrate the schemata \( S_{V_1} \) and \( S_{V_2} \) into \( S_V \) according to the method described above, we obtain an induced mapping \( f : \text{inst}(S_{V_1} \cup S_{V_2}) \rightarrow \text{inst}(S_V) \). As we deal with computable queries, \( f \) is the query mapping of some computable query \( q_f \). Taking \( q_V = q_f \circ (q_{V_1} \cup q_{V_2}) \), \( V \) becomes a view over \( (S, \Sigma) \) with schema \( S_V \) and defining query \( q_V \).

This approach to view integration also works in the more general situation, where the given views \( V_1 \) and \( V_2 \) are defined over different schemata \( (S_1, \Sigma_2) \) and \( (S_2, \Sigma_2) \), respectively.
4.2 Transformation Rules

In the following subsections we describe transformation rules in detail. All rules will be presented in the same way, i.e. we assume a given HERM schema \((S, \Sigma)\), but we only indicate parts of it. The resulting schema will be \((S_{new}, \Sigma_{new})\). The new types in the new schema will be marked with a subscript \(new\). With these conventions the rules will be self-explanatory.

All the rules in this subsection subsume the rules that have been stated implicitly or explicitly in former work by others. In addition, the rules will establish computational equivalence in all cases, but formal proofs – which are not too hard – have been omitted. So what we can expect from our approach is soundness, but not completeness. The non-completeness results from the fact that computational dominance would allow us to define more general rules. It is very unlikely that there will exist a complete set of rules. On the other hand, having used the notion of computational dominance permits openness, i.e. the set of rules can be extended, if such a need arises.

Schema Restructuring. The first group of rules addresses the aspect of schema restructuring which will be used in the homogenisation step 1 of our method.

**Rule 1.** Replace a tuple attribute \(X(A_1, \ldots, A_m)\) in an entity or relationship type \(R\) by the attributes \(A_1, \ldots, A_m\). The resulting type \(R_{new}\) will replace \(R\). For \(X(A'_1, \ldots, A'_n) \in key(R)\) with \(A_i \leq A'_i\) we obtain \(A'_1, \ldots, A'_n \in key(R')\).

This rule includes the simple case, where \(R\) is an entity type, which could be treated as a separate rule.

**Rule 2.** Replace a component \(r : R'\) in a relationship type \(R\) by lower level components and attributes. Let the new type be \(R_{new}\). For \(\text{comp}(R') = \{r_1 : R_1, \ldots, r_n : R_n\}\) we get \(\text{comp}(R_{new}) = \text{comp}(R) \cup \{r : R'\} \cup \{r_1 : R_1, \ldots, r_n : R_n\}\) composed from \(r_i\) and \(r\) and \(\text{attr}(R_{new}) = \text{attr}(R) \cup \text{attr}(R')\). In the case \(r : R' \in \text{key}(R)\) and \(\text{key}(R') = \{r_i : R_i, \ldots, r_k : R_k, A_1, \ldots, A_m\}\) we obtain \(\text{key}(R_{new}) = \text{key}(R) \cup \{r : R'\} \cup \{r_1 : R_1, \ldots, r_n : R_n\} \cup \{r_i : R_i, \ldots, r_k : R_k, A_1, \ldots, A_m\}\), otherwise we have \(\text{key}(R_{new}) = \text{key}(R)\).

It is easy to see how to simplify this rule in the case, where \(R'\) is an entity type. Again, this could be formulated by two separate rules.

**Rule 3.** Replace a cluster \(C = C_1 \oplus \cdots \oplus C_n\) with a cluster component \(C_i = C_{i_1} \oplus \cdots \oplus C_{i_m}\) by a new cluster \(C = C_1 \oplus \cdots \oplus C_{i_1} \oplus \cdots \oplus C_{i_m} \oplus C_{i+1} \oplus \cdots \oplus C_n\).

**Rule 4.** Replace a relationship type \(R\) with a cluster component \(r : C\) \((C = C_1 \oplus \cdots \oplus C_n)\) by a new relationship \(R_{new} = R_{new_1} \oplus \cdots \oplus R_{new_{n,new}}\) and new relationship types \(R_{i,new}\) with \(\text{comp}(R_{i,new}) = \text{comp}(R) \cup \{r_i : C_i\}\) and \(\text{attr}(R_{i,new}) = \text{attr}(R)\). For \(r : C \in \text{key}(R)\) we obtain \(\text{key} = \text{key}(R) \cup \{r : C\} \cup \{r : C_i\}\), otherwise take \(\text{key} = \text{key}(R)\).
In the case of the restructuring rules 1 – 4 we can always show that the original schema and the resulting schema are equivalent. The next rule only guarantees that the resulting new schema dominates the old one.

Rule 5. Replace a key-based inclusion dependency $R'[\text{key}(R)] \subseteq R[\text{key}(R)]$ by new relationship types $R'_{\text{new}}$ with $\text{comp}(R'_{\text{new}}) = \{t' : R', r : R\} = \text{key}(R'_{\text{new}})$ and $\text{attr}(R'_{\text{new}}) = \emptyset$ together with participation cardinality constraints

$$\text{card}(R'_{\text{new}}, R) = (0, 1) \text{ and } \text{card}(R'_{\text{new}}, R') = (1, 1).$$

The last two restructuring rules allow to switch between attributes and entity types and between entity and relationship types. These rules 6 and 7 guarantee schema equivalence.

Rule 6. Replace an entity type $E$ with $A \in \text{attr}(E)$ by $E_{\text{new}}$ such that $\text{attr}(E_{\text{new}}) = \text{attr}(E) - \{A\}$ holds. Furthermore, introduce an entity type $E'_{\text{new}}$ with $\text{attr}(E'_{\text{new}}) = \{A\} = \text{key}(E'_{\text{new}})$ and a new relationship type $R_{\text{new}}$ with $\text{comp}(R_{\text{new}}) = \{r_{\text{new}} : E_{\text{new}}, r'_{\text{new}} : E'_{\text{new}}\} = \text{key}(R_{\text{new}})$ and $\text{attr}(R_{\text{new}}) = \emptyset$. Add the cardinality constraints $\text{card}(R_{\text{new}}, E_{\text{new}}) = (1, 1)$ and $\text{card}(R_{\text{new}}, E'_{\text{new}}) = (1, \infty)$.

Rule 7. Replace a relationship type $R$ with $\text{comp}(R) = \{r_1 : R_1, \ldots, r_n : R_n\}$ and the cardinality constraints $\text{card}(R, R_i) = (x_i, y_i)$ by a new entity type $E_{\text{new}}$ with $\text{attr}(E_{\text{new}}) = \text{attr}(R) = \text{key}(E_{\text{new}})$ and $n$ new relationship types $R_{i,\text{new}}$ with $\text{comp}(R_{i,\text{new}}) = \{r_i : R_{i}, r : E_{\text{new}}\} = \text{key}(R_{i,\text{new}})$ and $\text{attr}(R_{i,\text{new}}) = \emptyset$. Replace the cardinality constraints by $\text{card}(R_{i,\text{new}}, R_i) = (1, y_i)$ and $\text{card}(R_{i,\text{new}}, E_{\text{new}}) = (1, \infty)$.

In the case of rule 7 explicit knowledge of the key of $R$ allows to sharpen the cardinality constraints.

Shifting Attributes. The second group of rules deals with the shifting of attributes. This will also be used in the homogenisation step 1 of our method. Rule 8 allows to shift a synonymous attribute occurring in two subtypes, i.e. whenever tuples agree on the key they also agree on that attribute, to be shifted to a supertype. This rule leads to a dominating schema. Conversely, rule 9 allows to shift an attribute from a supertype to subtypes, in which case schema equivalence can be verified.

Rule 8. For $\text{comp}(R_i) = \{r_i : R\}$ and $A_i \in \text{attr}(R_i) - \text{key}(R_i)$ ($i = 1, 2$) together with the constraint $\forall t, t' \cdot t[\text{key}(R_1)] = t'[\text{key}(R_2)] \Rightarrow t[A_1] = t'[A_2]$ replace the types $R, R_1$ and $R_2$ such that $\text{attr}(R_{\text{new}}) = \text{attr}(R) \cup \{A_i\}$, $\text{comp}(R_{i,\text{new}}) = \{r_i : R_{\text{new}}\}$ and $\text{attr}(R_{i,\text{new}}) = \text{attr}(R_i) - \{A_i\}$ hold.

Rule 9. For $\text{comp}(R_i) = \{r_i : R\}$ ($i = 1, \ldots, n$) and $A \in \text{attr}(R) - \text{key}(R)$ together with the constraint $\forall t \in R. \exists t' \cdot t[r_i] = t' \Rightarrow t[A]$ replace the types such that $\text{attr}(R_{\text{new}}) = \text{attr}(R) - \{A\}$, $\text{comp}(R_{i,\text{new}}) = \{r_i : R_{\text{new}}\}$ and $\text{attr}(R_{i,\text{new}}) = \text{attr}(R_i) \cup \{A\}$ hold.
The next two rules 10 and 11 concern the reorganisation of paths and the shifting of attributes along paths. In both cases we obtain a dominating schema. Rule 10 could be split into two rules dealing separately with binary and unary relationship types $R_n$.

**Rule 10.** For a path $P \equiv R_1 \cdots R_n$ and a relationship type $R$ with $r_n : R_n \in \text{comp}(R)$ together with path cardinality constraints $\text{card}(P, R_1) \leq (1,1) \leq \text{card}(P, R_n)$ replace $R$ such that $\text{comp}(R_{\text{new}}) = \text{comp}(R) - \{r_n : R_n\} \cup \{r_{1,\text{new}} : R_1\}$ with a new role $r_{1,\text{new}}$ holds.

**Rule 11.** For a path $P \equiv R_1 \cdots R_n$ with $A \in \text{attr}(R_n)$ and path cardinality constraints $\text{card}(P, R_1) \leq (1,1) \leq \text{card}(P, R_n)$ replace $R_1, R_n$ such that $\text{attr}(R_{1,\text{new}}) = \text{attr}(R_1) \cup \{A\}$ and $\text{attr}(R_{n,\text{new}}) = \text{attr}(R_n) - \{A\}$ hold.

**Schema Extension.** The third group of rules deal with the schema extensions. This either concerns new attributes, new subtypes or the simplification of hierarchies. These rules are needed in step 1 of our method.

**Rule 12.** Add a new attribute $A$ to the type $R$, i.e. $\text{attr}(R_{\text{new}}) = \text{attr}(R) \cup \{A\}$. In addition, the new attribute may be used to extend the key, i.e. we may have $\text{key}(R_{\text{new}}) = \text{key}(R) \cup \{A\}$.

If the new attribute $A$ introduced by rule 12 does not become a key attribute, we obtain a dominating schema.

The next two rules allow to introduce a new subtype via selection or projection on non-key-attributes. In both cases we have schema equivalence.

**Rule 13.** For a type $R$ introduce a new relationship type $R'_{\text{new}}$ with $\text{comp}(R'_{\text{new}}) = \{r : R\} = \text{key}(R'_{\text{new}})$ and add a constraint $R'_{\text{new}} = \sigma_{\varphi}(R)$ for some selection formula $\varphi$.

**Rule 14.** For a type $R$ and attributes $A_1, \ldots, A_n \in \text{attr}(R)$ such that there are no $B_i \in \text{key}(R)$ with $A_i \geq B_i$ introduce a new relationship type $R'_{\text{new}}$ with $\text{comp}(R'_{\text{new}}) = \{r : R\} = \text{key}(R'_{\text{new}})$ and $\text{attr}(R'_{\text{new}}) = \{A_1, \ldots, A_n\}$, and add a constraint $R'_{\text{new}} = \pi_{A_1,\ldots,A_n}(R)$.

The last rule 15 in this group allows to simplify hierarchies. Here again we must exploit $\mathcal{H}_{\text{ext}}$ to obtain dominance.

**Rule 15.** Replace types $R, R_1, \ldots, R_n$ with $\text{comp}(R_i) = \{r_i : R\} = \text{key}(R_i)$ and $\text{card}(R, R_i) = (0,1)$ ($i = 1, \ldots, n$) by a new type $R_{\text{new}}$ with $\text{comp}(R_{\text{new}}) = \text{comp}(R)$, $\text{attr}(R_{\text{new}}) = \text{attr}(R) \cup \bigcup_{i=1}^{n} \text{attr}(R_i)$ and $\text{key}(R_{\text{new}}) = \text{key}(R)$. 
Type Integration. The fourth group of rules deals with the integration of types in step 4 of our method. Rule 16 considers the equality case, rule 17 considers the containment case, and rule 18 covers the overlap case. Note that these transformation rules cover the core of the approaches in [13, 29, 14].

Rule 16. If \( R_1 \) and \( R_2 \) are types with \( key(R_1) = key(R_2) \) and we have the constraint \( R_1[\{key(R_1) \cup X\}] = f(R_2[\{key(R_2) \cup Y\}] \) for some \( X \subseteq comp(R_1) \cup attr(R_1) \), \( Y \subseteq comp(R_2) \cup attr(R_2) \) and a bijective mapping \( f \), then replace these types by \( R_{\text{new}} \) with \( comp(R_{\text{new}}) = comp(R_1) \cup (comp(R_2) - Y - key(R_2)) \), \( attr(R_{\text{new}}) = attr(R_1) \cup (attr(R_2) - Y - key(R_2)) \cup \{D\} \) and \( key(R_{\text{new}}) = key(R_1) \cup \{D\} \) and an optional new distinguishing attribute \( D \).

Rule 17. If \( R_1 \) and \( R_2 \) are types with \( key(R_1) = key(R_2) \) and the constraint \( R_2[\{key(R_2) \cup Y\}] \subseteq f(R_1[\{key(R_1) \cup X\}] \) holds for some \( X \subseteq comp(R_1) \cup attr(R_1) \), \( Y \subseteq comp(R_2) \cup attr(R_2) \) and a bijective mapping \( f \), then replace \( R_1 \) by \( R_{1,\text{new}} \) with \( comp(R_{1,\text{new}}) = comp(R_1) \), \( attr(R_{\text{new}}) = attr(R_1) \cup \{D\} \) and \( key(R_{\text{new}}) = key(R_1) \cup \{D\} \) and an optional new distinguishing attribute \( D \). Furthermore, replace \( R_2 \) by \( R_{2,\text{new}} \) with \( comp(R_{2,\text{new}}) \{r_{\text{new}} : R_{1,\text{new}}\} \cup comp(R_2) - Y - key(R_2) \), \( attr(R_{2,\text{new}}) = attr(R_2) - Y - key(R_2) \) and \( key(R_{2,\text{new}}) = \{r_{\text{new}} : R_{1,\text{new}}\} \).

Rule 18. If \( R_1 \) and \( R_2 \) are types with \( key(R_1) = key(R_2) \) such that for \( X \subseteq comp(R_1) \cup attr(R_1) \), \( Y \subseteq comp(R_2) \cup attr(R_2) \) and a bijective mapping \( f \) the constraints

\[
R_2[\{key(R_2) \cup Y\}] \subseteq f(R_1[\{key(R_1) \cup X\}], \\
R_2[\{key(R_2) \cup Y\}] \supseteq f(R_1[\{key(R_1) \cup X\}] \quad \text{and} \\
R_2[\{key(R_2) \cup Y\}] \cap f(R_1[\{key(R_1) \cup X\}] = \emptyset
\]

are not satisfied, then replace \( R_1 \) by \( R_{1,\text{new}} \) with \( comp(R_{1,\text{new}}) \{r_{1,\text{new}} : R_{\text{new}}\} \cup comp(R_1) - X - key(R_1) \), \( attr(R_{1,\text{new}}) = attr(R_1) - X - key(R_1) \) and \( key(R_{1,\text{new}}) = \{r_{1,\text{new}} : R_{\text{new}}\} \), replace \( R_2 \) by \( R_{2,\text{new}} \) with \( comp(R_{2,\text{new}}) \{r_{\text{new}} : R_{1,\text{new}}\} \cup comp(R_2) - Y - key(R_2) \), \( attr(R_{2,\text{new}}) = attr(R_2) - Y - key(R_2) \) and \( key(R_{2,\text{new}}) = \{r_{\text{new}} : R_{1,\text{new}}\} \) and introduce a new type \( R_{\text{new}} \) with \( comp(R_{\text{new}}) = comp(R_1) \cup comp(R_2), attr(R_{\text{new}}) = attr(R_1) \cup attr(R_2) \cup \{D\} \) and \( key(R_{\text{new}}) = key(R_1) \cup \{D\} \) and an optional new distinguishing attribute \( D \).

The rules 16-18 could each be split into several rules depending on \( f \) being the identity or not and the necessity to introduce \( D \) or not. In all cases we obtain dominance.

Rule 19 considers the case of a selection condition, in which case schema equivalence holds.

Rule 19. If \( R \) and \( R' \) are types with \( comp(R') \cup attr(R') = Z \subseteq comp(R) \cup attr(R) \) such that the constraint \( R' = \sigma_{\varphi}(\pi_Z(R)) \) holds for some selection condition \( \varphi \), then omit \( R' \).
Handling Integrity Constraints. The fifth group of rules to be applied in step 5 of our method concerns transformations originating from path inclusion constraints. Rule 20 allows us to change a relationship type. This rule leads to equivalent schemata. Rule 21 allows to introduce a relationship type and a join dependency. Finally, rule 22 handles a condition under which a relationship type may be omitted. Both rules 21 and 22 guarantee dominance.

Rule 20. If there are paths \( P \equiv R_1 - R - R_2 \) and \( P' \equiv R_2 - R' - R_3 \) with \( \text{comp}(R) = \{r_1 : R_1, r_2 : R_2\} \) and \( \text{comp}(R') = \{r_3 : R_3, r'_2 : R_2\} \) such that the constraint \( P[R_2] \subseteq P'[R_2] \) holds, then replace \( R \) in such a way that \( \text{comp}(R_{\text{new}}) = \{r_1 : R_1, r_{\text{new}} : R'\} \), \( \text{attr}(R_{\text{new}}) = \text{attr}(R) \) and \( \text{key}(R_{\text{new}}) = \text{key}(R) - \{r_2 : R_2\} \cup \{r_{\text{new}} : R'\} \) hold.

Rule 21. If there are paths \( P \equiv R_1 - R - R_2 \) and \( P' \equiv R_2 - R' - R_3 \) with \( \text{comp}(R) = \{r_1 : R_1, r_2 : R_2\} \) and \( \text{comp}(R') = \{r_3 : R_3, r'_2 : R_2\} \) such that the constraint \( P[R_2] = P'[R_2] \) holds, then replace \( R \) and \( R' \) by \( R_{\text{new}} \) such that \( \text{comp}(R_{\text{new}}) = \{r_1 : R_1, r_{\text{new}} : R_2, r_3 : R_3\} \), \( \text{attr}(R_{\text{new}}) = \text{attr}(R) \cup \text{attr}(R') \) and \( \text{key}(R_{\text{new}}) = (\text{key}(R) - \{r_2 : R_2\}) \cup \text{key}(R') - \{r'_2 : R_2\} \) hold. Add the join dependency \( R_{\text{new}}[r_1, r_{\text{new}}, r_3] \approx R_{\text{new}}[r_2, r_{\text{new}}, r_3] \subseteq R_{\text{new}}[r_1, r_{\text{new}}, r_3] \).

Rule 22. If there are paths \( P \equiv R_1 - R_2 - \cdots - R_n \) and \( P' \equiv R_1 - R - R_n \) with \( \text{comp}(R) = \{r_1 : R_1, r_n : R_n\} \) such that the constraint \( P[R_1, R_n] = P'[R_1, R_n] \) holds, then omit \( R \).

The final group of transformation rules 23-26 permits to handle remaining constraints such as functional dependencies, path functional dependencies, and join dependencies. All these constraints are described in detail in [30]. The rules refer to step 6 of our method.

Rule 23 handles vertical decomposition in the presence of a functional dependency. Rule 24 allows to simplify a key in the presence of a path functional dependency. Rule 25 introduces a new entity type in the presence of a path functional dependency. Finally, rule 26 replaces a multi-ary relationship type by binary relationship types in the presence of a join dependency. The four rules lead to dominating schemata.

Rule 23. If a functional dependency \( X \rightarrow A \) with a generalised subset \( X \) of \( \text{attr}(E) \) and an attribute \( A \in \text{attr}(E) - X \) holds on an entity type \( E \), but \( X \rightarrow \text{key}(E) \) does not hold, then remove \( A \) from \( \text{attr}(E) \) and add a new entity type \( E'_{\text{new}} \) with \( \text{attr}(E'_{\text{new}}) = X \cup \{A\} \) and \( \text{key}(E'_{\text{new}}) = X \).

Rule 24. For a path \( P \equiv R_1 - R - R_2 \) with \( \text{comp}(R) = \{r_1 : R_1, r_2 : R_2\} \) such that the path functional dependency \( X \rightarrow \text{key}(R_2) \) holds for a generalised subset \( X \) of \( \text{attr}(R_1) \) replace \( \text{key}(R) \) by \( \{r_1 : R_1\} \).

Rule 25. For a path \( P \equiv R_1 - \cdots - R_n \) such that the path functional dependency \( X \rightarrow A \) holds for a generalised subset \( X \) of \( \text{attr}(R_1) \) and \( A \in \text{attr}(R_n) \) add a new entity type \( E_{\text{new}} \) with \( \text{attr}(E_{\text{new}}) = X \cup \{A\} \) and \( \text{key}(E_{\text{new}}) = X \).
Rule 26. If $R$ is an $n$-ary relationship type with $\text{comp}(R) = \{r_1 : R_1, \ldots, r_n : R_n\}$ and $\text{attr}(R) = \emptyset$ such that the join dependency $R[r_1, r_2] \bowtie \cdots \bowtie R[r_1, r_n] \subseteq R[r_1, \ldots, r_n]$ holds, then replace $R$ by $n$ new relationship types $R_{1,\text{new}}, \ldots, R_{n,\text{new}}$ with $\text{comp}(R_{i,\text{new}}) = \{r_1 : R_1, r_i : R_i\} = \text{key}(R_{i,\text{new}})$ and $\text{attr}(R_{i,\text{new}}) = \emptyset$.

4.3 View Cooperation

View cooperation [30] provides an alternative to view integration in which the integrated view is only virtual. That is the constituting views are kept and exchange functions are designed to provide the same functionality as if the views were integrated.

Definition 4.2. Let $V_i = (S_{V_i}, q_{V_i}) (i = 1, 2)$ be views (on the same or different HERM schemata). $V_1$ cooperates with $V_2$ iif there are subschemata $S'_{V_i}$ of $S_{V_i}$ and functions $f_1 : \text{inst}(S'_{V_1}) \rightarrow \text{inst}(S'_{V_2})$ and $f_2 : \text{inst}(S'_{V_2}) \rightarrow \text{inst}(S'_{V_1})$, such that both $f_1 \circ f_2$ and $f_2 \circ f_1$ are the identity function.

Basically, view cooperation expresses that part of view $V_1$, exactly the one corresponding to the subschema $S'_{V_1}$, can be expressed by the part of view $V_2$ corresponding to the subschema $S'_{V_2}$.

Now, if we want to obtain a cooperation between given views $V_1$ and $V_2$, we may simply apply our view integration method to them using the same transformation rules. This will result in an integrated view $V = (S_V, q_V)$. With respect to this integrated view both $S'_{V_1}$ and $S'_{V_2}$ will be identified with a subschema $S'_{V_i}$. In particular we obtain functions $f'_1 : \text{inst}(S'_{V_1}) \rightarrow \text{inst}(S'_{V_2})$ and $g'_i : \text{inst}(S'_{V_i}) \rightarrow \text{inst}(S'_{V_i}) (i = 1, 2)$ with $g'_i \circ f'_i = \text{id}$ and $f'_i \circ g'_i = \text{id}$. Thus, $f_1 = g'_2 \circ f'_1$ and $f_2 = g'_1 \circ f'_2$ define the view cooperation functions.

4.4 Case Study

In order to demonstrate our approach to view integration and cooperation we first use a second HERM schema $S_2$, which is given by the HERM diagram in Figure 3. We may think of this schema being used by a mortgage broker who deals with mortgages from different banks. In addition, the assessment of the credibility of a customer may depend on securities owned by supporting relatives, so we add information about relatives to the schema. We omit the formal definition for this HERM schema.

Example 4.2. Let us first address the integration of the HERM schemata $S_1$ and $S_2$ from Figures 1 and 3. In these schemata we mainly have to deal with the homogenisation of types in the two schemata, i.e. with type restructuring, and with type integration.

First we homogenise $\text{Customer}$ appearing in both schemata adding attributes (rule 12) and turning the attributes name and address into tuple attributes with components first name and last name, and city and street address, respectively – which means to apply rule 1. We obtain the integrated type
Customer = (\emptyset, \{ customer\_no, name(first\_name, last\_name), address(city, street\_address), phone, date\_of\_birth \}, \{ customer\_no \})

Next we split Finance\_Transactions in \( S_2 \) into Income and Obligation according to the value of the attribute plus/minus. This results from applying rule 13. Furthermore, we can rename role names and turn the attribute account for both Income and Obligation into a component on\_account: Account.

The homogenisation of Mortgage\_Type in \( S_1 \) and Loan\_Type in \( S_2 \) requires not much more than the renaming of the attribute type to identification. The homogenisation of Mortgage in both schemata requires first to introduce in \( S_2 \) a new type Payment, which takes the components who: Customer and payments: Account from Mortgage, and add a component for: Mortgage. Similarly, we can split Security in \( S_1 \) into Available\_Security, for which we can drop the component for: Mortgage, and Security. In \( S_1 \) we can rename Owes to Payment and add a component payments: Account. Finally, we replace Mortgage in Payment by the cluster Loan.

The HERM diagram of the final resulting schema is shown in Figure 4.

The work in [16] contains further examples for schema integration.

The integration of schemata turns each view over one of the source schemata into a view over the integrated schema. What is changed is the defining query.

**Example 4.3.** Consider the view \( V_1 \) from Example 4.1. If schema \( S_1 \) is integrated with schema \( S_2 \) into schema \( S_{12} \) in Figure 4, the defining query for \( V_1 \) changes
as follows:

\[
C(i_c, c, n, b, O, L) \leftarrow \text{CUSTOMER}(i_c, (c, n, a, b, ph));
\]

\[
\hat{L}(\text{"mortgage"}, a, c) \leftarrow \text{MORTGAGE}(i_m, (i_t, m, a, d, p, b, c, o)),
\]

\[
\text{LOAN}(i_l, (\text{mortgage} : i_m)),
\]

\[
\text{PAYMENT}(i_p, (i_a, i_l, i_c)),
\]

\[
C(i_c, cc, nc, bc, O, L).
\]

\[
\hat{L}(\text{"personal"}, a, c) \leftarrow \text{PERSONAL\_LOAN}(i_p, (i_t, a, p, b, c, t)),
\]

\[
\text{LOAN}(i_l, (\text{personal} : i_p)),
\]

\[
\text{PAYMENT}(i_p, (i_a, i_l, i_c)),
\]

\[
C(i_c, cc, nc, bc, O, L).
\]

\[
\hat{O}(\text{"income"}, a, f) \leftarrow C(i_c, cc, nc, bc, O, L),
\]

\[
\text{INCOME}(i_c, (i_c, a, t, t', a, f)).
\]

\[
\hat{O}(\text{"obligation"}, a, f) \leftarrow C(i_c, cc, nc, bc, O, L),
\]

\[
\text{OBLIGATION}(i_c, (i_c, a, t, t', a, f));
\]

\[
\text{CUSTOMER\_Cr}(i, (c, n, h, O, L)) \leftarrow C(i_c, c, n, b, O, L).
\]
Example 4.4. Let us now address the integration of views. For this consider another view $V_2$ defined on the database schema $S_2$. Let the target schema $S_{V_2}$ consist again of a single entity-type

$$\text{CUSTOMER}_{\text{CR}} = (\emptyset, \{\text{customer\_no}, \text{name(first\_name, last\_name)}, \text{date\_of\_birth},$$

$$\text{inc\_oblig}((\text{type, amount, frequency})), \text{securities}((\text{type, object, value, self})),$$

$$\emptyset \})$$

Analogous to Example 4.1 we want to obtain the set of all customers with their customer number, name and date of birth, plus their set of incomes and obligations (each described by their type, amount and frequency of payment), plus their set of securities (each described by their type, object, value and whether it is a security owned by the customer or a supporting relative). The defining query will be built analogously to the one in Example 4.1 – we omit the details.

We also omit how this defining query will be translated into a query on the integrated schema $S_{12}$.

Then we basically integrate the target schemata of $V_1$ and $V_2$, which results in a schema $S_{V_{12}}$ with a single entity-type

$$\text{CUSTOMER}_{\text{CR}} = (\emptyset, \{\text{customer\_no}, \text{name(first\_name, last\_name)}, \text{date\_of\_birth},$$

$$\text{inc\_oblig}((\text{type, amount, frequency})), \text{securities}((\text{type, object, value, self})),$$

$$\text{loans}((\text{type, amount, account\_balance})), \emptyset \})$$

The defining query on $S_{12}$ is built analogously to the ones for $V_1$ and $V_2$. We just obtain a third set-valued attribute that has to be constructed by some fixed-point computation.

Let us finally look at view cooperation, i.e. we do not integrate the views $V_1$ and $V_2$ on schema $S_{12}$, but only determine view cooperation functions.

Example 4.5. The target schemata of the views $V_1$ and $V_2$ from Examples 4.1 and 4.4 share an equivalent subschema. Both $S_{V_1}$ and $S_{V_2}$ consist of a single entity-type

$$\text{CUSTOMER}_{\text{CR}} = (\emptyset, \{\text{customer\_no}, \text{name}, \text{date\_of\_birth},$$

$$\text{inc\_oblig}((\text{type, amount, frequency})), \emptyset \})$$

The only difference is that in $V_1$ the attribute name is a simple attribute, while in $V_2$ it is a tuple attribute name(first\_name, last\_name). The corresponding equivalent subschema of $S_{V_{12}}$ also contains this tuple attribute. Therefore, the cooperation functions $f_1 : \text{inst}(S_{V_1}) \rightarrow \text{inst}(S'_{V_1})$ and $f_2 : \text{inst}(S_{V_2}) \rightarrow \text{inst}(S'_{V_2})$ only affect the name attribute. That is, $f_2$ will concatenate the first\_name and the last\_name, while $f_1$ will decompose name according to some algorithm. For instance, if we assume that spaces are not allowed inside last names, we could simply search for the last space in a name string and take the substring preceding it for first\_name, and the string following the space for last\_name.
5 View Integration in Data Warehouses

In this section we apply our view integration method to data warehouses. For this we exploit the general idea of data warehouses, i.e. the separation of input from operational databases and output to data marts. This idea is illustrated using a three-tier architecture in Figure 5. In particular, we take up the idea from [17] to model OLAP applications by dialogue types [21].

The underlying idea is simply that the content presented to a user can be described by a value of a complex data type. In addition a user is offered operations – the OLAP operations in the case of data warehouses – that work on the presented value. This defines a dialogue object, and dialogue types arise from the usual abstraction. In addition, the values presented to the users depend on some database, so the dialogue types give rise to views that are extended by operations. We briefly illustrate these concepts, then extend our view integration method to this case.

![General Data Warehouse Architecture](image)

The fact that view integration arises within the area of data warehousing arises from using the dialogue types approach to model the data marts for the OLAP applications. In developing an OLAP application we would first collect the user needs including ways how to work with a system – see [21] for more details. This gives a collection of views, while the warehouse itself is the result of integrating these views. The problems of deciding on materialised views that
support the data marts efficiently and choosing view cooperation as an alternative are orthogonal to this integration problem.

5.1 Data Warehouse Architecture

On the bottom tier we have operational databases set up for purposes that are of no particular interest for the data warehouse. However, we assume that the data stored in the data warehouse is extracted from these operational databases. In other words, the input of data into the data warehouse is defined by update- or refresh operations, which take data from these operational databases and insert them into the data warehouse. In particular, these refresh-operations can be formulated by queries on the operational databases. Assuming that the HERM has been used for the operational databases, we can express the refresh-operations in (extended) HERM algebra or calculus. It is not very likely that the structure of a data warehouse will require more complicated (computable) queries.

In general, if there is some conflict between these operational data, i.e. data from different operational databases has to be integrated before it can be inserted into the data warehouse, we need an integrator component [36]. However, we may assume that this integrator is part of the extraction functions.

The second tier of the architecture in Figure 5 is made up by the data warehouse itself. In principle, we just have a database system here with the only differences that we may assume a simpler schema, i.e. star or snowflake schema, and we do not have to consider complex transactions. The only write-operations are the refresh operations that connect the data warehouse to the operational databases. All other operations only read data from the data warehouse. In fact, we just build views for the “data marts” or dialogue objects that are used as the OLAP interface. So, the view construction operations are the only ones that link the middle tier to the top tier, which deals with OLAP. So again, the major functionality is expressed by views.

The top tier itself is constructed out of the dialogue objects and the OLAP operations working on them. This tier realises the idea of using dialogue objects for this purpose. The general idea from [21] is that each user has a collection of open dialogue objects, i.e. data marts. At any time we may get new users, and each user may create new dialogue objects without closing the existing ones. Thus, we maintain the lists of users and their data marts in the system. Part of the functionality of the OLAP tier deals with adding and removing users and data marts. In particular, if a user leaves the system, all data marts owned by him/her must be removed as well.

The major functionality, however, deals with running operations on existing data marts or creating new data marts. In the latter case we have to use a view creation query as described for the middle tier. In this case we choose a new identifier for this data mart / dialogue object, and initialise its data content. If the user selects some of the data of the data mart and an operation other than quit (i.e. the user does not want to leave the system), close (i.e. the user does not want to finish work on the current data mart) or open (i.e. no new data mart is
to be created), then we request to receive additional input from the user, before the selected operation will be executed.

5.2 Dialogue Types

We have seen that the major functionality on the OLAP tier is expressed by views that are extended by operations, which is exactly the idea underlying dialogue types. However, we simplify the original definition from [21] omitting the subtle distinction between hidden and visible parts. We also omit hierarchies of dialogue types.

**Definition 5.1.** A dialogue type $D$ over a HERM schema $(S, \Sigma)$ consists of a view $V_D = (S_D, q_D)$, in which $S_D$ consists of a single entity type $E$, and a set $O$ of dialogue operations. Each dialogue operation (d-operation for short) in $O$ consists of

- an operation name $op$,
- a list of input parameters $i_1: D_1, \ldots, i_k: D_k$ with domain names $D_i$,
- an (optional) output domain $D_{out}$,
- a subattribute $sel$ of the representing attribute $X_E$, and
- a d-operation body, which is built from usual programming constructs operating on instances over $(S, \Sigma)$ and constructs for creating and deleting dialogue objects.

Whenever we are given an instance $db$ over $(S, \Sigma)$, the defining query $q_D$ produces an instance over $S_D$, i.e. a set of pairs $(i, v)$ with $i \in ID$ and $v \in \text{dom}(X_E)$, each of which will be called a dialogue object of type $D$. At any time only a subset of $q_D(db)$ will be available, the set of active dialogue objects. These represent the active user dialogues in an abstract way.

The presented value $v$ may be projected to $\pi^{X_E}_{sel}(v)$, which represents the data that must be selected by the user as a prerequisite for executing operation $op$. Once these data are selected and the operation $op$ is started, further input for the parameters $i_1, \ldots, i_k$ will be requested from the user – using e.g. so-called dialogue boxes [21] – and the execution of $op$ will update the database $db$ and result in a new set of active dialogue objects.

5.3 Transformation Rules for Dialogue Types

As user dialogues are an invaluable source of information in requirements engineering, we may usually assume that we know about dialogue objects before the defining queries and the underlying database schema is fixed. Therefore, view integration is an unavoidable design task. Integrating the views that underly the dialogue types leads to a design of the data warehouse. In addition, we will always be confronted with the desire to rearrange the “data marts”, i.e. the dialogue types that define the OLAP functionality. This problem also appears in database applications other than OLAP.
Therefore, the additional problem is to adapt the d-operations that are used for the functionality presented to a database or data warehouse user. For this we define further transformation rules. However, these additional transformation rules have to be understood as follow-on rules for the case that one of the rules 1-26 is not just applied to schemata or views, but to dialogue types. Thus, we obtain additional changes to the selection attribute \( sel \) and the body of d-operations.

**Rule 27.** In case rule 1 is applied to a dialogue type, whenever \( X(A'_1, \ldots, A'_k) \) \((k \leq m)\) appears in \( sel \) or in the body of a d-operation, replace it by \( A'_1, \ldots, A'_k \).

**Rule 28.** In case rule 2 is applied to a dialogue type, whenever \( r \) appears in \( sel \) or in the body of a d-operation, replace it by \( r_1^{(r)}, \ldots, r_n^{(r)} \).

We may ignore rules 3 and 4, as clusters have not been allowed in dialogue types. There is also nothing to add for rule 5, as this introduces a new type, so operations have to be defined for that type.

**Rule 29.** In case rule 6 is applied to a dialogue type omit \( A \) in \( sel \) or in the body of a d-operation, whenever it appears.

**Rule 30.** In case rule 6 is applied to a dialogue type omit \( r_1, \ldots, r_n \) in \( sel \) or in the body of a d-operation, whenever it appears.

These extensions capture the first group of rules dealing with schema restructuring. For the second group of rules dealing with the shifting of attributes we obtain the following extension rules in case the rules are applied to dialogue types.

**Rule 31.** In case rule 8 is applied to a dialogue type omit \( A_i \) in \( sel \) and the body of operations associated with \( R_i;new \), whenever it appears in \( sel \) or the body of an operation associated with \( R_i \).

**Rule 32.** In case rule 9 is applied to a dialogue type omit \( A_i \) in \( sel \) and the body of operations associated with \( R_{new} \), whenever it appears in \( sel \) or the body of an operation associated with \( R \).

Note that the last two extension rules have no effect on \( R_{new} \) or \( R_{i;new} \), as the extension of the selection attribute or the body of an operation has to be defined for these new types.

**Rule 33.** In case rule 10 is applied to a dialogue type replace \( r_n \) by \( r_{1,new} \) in \( sel \) and the body of operations associated with \( R_{new} \).

**Rule 34.** In case rule 11 is applied to a dialogue type omit \( A \) in \( sel \) and the body of operations associated with \( R_{n,new} \).

For the third group of rules, i.e., rules 12-15 dealing with schema extension we cannot define reasonable extension rules for dialogue types, as we always have to deal with completely new types. The same applies to rules 16-19, i.e. the group of rules dealing with type integration.

Finally, for the group of rules dealing with integrity constraints only rules 20, 21 and 23 give rise to the following three extension rules for dialogue types.
Rule 35. In case rule 20 is applied to a dialogue type replace \( r_2 \) by \( r_{\text{new}} \) in \( sel \) and the body of operations, whenever it appears.

Rule 36. In case rule 21 is applied to a dialogue type replace \( r_2 \) by \( r_{2,\text{new}} \) in \( sel \) and the body of operations, whenever it appears.

Rule 37. In case rule 23 is applied to a dialogue type remove \( A \) in \( sel \) and the body of operations, whenever it appears.

5.4 Case Study

Let us continue the case study from the previous section. What we have to do is to study how operations associated with a view will be affected by the view integration process.

Example 5.1. Consider the view \( V_1 \) on schema \( S_1 \) from Example 4.1. Let us associate a d-operation \( \text{show}_\text{security} \) with this view turning it into a dialogue type. The idea behind this operation is that a user who gets a presentation of a customer together with his/her obligations and loans, checks the securities for a mortgage in the list. So the user will select one of the loans, a mortgage, then choose the operation \( \text{show}_\text{security} \), and receive a presentation of a security list, i.e. a new dialogue object will be opened. That is, the d-operation is defined as

\[
\text{show}_\text{security}[\text{mortgage\_no}](\cdot) = \text{d-open} (\text{Security\_List}[\text{mortgage\_no}])
\]

This is a very simple d-operation, which only opens a d-object that is specified by another dialogue type \( \text{Security\_List} \) and the value for the key attribute \( \text{mortgage\_no} \). We omit the definition of this dialogue type.

The integration of schemata \( S_1 \) and \( S_2 \) and the integration of views \( V_1 \) and \( V_2 \) has only a marginal effect on this d-operation, which becomes a d-operation for the integrated view. As we only open a d-object, the change is already reflected in adapting the defining query for the underlying view of the dialogue type \( \text{Security\_List} \).

Let us consider another d-operation \( \text{loan}_\text{update} \) with selection attribute \( sel = \text{loan\_no} \), i.e. we request again that a loan, this time a personal loan, be selected from the list of loans of a customer. For this operation we also request additional input from a user, which is specified by the parameters \( \text{amount} \) and \( \text{interest} \), both with domain \( \text{Decimal} \), and terms with domain \( \text{String} \). So we may specify

\[
\text{loan}_\text{update}[\text{loan\_no}](\text{amount}:\text{Decimal},\text{interest}:\text{Decimal},\text{terms}:\text{String}) =
\begin{align*}
\text{let } \& (i, (t', l, a, p, b, e, t)) = \mathbf{L}x \in \text{PERSONAL\_Loan}. \\
& x.\text{loan\_no} = \text{loan\_no}
\end{align*}
\]

in \( \text{PERSONAL\_Loan} : \& (i, (t', l, \text{amount}, \text{interest}, b, e, \text{terms})) \)

That is, we select the data about the loan with the selected loan number from the database, which is expressed by “the unique \( x \ldots \)”, written as \( \mathbf{L}x \in \text{PERSONAL\_Loan} \ldots \), then update \( (; \&) \) this tuple in the database using the
input provided by the user, i.e. the values of the parameters amount, interest and terms.

In adapting this d-operation to the integrated view, we have to replace the specification of values in PERSONAL_LOAN according to the changes to the integrated schema.

6 View Integration in Web Information Systems

A web information system (WIS) is a database-backed information system that is realized and distributed over the web with user access via web browsers. Information is made available via pages including a navigation structure between them and to sites outside the system. Furthermore, there should also be operations to retrieve data from the system or to update the underlying database(s).

The methodology for WIS design in [22, 25] emphasises abstraction layers and the co-design of structure, operations and interfaces. As WISs are open systems in the sense that everyone who has access to the web may turn up as a user, their design requires a clear picture of the intended users and their behaviour. This includes knowledge about the used access channels and end-devices. At a high level of abstraction this first leads to storyboarding, an activity that addresses the design of underlying application stories. Storyboarding first describes a story space by scenes and actions on these scenes. Furthermore, it describes the actors in these scenes, i.e. groups of users of the WIS. Actor modelling leads to roles, profiles, goals, preferences, obligations and rights. Finally, the actors are linked to the story space by means of tasks.

Further on in the development process the scenes in the story space have to be adequately supported. For this the methodology focuses on media types, which cover extended views, adaptivity and hierarchies. In a nutshell, a media type is an extended view on some underlying database schema. These views are built in a way that they capture the complex content and navigation structure that is to be presented to a user. In order to capture the navigation structure we use abstract identifiers in the views, both for having a unique handle for each object in the result and for being able to reference this object. So the defining queries must be powerful enough to create these identifiers. As a consequence, views are no longer independent from each other in the sense that creating one view may require to create another one simultaneously. This is a well known problem from querying object oriented databases, for which the solution by IQL [1] can be adopted. An alternative would be to use the extended query algebra from [25].

The view integration (or cooperation) problem for WISs arises in the same way as the one for data warehouses. Storyboarding results among others in a collection of elementary scenes, each of which has to be supported by a media type, i.e. an extended view. Defining these views gives a collection of views, while the actual database support requires the integration of these views.

Adaptivity to users, channels and end-devices mainly concerns the question, whether all information or only the most important part of it is to be presented
to a user. By specifying on a conceptual level what these “most important” parts are and which parts have to be kept together we may then leave the technical realisation of adaptivity to an algorithmic solution. Hierarchies enable the presentation of information at different levels of granularity allowing a user to switch between these levels. Such hierarchies are common in OLAP systems and the principles can be borrowed from there. We will not deal with hierarchies in this article.

Similar to OLAP systems it is likely that the content that is to be made available to users is modelled, before an underlying database schema and thus defining queries for media types have been defined. This leads unavoidably to a view integration problem. In addition, we have to cope with the implications for operations and for adaptivity – postponing the hierarchies to later. Operations can be dealt with in the same way as for data warehouses and OLAP systems.

The transformation rules in Section 4 already capture the core of the integration process, the integration of the views. So we only have to look at the extensions that arise from media types. Operations have already been dealt with in Section 5, so we are left with the adaptivity, i.e. we just explain in this section how the rules have to be extended to be applicable to media types. Thus, we may concentrate on the aspect of adaptivity.

6.1 Extended Views for Web Information Systems

At the core of a media type we have the same idea as for dialogue types with some subtle distinctions. The idea is the same as for the dialogue types: The content of a web page can be described by a complex value of some type, which may contain references to other values. These references abstract from links between pages. Then abstract to types and add operations. This leads first to the notion of interaction type as defined (in a simplified form) next.

**Definition 6.1.** An interaction type $I$ over a HERM schema $(\mathcal{S}, \Sigma)$ consists of a view $V_I = (\mathcal{S}_I, q_I)$, and a set $O$ of dialogue operations.

The difference between interaction and dialogue types is that the view schema $\mathcal{S}_I$ of an interaction type may be much more complicated. In particular, we permit references (or roles) between the interaction objects of any type in $\mathcal{S}_I$ that result from applying the defining query $q_I$ to an instance over $(\mathcal{S}, \Sigma)$. This usually requires $q_I$ to be written in a highly expressive query language. As already indicated above, we require languages that create abstract identifiers. It is known from [1] that this identifier creation may involve computing a fixed point, which is more expressive than what we would use in simpler SQL-like query languages.

Furthermore, for each interaction object $(i, v)$ in $q_I(db)$ we interpret the abstract identifier $i$ as a surrogate for a URL address. In this way the queries used in interaction types already define the navigation structure of the WIS. This is a fundamental difference to work by others as e.g. in [4], where views are only used to extract data, whereas the navigation structure is added later on in a separate design step.
Apart from these subtle differences, media types extend interaction types by cohesion in order to enable adaptivity. *Cohesion* introduces a controlled form of information loss exploiting the partial order $\geq$ on nested attributes.

**Definition 6.2.** If $X_M$ is the representing attribute of an interaction type $M$ and $\text{sub}(X_M)$ is the set of all nested attributes $Y$ with $X_M \geq Y$, then a preorder $\preceq_M$ on $\text{sub}(X_M)$ extending the order $\geq$ is called a *cohesion preorder*.

Large elements in $\text{sub}(X_M)$ with respect to $\preceq_M$ define information to be kept together, if possible. Clearly, $X_M$ is maximal with respect to $\preceq_M$. This enables a controlled form of information decomposition [25]. So we obtain the following (simplified) definition of a media type.

**Definition 6.3.** A *media type* is an interaction type $M$ together with a cohesion preorder $\preceq_M$.

As media types are extended views over some database schema, any instance of such a schema defines a set of objects for the media types, which we call media objects. The difference between these media objects and the interaction objects that we considered first is that the existence hierarchies lead to different versions of these objects, while adaptivity replaces a single object by an interlinked sequence of objects, i.e. the information is fragmented. Nevertheless, these versions and sequences are determined by the media type, and for our purpose of integrating these types we do not have to look at the objects at all.

[25] contains an algorithmic approach to adaptivity based on cohesion preorders. This algorithm is not relevant for our purposes here. In that article also alternatives to cohesion preorders, which consist of proximity values, but lead to the same results, are discussed.

### 6.2 Transformation Rules for Media Types

The addition of cohesion preorders requires further extensions to our view integration methodology. We now need additional transformation rules dealing with the impact of the view integration on the cohesion. We may dispense with discussing operations as these have already been captured by the rules for dialogue types. So the following rules are either extension rules to the basic transformation rules 1-26 or to the rules 27-37 for dialogue types.

**Rule 38.** In case rules 1 and 27 are applied to a media type, if $X(A'_1, \ldots, A'_k)$ ($k \leq m$) appears in $Y \in \text{sub}(X_M)$, replace it by $A'_1, \ldots, A'_k$.

**Rule 39.** In case rules 2 and 28 are applied to a media type, whenever $r$ appears in $Y \in \text{sub}(X_M)$, replace it by $r_1^{(r)}, \ldots, r_n^{(r)}$.

Same as for dialogue types we may ignore rules 3, 4 and 5 for media types.

**Rule 40.** In case rules 6 and 29 are applied to a media type omit $A$ in $Y \in \text{sub}(X_M)$, whenever it appears.
Rule 41. In case rules 7 and 30 are applied to a dialogue type omit $r_1, \ldots, r_n$ in $Y \in \text{sub}(X_M)$, whenever it appears.

These extensions capture the first group of rules dealing with schema restructuring. For the second group of rules dealing with the shifting of attributes we obtain the following extension rules in case the rules are applied to media types.

Rule 42. In case rules 8 and 31 are applied to a media type omit $A_i$ in $Y \in \text{sub}(X_M)\text{ associated with } R_{i,\text{new}}$, whenever it appears.

Rule 43. In case rules 9 and 32 are applied to a media type omit $A_i$ in $Y \in \text{sub}(X_M)\text{ associated with } R_{\text{new}}$, whenever it appears.

As for dialogue types the last two extension rules have no effect on $R_{\text{new}}$ or $R_{i,\text{new}}$, as the extension of the cohesion preorder has to be defined for these new types.

Rule 44. In case rules 10 and 33 are applied to a media type replace $r_n$ by $r_{1,\text{new}}$ in $Y \in \text{sub}(X_M)$ and extend the now incomplete cohesion preorder.

Rule 45. In case rules 11 and 34 are applied to a dialogue type omit $A$ in $Y \in \text{sub}(X_M)$ and extend the resulting flawed cohesion preorder.

For the third group of rules, i.e. rules 12-15 dealing with schema extension only rules 12 and 15 give rise to reasonable extension rules for media types. This defines the following two extension rules.

Rule 46. In case rule 12 is applied to a media type, add $A$ to all $Y \in \text{sub}(X_M)$ associated with $R_{\text{new}}$.

Rule 47. In case rule 15 is applied to a media type, define the Cartesian product of the cohesion preorders and extend the resulting flawed cohesion preorder.

For the group of rules dealing with type integration, i.e. rules 16-19, no reasonable extension rules for media types can be defined, as we deal with new types. Finally, for the group of rules dealing with integrity constraints again only rules 20, 21 and 23 give rise to the following three extension rules for dialogue types.

Rule 48. In case rules 20 and 35 are applied to a media type replace $r_2$ by $r_{\text{new}}$ in $Y \in \text{sub}(X_M)$, whenever it appears.

Rule 49. In case rules 21 and 36 are applied to a media type replace $r_2$ by $r_{2,\text{new}}$ in $Y \in \text{sub}(X_M)$, whenever it appears.

Rule 50. In case rules 23 and 37 are applied to a media type remove $A$ in $Y \in \text{sub}(X_M)$, whenever it appears.
6.3 Case Study

Let us continue the case study from the previous sections. What we have to do is to study how a cohesion preorder associated with a view will be affected by the view integration process.

**Example 6.1.** Consider the view $V_1$ on schema $S_1$ from Example 4.1. The representing attribute of the type $\text{CUSTOMER}_{CR}$ is a tuple attribute with five components, two of which are set attributes. As a shortcut we may write $(C, N, D, O \{(T_1, A_1, F)\}, L \{(T_2, A_2, B)\})$ for this attribute.

Here $C$ represents customer no, $N$ the name, $D$ the date of birth, and $O$ and $L$ the income-obligations and loans, respectively. If this is used as the basis of a media type $M$, it is easy to calculate that there are 648 attributes in $\text{sub}(X_M)$. As this is a bit too big for our purposes here, let us concentrate only on one component, say on the attribute $O \{(T_1, A_1, F)\}$, for which $T_1, A_1$ and $F$ represent the attributes type, amount and frequency, respectively.

The corresponding lattice is shown in Figure 6. Thus, we may define the following cohesion preorder:

$$O \{(T_1, A_1, F)\} \preceq O \{(T_1, A_1)\} \preceq O \{(T_1, F)\} \preceq O \{(T_1)\} \preceq O \{(A_1, F)\} \preceq O \{(A_1)\} \preceq O \{()\} \preceq ()$$

If we now remove the attribute $F$, this preorder reduces to

$$O \{(T_1, A_1)\} \preceq O \{(T_1)\} \preceq O \{(A_1)\} \preceq O \{()\} \preceq ()$$
7 Conclusion

In this article we revisited schema and view integration and cooperation. We presented a method for schema (or view) integration following the framework in [16], i.e. we first “clean” given schemata by removing name conflicts, synonyms and homonyms, then we add inter-schema constraints, and to this schema we then apply formal equivalence transformation or augmentation rules. The transformation and augmentation rules are correct in the sense that they will always result in a new schema / view that is equivalent to the original one or dominates it. For this we introduced a new concept of schema equivalence and dominance based on computable queries.

This new notion of equivalence and dominance subsumes all reasonable existing ones, so we do not lose valuable transformation rules. On the other hand it is rather general, so that the set of transformation rules can be extended if needed without violating schema equivalence. It is an open problem to characterise exactly, which integration problems would require which notion of dominance and equivalence. The work in this article did not aim at solving this interesting theoretical problem. We concentrated instead on finding a reasonable approach to view integration and cooperation that is theoretically founded, but pragmatically oriented.

In follow-on steps we applied the view integration and cooperation framework to databases, data warehouses and web information systems. In the first two cases the key concept is that of a dialogue type, which is a view extended by operations. In the last case the key concept is that of a media type, which further extends dialogue types by cohesion in order to facilitate adaptivity. In all three cases the integration of the extended views requested additions to the transformation and augmentation rules.

The extension of view integration and cooperation to data warehouses and web information systems is completely new. So far, the relevant work in the literature concentrated exclusively on the structural aspect, leaving the implications to view extensions aside. As the extensions to the views are the core contribution of dialogue types (for the case of data warehouses) and media types (for web information systems), respectively, it is important to have a pragmatic view integration and cooperation method at hand. Without such a method the development methodology would be incomplete.

Thus, the presented approach to view integration and cooperation turns out to be general enough to be applicable to a large class of data-intensive information systems. By means of the areas for which we demonstrated this applicability, it contributes to a decisive aspect in systems development. We demonstrated this applicability by a case study for all three areas. However, the integration of view integration / cooperation into an overall framework for systems development has been left out of this article. For dialogue-oriented enterprise information systems we refer to [21], and for WISs to [25].

On a more theoretical basis the new concept of schema dominance and equivalence can be the subject of a deeper investigation. In particular, it opens the possibility to develop even more powerful transformation and augmentation rules.
and to investigate the implications on complexity. Such problems are left for future research.

References


Pragmatics of Storyboarding for Web Information Systems I: Usage Analysis

Klaus-Dieter Schewe and Bernhard Thalheim

1 Massey University, Information Science Research Centre
Private Bag 11222, Palmerston North, New Zealand
k.d.schewe@massey.ac.nz

2 Christian Albrechts University Kiel, Department of Computer Science
Oshausenstr. 40, D-24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de

Abstract. On a high level of abstraction a Web Information System (WIS) can be described by a storyboard, which in an abstract way specifies who will be using the system, in which way and for which goals. While syntax and semantics of storyboarding has been well explored, its pragmatics has not. This paper contributes the first step towards closing this gap by analysing the usage of WISs. Starting from a classification of intentions we first present life cases, which capture observations of user behaviour in reality. We discuss the facets of life cases and present a semi-formal way for their documentation. Life cases can be used in a pragmatic way to specify a story space, which is an important component of a storyboard. In a second step we complement life cases by user models that are specified by various facets of actor profiles that are needed for them. We analyse actor profiles and present a semi-formal way for their documentation. We outline how these profiles can be used to specify actors, which are an important component of a storyboard. Finally, we analyse contexts and the way they impact on life cases, user models and the storyboard.

1 Introduction

A Web Information System (WIS) is an information system that can be accessed through the world-wide-web. On a high level of abstraction a WIS can be described by a storyboard [12], which in an abstract way specifies who will be using the system, in which way and for which goals. In a nutshell, a storyboard consists of three parts:

- a story space, which itself consists of a hierarchy of labelled directed graphs called scenarios, one of which is the main scenario, whereas the others define the details of scenes, i.e. nodes in a higher scenario, and a plot that is specified by an assignment-free process, in which the basic actions correspond to the labels of edges in the scenarios,
- a set of actors, i.e. abstractions of user groups that are defined by roles, which determine obligations and rights, and user profiles, which determine user preferences,
- and a set of tasks that are associated with goals the users may have.

In addition, there are many constraints comprising static, dynamic and deontic constraints for pre- and postconditions, triggering and enabling events, rights and obligations of roles, preference rules for user types, and other dependencies on the plot. Details of storyboarding have been described in [12]. An overview of our method for the design of WISs was presented in [11].

While syntax and semantics of storyboarding has been well explored, its pragmatics apart from the use of metaphors [15] has not. Pragmatics is part of semiotics, which is concerned with the relationship between signs, semantic concepts and things of reality. This relationship
may be pictured by the so-called semiotics triangle. Main branches of semiotics are syntactics, which is concerned with the syntax, i.e. the construction of the language, semantics, which is concerned with the interpretation of the words of the language, and pragmatics, which is concerned with the current use of utterances by the user and context of words for the user. Pragmatics permits the use of a variety of semantics depending on the user, the application and the technical environment. Most languages defined in Computer Science have a well-defined syntax; some of them possess a well-defined semantics; few of them use pragmatics through which the meaning might be different for different users.

Syntactics is often based on a constructive or generative approach: Given an alphabet and an set of constructors, the language is defined as the set of expressions that can be generated by the constructors. Constructions may be defined on the basis of grammatical rules.

Semantics of generative languages can be either defined by meta-linguistic semantics, e.g. used for defining the semantics of predicate logics, by procedural or referential semantics, e.g. operational semantics used for defining the semantics of programming languages, or by convention-based semantics used in linguistics. Semantics is often defined on the basis of a set of relational structures that correspond to the signature of the language.

Pragmatics has to be distinguished from pragmatism. Pragmatism means a practical approach to problems or affairs. According to Webster [16] pragmatism is the “balance between principles and practical usage”. Here we are concerned with pragmatics, which is based on the behaviour and demands of users, therefore depends on the understanding of users.

The six characteristics of WISs that were discussed in [12] can be mapped to conceptual structures that are used for storyboard specification:

1. We start with the characteristics used for the strategic layer. Main specification elements used are intention and mission. They are mapped to metaphors, general goals, rhetorical figures, and patterns and grids of web pages discussed later.
2. The scenarios reflect the utilisation by actors, for which we envision a number of stories that correspond to real use. These scenarios may be captured through observation of reality. Story spaces and plots are recorded in various level of detail through the methods discussed in [12]. The stories are reflected in the storyboard.
3. Content specification is the basis for the media types, i.e. data types and their functions, which will be introduced in part III. It combines data specification with user requirements and is reflected in the content portfolio.
4. Functionality is provided by the media types as required by the storyboard. Typical standard functions are navigation, retrieval (search), support functions, and feedback facilities.
5. Context is based on tasks, history, and environment. We use the specification of context for restructuring and functionality enhancement, which will form the basis of XSL transformations and the onion approach that is discussed in the last part of the book.
6. Presentation depends on the intention, the provider, the technical environment available and the users the WIS is targeting at. Presentation results in the layout and the playout of the WIS. Layout requires the development of multimedia presentations for each page. Playout additionally requires the development of functionality that supports visits of users depending on the story they are currently following to achieve their goals. Layout and playout integrate the chosen metaphors; they depend on chosen page patterns and grids as well as on quality requirements.

Conceptual structures and their association are depicted in Figure 1. We may separate the syntactics and pragmatics layers. Arrows are used for representing part-of- or uses- or relates-
associations. For instance, the story is based on the user and the functions. Information metaphors relate content to information.

![Diagram](image)

**Fig. 1.** The Web Utilisation Space Based On the Characteristics of WIS

We use the notions of information and content in a specific manner. *Information* as processed by humans, is carried by *data* that is perceived or noticed, selected and organized by its receiver, because of his subjective human interests, originating from his instincts, feelings, experience, intuition, common sense, values, beliefs, personal knowledge, or wisdom, simultaneously processed by his cognitive and mental processes, and seamlessly integrated in his recallable knowledge. Content is complex and ready-to-use data. Content management systems are information systems that support extraction, storage and delivery of complex data. Content may be enhanced by *concepts* that specify the semantic meaning of content objects and by *topics* that specify the pragmatic understanding of users.

Therefore, information is directed towards pragmatics, whereas content may be considered to highlight the syntactical dimension. If content is enhanced by concepts and topics, then users are able to capture the meaning and the utilisation of the data they receive. In order to ease perception we use *metaphors*. Metaphors may be separated into those that support perception of information and into those that support usage or functionality.

Users are reflected by actors that are abstractions of groups of users. Pragmatics and syntaxics share data and functions. The functionality is provided through functions and their representations. The web utilisation space depends on the technical environment of the user. It is specified through the layout and the playout. Layout places content on the basis of a data representation and in dependence of the technical environment. Playout is based on functionality and function representations, and depends on the technical environment.

We may base the pragmatics of storyboarding on two ingredients that are easy to develop:

**The utilisation portfolio of the WIS:** The utilisation portfolio consists of
- tasks users might wish to accomplish within the system,
- goals of user groups, i.e. actors, and
- actors that might use the web information system.

**The profiles of the users:** Profiles of users describe their abilities, skills, knowledge, and other dimensions. We map user profiles to preference rules for utilisation of systems.
We first consider these basic questions. They are related to the strategic and the business layers, so they also have an impact on the storyboard. Refinement and utility take us to the conceptual layer.

In this article we approach the analysis of WIS usage as the first important part of storyboarding pragmatics. After a brief review of related work in the literature in Section 2 we first look at intentions that are associated with the WIS, and classify them using facets for purpose, intent, time and representation, the first of which was already discussed in the strategic model of WISs [7]. These facets of intentions form the basis for the following three sections, in which we introduce life cases, user models and contexts. Section 4 is devoted to the discussion of life cases, which capture observations of user behaviour in reality. We discuss the facets of life cases that impact on the design of storyboards and present a semi-formal way for their documentation. Life cases can be used in a pragmatic way to specify the story space. The work on life cases extends a previous conference publication [9]. In Section 5 we complement life cases by user models that are specified by user and actor profiles, and actor portfolios. We analyse actor profiles and present a semi-formal way for their documentation. The actor portfolios are used to get a better understanding of the tasks associated with the WIS. The work on user models extends a previous conference publication [10]. Finally, in Section 6 we analyse contexts and the way they impact on life cases, user models and the storyboard. In Section 7 we conclude with a brief summary and a discussion of open issues.

2 Related Work

Storyboarding and also the preceding strategic modelling of WIS [7] are unique to our approach to WIS modelling. Other approaches to WIS engineering such as the object oriented OOHDM [4, 8, 14], WebML [1], HERA [5] and variants of UML [2, 6] concentrate on providing models of content, navigation and interaction by means of extended views, which in our own work is captured by so-called media types [12]. Only WSDM [3] emphasises the additional need for a mission statement and a high-level description of processes and users in the WIS.

to be complemented

3 Facets of Intention

The description of intention is based on a clear understanding of aims and targets of the WIS including a specification of long-range and short-term targets, expected visitors, characteristics of this audience, tasks performed by the users, necessary and unnecessary content, and finally an understanding of restrictions of usage.

Utilisation scenarios are developed on the basis of intentions. An intention specifies what one purposes to accomplish or do, i.e. one has in mind to do or bring about. It has four facets that are based on the general characteristics and questions discussed in [7]:

Purpose facet: The purpose of stakeholders specifies an anticipated outcome that is intended or that guides the planned actions. It may be based on two additional pieces of information:

- The aims specify what is intended to be attained and what is believed to be attainable.
- The objectives are more general than the aims. They specify something toward which an effort is directed, e.g. goals or ends of an action.
The purpose is already specified at the strategic layer. It depends on the audience intended for the WIS, and is influenced by the mission that is determined by the WIS provider.

**Intent facet:** The intent suggests a clearer formulation or greater deliberateness. It distinguishes between the following two aspects:

- The targets of stakeholders specify the steps of a plan and its termination or satisfaction conditions.
- The object of stakeholders is related to wishes or needs of users, and specifies an effort or activity of a user in order to satisfy the need.

The intent facet is related to tasks the user wants to accomplish. The intent may be ordered into major intents specifying the main uses of the system and minor intents that may be only partially of interest. Intents may be generalized to themes that represent classes of intents.

**Time facet:** The time facet is used for specification of the general time restrictions such as

- the design, which implies a more carefully calculated plan, and
- the end, which stresses the intended effect of an action, often in distinction or contrast to the action or means as such.

Time facets may be very general. In this case, we use occasions to represent an entire class of time frames.

**Representation facet:** Intentions are described or annotated through utterances, words or pictures. The word representation is related to word fields used in the strategic model [7]. The icon representation is based on metaphors. Word fields may be specialised to concept fields discussed later in this chapter. Representation is deeply dependent on the cultural environment and the community that uses the WIS. Intentions can be supported by providing stimuli that rouse or incite activity.

Intentions may be restricted by a scope. The scope allows to concentrate on the main aspects. Typical restrictions are the cultural environment, education or other profile properties of potential users, or specific time facets for utilisation of the WIS.

The first two facets of intention have a general form (objective, object) and a more concrete form (aim, target). The purpose facet depends on the mission and the audience, whereas the intent facet depends on the tasks. Therefore, the first facet specifies the ‘what’, the second the ‘how’, and the third facet the ‘when’ of an intention. We may either concentrate on a more general specification or a more concrete one.

The different facets may be considered separately or altogether in a condensed form. The detailed consideration is necessary, whenever a fastidious audience requires sophisticated content. High demands come into consideration due to the content provided, the community that must be satisfied, the sophisticated functionality required for the WIS, or the high attention the WIS gains.

*Example 1.* The purpose facet is often considered to cover ‘soft intentions’ or by ‘business rules’. In an information service a user may be interested or becoming more interested in some content. The intent facet is different and mainly driven by tasks the user tries to solve.

Similarly, we may derive a number of intentions a user has in mind whenever s/he visits an edutainment site:

**Aim:** An aim of an edutainment user may be to obtain a certificate that proofs the success of learning. Aims of providers of edutainment sites might also be binding the learner to
some products such as the software of a supporter of the website. Typical general aims within an edutainment site are to grant equal rights to everybody; nobody should have higher rights than other learners.

**Objectives:** Typical objective of using an edutainment site are greater ease of learning, greater pleasure or satisfaction during learning, and more fun. Another general objective is security and privacy. Edutainment applications also host confidential information, e.g. information on progress and errors. The learners need to be sure of the security and privacy of their data. They should be informed of the security precautions.

The *audience* of an edutainment site may be rather unspecific or more specific such as students of a college or analysts of a bank.

The *mission* of an edutainment site is to support learning by providing easy-to-grasp knowledge. At the same time, the provider may be interested in increased visitor-to-customer conversion rate, increased number of returning customers, and increased revenue.

The intent is related to the more concrete tasks a user has in mind while visiting a WIS.

*Example 2.* Intents come directly from analysing the needs and the demands of users. Some possible *tasks* a user may want to accomplish in an edutainment site are the following:

- A user realises that certain knowledge or information is necessary for solving a task. For instance, an analyst of a bank wants to know what the changes are in the customer community that led to a rise of faulty credits.
- A student in a college needs to know methods for analysing data. In this case, the student is interested in a data mining course that provides material on his/her educational level, permits learning the content interactively, and is comparable to the material the student is currently working with in the college.

Possible intents in an edutainment site are the following:

**Objects:** There are a number of objects such as faster task completion, successful completion of more tasks, or commission of fewer errors.

**Targets:** In the bank analyst case, the analyst needs to know data mining methods, to capture the achievements and the disadvantages and pitfalls of such methods.

The *themes* in the bank analyst case could be the interest in learning association methods, preprocessing of data, and prediction methods.

The time facet represents general time restrictions such as the time or the interval for visits of the WIS, the time and the interval of repeated visits, or the temporality that may force a user to leave the site.

*Example 3.* Visitors of a learning site can be characterised by their time dimension, i.e. the time they access and use the site, the access pattern they prefer, and the response time they request from the site.

**Design:** An edutainment site may allow a visitor to use the material on one shot or to use it step by step. In the latter case, didactics for the elaboration of material becomes an issue of WIS development. Step-by-step learning depends on the cultural environment of the user, the collaboration with other learners, and the interactive facilities of the site.
Furthermore, edutainment sites may be used over a longer time period. Therefore, context-sensitive help becomes important, because some actions or scenes may not be obvious to all users. Making such help available, e.g. through pop-up windows ensures that the learners do not lose time and interest.

**End:** Each action of a user leads to some effect that is either made visible to the user, recorded, or changes the state of the learning process itself. The ends of actions must be checked against the intended effect. This kind of consistency check is supported by acceptance conditions used for explicit checkout from scenes.

For edutainment sites, typical *occasions* may be the sudden requirement to explore some knowledge or to prepare a piece of homework. Another occasion might be the sudden demand in quality information due to some news popping up somewhere else.

The representation facet represents the flavour or atmosphere of a site as discussed in [7]. This facet is largely determined by the other three facets.

**Example 4.** An edutainment site that supports bank analysts should strive for an aesthetic and minimalist design. The main target for the representation is to support work. The site needs to ensure a clean and understandable layout. All irrelevant features must be omitted. At the same time an informative feedback is required. The site needs to be consistent throughout. The analysts want to control their way of working. At the same time they become experts of the site and thus need shortcuts and accelerators. Furthermore, they are restricted in their attention to the site, so they need a facility to recover from errors.

An edutainment site oriented to support pupils may follow completely different design principles. Pupils need to be attracted by graphical elements they like, they face in every days life, and which make using the site a pleasing experience. Pupils have a less-trained short-term memory. So, the short-term memory load is reduced, if pupils can recognize what they need to know from visible elements of the site.

The set of intentions we should consider may be rather large. Moreover, the faceted representation may become too difficult to manage and to satisfy. The order we may use depends on the audience and on the provider. The audience is oriented towards solving tasks. The provider has a ‘mission’. We can use these two restrictions to figure out which kind of website supports this profile, to harmonize our understanding with the corporate identity, to order the target realisation (short-term, long-term), and finally to develop an understanding how the site will look like in two years from now on.

**Example 5.** Aims and targets can be ordered. We can use the mission and goals of the provider and compare them with the tasks the provider wants to support for different audiences. For instance, we may range the priority using a scale from 1 (highest) to 5 (lowest).

<table>
<thead>
<tr>
<th>aims and targets</th>
<th>community customer</th>
<th>casual customer</th>
<th>reseller</th>
<th>hunter/gatherer</th>
<th>kids</th>
<th>students</th>
</tr>
</thead>
<tbody>
<tr>
<td>selling books</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>informing on books</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>reviewing books</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>selling bestsellers</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>developing a community</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
For business or information sites we obtain the following order:

1. primary intentions of the provider;
2. primary necessities and demands of the customers;
3. secondary aims of the provider;
4. stimuli for customers to revisit the system and to buy more;
5. audience education for the customer with our customer community;
6. beauty for customers;
7. customers self-realisation.

The specification of the intended audience of the WIS is based on a weighted list of kinds of users, their aims, a comparison of the interaction with users we have so far, and a specification of aims of the audience and reasons for re-visiting.

**Example 6.** Let us consider an information service that supports traveling, e.g. based on the brand

\[(\text{City, Hotel})^\text{information}_2(\text{Tourist, Investor})^\text{inform, book, invest}.\]

This pattern is based on the following coarse specification of intention:

- **Purpose:** attraction of visitors and building up customer relations based on a *mission* that is centered around area information and attractions.
- **Intent:** travel convenience, hotel room sales and more depending on the *theme* travel support.
- **Occasion:** utilisation of links from the site or other associated sites.
- **Metaphors** used for presentation: survey and collecting basket.

The intention reflects tasks such as travel preparation, supporting visitors of an area, and general information for citizens and tourists.

**Example 7.** The intention *prepare for a visit* is based on a number of intents such as addressing visitors beforehand for some purpose, use, or activity. It includes the aim to put the visitor of the WIS in a proper state of mind. A task associated with preparation may be to work out the details of a plan in advance. The task may be extended by putting pieces of information together into a compound form or preparing a report. The time facet may range from ‘now’ until ‘next opportunity’. Typical metaphors supporting this intention are baskets, descriptions, and cultural journals.

The intention *prepare for a cinema visit* extends the intention of *prepare for a visit* by a certain content, a refinement of the time facet to the next possible visit, and a refinement of the purpose facet now targeting at entertainment.

We may use a graphical language for the representation of intentions. The language we are using extends the task-goal graphs considered in [7]. Tasks are represented by rectangles. Goals are represented by diamonds. We add now purposes, intents, data types that may be used for task completion, and the time and representation facets. Since purposes and intents refine tasks, we may place those into the tasks. Purposes, e.g. aims are represented by rounded rectangles. For objectives we use clouds. Resources, i.e. media types are represented by document icons. Intents are represented by hexagons. In addition, we may add the time facet, the representation facet, and supporting databases. Figure 2 displays a graphical presentation of the intention *prepare for a visit*. We have added links that show the dependence among the facets.
Fig. 2. The graphical representation for the intention *prepare for a visit*

We can combine the description of intentions in a semi-formal way as follows. The items in the list are optional except the first one.

**Intention space:** *(intention name)*

**Purpose:** *(outcome description)*

**Aims:** *(list of aims)*

**Objectives:** *(list of objectives)*

**Intents:** *(outcome description)*

**Targets:** *(list of weighted targets)*

**Objects:** *(list of weighted objects)*

**Themes:** *(class of intents)*

**Time:** *(outcome description)*

**Design:** *(general flow)*

**End:** *(effects, termination conditions)*

**Occasion:** *(list of objectives)*

**Presentation:** *(general style guide)*

**Atmosphere:** *(general description of atmosphere)*

**Metaphors:** *(list of metaphors)*

**Based On:** *(tasks, audience, mission, goal)*
4 Life Cases

Life cases allow to overcome the information overload and lost in the hyperspace syndromes typically observed for WISs. For completion of tasks users need the right kind of data at the right moment of time, in the right dose, of the right form, in the complete extent, and within the restrictions agreed upon in advance. Moreover, users are limited in their abilities for verbalisation and digestion of data, and by their habits, practices, and cultural environment.

These limitations may cause intellectual overburdening of users. Most systems that require sophisticated learning courses for their exploration and utilization did not consider these limitations and did not cope with real life situations. The approach we use for avoiding overload is based on observation of real applications before developing the system.

We may extract life cases from observations in reality, which are very useful source, whenever a WIS is going to be developed from scratch or is required to provide a ‘natural’ behaviour. In the latter case users are not required to learn the behaviour of the WIS. Instead, the user can continue using a ‘classical’ behavioural pattern.

*Example 8.* As a motivating example let us consider the life case *relocation* of a person, which consists of

- the change of basic relocation data including the possible removal of data on the old location,
- the change of official documents such as the passport,
- the optional change of relation enhancements such as the registration of pets, relocation of cars,
- the change of personal specific data such as family enhancements, or relationships to religious bodies,
- the change of data for additional relocation announcements such as tax, insurance changes, and
- specific additional tasks such as applications for housing allowances.

The person acts in the role of an *issuer*. We observe that *relocation* is enhanced by the profile of the issuer, by the specific tasks related to the *relocation* of the issuer, by specific laws and regulations, and by advanced functionality required for associating the life case with other life cases.

The life case *relocation* consists of steps such as change of address data, change of data for associated people, change of registration data for cars, pets, etc., change of specific data, e.g. data for public authority responsible for aliens, change of data for social aid, etc. These steps are bundled together due to their relationship to one person and to one life case. The associations may be represented by adhesion of different steps, e.g. representing the association of steps by a hypergraph, e.g. the one in Figure 3.

4.1 The Concept of Life Case

*Life cases* are characterized by:

**Observations:** We are interested in the collection and assessment of behaviour relating to the specific application. This would typically involve an observation of behaviour of users in real environments, including a background check that relates usage to intentions, goals or tasks.
**Processes:** This involves arranging all the actions observed in an application into a main logical and coherent pattern. In some case, deviations of the main pattern must be modelled through exceptions. In most cases, we can use parallel execution of independent actions.

**Assessment:** This involves the reconstruction of the sequence of actions and specific behaviour of users. This will aid in understanding the role each individual has within the story. It assists in developing the user profile.

**Individual profiles:** A list of background including physical, and behavioural characteristics of individuals is conducted. This list can also be used for deriving the most appropriate interview technique we discuss below.

**Interpersonal coherence:** A variation in the activity will relate to variations of other life cases.

**Significance of time and place:** The choices made also depend on mobility, surrounding, and schedules of events of interest.

**Characteristics of the life case:** Individuals using a service may be grouped by characteristics. Based on this grouping a similar behaviour is observed.

**Experience and skills:** Individuals may have their own experience with services provided in real life and thus use different behavioural pattern of service employment.

In general, life case studies are designed to produce a picture of service employment that is as accurate as possible. Determining why, how, where, when and why a service is called using what data provides a better picture for utilisation scenario. As life cases are used for quality control, we must carefully record our observations. We either use a natural language specification or a semi-formal one as described later.

**Example 9.** Let us consider the support of hotel search within an information service. In this case we may observe the behaviour of individuals in travel agencies while seeking for hotels. We observe that in most cases search based on associations is preferred over search by main properties such as name, address or facilities. Associations may be of a large variety, e.g. convenience to reach a hotel, location in certain maps, places of interest, or events that have caused the search. Other search criteria may be bargain or bundled offers. At the same time we observe that hotel search is combined with other intentions of users such as visiting cultural institutions. It may depend on results for search of other individuals.

Another example of a life case study is the booking of train tickets depending on individuals, offers of railway companies, circumstances of individuals are using trains, etc.

4.2 Life Case Development

Life cases may be developed and interpreted step by step:

**Fig. 3.** Hierarchically ordered life case steps for relocation
1. The first step during life case collection is the survey of possible application cases we might consider. The observations could have more than one meaning and may follow a variety of user-related goals. In this case we consider the most likely meaning.

2. The second step involves a deep assessment of the life cases. We extract the different execution orders, the relationship among actions, and the roles individuals play these actions.

3. The third step extracts those life case characteristics that are distinguishing features and are relevant for time and location. At the same time we search for similarities within these life cases.

4. The final step is concerned with the characterization of potential users, their behavioural patterns, and their roles in various stages of the life case.

Collectively, this information will produce a picture of the life case we are intending to support by a WIS. This may produce further life cases, or may aid in reducing the amount of development. It may result in a prioritisation of life cases under consideration, assist in the linkage of potentially related life cases, assist in assessing the potential of the WIS development, provide the WIS developers with relevant leads and strategies, and keep the WIS development on track and undistracted. These life case are mapped to scenarios in the sequel. The integration of scenarios can also be based on life cases.

Example 10. Let us consider an information service for a city or a region, which supports a number of life cases:

- Attracting visitors: The information we are providing is based on some information need visitors may have. The life case follows those that we observe during marketing activities.
- Inhabitants information: The life case is based on selected information chunks that may be of interest to individuals, an ordering of the corresponding available information, and a derivable personal newspaper.
- Informing tourists: The life case is similar to those observed in city information centres.
- Providing official city information to inhabitants: This life case follows the message board metaphor that is used for newspaper information.

During the development of city information services such as the service www.cottbus.de we have made a life case analysis for city information services and detected around 30 different life cases. We can categorize these life cases by the content and information demand of individuals. In this case we distinguish:

- life cases related to tourist content for inhabitants permanently living in the city, for inhabitants temporarily living in the city, for short-term tourists and business people on short term trip, for vacationists planning a trip for more than 5 days, for teenagers with uncertain wishes, for seniors with particular interests, for week-end visitors, for visitors of highlights, festivals, etc.;
- life cases related to official content for inhabitants on search through directory of public authority, for inhabitants on search for emergency cases, for inhabitants orienting themselves before applying at public offices, for inhabitants interested in problems of city management, etc.;
- life cases related to business content, e.g. for investors considering engagement in the area.
For the collection and development of life cases it is normally a good idea to interview the personnel currently providing the service.

Life case should be checked against sufficiency. As they may have a variety of traces, we add two stages to life case analysis:

**Exploration:** The actual life case is provided to WIS customers and incorporated into their processes. Life case exploration can be conducted in normal environments of the user such as home or work. Then users can show the actions while they are explaining their aims and targets.

**Apprehension:** Finally, cross checking of life cases and utilisation scenario is used for correction, extension and affirmation of the developed utilisation scenarios.

During life case elicitation and specification users are confronted with sketches of scenarios. We ask what users think about these conceptualisations. The feedback is used for the refinement of life cases. We may use affinity diagramming, in which case we arrange the individual conceptions we discover on a blackboard, associate them, and discuss their relationship to the general characteristics of WISs. Another technique is based on card sorting similar to the model-view-control cards used in software engineering. In this case cards represent simple life situations. Their associations are then used for organising the conceptions. These sketches can also be used for discussing deviations from the normal case and for search of exceptional life cases. Instead of directing users to specific life cases we can show them two or more alternatives for the problem solution.

Life case analysis goes beyond task analysis. It is simple and easy to carry out, lightweight and straightforward to understand, and yet quite accurate in identifying the real demands users of a WIS might have. While observing real life cases and mapping them to life cases that must be supported by the WIS we carry out a user-centered development. Additionally, we may identify and resolve conflicting or competing intentions. These conflicts can be resolved on the basis of life cases by accommodating both intentions, i.e. fulfilling the stakeholder intentions and supporting the demand of users.

Besides observation of real life situations life case detection can also been based on interview techniques. Research on artificial intelligence has also resulted in a number of interview techniques:

- Unstructured interviews can be considered as an informal and explorative conversation on goals a user is following and on tasks a user has to complete. Unstructured interviews observe the rules that are applied to brainstorming. All interruptions should be avoided. The questions asked must be short and simple to answer and should be open-ended questions. Interview partners should share their thoughts and experiences. There is no judgement, confrontation or condescension. Users should not be lead toward a certain scenario. Unstructured interviews should only be interrupted if clarification is required. They need consecutive feedback in order to give the interviewees the impression that the interviewer is listening. While the interviewees are talking, everything that seems to be important is written down and used for later questions.

- Structured interviews are based on query plans. These plans can be based on the general characteristics of WISs. We start with easy questions on data and main functions and continue with the life cases of their utilization. Context and intentions are more difficult to capture. Structured interviews are also based on features, which can be shown to users for a rating of their importance.
– Life cases can also be detected by observing and critically analysing products of competitors. Elicitation of life cases from existing solutions is based on specifications, results from interviews with business users, excerpts from documents and spreadsheets, analyzed messaging and transactions, knowledge on the solutions that are currently used, metadata and context information on the utilization within the current framework, and third party information. As reasons for restrictions cannot be captured, the copying of already existing solutions can only be used in exceptional situations.

– Users may also use protocols of “loud thinking”, in which case they are provided with a real-life problem of a kind that they deal with during their working life, and asked to solve it. They imagine that they are solving the problem presented to them. As they do so, they are required to describe each step and the reasons for doing what they do. The transcript of their verbal account is the protocol. In this case, they work and explain their current work. These protocols can be the basis for capturing a scenario. This interview technique should be combined with other interview techniques.

– There are other interview techniques that might be useful for life case elicitation such as the laddering or grid techniques. In these cases, a hierarchical structure of the application domain is formed by asking questions designed to move up, down, and across the hierarchy.

4.3 Semi-Formal Representation of Life Cases

Although it is sufficient for life cases to be stated in natural language, we may use a semi-formal representation instead using the following template:

Life case: (life case name)
Characterisation: (outcome description)
    Tasks: (list of user tasks)
    Problems: (list of problems)
    Background: (general characterisation)
    Objectives: (list of objectives)
Life case flow: (general graphical description)
Milestones: (graph of milestones)
    Content consumed: (consumed content items)
    Content produced: (produced content items)
Themes: (class of intents)
Figures: (actors list)
    Expected profile: (general profile description)
    Collaboration: (general collaboration description)
Context: (general context description)
    Time: (temporality limitations)
    Place: (assignment of places)
Legacy: (names of documents)
WIS: (general WIS context)
Representation: (general behavior)
Approaches: (general description of approaches)

This template captures the following components of life cases:
**Characterisation:** Life cases are characterised on the basis of strategic issues, the problem statement, background and objectives for the life case, the methodology that is used for solving the life case, and by describing the basic modules that are used for solving the life case. The characterisation is harmonized with intentions, tasks and goals.

**Life case flow:** The life case flow combines the observations we made, the processes involved, and the data which are consumed or produced. The life case flow is mapped to a scenario.

**Figures:** We develop profiles, especially those of individuals, as part of the WIS utilization portfolio, and interpersonal coherence specifications.

**Context:** Time and location is explicitly described for life cases. The applicability of life cases is restricted by regulations, laws, and orders. These restrictions are seldom given in an explicit form. In addition, the context of life cases is given by the provider, the intended audience, the utilization history, and the availability of data due to the technical environment. We also use this information for the context specification.

**Requirements:** Life cases are restricted by habits, general approaches, good practices, and boundary conditions for their application. They presuppose experiences and skills of the users involved.

Note that life case we intend to support by a WIS can be completely different in real life. Sometimes we need a complete reorganisation of the business activities. In this case we should not map the real life case to a suite of associated scenarios, but rather envision a better organisation of the tasks and goals and then map these to a new envisioned hypothetical life case.

**Example 11.** For illustration let us consider a typical life case that is currently based on mailing and delivery services. Such a life case can be often observed, when a number of institutions is involved in the actual life case and these institutions collaborate loosely using a number of default and exceptional cases. Often, the latter ones are not explicitly stated, but belong to the service context.

For instance, a fee applies to kindergarten or day nurseries. This fee depends on the income of the parent(s) or legal guardians, their social status, the regulations of the day nursery, and the support of the government. Additionally, exemption of fees may be applied for. So, the observed life case consists in the following steps:

- Parents or legal guardians are filling in an application provided by the day nursery. This application is mailed to the youth welfare department.
- The youth welfare department checks the application. They use data provided by other city officials. The result can be the refusal or acceptance of the application, or the request to extend the application by further data.
- In the last case a new application form is send back to the applicant(s) that will be the basis for calculation of the fees.
- The applicants are providing additional data, e.g. data about their income. This new application form is mailed to the youth welfare department.
- The youth welfare department calculates now the fees that are applicable in the (special) case.
- The decision of the youth welfare department is mailed back to the applicants.
- The applicants may accept the decision made or may file an objection against the decision. The objection is mailed again. It may cause another partial or full assessment.
This life case should not be mapped to a supported life case. Beside unreliability of mailing services a good number of exceptions can be observed. We may use another approach that has been implemented within the Cottbus university service system. Here the applicant becomes the owner of the application and as such all new data associated with the application are made visible to the applicant. Whenever new essential data are added to the application the applicant is notified about it. If a reaction is necessary then the notification is extended by a deadline tracking life case. The applicant opens a number of views to other actors with rights to see, to update or to extend data. Concurrently, the youth welfare department uses another document tracking system. This system is based on a blackboard approach, i.e. all documents which are currently under consideration are placed onto the blackboard and are ranked by their priority and deadlines. The document tracking system is supported by a work support system. This system contains all necessary regulations, a number of samples, and a data acquisition system that extracts, transforms, and loads data from other collaborating systems. The work support system provides also support for handling exceptional cases.

Thus, the new life case consists of an application delivery and tracking life case, a document handling and assessment life case, and a planning, scheduling and work support life case. These life cases are mapped to scenarios, which are further combined into a single scenario. The three specific life cases become views on the combined one. Furthermore, we observe that the new life case is far more general than the previous one and could be the basis for a generic application processing WIS.

5 User Models

User modelling has changed the development to human-computer interfaces, and allows to tailor systems to the users, their needs, abilities, and preferences. User modelling is based on the specification of the user profile that addresses the characterization of the user, and the specification of the user portfolio that describes the user's tasks, involvement, and collaboration on the basis of the mission of the WIS.

5.1 User Profiles

In general, user profiles are specified through the education profile based on an insight into the knowledge, skills, and abilities of potential users, the work profile with a specification of the specific work approaches of users, and the personality profile representing the specific properties of users.

Education Profiles. The education profile is characterized by properties obtained during education, training, and other educational activities, i.e. education, capabilities, and application knowledge.

The education is characterised by the technical and professional training a user got. Technical training emphasises the understanding and practical application of basic principles of science and mathematics. Professional training places major emphasis upon the theories, understanding, and principles in such fields as science, and engineering. It results in erudition, knowledge, and literacy.

Capabilities of the user for task solutions are based on the attainment of proficiency in skills such as understanding the problem area, reasoning capabilities on analogy, realizing
variations of the problem solution, solving and handling problems, communication abilities, abilities for explaining results and solutions, and abilities for integration of partial problem solutions.

Application knowledge depends on the application type, the application domain, the application structuring by subjects, the direct structure, the description, the context and environment of the description, the parameters of the application, and application functions:

- The application type covers configuration, production, construction, design, interpretation, error analysis, repair by replacement, rebuilding, and construction, planning, supervision, monitoring, prediction, classification, recognition, instruction, training, consulting, coordination, control, judgement, and evaluation.
- The application domain can be as diverse as science, engineering, e.g. construction, computer engineering, mechanical engineering, telecommunication, energy, environmental, aviation, etc., institutions, government, business, e.g. accounting, administration, finance, production, personnel, procurement, distribution, development, design, material, research, planning, management, etc., military, or journalism.
- Application structuring by the subject refers to title, purpose, domain, evaluation, basic knowledge, and terminology.
- The direct structure captures kinds of associations, e.g. nets, clusters, relations, dimensions, hierarchies, etc., deep or narrow structuring with or without regularity, and internal structuring of the application.
- Application structuring by description addresses the representation with depth, width, level and dimension, solutions with size and complexity, kinds of handling, e.g. procedural, declarative, graphical, combined or mixed, and naming.
- The context of the description deals with associations to the user, applications, space and time, dependence and evaluation by the general context, reasons, purposes, and particularities.
- The environment of the description refers to the immersion of the application in terms of search and solution space.
- Parameters of the application can be time, space and others.
- Application functions are as diverse as accumulate, observe, form, structure, judge, weight, summarize, aggregate, adjust, prove, prefer, determine, estimate, predict, calculate, derive, transfer, control, check, plan, solve, treat, combine, analyse, decide, execute, and feedback.

Work Profiles. The work profile is mainly characterized by the task solving knowledge and skills in the application area, i.e. task expertise and experience, system experience, and information and interaction profiles.

The task expertise describes the exact and partial knowledge of data, procedures, algorithms, functions, constraints, associations, frames, graphs, schedules, rules, calculi, examples, objects, diagrams, etc. Partial knowledge is characterised by vagueness expressed via uncertainty, incompleteness, inexactness, inconsistency, and heuristics, variants of representations using evidence, evaluation, probability, confirmation, certainty, belief, confidence, support, rough sets, etc., calculi, comments, constraints, etc.

The task experience identifies both positive experience, e.g. applicable knowledge, strategies, models and theories, and negative experience, e.g. development, support or knowledge deficits, cooperation and planning support reduction, capability gaps, lack of effectiveness, insufficient competences, etc. In addition, the users' experience in coping with errors, self-organization, planning, and abstraction can be specified.
The system experience depends on the systems to be explored and used. It is limited by the user's abilities to cope with information delivered by a system and system-supported collaboration, and to work with systems in parallel to working without system support.

The information profile is based on the information needs of a user discussed below, i.e. the intentions in approaching the system, the amount of information a user can cope with, and the complexity of information a user can handle. The information profile can be modelled by database schemata that are extended by a user's preferred way of browsing through information.

The interaction profile of a user is determined by his frequency, intensity, and style of utilization of the WIS. Users might be experienced in working with several workplaces or with only one, e.g. a single browser. The interaction profile supports a flexible communication, cooperation, and coordination of a given user.

**Personality Profiles.** The personality profile of a user characterizes his/her general properties, his/her preferences in dealing with the WIS, and his/her polarity profile, which addresses psychological properties of a user.

General properties of the user or a user group are the status in the enterprise, community etc., formal properties such as name, address, password, etc., the context for performing a task such as position, allocated tasks in development, distribution and application, task area and customer properties, the psychological and sensory properties capturing vision, hearing, motoric control, information processing, speed of information uptake, attitude and anxiety, the background and personality factors capturing intentions, motivation, expectations, emotions, etc., training and education, behavioural patterns such as acceptance of system properties and methods, required guidance by help and tutoring systems, and user type, i.e. whether is casual, user, knowledgeable or expert.

For characterization of user preferences we may concentrate on those that are important for WIS development such as preferences for input and output devices and dialogues. For input and output devices this captures the handling of types such as text, picture, graphics, audio and video and the required support for these, the preference of specific forms, e.g. character-, line-, window- or form-orientation with presentation preferences for colour, shape, size, font, format, contrast, etc., the required guidance and help during input or output capturing tutoring, explanations, alternatives, etc., command preferences, e.g. the command style, and the understanding of the input and output tasks depending on the level and capabilities.

Preferences for the design of dialogues capture dialogue properties such as the information load, flexibility, complexity, expressive power, initiated actions, consistency, feedback, etc., dialogue forms and styles such as conversation, question/answer form, input/act forms and various presentation styles, dialogue structuring, e.g. closed dialogues, open discussions, undirected querying, interrogation, etc., dialogue control on amount, quantity, relevance, representation, etc., and dialogue support, e.g. by tools.

**Polarity Profiles.** We may draw special attention to the polarity profile that is a part of the personality profile. The polarity profile governs the development of layout and partially also playout. So, we may classify users according to the scale in the following table. Six different categories as by the table below may be used for yhe characterisation of users. These categories may vary and further refined using facets.
<table>
<thead>
<tr>
<th>businesslike - lyrical</th>
<th>conventional - inventive</th>
</tr>
</thead>
<tbody>
<tr>
<td>unemotional - soulful</td>
<td>common - extreme</td>
</tr>
<tr>
<td>rational - sensitive</td>
<td>coated - cool</td>
</tr>
<tr>
<td>considerate - sensual</td>
<td>reliable - exceptional</td>
</tr>
<tr>
<td></td>
<td>conventional - bohemian</td>
</tr>
<tr>
<td>classical - modern</td>
<td>traditional - vanguard</td>
</tr>
<tr>
<td>discrete - noisy</td>
<td>old - young</td>
</tr>
<tr>
<td>ageless - modern</td>
<td>colorless - gaudy</td>
</tr>
<tr>
<td>calm - disquietingly</td>
<td>even-tempered - exciting</td>
</tr>
<tr>
<td>noncommittal - intrusive</td>
<td>familiar - uninformed</td>
</tr>
<tr>
<td></td>
<td>spiritless - lively</td>
</tr>
<tr>
<td>tough - tender</td>
<td>rustic - cosmopolitan</td>
</tr>
<tr>
<td>harsh - mawkish</td>
<td>natural - artificial</td>
</tr>
<tr>
<td>geometric - bloomy</td>
<td>coltish - serious</td>
</tr>
<tr>
<td>firm - mellow</td>
<td>simple - complex</td>
</tr>
<tr>
<td>robust - delicate</td>
<td>difficult - easy</td>
</tr>
<tr>
<td>angled - round</td>
<td>graceful - gracile</td>
</tr>
</tbody>
</table>

These polarity properties may be represented by Kiviat graphs. For instance, Figure 4 is commonly known for characterising people from the Bavaria region in Germany. This self-perception may be used for the development of style-guides or patterns for layout of web pages.

Fig. 4. The self-perception of the Bavarian

**Representation of User Profiles.** User profiles can also be described in a semi-formal way as follows:
User profile: (user profile name)
Education profile: (general description)
Education: (list of degrees)
Capabilities: (list of skills)
Knowledge: (description of knowledge in the application)
Work profile: (general description)
Task expertise: (description of knowledge)
Task experience: (positive and negative experience)
System experience: (experience with infrastructure planned)
Information profile: (information need)
Interaction profile: (interaction properties)
Personality profile: (general description)
General properties: (list of user properties)
Preferences: (list of input/output/dialogue preferences)
Polarity profile: (list of characteristics)
Derivable profiles: (profile description and enforcement)
Security profile: (access control and privacy)
Safety profile: (safety requirements)
Based On: (user goals)
Based On: (user types)

5.2 Actor Profiles

We group users by their information demand and requested content, their utilisation patterns, and their specific utilisation and context. The abstraction of a group of users is called an actor. Actors are characterized by profiles and portfolios. Same as for users actor profiles consist of the education, work and personality profiles. Profiles are used for the derivation of preferences. These preferences are used for adaptation of scenes to specific actors. We look at the actor portfolios separately in the next subsection.

Actors may act in the role of an end-user, an antagonist, or an inactive non-player character. The three categories are commonly described by their properties, their abilities, subjects of their actions, and by changes they are imposing.

The profile is also used for the derivation of the security profile of the actors. It is specified on the basis of tasks to be performed by the actors, roles they play, and actions that are permitted or forbidden. The security profile includes all aspects of security, i.e.

- **access control** limiting access only to authorised individuals by applying policies for identification, authorization, and authentication,
- **privacy support** by preventing unauthorized parties from obtaining sensitive information, e.g. through support of anonymity and confidentiality.

This profile may be extended by the safety profile.

An actor takes part in some scenes of the storyboard. S/he is characterised by the character, location, actions, intention, and circumstances, i.e. what happened before s/he came there. He must discern the actions he is taking as well as the rhythms of the work as a whole, and he must determine what adjustments must be made in his/her storyboard for each of the other characters. Though actors are abstractions of user groups, we may characterize them by these abstract properties.
Example 12. Let us consider an edutainment site. Based on the profile we can derive a number of preferences of learners. The background knowledge leads to different speed and reception of content by the learner. Work abilities and habits influence current work. Learning styles have many many facets depending on the profile of the learner. The social environment with cultural and psychological differences, the support for treatment of history of the learning process without annoying repetitions, and the the content change management with or without refresh are mapped to preferences we apply to the storyboard.

The profiles of actors can be specified using the following template:

Actor profile: (actor profile name)
Grouping criteria: (characteristics of grouping of users)
Information demand: (general description)
Utilisation pattern: (general description)
Specific utilisation: (general description)
Actor context: (general description)

Profiles can also be used for derivation of quality requirements of users.

Example 13. For an e-commerce WIS we may derive a number of quality requirements depending on which kind of business the WIS is supporting. For instance, if the system is oriented towards selling after advertisement and presentation then actors that are acting as customers are interested in ease of use, speed, accessibility, previewing, support, community, safety, security, and guidance. If the WIS is supporting configuration sale, e.g. for travel, cars then customers are interested in customization, feedback, flexibility, previewing, background, contracts, safety, security, privacy, and guidance. If the WIS is supporting direct sale or reselling, e.g. auctions, first-in-first-out sale, or remnant sale then the quality criteria that should be supported are ease of use, accessibility, preview, fun/community, alert, and privacy.

5.3 Actor Portfolios

A portfolio is determined by the responsibilities one has and is based on a number of targets. The actor portfolio within an application is thus based on

– a set of tasks a actor has or intents to complete and for which solution the actor has the authority and control,
– a description of involvement within the task solution, and
– a collaboration that is necessary for solving the task.

Task modelling means to understand what a user wants to accomplish while visiting the WIS. At the same time, task analysis may lead to a reorganisation of the work processes to be supported. The WIS should support streamlining and augmenting what users do. Task analysis leads to a description of things users do and those they need to know. It does not specify how the task is accomplished. The tasks supported by a WIS need to be representative for the application, important within the application, and completely supported. Task support can be tailored depending on the profile and the context of the actors.
**Tasks.** A task is a usually assigned piece of work, which often has to be finished within a certain time by a user or a set of users whose duty is its completion. It implies work imposed by a user in authority and obligation or responsibility to perform. A task may consist of subtasks, so we assume that tasks can be constructed on the basis of elementary tasks. Thus, a task is characterized by:

**Problem statement:** Tasks are associated to problems, for which often a class of solution strategies is provided. Additionally, problems often require collaboration with the local and global systems and with other actors.

**Target states:** After successfully completing a task we may observe a change in the state. Target states are specified by means of target conditions. Some of the target conditions can be optional. If no optional conditions are given then all conditions are obligatory. Target states correspond to intents.

**Initial states:** The necessity for tasks enactment is based on the insufficiency of the current state of affairs. Additionally, task enactment conditions may specify, under which circumstances we can start task execution.

**Profile presupposed for task completion:** The completion of a task requires skills, experience and knowledge that must be presupposed by the user, whenever the task is going to be activated. Tasks may be embedded into a certain organizational context. The profile also presupposes a certain technical environment, e.g. communication, information, and workspace systems.

**Instruments for task completion:** Task enactment is supported by instruments such as actions and data. Problems are solved on the basis of an information demand and within a class of functions that might be used for task solution. Leteron the information demand is mapped to database views or media objects. The function utilization is organized on the basis of workflows.

**Collaboration profile:** A task may be allocated to a single user or a group of collaborating users. In the latter case, the subtasks for each user, the obligations, the time and other restrictions must be given.

**Auxiliary conditions:** The settling of tasks may be restricted. Typical auxiliary conditions are based on rights for the direct handling and retrieval, roles of the antagonist and the protagonist, and obligations required whenever settling a task.

We extend tasks by a general task context that consists of the releasing actor(s), the actors to which task completion and results of completion are reported, the organisational unit that calls for task completion, the activities that are required by users, the aids that can be used for task completion, and the workspace that is supporting task completion. The latter is mapped to session support or to temporary workplaces of users.

Tasks are associated with intents a user or a group of users have. The intents are either objects or targets [9]. In the latter case, we can directly map the task specification to corresponding targets.

**Example 14.** In a learning WIS for data mining learners have very different skills, very different background knowledge, very different approaches to learning, and want to use learning systems whenever this is really necessary. Let us consider tasks to understand the essentials of data mining and seeks for an algorithm that serves his needs.

1. The learner searches first algorithms that might be applicable within the application under consideration.
2. Next the user needs to understand the algorithm that seems to be the most appropriate. Understanding also includes learning the interpretation of results obtained by applying one of the algorithms to the learners data. For that, the learner may first look over demonstrations and illustrations for the chosen algorithm.

3. Now the learner experiments with the data. Data are selected and configured for the algorithm. The algorithm is executed and results are obtained.

4. Finally, the results are explained to the learner. S/he may now continue by selecting another method or algorithm and obtain additional insight into the data to be analyzed.

We may map these subtasks to the intents: exploring data mining algorithms, understanding how to prepare data, learning to interpret results obtained.

At the same time this task may be required by a life case [9]. For instance, the learner works as a clerk in a bank and tries to figure out which loans have become faulty and what is the reason for it. In this case, the clerk becomes a learner and tries to learn data mining as much as it is needed for the life case.

**Task Execution.** The task execution model defines what, when, how, by whom, and with which data work has to be done:

- Task activities define how the work is actually done, e.g. by instructing the user or by invoking an application.
- The control flow defines the sequence of activity execution. Depending on the specific WIS several control constructs are available such as decision, alternative, parallelism, loops, start/exit conditions, and different kinds of constraints such as deadlines.
- The data flow specifies how data flows through the task. Depending on the WIS different concepts exist such as input/output-container of activities or local/global variables.

The result of execution is the state of affairs reached and the satisfaction of target conditions. Task enactment is supported by pre-determined plans. These plans are scenarios which have a number of stories they are supporting. We may use blackboards for recording open targets. Whenever a target is completed and a task is considered to be successfully fulfilled then we delete this task from the blackboard. The involvement of actors within the task solutions is based on the specification of the role an actor plays, the part the actor plays within the scenario, and the rights and obligations an actor has within the given role.

The role specifies the behaviour expected of an actor. A role is a comprehensive pattern of behaviour and serves as a strategy for coping with recurrent situations and dealing with the roles of others. A role remains relatively stable, even though different users occupy the position. A user may have a unique style of role execution, but this is exhibited within the boundaries of the expected behaviour of the actor. Role expectations include both actions and qualities: for instance, in real life a teacher may be expected not only to deliver lectures, assign homework, and prepare examinations but also to be dedicated, concerned, honest, and responsible.

There are two types of roles: declarative and contextual ones. The former ones declare that an actor is playing a particular role, e.g. an actor being identified and as an employee. Contextual roles show how an actor acts within the context of a scenario and how an actor is involved within the context of another scenario. Declarative roles may be modelled by associating the actor to a role type. Contextual roles are modelled by associating an actor with
the work effort the actor is assigned to and a role type describing the involvement of the actor. The role type provides a description of the role and can be hierarchically structured. Roles may also be hierarchically structured. At the same time roles may be played in collaboration.

**Example 15.** In an e-business application we can distinguish roles such as worker, customer, and roles within the distribution scenario. The worker role entails roles such as contractor, contact, employee, and sponsor. The customer role takes part in bills to customer, ship to customer, and end-user customer. Within the distribution scenario we may distinguish active and pro-active roles, i.e. the distributor and an agent. These roles can again be distinguished into association, competitor, partner, prospect, referer, regulatory agency, shareholder, supplier, and website visitor.

At the same time several roles are played by people or organizations within the context of project management. A person or organization may be the sponsor, worker, manager, lead, advisor or quality assurance manager for a project.

Users usually occupy several positions, which may or may not be compatible with one another. So actors may play several roles at the same time. In this case, they must have a combined view of the application and a combined access to their capabilities. Therefore, the WIS must provide the ability to set-up and manage these roles, and to combine portfolios so that the application maintains to be consistent for each actor.

The part associates the atmosphere, mission, objectives, and objects with the involvement of the actor in a scenario. It is refined, when context of the involvement is taken into consideration. The part is based on a categorization of the behavior of users. We may derive that users tackling some problem do not like to wait. The part is described through the kind of interaction actors are preferring for the current tasks and the behaviour of actors we need to support. Therefore, the behavioral stereotype is described through the part.

A role entails certain obligations and rights. Obligations are obligatory tasks, conduct, service, or functions that arise from the users role under specified conditions. Rights are granted as a peculiar benefit, advantage, or favor to a role. We specify right and obligations as conditions that are mapped to conditions on actions an actor can call in a scenario. Obligations and rights include such for functionality, for collaboration, for content, and for dissemination of such to other actors. Obligations and rights may include also the specification of obligations, permissions and prohibitions in the case of exceptional situations.

**Collaboration.** Actors can collaborate for solving tasks. Such a collaboration is specified on the basis of the storyboard, the style of cooperation and the coordination facilities, and the roles of the partners, their responsibilities, their rights, and the protocols they may rely on. Collaboration is separated into communication, coordination, and cooperation. Communication can be based on classical architectures supporting communicating systems. Cooperation is based on an explicit handling of the work processes, the work organization, and on working spaces for collaborating partners. Cooperation can be specified by storyboards. Coordination supports the consistency of work products, of work progress, and is supported by an explicitly specified contract that depends on intentions, on the users community and on the system supporting collaboration.

**Example 16.** Collaborative learning is a teaching strategy in which small learning groups of learners, each of different levels of ability, use a variety of learning activities to improve
their understanding of a subject. Each member of a learning group is responsible not only for learning what is taught, but also for helping group members to learn, thus creating an atmosphere of joint achievement. Learners may apply the knowledge they have obtained during learning steps. Each student may act in different groups in three different roles: the role of an exercise team member, the role of an advisor or expert and the role of an evaluator, who marks and grades the results of the team. The collaborating parties have their own cooperation, communication and coordination space. For instance, the evaluator and the teams share a group communication place and store their results in an answer delivery box. The evaluator delivers assignments to exercise teams through the assignment box.

**Actor Portfolios and Life Cases.** Portfolios refine the specification of life cases, which are collected by observing real life situations and the data and action flow in them. The life case description allows to deduct the demands for data, functions, performance, and supporting aids such as workspace and workplace support. The workspace is determined by the portfolio, its support by the profile of the actors involved.

*Example 17.* Let us consider a life case relocation. The portfolio for this life case is based on a number of tasks such as: change basic data of issuer, provide a view on the data to all other interested parties, initiate tasks of associated life cases, archive old data and current changes, handle exceptional tasks, and change data in official documents. The life case is complex due to the large variety of subtasks, the initiation of which depends on the issuer. These tasks are associated with subtasks such as checking, whether all necessary documents have been supplied, special handling must be applied, etc.

At the same time we need to consider the involvement of at least four different actors. The issuer raises the change of data task handled by civil servants. If the issuer raises exceptions for data transfer then special support is required by an actor that acts as a data protection official. Finally, the data on the relocation are delivered to a number of actors that play the role of official bodies or support organizations. The security profiles of the issuer and the civil servants require a sophisticated data management, e.g. tax and social security data must be kept in isolation from each other.

The mindmap in Figure 5 summarises a part of the life case relocation, the tasks related to basic changes, to changes of associated parties, to change of data ownership, and to tasks of associated life cases. The actors participating in the life case are restricted to the four main types of actors. Their subtasks are different from the tasks of the issuer.

During analysis of user behaviour a number of principles needs to be observed. These principles form the *portfolio restrictions.*

**Non-determinism of user behaviour:** We need to cope with all possible options that are plausible. We do not assert that an option for execution will be followed. The various user intents are stored on a blackboard that is used to record tasks and subtasks to be completed.

**Intention-based behaviour:** We may assume that a user intermittently terminate actions as soon as their aims and their targets have been achieved. A typical implementation support for this principle is realized through the submit button. After submitting data or after hitting the submit button the user starts a new scenario or another scene under the assumption that the current intention has been achieved.
**Task-oriented termination:** We may assume that a user will terminate the current interaction when their whole task has been achieved. In achieving a target, subsidiary tasks might be generated. Termination is based on termination conditions. For instance, the user of an ATM machine expects that with termination the bank card is returned. Therefore, the user will have the card after termination of the ATM scenario.

**Reactive behaviour:** Users of a WIS are reacting to stimuli or messages of the WIS the user is observing. Rather than specifying each reaction in a separated form we use generic functions for the specification of this behaviour. Reactive behaviour can be modelled on the basis of observation-reaction pairs.

**Collaboration target behaviour:** The user enters an interaction with the knowledge of the task dependent on the targets that must be discharged. The collaboration target is based on a pre-determined plan that is known in advance to the user. Once a target has been achieved the user does not expect to need to repeat the actions again.

**Separation of mental and physical actions:** Whenever the user commits to taking a physical action, then this action cannot be revoked after a certain point. Before doing the physical action the user generates the signal sent from the brain to the motoric system. We supply the user model by a guard-action pair linking mental actions with physical actions.

**No-option-based completion:** A user may abort a scenario when there is no apparent action the user can take that would help to complete the task. If e.g. the user of a railway ticketing system does not find the appropriate option, then the user might give up and

![Fig. 5. The life case relocation with basic tasks, associated tasks, and actors](image-url)
might assume that the intentions cannot be achieved. We model this opportunity by a no-option-based addendum that is added to each scene where a user may leave the story.

Relevance: A user chooses an action only, if this action seems to be relevant for achieving the currently considered targets.

These principles can be mapped to control rules of the storyboard. For instance, goal-based behaviour is mapped to acceptance conditions of scenes. The intentional non-determinism is reflected in non-deterministic scenarios. Task-oriented termination is mapped to invariants on stories, scenes and scenarios. Reactive behaviour is represented through the transitions between scenes. The user moves with an action from one scene to the next. To do so the action must be released by the system. We use collaboration targets for the derivation of actions that must be performed by the user. The necessity of actions can be modelled on the basis of deontic logics as described in the previous chapters. They are modelled through action guards. After a target has been achieved it is removed from the task blackboard. The guard-action pairs linking mental and physical actions are also used for refinement of the storyboard. For instance, if a user is not reacting within a certain time frame, then we may assume that the current guard-action is not valid and that the user needs action support or action feedback. The no-option-based addendum is added to each scene as a default addition. We may refine this addendum by actions that a user uses randomly after becoming confused. A possible refinement of the addendum is the incorporation of a wait function. Relevance is a quality criteria we apply during check of each scene. Only relevant actions may be fired by a user.

Representation of Portfolios. Portfolios can be specified in a semi-formal way using the following template:
<table>
<thead>
<tr>
<th>Actor portfolio:</th>
<th>(actor portfolio name)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task:</td>
<td>(general description)</td>
</tr>
<tr>
<td>Extension of:</td>
<td>(General characterisation of tasks)</td>
</tr>
<tr>
<td>Characterisation:</td>
<td>(general description)</td>
</tr>
<tr>
<td>Initial state:</td>
<td>(characterisation of the initial state)</td>
</tr>
<tr>
<td>Target state:</td>
<td>(characterisation of the target state)</td>
</tr>
<tr>
<td>Profile:</td>
<td>(profile presupposed for solution)</td>
</tr>
<tr>
<td>Instruments:</td>
<td>(list of instruments for solution)</td>
</tr>
<tr>
<td>Collaboration:</td>
<td>(specification of collaboration required)</td>
</tr>
<tr>
<td>Auxiliary:</td>
<td>(list of auxiliary conditions)</td>
</tr>
<tr>
<td>Execution:</td>
<td>(list of activities, control, data)</td>
</tr>
<tr>
<td>Result:</td>
<td>(final state, satisfied target conditions)</td>
</tr>
<tr>
<td>Actors involvement:</td>
<td>(general description)</td>
</tr>
<tr>
<td>Role:</td>
<td>(description of role)</td>
</tr>
<tr>
<td>Part:</td>
<td>(behavioural categories and stereotypes)</td>
</tr>
<tr>
<td>Collaboration:</td>
<td>(general description)</td>
</tr>
<tr>
<td>Communication:</td>
<td>(protocols and exchange)</td>
</tr>
<tr>
<td>Coordination:</td>
<td>(contracts and enforcement)</td>
</tr>
<tr>
<td>Cooperation:</td>
<td>(flow of work)</td>
</tr>
<tr>
<td>Restrictions:</td>
<td>(general description)</td>
</tr>
<tr>
<td>Actor restrictions:</td>
<td>(general description)</td>
</tr>
<tr>
<td>Environment:</td>
<td>(general description)</td>
</tr>
<tr>
<td>Based On:</td>
<td>(life cases)</td>
</tr>
<tr>
<td>Based On:</td>
<td>(intentions)</td>
</tr>
<tr>
<td>Based On:</td>
<td>(general tasks, audience, mission, goal)</td>
</tr>
</tbody>
</table>

**Portfolio Context.** By explicit specification of the *portfolio context* we have a chance to cope with “errors” a user can make. Typical examples are the unnecessary repetition of an action, jumping back by the using the back button of the browser, reversing the order of actions, omitting or delaying actions, replacing one action by another, executing an additional action that is not required or unrelated to the current targets. As it is infeasible to support users in avoiding all these “errors”, we specify the context together with the portfolio.

The context is used for the development of production rules for WIS implementation. A typical list used in some of our projects consists of:

**Rules for blackboard control:** The scenario is completed whenever the blackboard of tasks becomes empty. The scene data may be logged or removed.

**Rules for information on actions:** Actions that are not initiated by the user but are of interest to him/her need to be reported to the user.

**Rules for control flow reporting:** Users need a clear indication, if the order of actions can be initiated by the user is not arbitrary. For example, a common practice is the utilization of a ‘next’ button. If the user does not know what that means then s/he is left in confusion.

**Rules for control flow within scenarios:** Although a user can choose among several options for actions we need to ensure that options are narrowed down to those that lead to meaningful stories. So, options that are available to the user are reduced at run-time.
Refinement of Life Cases. Profiles and portfolios are used for the refinement of life cases. So far, life cases are specified by a general characterisation, the life case flow, figures, context, and requirements. With the availability of the user model, i.e. the profiles and portfolio of users and actors, we may extend the life case specification by:

Tasks associated with the life case: Life cases are normally not raised in isolation, but are associated with tasks that correspond to problems an actor intends to solve. These tasks have their own dependencies. So, we can associate with a life case other corresponding life cases. This association is mainly used for an extension of the life case under consideration.

Involvement of other actors and collaboration with them: The life case flow is refined by the cooperation specification of all collaborating actors. The involvement includes roles, rights, and obligations. Each actor plays a certain part that is restricted by time, data, processes, and specific restrictions of the portfolio.

Restrictions due to the profiles of participating actors: Due to the profiles of actors life cases are extended in order to support their specific requirements. The personality profiles are used for the generation of requirements according to the atmosphere of the WIS. Security profiles restrict the involvement of actors.

Context specialisation: The general context specified for a life case can be adapted according to the portfolio and the profiles of participating actors.

Example 18. The life case relocation can be refined using a portfolio that is specific for the state Schleswig-Holstein or for a city such as Cottbus. If there are associated life cases or special cases, then another set of tasks is raised and the issuer is send to a number of offices. The city information system of Cottbus supports such life cases, e.g. the one depicted in Figure 5. In this case, the life case relocation is extended by all its associated life cases. In Cottbus one agency is handling all tasks for the issuer. So, the issuer uses the collaboration with the city inhabitants agency. In the background this agency collaborates with all other agencies and recipients.

The life case relocation can be raised by any citizen. So, the profiles are rather general. We cannot expect that an issuer has a specific education profile, but we may expect that the issuer has a personality profile. This can be used for the life case flow, e.g. for checking first the existence of documents that are necessary and then raising the basic changes.

This refinement of life cases is reflected in an extension of the life case template:

- Life case: \( \{ \text{life case name} \} \)
- Tasks: \( \{ \text{list of tasks associated with the life case} \} \)
- Actors involvement: \( \{ \text{description of actors involvement} \} \)
- Profile restrictions: \( \{ \text{list of restrictions due to actors profile} \} \)
- Context specialisation: \( \{ \text{context embedding} \} \)

Refinement of life cases permits to derive requirements for the WIS development. These requirements support the designer by providing an input information to sketch the detailed content structure, the interface frames and grids, and the main functionality. Content developers in turn can define the main content demand needed for the WIS. The information we gain will impact on the content the WIS is going to provide, the structuring of this content, the access paths to content, the navigation, the presentation, the user operation, the WIS
operation, and the interaction. Furthermore, these life cases may be used as a rich source for usability evaluation. We may also use these refinements for discussing non-functional requirements, which are usually expressed on the basis of user expectations.

The actor profile and the actor portfolio are also used for the derivation of requirements to the WIS.

**Workspace:** Actors need their own workspace in order to complete their tasks. This workspace may be provided temporarily, can be partially archived, and can be used for other tasks in the portfolio of the actor. Depending on the involvement of actors who collaborate to solve a task the workspace can also be accessible to other actors.

**Session support:** Actors are supported by session-defined media objects, i.e. certain content and functionality as defined later in part III. These media objects are generated whenever tasks are started, and may be adapted to the profile of the actors.

**Collaboration infrastructure:** Collaboration is based on communication, coordination, and cooperation of involved actors. In order to support collaboration an infrastructure is required that is extended or shrunked when new tasks are started or tasks are completed.

*Example 19.* The workspaces for the life case *relocation* depicted in Figure 5 are different for the actors. The issuer does not need any workspace. The agency actors have a workspace in which a number of tasks may be bundled. Their workspace can be organized as a blackboard. The actor supporting data protection may use a completely different workspace that is based on monitor techniques. The issuer is supported by session objects that persist until all tasks of the life case are completed. Using this session support the issuer can resume the tasks at any time, e.g. by providing new data. The actors of the agencies collaborate in accordance with the legal restrictions and regulations.

6 Determination of Context

Taking the commonly accepted meaning a context characterises the situation in which a user finds him/herself at a certain time in a particular location. In this sense context is usually defined only statically referring to the content of a database. Only very few attempts have been made so far to consider context of scenarios or stories.

More generally, we consider context as everything that surrounds a utilisation situation of a WIS by a user and can throw light on its meaning. Therefore, context is characterised by interrelated conditions for the existence and occurrence of the utilisation situation such as the external environment, the internal state, location, time, history, etc. For WISs we need to handle the mental context that is based on the profile of the actor or user, the storyboard context that is based on the story leading to a situation, the data context that is based on the available data, the stakeholder context, and the collaboration context. These different kinds of contexts have an influence on the development of the storyboard and must thus be considered for the development of the WIS.

*Example 20.* Let us consider a travel information system. It is often desirable to resolve the context of utterances. While booking an airline ticket to *London* the user may be asked for the airport code, for which s/he has a choice between LGW (London Gatwick), LHR (London Heathrow), LON (all London airports in UK), STN (London Stansted), and YXU (London,
Ontario, Canada). The context of the travel request can be used to exclude the last option.
The context of the airline used so far can be used to exclude two of the others. This context
injection is based on the story environment and on content data.

**Context Space.** When determining context we already know the major life cases we would
like to support, the intentions associated with the WIS, the user and actor characterisation
on the basis of profiles and portfolios, and the technical environment we are going to use.
These restrictions enable a more refined understanding of context within a WIS.

The work in [?] characterises a WIS by six intertwined dimensions, one of which is context.
We now relate context to the other dimensions, i.e. to the intentions, the usage, the content,
the functionality, and the presentation. As presentation resides on a lower level of abstraction,
it does not have an impact on context. Content and functionality will be used for context
refinement, which we address later. So, we first concentrate on intention and usage. The user
model, the specified set of life cases, and the intention can be used for a disambiguation of
the meaning and an injection of context. In doing so we distinguish the following facets of
context:

**Actor context:** The WIS is used by actors for a number of tasks in a variety of involvements
and well understood collaboration. These actors impose their quality requirements on
the WIS usage as described by their security and privacy profiles. They need additional
auxiliary data and auxiliary functions. The variability of use is restricted by the actor's
context, which covers the actor's specific tasks and specific data and function demand, and
by chosen involvement, while the profile of actors imposes exceptions. The involvement
and collaboration of actors is based on assumptions of social behaviour and restrictions
due to organisational decisions. These assumptions and restrictions are components of the
actor's context.

**Storyboard context:** The meaning of content and functionality to users depends on the
stories, which are based on scenarios that reflect life cases and the portfolios of users or
actors. According to the profile of these users a number of quality requirements such as
privacy, security and availability must be satisfied. The actor's scenario context describes
what the actor needs to understand in order to efficiently and effectively solve his/her
tasks in the actual portfolio. The actor's determine the policy for following particular
stories.

**System context:** The WIS is developed to support a number of intentions. The purposes
and intents lead to a number of decisions on the WIS architecture, the technical envi-
ronment, and the implementation. The WIS architecture has an impact on its utilisation,
which often is only implicit and thus leads to not understandable systems behaviour. The
technical environment restricts the user due to restrictions imposed by server, channel and
client properties. Adaptation to the current environment is defined as context adaptation
to the current channel, to the client infrastructure and to the server load. At the same
time a number of legal decisions based on regulations, laws and business rules have been
incorporated into the WIS.

**Temporal context:** The utilisation of a scene by an actor depends on his/her history of
utilisation. Actors may interrupt and resume their activities at any moment of time. As
they may not be interested in repeating all previous actions they have already successfully
completed, the temporal context must be taken into account. Due to availability of content
and functionality the current utilisation may lead to a different story within the same scenario.

We will discuss these various facets of context in more detail later in this section. This entire information forms the context space, which brings together the storyboard specification and the contextual information. Typical questions that are answered on the basis of the context space are:

- What content is required by the context space?
- What functionality is required by the context space?
- What has to be changed for the life cases, the storyboard, etc., if context is considered?

As outlined above the context space is determined by the actors, the scenarios, the WIS itself, and the time. It leads to a specialisation of the content, structuring and functionality of the scenes. We will discuss this specialisation in the last part of this chapter.

Context is associated with desirable properties of the WIS such as quality criteria and security and privacy requirements. Quality criteria such as suitability for the users or learnability provide obligations for the WIS development process. Though these criteria are rather fuzzy, they lead directly to a number of implementation obligations that must be fulfilled at later stages, i.e. within the development on the implementation layer.

For instance, learnability means comprehensibility, i.e. the WIS must be easy to use, remember, capture and forecast. This requires clarity of the visual representation, predictability, directness and intuitiveness. These properties allow the user to concentrate on the tasks. The workflows and the discourse structure correspond to the expectations of the users and do not lead to surprising situations. They can be based on metaphors and motives taken from the application domain. In the same way other quality criteria can also be mapped to development obligations.

Other properties that may be associated with context refer to the potential utilisation for other tasks outside the scope of the storyboard. In this case we do not integrate the additional tasks into the storyboard, but instead support these tasks, if this in accordance with our intentions. For instance, we might expect further visits targeting at core concerns of the WIS.

*Example 21.* Sometimes customers may want to use a WIS for a purpose that does not meet the system’s mission statement. For example, a customer may use a banking WIS to learn about the loan business, or a bookshop WIS to learn English. Clearly, the larger the gap between the actual customer’s intention and the system’s mission statement is, the higher the expected costs will be for supporting such customers. If it can be expected that some customers will interact with the WIS in a ‘non-standard’ way, a decision has to be made whether to support such intentions or not. This implies a modification of the anticipated information space. It shows that our focus on a business model for context modelling is not always a severe restriction.

We may consider three additional context facets:

**Provider context:** Providers are characterised by their mission, intentions, and specific policies. Additionally, terms of business may be added. Vendors need to understand how to run the WIS economically. Typical parts of this context are intentions of the provider,
themes of the website, mission or corporate identity of the site, and occasion and purpose of the visits of actors. Thus, providers may require additional content and functionality due to their mission and policy. They may apply their terms of business and may require a specific layout and playout.

Based on this information, the WIS is extended by provider-specific content and functionality. The storyboard may be altered according to the intentions of the provider, and life cases may be extended or partially supported. Provider-based changes to portfolios are typical for WISs in e-government and e-business applications.

**Developer context:** The WIS implementation depends on the capability of the developer. Typically we need to take into account the potential environment, e.g. hard- and software, communication channels, the information systems that are to be incorporated, especially the associated databases, and the programming environment developers use.

**Organisational and social context:** The organisation of task solutions is often already predetermined by the application domain. It follows organisational structures within the institutions involved. We captured a part of these structures already on the basis of the portfolio and modelled it by collaboration. The other parts form the organisational context. Collaboration of partners consists of communication, coordination, and cooperation. Cooperation is based on cooperativity, i.e. the disposition to act in a way that is best helpful for the collaboration partners, taking their intentions, tasks, interests and abilities into account. At the same time, collaboration is established in order to achieve a common goal. Actors choose their actions and organise them such that their chances of success are optimised with respect to the portfolio they are engaged in. Additionally, the social context may be taken into account, which consists of interactive and reactive pressures. Typical social enhancements are socially indicated situations such as welcome greetings, thanking, apologising, and farewell greetings.

**Actor Context.** Let us next take a deeper look into the facets of the context space, i.e. examining actor, storyboard, system and temporal context in more detail. The context of an actor is based on his/her intentions. According to the actor's profile s/he needs support to fulfil the expectations with respect to the quality of information and work. The social and intellectual interests of the actor may also be part of the actor's context. The actor's profile may be used for a refinement of the actor's context leading to the following four specific kinds of context:

**Actor projection context:** Actors may act on their expectations. In this case, they intentionally drop portions of content or functionality and project the current content and functionality to the "normal" case. This projection leads to an implicit context. For instance, within a travel scenario actors are expected to behave like travellers. Another kind of projection is parameter suppression, in which case content or functionality may be dropped or is not noticed whenever it becomes partially irrelevant.

**Actor approximation context:** Often actors need first a condensed or approximated information that may be refined later. Typical such approximations are attribute value approximations or structural approximations. For instance, the former ones may allow the WIS to provide first an approximate value for the orientation of the user. A common misuse of approximation is pricing by "starting from". Structural approximation permits the use of the same symbol for the original object and an abstraction, hence enables the usage of simpler representations.
**Actor ambiguity context:** Sometimes the reference of a symbol can be unambiguous within a narrow scope, in which certain limitations apply, but ambiguous in a larger scope without the limitations. A typical unambiguous symbol is the ‘next’ button in case the next scene lies within the expectations of the actor. Another use of ambiguity can be made by choosing less expressive textual representations. For instance, in a loan application there is no need to clarify that the word ‘bank’ denotes a financial institution.

**Actor mental context:** The mental context captures attitudes and knowledge of actors or other kinds of alternative states of affairs such as fiction and user expectations. This context is described in terms of provenance, i.e. relating to real life cases or to expected life cases. Expectations of actors or users can be combined with other more general requirements. The knowledge of the mental context will remain highly incomplete. However, it provides a handle for incorporating users’ and actors’ expectations.

The actor context is intellectual as well as existential. It contributes to enabling the scenarios and the corresponding stories under consideration. The intellectual part is based on the profile of the actor, on habits, traditions, knowledge, experience, etc. It may be also based on the quality requirements an actor is imposing. The actor context restricts the users that might use the WIS, the way the system will be used, and the portfolios. It is based on the intentions for using the system and the portfolio of the actor, e.g. tasks, involvement, and the collaborations the actor is involved in. The existential part is also related to the portfolio under consideration. It is related to the data and functions currently available or provided, and the technical environment.

The specification of this specific actor context becomes necessary whenever we want to support the work of actors that is close to human communication. Human communication exploits the context often to an extreme degree, leaving many things implicit. We do not need a complete decontextualisation as long as the actor can interpret the content and functionality that is provided by the WIS. Contexts provide a mechanism by which we can use the simplest presentation, content and functionality, i.e. the ones that makes the fewest distinctions most of the time while transcending to more expressive presentation, content and functionality only when needed.

Due to these contextual abilities we may restrict presentation, content and functionality to those features that are absolutely necessary. These restrictions may result in presentation principles such as sparse utilisation of additional and not directly necessary content or functionality or economy in utilisation of colours, multimedia objects, and texture.

*Example 22.* Let use Example 18 for an illustration of this principles. An issuer of the relocation life case expects that his personal and identification data are already sufficient for providing him/her all necessary details. So, the context in which the issuer reacts is based on projection and ambiguity context. If we use the information the passport office provides as public information for the city office, then we can adapt the life case directly to the current one. At the same time, the visit of the issuer might be not the first such in his/her life. So, we can now use the information on previous life cases for scaling the life case to the expectations the issuer has.

The adaptation requires some background knowledge on the handling of life cases in other cities, previous visits, and the profile of the issuer. We may then use a number of questions to figure out, which further adaption or refinement of the life case is applicable. Since some data on the issuer cannot be stored in the system due to regulations and laws we need to
repeatedly obtain these data. So, the data we need to capture within the life case are extended by data we need for figuring out which specific life case is under consideration. At the same time we may use this context information for adapting functionality that is provided.

This specific actor context is combined with the portfolio restrictions. Actors with a non-deterministic behaviour do not use high ambiguity or deep projection. At the same time, their mental context and their approximation context must be rather sophisticated. Actors acting more on intention intensively use all four kinds of actor contexts. Task-oriented and reactive behaviour requires support for mental context. Actors acting in collaborations need additional support for their common disambiguation. If actors do not complete their tasks within one session, they need a well-prepared projection context for the case that they resume their tasks. We shall later map these requirements to adaptation rules and control rules for adaptation.

**Storyboard Context.** Context has also a storyboard dimension. The actor’s context must be combined with the storyboard, life case and portfolio contexts. The latter two selectively condition the situational interest of the actor and the relevance of the current scene for the actor. Based on the relevance we may identify and use properly all the content that should exert the evolution of the current story. We may now use this information for extracting whether a sequence rule, i.e. a rule of the form $s_1 \Rightarrow s_2$ requiring that a visit of scene $s_1$ should be followed by a visit of scene $s_2$ can be applied to the current system usage. The rule may hold in general, but is considered to be not applicable if the existence of process $p_1$ leading to scene $s_1$ does not have a bearing on the existence of process $p_2$ leading to scene $s_2$. Therefore, the incorporation of context and the derivation of relevance has mainly to do with selecting the best story for the user and is thus used for the adaptation of content and functionality.

The storyboard context can be used for deriving the most appropriate content. We aim at delivering the right data to the right actor with the right tools and scope at the right time. As the storyboard context provides a good source for adaptation of content and functionality to the current stage of the scenario we collect context within the storyboard and add this context information to the context space. This context allows a treatment of the expectations of the actor. Therefore, each scene in a scenario is provided with a pre-scene context, scene context, and a post-scene context.

- The pre-scene context consists of all content that has already been delivered to the actor before the appearance at the actual scene. This information can be used to reduce content delivery for scenes. At the same time, this content can be stored in condensed form and made available to the actor when needed, i.e. the actor can revisit the old content whenever this seems to be necessary. Classical browsers only provide a strictly sequential ‘back’ button for this kind of history management. The pre-context of a scene thus contains all valuable content that is collected during a story, and guarantees the availability of this content when needed.
- The post-scene context consists of a potential playout of scenes that can be entered after the current scene. If an actor needs some information on the next actions, then this context information can be used. This information is valuable for those actors who intend to drop out of the system. It is also a part of the help information. The post-scene context can be enriched by meta-data describing the content that is provided in the next steps or the
data to be produced by the actor. In this case, an intelligent interface may forecast the information need for further steps of the storyboard.

- Each scene may also be enriched by superimposed meta-data on the scene, which include everything that could be referenced within the expected consumed and produced data. Typical such references are collaborating actors, retrieval or update data of the current content, and details taken from the log of the current story. Finally, the scene context may include administrative data such as identification of content currently under consideration. Scene context is enhanced by generic scene information, which can be based on intentions of the WIS. For instance, adverts may be attached to each of the scenes. Default information serves as an exception handling for scenes. If content or functions are currently not available, then default data are provided.

**System Context.** The system context is determined by the content and the functions that are provided by the web information system. It consists of at least the following four parts:

**Source and acquisition:** Source and acquisition is an orthogonal dimension of the WIS. A WIS is supported by media objects that belong to media types as we will explore in detail in part III. In a nutshell, a media object is defined by an extended view on some underlying database, which can then serve for provision of content and functionality of an elementary scene. The databases used for the generation of content form the context of the scenario. We may associate with each scenario the subschemata of these databases that are used for generation of consumed data or for integration of data produced by actors in the scenario.

**Associated content:** The data that are used for consumed and produced information do not exist in isolation. They are usually associated with other data on the basis of integrity constraints or existence constraints, in particular existence constraints are often not explicitly represented as such, but are embedded into the database schemata used. For instance, we usually associate with objects collected in relationship classes those objects collected in the component classes on which they are based. In this case, we assume that objects in component classes of the relationship type co-exist with objects in the relationship class. We need to consider the environment of content that is currently under consideration together with the data that are associated with this content.

**Supported functionality:** Functions supporting the actions in scenarios are provided by the WIS. These functions have their own control environment. Typical such control mechanisms are logging, concurrency control, and recovery management.

**Security:** Security concepts describe *encipherment* and *encryption* (keys, authentication, signatures, notarisation, routing control, access control, data integrity, and traffic padding) for data exchange.

**Temporal Context.** The temporal context appears in a number of variants, e.g. storage time, validity time, display time, user-defined time, transaction time, etc. The temporal context is applicable in a number of combinations. Sometimes it is necessary to use all of them, but often it is observed that only one variant of this context is necessary.

Versions show the life cycle of the objects under consideration. As scenarios will have their own life cycle we cannot assume that database changes are directly enforced on websites. Moreover, it may be useful to provide the old content as long as an actor continues with the same story. Versions can often be systematically structured by *database system phases*:
– The \textit{initialization phase} permits the development of objects storing initial information. Integrity constraints are applicable in a limited form.
– The \textit{production phase} is the central phase, which consists of runtime querying, modification, transaction management, etc.
– The \textit{maintenance phase} is used in productive database applications for clarification of soft constraints, maintenance of constraints that have been cut out from runtime maintenance, and changing the structuring and functionality of the entire database system. Maintenance phases are used in data warehouse applications for recharging the data warehouse with actual information.
– The \textit{archiving phase} is used for archiving the content of a database in a form that data relevant for historical information can be easily retrieved. No data modification is permitted; the only modification operation is to load new changes to the archive.

\textbf{Representation of Contexts.} We may now combine this context information using the following semi-formal template:

\begin{itemize}
    \item \textbf{Context:} \textit{(context name)}
    \item \textbf{Extension of:} \textit{(General context)}
    \item \textbf{Actor context:} \textit{(general description)}
    \item \textbf{Projection context:} \textit{(expectations)}
    \item \textbf{Approximation context:} \textit{(condensations and abstractions)}
    \item \textbf{Ambiguity context:} \textit{(scope)}
    \item \textbf{Mental state context:} \textit{(general description)}
    \item \textbf{Characterisation:} \textit{(general description)}
    \item \textbf{Storyboard context:} \textit{(general description)}
    \item \textbf{Pre-scene context:} \textit{(history of usage)}
    \item \textbf{Post-scene context:} \textit{(potential continuation)}
    \item \textbf{Scene context:} \textit{(superimposed meta-data)}
    \item \textbf{WIS context:} \textit{(general description)}
    \item \textbf{Source and acquisition:} \textit{(system environment)}
    \item \textbf{Associated content:} \textit{(content environment)}
    \item \textbf{Supported functionality:} \textit{(function environment)}
    \item \textbf{Security:} \textit{(required security functionality)}
    \item \textbf{Temporal context:} \textit{(general description)}
    \item \textbf{Versioning:} \textit{(general description)}
    \item \textbf{Development phase:} \textit{(general description)}
    \item \textbf{Provider context:} \textit{(general description)}
    \item \textbf{Developer context:} \textit{(general description)}
    \item \textbf{Organisational and social context:} \textit{(general description)}
    \item \textbf{Based On:} \textit{(life cases, portfolio)}
    \item \textbf{Based On:} \textit{(scenarios)}
    \item \textbf{Based On:} \textit{(general tasks, audience)}
    \item \textbf{Based On:} \textit{(mission, goals)}
\end{itemize}

\textbf{Lifting Relations.} Context evolves for actors, scenarios, systems, and over time. We model the relation between different contexts by \textit{lifting relations}. Properties that are valid for a
certain context may be lifted to another context. This transfer can be based on local model semantics.

For this recall that a context is determined by actor, storyboard, system and temporal contexts. So let $A$ denote the set of actors, $S$ the set of scenarios, $W$ the set of system characteristics, and $T$ the set of time units. Then we can take a subset $C \subseteq A \times S \times W \times T$ to represent a set of contexts. Furthermore, we use a family of contexts $\{ C_i \in C | i \in I \}$ and a family of statement sets (or theories) $\{ T_i | i \in I \}$ that are associated with these contexts. Of course, the theory $T_i$ describes the properties of the context $C_i$.

On these grounds we may use local models $M_{i,j}$ for each of these statement sets assuming that the models we consider are enumerated by the second index. More precisely, the models $M_{i,j}$ determine the meaning of content drawn from a language $L$ for describing content in view of context $C_i$. That is, we use a partial mapping $\Psi : L \times C \rightarrow M$, where $M$ denotes a set of pre-determined meanings for content in $L$.

We may now distinguish the formula $\alpha$ occurring context $C_i$ from the same formula occurring in another context by considering the context index $i$, i.e. we consider pairs $(\alpha, i)$. Lifting relations can be modelled by rules of the form

$$\frac{(\alpha_1, i_1) \ldots (\alpha_n, i_n)}{(\alpha, i)} \varphi$$

stating that the formulae $(\alpha_1, i_1) \ldots (\alpha_n, i_n)$ can be lifted to $(\alpha, i)$ under the side condition $\varphi$. In addition, a compatibility relation among local models is introduced similar to logics that capture possible world semantics. This compatibility relation is used for entailment and satisfiability. This approach allows us to reason locally and then to transfer the knowledge we gained to other contexts.

Based on this coarse clarification of basic notation we develop a number of facilities and extend the specification of the WIS:

**Context space:** The content context space is defined on the basis of the content $C$, scenarios $S$ and actors $A$. In Example 20 we could use information on the travel and on the airline to exclude options that seem to be less likely. The content context space of a WIS for a given content-meaning pair $c, m$ consists of precisely those contexts, under which the particular content will have that particular meaning, i.e.

$$C(c, m) = \{(a, s, w, t) \in C | \Psi(c, (a, s, w, t)) = m \}.$$ 

Adaptation of content, functionality, and scenarios to the context that is currently available is based on context infusion. Applying transformation rules we change content, functions, and the presentation. Therefore, we use a context specification for the development of enforcement rules. These rules may restrict scenarios to more specific ones, extend or shrink content, and extend or remove functions.

**Life case extension and specialisation:** The general life case specification can often be specialised, if context is explicitly injected. We need both the more general life cases and the contextualised ones. Whenever the WIS is revised or extended, we can return to more general life cases and generate another contextualisation. Typical specialisations concern changes in the life case flow. We may specialise the data consumed by an actor in dependence of the actor’s context. If we know that actors need special auxiliary information or conversely actors became more knowledgeable during the utilisation of the WIS, then we
may adapt the data provided for consumption. At the same time, we can specialise the figures according to the given context. In the same way spatial and temporal information provide a basis for refinement of life cases.

Life cases may be extended to requirements that were collected in the context space. The content context may require a more elaborated content to be provided. The supported functionality may require additional functions, content, or a specific presentation. Intentions may be more specific under consideration of context. For instance, if we want to support a certain usage of a WIS that was not originally intended but became important in order to maintain frequent visits, then the original life case is extended by those associated life cases.

**Development of a context manager:** Context is also bound to scenes and thus evolves within a story. We may expect that content enhances context. For this reason, we introduced the pre-scene context. Therefore, a subsystem that manages the context is needed. This context manager uses the lifting rules introduced above for transferring context to context for scenes, collaborating actors, and the WIS as such. The system also supports the rule-based development of logics over time. We cannot require that the rule system is complete, but it must be consistent. A useful property is commutativity, i.e. the results of firing rules does not depend on their order. The context management system enhances the dialogue management system by adapting and specialising the presentation and injecting context into it.

*Example 23.* A typical context extension to functionality is associated with the problem to avoid that users trap into losing-track situations. Such situation can be detected based on the user's behaviour, e.g. invoking the help function repeatedly on similar topics, repeatedly positioning on particular locations and performing similar operations on similar data, excessively navigating through information space without invoking any reasonable functionality, looking repeatedly for FAQs on similar topics, attempting to enter a discussion forum, and sending email to the site administrator.

User aid that can be provided for losing-track situations is giving access to a thesaurus of the subsystem the user is accessing. Furthermore, the respective business model may be exposed to the user together with an explanation that is adapted to a particular user type. Similarly, access to a FAQ list suitable for the user and the accessed subsystem may be given. Furthermore, improved search facilities and examples targeting at the subsystem accessed may be provided.

**Adaptivity.** The idea of adaptivity is to equip the system with enough additional information and rules that would render it possible to engender the right content and functionality for the current situation. That is, the system is supposed to act according to the dictum 'you take care of the specification, and the system will take care of itself and adapt to the current use'.

Two content objects \(c_1, c_2\) are *synonymous* in the context \(C_i \in \mathcal{C}\) iff \(\psi(c_1, C_i) = \psi(c_2, C_i)\). They are *totally synonymous* iff \(\psi(c_1, C_i) = \psi(c_2, C_i)\) holds for all contexts \(C_i \in \mathcal{C}\). They are *epistemically synonymous* within a scenario \(s\) for an actor \(a\) iff \(\psi(c_1, C_i) = \psi(c_2, C_i)\) holds for all contexts \(C_i \in \mathcal{C}\) associated with \(a\) and \(s\).

Applications often require adaptation of processing context, e.g. to

- actual environments such as client, server, and current communication channel,
Consider for instance e-learning or e-government websites discussed in [7] and [13]. Citizens may apply for a primary place of residence. In this case, their passport must be changed; otherwise, no change is required. Citizens with school-age children may have to complete additional documents. Completed documents may be decomposed into a suite of documents due to legal restrictions, e.g. by a data protection act requiring that data for city officials and service offices such as the unemployment agency must be separated.

Depending on the role of users, story completion may be scheduled sequentially for some users or in parallel for others. For instance, clerks in a city office may consider documents in parallel, while citizens complete their documents in a sequential mode.

Example 24. Adaptivity may be required at run-time. For instance, people with foreign citizenship may be required to apply for a residence permit. Users may require a varying support depending on the environment that is used for the completion of documents. Users should be supported whenever they are interrupted during task completion.

These requirements lead directly to the requirement to develop a facility for mutable, adaptable scenarios for different users, portfolios, and contexts. We shall return to this requirement after introducing templates in the next section. It is our target to develop generic scenarios that can be refined to scenarios by injecting context. This approach is more widely used for WISs than one would expect. For instance, almost all information sites of cities and regions provide a very similar hotel or event search. The reason is not the existence of a development monopoly but rather the evolution of these search facilities to semi-standards. These standards are not officially agreed, but have been formed by copying successful solutions.

7 Conclusion

to be complemented

References

9. Schewe, K.-D., and Thalheim, B. Life cases: An approach to address pragmatics in the design of web information systems. submitted for publication.
10. Schewe, K.-D., and Thalheim, B. User models: A contribution to pragmatics of web information systems design. submitted for publication.
Context Analysis: Toward Pragmatics of Web Information Systems Design

Hui Ma\(^1\) Klaus-Dieter Schewe\(^1\) Bernhard Thalheim\(^2\)

\(^1\)Massey University, Department of Information Systems & Information Science Research Centre Private Bag 11 222, Palmerston North, New Zealand, email: h.ma\(\text{[dot]}\)k.d.schewe\(\text{[dot]}\)massey.ac.nz

\(^2\)Christian Albrechts University Kiel, Department of Computer Science Olshausenstr. 40, 24098 Kiel, Germany, email: thalheim\(\text{[dot]}\)is\(\text{[dot]}\)informatik\(\text{[dot]}\)uni-kiel.de

Abstract

On a high level of abstraction a Web Information System (WIS) can be described by a storyboard, which in an abstract way specifies who will be using the system, in which way and for which goals. While syntax and semantics of storyboarding has been well explored, its pragmatics has not. This paper contributes context analysis as a step towards closing this gap. We classify various aspects of contexts related to actors, storyboard, system and time, which make up the context space, then analyse each of these aspects in detail. This is formally supported by lifting relations. Finally, we analyse how contexts impact on life cases, user models and the storyboard.

1 Introduction

A Web Information System (WIS) is an information system that can be accessed through the world-wide-web. On a high level of abstraction a WIS can be described by a storyboard (Schewe & Thalheim 2005b), which in an abstract way specifies who will be using the system, in which way and for which goals. In a nutshell, a storyboard consists of three parts:

- a story space, which itself consists of a hierarchy of labelled directed graphs called scenarios, one of which is the main scenario, whereas the others define the details of scenes, i.e. nodes in a higher scenario, and a plot that is specified by an assignment-free process, in which the basic actions correspond to the labels of edges in the scenarios,
- a set of actors, i.e. abstractions of user groups that are defined by roles, which determine obligations and rights, and user profiles, which determine user preferences,
- and a set of tasks that are associated with goals the users may have.

In addition, there are many constraints comprising static, dynamic and deontic constraints for pre- and postconditions, triggering and enabling events, rights and obligations of roles, preference rules for user types, and other dependencies on the plot. Details of storyboarding have been described in (Schewe & Thalheim 2005b). An overview of our method for the design of WIS was presented in (Schewe & Thalheim 2005a).

While syntax and semantics of storyboarding has been well explored, its pragmatics apart from the use of metaphors (Thalheim & Dusterhoft 2000) has not. Pragmatics is part of semiotics, which is concerned with the relationship between signs, semantic concepts and things of reality. This relationship may be pictured by the so-called semiotics triangle. Main branches of semiotics are syntactics, which is concerned with the syntax, i.e. the construction of the language, semantics, which is concerned with the interpretation of the words of the language, and pragmatics, which is concerned with the current use of utterances by the user and context of words for the user. Pragmatics permits the use of a variety of semantics depending on the user, the application and the technical environment. Most languages defined in Computer Science have a well-defined syntax; some of them possess a well-defined semantics; few of them use pragmatics through which the meaning might be different for different users.

Syntactics is often based on a constructive or generative approach: Given an alphabet and a set of constructors, the language is defined as the set of expressions that can be generated by the constructors. Constructions may be defined on the basis of grammatical rules.

Semantics of generative languages can be either defined by meta-linguistic semantics, e.g. used for defining the semantics of predicate logics, by procedural or referential semantics, e.g. operational semantics used for defining the semantics of programming languages, or by convention-based semantics used in linguistics. Semantics is often defined on the basis of a set of relational structures that correspond to the signature of the language.

Pragmatics has to be distinguished from pragmatism. Pragmatism means a practical approach to problems or affairs. According to Webster (Web 1991) pragmatism is the “balance between principles and practical usage”. Here we are concerned with pragmatics, which is based on the behaviour and demands of users, therefore depends on the understanding of users.

The six characteristics of WISs that were discussed in (Schewe & Thalheim 2005b) can be mapped to conceptual structures that are used for storyboard specification:

1. We start with the characteristics used for the strategic layer. Main specification elements used are intention and mission. They are mapped to metaphors, general goals, rhetorical figures, and patterns and grids of web pages discussed later.

2. The scenarios reflect the utilisation by actors, for which we envision a number of stories that correspond to real use. These scenarios may be captured through observation of reality. Story spaces and plots are recorded in various level of
3. Content specification is the basis for the media types, i.e. data types and their functions, which will be introduced in part III. It combines data specification with user requirements and is reflected in the content portfolio.

4. Functionality is provided by the media types as required by the storyboard. Typical standard functions are navigation, retrieval (search), support functions, and feedback facilities.

5. Context is based on tasks, history, and environment. We use the specification of context for restructuring and functionality enhancement, which will form the basis of XSL transformations and the onion approach (Binemann-Zdanowicz, Kaschek, Schewe & Thalheim 2004).

6. Presentation depends on the intention, the provider, the technical environment available and the users the WIS is targeting at. Presentation results in the layout and the playout of the WIS. Layout requires the development of multimedia presentations for each page. Playout additionally requires the development of functionality that supports visits of users depending on the story they are currently following to achieve their goals. Layout and playout integrate the chosen metaphors; they depend on chosen page patterns and grids as well as on quality requirements.

Conceptual structures and their association are depicted in Figure 1.1. We may separate the syntactics and pragmatics layers. Arrows are used for representing part-of- or uses- or relates-associations. For instance, the story is based on the user and the functions. Information metaphors relate content to information.

We use the notions of information and content in a specific manner. Information as processed by humans, is carried by data that is perceived or noticed, selected and organized by its receiver, because of his subjective human interests, originating from his instincts, feelings, experience, intuition, common sense, values, beliefs, personal knowledge, or wisdom, simultaneously processed by his cognitive and mental processes, and seamlessly integrated in his recallable knowledge. Content is complex and ready-to-use data. Content management systems are information systems that support extraction, storage and delivery of complex data. Content may be enhanced by concepts that specify the semantic meaning of content objects and by topics that specify the pragmatic understanding of users.

Therefore, information is directed towards pragmatics, whereas content may be considered to highlight the syntactical dimension. If content is enhanced by concepts and topics, then users are able to capture the meaning and the utilisation of the data they receive. In order to ease perception we use metaphors. Metaphors may be separated into those that support perception of information and into those that support usage or functionality.

Users are reflected by actors that are abstractions of groups of users. Pragmatics and syntactics share data and functions. The functionality is provided through functions and their representations. The web utilisation space depends on the technical environment of the user. It is specified through the layout and the playout. Layout places content on the basis of a data representation and in dependence of the technical environment. Playout is based on functionality and function representations, and depends on the technical environment.

The information transfer from a user A to a user B depends on the users A and B, their abilities to send and to receive the data, to observe the data, and to interpret the data. Let us formalise this process. Let \( s_X \) denote the function user by a user \( X \) for data extraction, transformation, and sending of data. Let \( r_X \) denote the corresponding function for data receiveal and transformation, and let \( \alpha_X \) denote the filtering or observation function. The data currently considered by \( X \) is denoted by \( D_X \). Finally, data filtered or observed must be interpreted by the user \( X \) and integrated into the knowledge \( K_X \) a user \( X \) has. Let us denote by \( i_X \) the binary function from data and knowledge to knowledge. By default, we extend the function \( i_X \) by the time \( t_X \) of the execution of the function.

Thus, the data transfer and information reception (or briefly information transfer) is formally expressed it by

\[
I_B = i_B(o_B(r_B(s_A(D_A))), K_B, t_X).
\]

Figure 1.1: The Web Utilization Space Based On the Characteristics of WIS

Figure 1.2: Dimensions of understanding messages
In addition, time of sending, receiving, observing, and interpreting can be taken into consideration. In this case we extend the above functions by a time argument. The function $s_X$ is executed at moment $t_{s_X}$ with $r_X(t_X)$, and $o_X$ at $t_{o_X}$. We assume $t_{s_X} \leq t_{r_B} \leq t_{o_B}$ for the time of sending data from $A$ to $B$. The time of a computation $f$ or data consideration $D$ is denoted by $t_f$ or $t_D$, respectively. In this extended case the information transfer is formally expressed it by

$$I_B = i_B(o_B(r_B(s_X(D_A,t_{s_X}),t_{r_B}),t_{o_B}),K_B,t_{o_B})$$

The notion of interpretation extends the dimensions of understanding of message displayed in Figure 1.2 to a web communication act that considers senders, receivers, their knowledge and experience. Figure 1.3 displays the multi-layering of communication, the influence of explicit knowledge and experience on the interpretation.

The communication act is specified by

the communication message with the content or content chunk, the characterisation of the relationship between sender and receiver, the data that are transferred and may lead to information or misinformation, and the presentation, the sender, the explicit knowledge the sender may use, and the experience the sender has, and the receiver, the explicit knowledge the receiver may use, and the experience the receiver has.

In this paper we approach the analysis of WIS usage as the first important part of storyboarding pragmatics. WIS usage analysis consists of three parts:

1. **Life cases** capture observations of user behaviour in reality. They can be used in a pragmatic way to specify the story space. The work on life cases was reported in a previous publication (Schewe & Thalheim 2007a).

2. **User models** complement life cases by specifying user and actor profiles, and actor portfolios. The actor portfolios are used to get a better understanding of the tasks associated with the WIS. The work on user models was reported in a previous publication (Schewe & Thalheim 2006).

3. **Contexts** complement life cases and user models by characterising the situation in which a user finds him/herself at a certain time in a particular location. We classify various aspects of contexts related to actors, storyboard, system and time, which make up the context space, then analyse each of these aspects in detail. This is formally support by lifting relations.

After a brief overview of the literature in Section 2 we approach the specification of contexts in Section 3. Formal aspects of contexts are then dealt with in Section 5. In Section 6 we conclude with a brief summary and a discussion of open issues.

## 2 Related Work

Storyboarding and also the preceding strategic modelling of WIS (Moritz, Schewe & Thalheim 2005) are unique to our approach to WIS modelling. Other approaches to WIS engineering such as the object oriented OOHDLM (Guell, Schwabe & Vilain 2000, Rossi, Schwabe & Lyardet 1999, Schwabe & Rossi 1998), WebML (Ceri, Fraternali, Bongio, Brambilla, Comai & Matera 2003), HERA (Houben, Barna, Frasincar & Vdovjak 2003) and variants of UML (Conallen 2003, Lowe, Henderson-Sellers & Gu 2002) concentrate on providing models of content, navigation and interaction by means of extended views, which in our own work is captured by so-called media types (Schewe & Thalheim 2005b). WSDM (De Troyer & Leune 1998) emphasises the additional need for a mission statement and a high-level description of processes and users in the WIS. Quite often high-level modelling of WISs is subject to UML-based methods, in particular variants of use cases. The rationale underlying our work is that this is far too little to capture strategic and usage issues of WISs.

The integration of goals and soft goals into the information systems development process has been proposed by (Mylopoulos, Fuxman & Giorgini 2000, Giorgini, Mylopoulos, Nicchiarelli & Sebastiani 2002). The detailed distinction of intentions as targets, objects, objectives and aims is based on linguistics (Web 1991). The integration of the temporal dimension is necessary because of the information systems context. The extension by the representational dimensions has been made in order to specify aspects of WISs.

Contextual modelling has found its own research community (Bouquet, Serafini, Bresilion, Benerecetti & Castellani 1999), but very little work has been done to integrate contextual modelling into WIS design. The work in (Mylopoulos & Motschnig-Pitrik 1995) uses contexts as an approach to modularise conceptual models. The most advanced attempt to modelling context is the work on contextual information bases (CIB) (Akaishi, Spyrratos & Tanaka 2002, Theodorakis, Analyti, Constantinopoulos & Spyrratos 1999, Theodorakis, Analyti, Constantinopoulos & Spyrratos 1999). Roughly speaking, a CIB-context associates objects with a name and an optional reference to another CIB-context. Thus, both the name and the reference depend on the usage history. As shown in (Kaschek, Schewe, Thalheim & Zhang 2004) CIB-contexts in a slightly generalised form can be combined with media types to provide a formal model of usage context. For this, instead of only associating merely a name with a location $L$ a location, i.e. a complex value, is associated with it.

## 3 Context Determination

Taking the commonly accepted meaning a context characterises the situation in which a user finds him/herself at a certain time in a particular location. In this sense context is usually defined only statically referring to the content of a database. Only very few attempts have been made so far to consider context of scenarios or stories.

More generally, we consider context as everything that surrounds a utilisation situation of a WIS by a user and can throw light on its meaning. Therefore, context is characterised by interrelated conditions for the existence and occurrence of the utilisation situation such as the external environment, the internal state, location, time, history, etc. The WIS is therefore required to handle the mental context that is based on the profile of the actor or user, the storyboard context that is based on the story leading to a situation, the data context that is based on the available data, the stakeholder context, and the collaboration context. These different kinds of contexts have an influence on the development of the storyboard and must thus be considered for the development of the WIS.

**Example 3.1** Let us consider a travel information system. It is often desirable to resolve the context of utterances. While booking an airline ticket to London...
3.1 Context Space

When determining context we already know the major life cases we would like to support, the intentions associated with the WIS, the user and actor characterisation on the basis of profiles and portfolios, and the technical environment we are going to use. These restrictions enable a more refined understanding of context within a WIS.

The context space is determined by six intertwined dimensions, one of which is context. The context of the travel request can be used to exclude the last option. The context of the airline used so far can be used to exclude two of the others. This context injection is based on the story environment and on content data.

System context: The WIS is developed to support a number of intentions. The purposes and intents lead to a number of decisions on the WIS architecture, the technical environment, and the implementation. The WIS architecture has an impact on its utilisation, which often is only implicit and thus leads to not understandable systems behaviour. The technical environment restricts the user due to restrictions imposed by server, channel and client properties. Adaptation to the current environment is defined as context adaptation to the current channel, to the client infrastructure and to the server load. At the same time a number of legal decisions based on regulations, laws and business rules have been incorporated into the WIS.

Temporal context: The utilisation of a scene by an actor depends on the actor’s context. Actors may interrupt and resume their activities at any moment of time. As they may not be interested in repeating all previous actions they have already successfully completed, the temporal context must be taken into account. Due to availability of content and functionality the current utilisation may lead to a different story within the same scenario.

We will discuss these various facets of context in more detail later in this section. This entire information forms the context space, which brings together the storyboard specification and the contextual information. Typical questions that are answered on the basis of the context space are:

- What content is required by the context space?
- What functionality is required by the context space?
- What has to be changed for the life cases, the storyboard, etc., if context is considered?

As outlined above the context space is determined by the actors, the scenarios, the WIS itself, and the time. It leads to a specialisation of the content, structuring and functionality of the scenes.

Context is associated with desirable properties of the WIS such as quality criteria and security and privacy requirements. Quality criteria such as suitability for the users or learnability provide obligations for the WIS development process. Though these criteria are rather fuzzy, they lead directly to a number of implementation obligations that must be fulfilled at later stages, i.e. within the development on the implementation layer.
For instance, learnability means comprehensibility, i.e., the WIS must be easy to use, remember, capture and forecast. This requires clarity of the visual representation, predictability, directness and intuitiveness. These properties allow the user to concentrate on the tasks. The workflows and the discourse structure correspond to the expectations of the users and do not lead to surprising situations. They can be based on metaphors and motives taken from the application domain. In the same way other quality criteria can also be mapped to development obligations.

Other properties that may be associated with context refer to the potential utilisation for other tasks outside the scope of the storyboard. In this case we do not integrate the additional tasks into the storyboard, but instead support these tasks, if this in accordance with our intentions. For instance, we might expect further visits targeting at core concerns of the WIS.

**Example 3.2** Sometimes customers may want to use a WIS for a purpose that does not meet the system’s mission statement. For example, a customer may use a banking WIS to learn about the loan business, or a bookshop WIS to learn English. Clearly, the larger the gap between the actual customer’s intention and the system’s mission statement is, the higher the expected costs will be for supporting such customers. If it can be expected that some customers will interact with the WIS in a ‘non-standard’ way, a decision has to be made whether to support such intentions or not. This implies a modification of the anticipated information space. It shows that our focus on a business model for context modelling is not always a severe restriction.

### 3.2 Additional Aspects

We may consider three additional context facets:

**Provider context:** Providers are characterised by their mission, intentions, and specific policies. Additionally, terms of business may be added. Vendors need to understand how to run the WIS economically. Typical parts of this context are intentions of the provider, themes of the website, mission or corporate identity of the site, and occasion and purpose of the visits of actors. Thus, providers may require additional content and functionality due to their mission and policy. They may apply their terms of business and may require a specific layout and playlist.

Based on this information, the WIS is extended by provider-specific content and functionality. The storyboard may be altered according to the intentions of the provider, and life cases may be extended or partially supported. Provider-based changes to portfolios are typical for WISs in e-government and e-business applications.

**Developer context:** The WIS implementation depends on the capability of the developer. Typically we need to take into account the potential environment, e.g., hard- and software, communication channels, the information systems that are to be incorporated, especially the associated databases, and the programming environment developers use.

**Organisational and social context:** The organisation of task solutions is often already predetermined by the application domain. It follows organisational structures within the institutions involved. We captured a part of these structures already on the basis of the portfolio and modelled it by collaboration. The other parts form the organisational context. Collaboration of partners consists of communication, coordination, and cooperation. Cooperation is based on cooperativity, i.e., the disposition to act in a way that is best helpful for the collaboration partners, taking their intentions, tasks, interests and abilities into account. At the same time, collaboration is established in order to achieve a common goal. Actors choose their actions and organise them such that their chances of success are optimised with respect to the portfolio they are engaged in. Additionally, the social context may be taken into account, which consists of interactive and reactive pressures. Typical social enhancements are socially indicated situations such as welcome greetings, thanking, apologising, and farewell greetings.

### 4 Details of Context Specifications

Let us next take a deeper look into the facets of the context space, i.e., examining actor, storyboard, system and temporal context in more detail.

#### 4.1 Actor Context

The context of an actor is based on his/her intentions. According to the actor’s profile s/he needs support to fulfill the expectations with respect to the quality of information and work. The social and intellectual interests of the actor may also be part of the actor’s context. The actor’s profile may be used for a refinement of the actor’s context leading to the following four specific kinds of context:

**Actor projection context:** Actors may act on their expectations. In this case, they intentionally drop portions of content or functionality and project the current content and functionality to the “normal” case. This projection leads to an implicit context. For instance, within a travel scenario actors are expected to behave like travellers. Another kind of projection is parameter suppression, in which case content or functionality may be dropped or is not noticed whenever it becomes partially irrelevant.

**Actor approximation context:** Often actors need first a condensed or approximated information that may be refined later. Typical such approximations are attribute values and structural approximations. For instance, the former ones may allow the WIS to provide first an approximate value for the orientation of the user. A common misuse of approximation is pricing by “starting from”. Structural approximation permits the use of the same symbol for the original object and an abstraction, hence enables the usage of simpler representations.

**Actor ambiguity context:** Sometimes the reference of a symbol can be unambiguous within a narrow scope, in which certain limitations apply, but ambiguous in a larger scope without the limitations. A typical unambiguous symbol is the ‘next’ button in case the next scene lies within the expectations of the actor. Another use of ambiguity can be made by choosing less expressive textual representations. For instance, in a loan application there is no need to clarify that the word ‘bank’ denotes a financial institution.

**Actor mental context:** The mental context captures attitudes and knowledge of actors or other
kinds of alternative states of affairs such as fiction and user expectations. This context is described in terms of proving or relating to real life cases or to expected life cases. Expectations of actors or users can be combined with other more general requirements. The knowledge of the mental context will remain highly incomplete. However, it provides a handle for incorporating users’ and actors’ expectations.

The actor context is intellectual as well as existential. It contributes to enabling the scenarios and the corresponding stories under consideration. The intellectual part is based on the profile of the actor, on habits, traditions, knowledge, experience, etc. It may also be based on the quality requirements an actor is imposing. The actor context restricts the users that might use the WIS, the way the system will be used, and the portfolios. It is based on the intentions for using the system and the portfolio of the actor, e.g. tasks, involvement, and collaborations the actor is involved in. The existential part is also related to the portfolio under consideration. It is related to the data and functions currently available or provided, and the technical environment.

The specification of this actor context becomes necessary whenever we want to support the work of actors that is close to human communication. Human communication exploits the context often to an extreme degree, leaving many things implicit. We do not need a complete decontextualisation as long as the actor can interpret the content and functionality that is provided by the WIS. Contexts provide a mechanism by which we can use the simplest presentation, content and functionality, i.e. the ones that makes the fewest distinctions most of the time while transcending to more expressive presentation, content and functionality only when needed.

Due to these contextual abilities we may restrict presentation, content and functionality to those features that are absolutely necessary. These restrictions may result in presentation principles such as sparse utilisation of additional and not directly necessary content or functionality or economy in utilisation of colours, multimedia objects, and texture.

**Example 4.1** Let us use relocation as an example for an illustration of this principles. An issuer of the relocation life case expects that his personal and identification data are already sufficient for providing him/her all necessary details. So, the context in which the issuer reacts is based on projection and ambiguity context. If we use the information the passport office provides as public information for the city office, then we can adapt the life case directly to the current one. At the same time, the visit of the issuer might be not the first such in his/her life. So, we can now use the information on previous life cases for scaling the life case to the expectations the issuer has. The adaptation requires some background knowledge on the handling of life cases in other cities, previous visits, and the profile of the issuer. We may then use a number of questions to figure out, which further adaption or refinement of the life case is applicable. Since some data on the issuer cannot be stored in the system due to regulations and laws we need to repeatedly obtain these data. So, the data we need to capture within the life case are extended by data we need for figuring out which specific life case is under consideration. At the same time we may use this context information for adapting functionality that is provided.

This specific actor context is combined with the portfolio restrictions. Actors with a non-deterministic behaviour do not use high ambiguity or deep projection. At the same time, their mental context and their approximation context must be rather sophisticated. Actors acting more on intention intensively use all four kinds of actor contexts. Task-oriented and reactive behaviour requires support for mental context. Actors acting in collaborations need additional support for their common disambiguation. If actors do not complete their tasks within one session, they need a well-prepared projection context for the case that they resume their tasks. We shall later map these requirements to adaptation rules and control rules for adaptation.

### 4.2 Storyboard Context

Context has also a storyboard dimension. The actor’s context must be combined with the storyboard, life case and portfolio contexts. The latter two selectively condition the situational interest of the actor and the relevance of the current scene for the actor. Based on the relevance we may identify and use properly all the content that should exert the evolution of the current story. We may now use this information for extracting whether a sequence rule, i.e. a rule of the form \( s_1 \rightarrow s_2 \), is required that a visit of scene \( s_1 \) should be followed by a visit of scene \( s_2 \) can be applied to the current system usage. The rule may hold in general, but is considered to be not applicable if the existence of process \( p_1 \) leading to scene \( s_1 \) does not have a bearing on the existence of process \( p_2 \) leading to scene \( s_2 \). Therefore, the incorporation of context and the derivation of relevance has mainly to do with selecting the best story for the user and is thus used for the adaptation of content and functionality.

The storyboard context can be used for deriving the most appropriate content. We aim at delivering the right data to the right actor with the right tools and scope at the right time. As the storyboard context provides a good source for adaptation of content and functionality to the current stage of the scenario we collect context within the storyboard and add this context information to the context space. This context allows a treatment of the expectations of the actor. Therefore, each scene in a scenario is provided with a pre-scene context, scene context, and a post-scene context.

The pre-scene context consists of all content that has already been delivered to the actor before the appearance of the actual scene. This information can be used to reduce content delivery for scenes. At the same time, this content can be stored in condensed form and made available to the actor when needed, i.e. the actor can revisit the old content whenever this seems to be necessary. Classical browsers only provide a strictly sequential ‘back’ button for this kind of history management. The pre-context of a scene thus contains all valuable content that is collected during a story, and guarantees the availability of this content when needed.

The post-scene context consists of a potential playout of scenes that can be entered after the current scene. If an actor needs some information on the next actions, then this context information can be used. This information is valuable for those actors who intend to drop out of the system. It is also a part of the help information. The post-scene context can be enriched by metadata describing the content that is provided in the next steps or the data to be produced by the actor. In this case, an intelligent interface may forecast the information need for further steps of the storyboard.
Each scene may also be enriched by superimposed meta-data on the scene, which include everything that could be referenced within the expected consumed and produced data. Typical such references are collaborating actors, retrieval or update data of the current content, and details taken from the log of the current story. Finally, the scene context may include administrative data such as identification of content currently under consideration.

Scene context is enhanced by generic scene information, which can be based on intentions of the WIS. For instance, adverts may be attached to each of the scenes. Default information serves as an exception handling for scenes. If content or functions are currently not available, then default data are provided.

4.3 System Context

The system context is determined by the content and the functions that are provided by the web information system. It consists of at least the following four parts:

Source and acquisition: Source and acquisition is an orthogonal dimension of the WIS. A WIS is supported by media objects that belong to media types as we will explore in detail in part III. In a nutshell, a media object is defined by an extended view on some underlying database, which can then serve for provision of content and functionality of an elementary scene. The databases used for the generation of content form the context of the scenario. We may associate with each scenario the subschemata of these databases that are used for generation of consumed data or for integration of data produced by actors in the scenario.

Associated content: The data that are used for consumed and produced information do not exist in isolation. They are usually associated with other data on the basis of integrity constraints or existence constraints, in particular existence constraints are often not explicitly represented as such, but are embedded into the database schemata used. For instance, we usually associate with objects collected in relationship classes those objects collected in the component classes on which they are based. In this case, we assume that objects in component classes of the relationship type co-exist with objects in the relationship class. We need to consider the environment of content that is currently under consideration together with the data that are associated with this content.

Supported functionality: Functions supporting the actions in scenarios are provided by the WIS. These functions have their own control environment. Typical such control mechanisms are logging, concurrency control, and recovery management.

Security: Security concepts describe *encipherment* and *encryption* (keys, authentication, signatures, notarisation, routing control, access control, data integrity, and traffic padding) for data exchange.

4.4 Temporal Context

The temporal context appears in a number of variants, e.g. storage time, validity time, display time, user-defined time, transaction time, etc. The temporal context is applicable in a number of combinations. Sometimes it is necessary to use all of them, but often it is often observed that only one variant of this context is necessary.

Versions show the life cycle of the objects under consideration. As scenarios will have their own life cycle we cannot assume that database changes are directly enforced on websites. Moreover, it may be useful to provide the old content as long as an actor continues with the same story. Versions can often be systematically structured by *database system phases*:

The *initialization phase* permits the development of objects storing initial information. Integrity constraints are applicable in a limited form.

The *production phase* is the central phase, which consists of runtime querying, modification, transaction management, etc.

The *maintenance phase* is used in productive database applications for clarification of soft constraints, maintenance of constraints that have been cut out from runtime maintenance, and changing the structuring and functionality of the entire database system. Maintenance phases are used in data warehouse applications for recharging the data warehouse with actual information.

The *archiving phase* is used for archiving the content of a database in a form that data relevant for historical information can be easily retrieved. No data modification is permitted; the only modification operation is to load new changes to the archive.

4.5 Context Templates

We may now combine this context information using the following semi-formal template:

```
Context:
  Extension of:
  Actor context:
    Projection context:
    Approximation context:
    Ambiguity context:
    Mental state context:
    Characterisation:
  Storyboard context:
    Pre-scene context:
    Post-scene context:
  Scene context:
    WIS context:
      Source and acquisition:
      Associated content:
      Supported functionality:
      Security:
    Temporal context:
      Versioning:
      Development phase:
    Provider context:
    Developer context:
    Organisational and social context:
  Based On:
```

5 Formal Aspects of Context Modelling

Context evolves for actors, scenarios, systems, and over time. We model the relation between different contexts by lifting relations. Properties that are valid for a certain context may be lifted to another context. This transfer can be based on local model semantics.
For this recall that a context is determined by actor, storyboard, system and temporal contexts. So let \( \mathcal{A} \) denote the set of actors, \( \mathcal{S} \) the set of scenarios, \( \mathcal{W} \) the set of system characteristics, and \( \mathcal{T} \) the set of time units. Then we can take a subset \( \mathcal{C} \subseteq \mathcal{A} \times \mathcal{S} \times \mathcal{W} \times \mathcal{T} \) to represent a set of contexts. Furthermore, we use a family of contexts \( \{ C_i \in \mathcal{C} | i \in I \} \) and a family of statement sets (or theories) \( \{ T_i | i \in I \} \) that are associated with these contexts. Of course, the theory \( T_i \) describes the properties of the context \( C_i \).

5.1 Lifting Relations

On these grounds we may use local models \( M_{i,j} \) for each of these statement sets assuming that the models we consider are enumerated by the second index. More precisely, the models \( M_{i,j} \) determine the meaning of content drawn from a language \( \mathcal{L} \) for describing content in view of context \( C_i \). That is, we use a partial mapping \( \Psi : \mathcal{L} \times C_i \rightarrow M_i \), where \( M_i \) denotes a set of pre-determined meanings for content in \( \mathcal{L} \).

We may now distinguish the formula occurring context \( C_i \) from the same formula occurring in another context by considering the context index \( i \), i.e. we consider pairs \( (i, \cdot) \). Lifting relations can be modelled by rules of the form

\[
\frac{(c_1, \cdot, \ldots, c_n, \cdot) \in \varphi \quad i \in I}{(c_1, i_1), \ldots, (c_n, i_n)}
\]

stating that the formulae \((c_1, \cdot, \ldots, c_n, \cdot)\) can be lifted to \((c, i)\) under the side condition \( \varphi \). In addition, a compatibility relation among local models is introduced similar to logics that capture possible world semantics. This compatibility relation is used for entailment and satisfiability. This approach allows us to reason locally and then to transfer the knowledge we gained to other contexts.

Based on this coarse clarification of basic notation we develop a number of facilities and extend the specification of the WIS:

**Context space:** The content context space is defined on the basis of the content \( C \), scenarios \( \mathcal{S} \) and actors \( \mathcal{A} \). In Example 5.1 we could use information on the travel and on the airline to exclude options that seem to be less likely. The context content space of a WIS for a given content-meaning pair \( (c, m) \) consists of precisely those contexts, under which the particular content will have that particular meaning, i.e.

\[
\mathcal{C}(c, m) = \{(a, s, w, t) \in \mathcal{C} | \Psi(c, (a, s, w, t)) = m\}.
\]

Adaptation of content, functionality, and scenarios to the context that is currently available is based on context infusion. Applying transformation rules we change content, functions, and the presentation. Therefore, we use a context specification for the development of enforcement rules. These rules may restrict scenarios to more specific ones, extend or shrink content, and extend or remove functions.

**Life case extension and specialisation:** The general life case specification can often be specialised, if context is explicitly injected. We need both the more general life cases and the contextualised ones. Whenever the WIS is revised or extended, we can return to more general life cases and generate another contextualisation. Typical specialisations concern changes in the life case flow. We may specialise the data consumed by an actor in dependence of the actor’s context. If we know that actors need special auxiliary information or conversely actors become more knowledgeable during the utilisation of the WIS, then we may adapt the data provided for consumption. At the same time, we can specialise the figures according to the given context. In the same way spatial and temporal information provide a basis for refinement of life cases.

Life cases may be extended to requirements that were collected in the context space. The content context may require a more elaborated content to be provided. The supported functionality may require additional functions, content, or a specific presentation. Intentions may be more specific under consideration of context. For instance, if we want to support a certain usage of a WIS that was not originally intended but became important in order to maintain frequent visits, then the original life case is extended by those associated life cases.

**Development of a context manager:** Context is also bound to scenes and thus evolves within a story. We may expect that content enhances context. For this reason, we introduced the previous scene context. Therefore, a subsystem that manages the context is needed. This context manager uses the lifting rules introduced above for transferring context to context for scenes, collaborating actors, and the WIS as such. The system also supports the rule-based development of logics over time. We cannot require that the rule system is complete, but it must be consistent. A useful property is commutativity, i.e. the results of firing rules does not depend on their order. The context management system enhances the dialogue management system by adapting and specialising the presentation and injecting content into it.

**Example 5.1** A typical context extension to functionality is associated with the problem to avoid that users trap into losing-track situations. Such situations can be detected based on the user’s behaviour, e.g. invoking the help function repeatedly on similar topics, repeatedly positioning on particular locations and performing similar operations on similar data, excessively navigating through information space without invoking any reasonable functionality, looking repeatedly for FAQs on similar topics, attempting to enter a discussion forum, and sending email to the site administrator.

User aid that can be provided for losing-track situations is giving access to a thesaurus of the subsystem the user is accessing. Furthermore, the respective business model may be exposed to the user together with an explanation that is adapted to a particular user type. Similarly, access to a FAQ list suitable for the user and the accessed subsystem may be given. Furthermore, improved search facilities and examples targeting at the subsystem accessed may be provided.

\[ \diamond \]

5.2 Adaptivity

The idea of adaptivity is to equip the system with enough additional information and rules that would render it possible to engender the right content and functionality for the current situation. That is, the system is supposed to act according to the dictum ‘you take care of the specification, and the system will take care of itself and adapt to the current use’.

Two content objects \( c_1, c_2 \) are synonymous in the context \( C_i \in \mathcal{C} \) iff \( \psi(c_1, C_i) = \psi(c_2, C_i) \). They are
totally synonymous iff $\psi(c_1, C_i) = \psi(c_2, C_i)$ holds for all contexts $C_i \in \mathcal{C}$. They are epistemically synonymous within a scenario $s$ for an actor $a$ iff $\psi(c_1, C_i) = \psi(c_2, C_i)$ holds for all contexts $C_i \in \mathcal{C}$ associated with $a$ and $s$.

Applications often require adaptation of processing context, e.g. to actual environments such as client, server, and current communication channel, user rights, roles, obligations, and prohibitions, content required for the portfolio of the current user, the actual user with his/her preferences, the level of task completion depending on the user, and the user’s completion history.

Consider for instance e-learning or e-government websites discussed in (Moritz et al. 2005) and (Schewe, Thalheim, Binemann-Zdanowicz, Kaschek, Kuss & Tschiedel 2005). Citizens may apply for a primary place of residence. In this case, their passport must be changed; otherwise, no change is required. Citizens with school-age children may have to complete additional documents. Completed documents may be decomposed into a suite of documents due to legal restrictions, e.g. by a data protection act requiring that data for city officials and service offices such as the unemployment agency must be separated.

Depending on the role of users, story completion may be scheduled sequentially for some users or in parallel for others. For instance, clerks in a city office may consider documents in parallel, while citizens complete their documents in a sequential mode.

**Example 5.2** Adaptivity may be required at runtime. For instance, people with foreign citizenship may be required to apply for a residence permit. Users may require a varying support depending on the environment that is used for the completion of documents. Users should be supported whenever they are interrupted during task completion.

These requirements lead directly to the requirement to develop a facility for *mutable*, *adaptable scenarios* for different users, portfolios, and contexts. We shall return to this requirement after introducing templates in the next section. It is our target to develop generic scenarios that can be refined to scenarios by injecting context. This approach is more widely used for WISs than one would expect. For instance, almost all information sites of cities and regions provide a very similar hotel or event search. The reason is not the existence of a development monopoly but rather the evolution of these search facilities to semi-standards. These standards are not officially agreed, but have been formed by copying successful solutions.

**6 Conclusion**

In this paper we approached the pragmatics of Web Information Systems (WIS) design focusing on the method of storyboarding that is an integral part of the codesign approach to WIS design (Schewe & Thalheim 2005a, Schewe & Thalheim 2005b). A storyboard specifies in an abstract way who will be using the WIS, in which way, and for which goals. Thus, the specification of a storyboard captures the navigation paths, i.e. the stories through the “scenes” of the WIS, the action scheme associated with the stories, the actors appearing in the scenes, and the tasks the actors accomplish. In addition, there are various static, dynamic and deontic constraints governing the storyboard.

While syntax and semantics of storyboarding have been well explored, its pragmatics has not. While many methods for WIS design emphasise content modelling, we start from the very fundamental observation grounded in semiotics that content refers to a syntactic dimension, whereas a pragmatic dimension requires dealing with information. This led to the objective to investigate in depth intentions associated with a WIS. The facets of intention arising from this form the basis for our technical development in this paper dealing with life cases, user models, and contexts.

Life cases capture observations in reality, which by means of abstraction can be used to derive scenarios for the storyboard. Integrating these scenarios provides a method for storyboarding. User models are by user and actor profiles, and actor portfolios. The latter ones provide a better understanding of the tasks associated with the WIS. Contexts can be classified according to how they impact on the life cases, the user models, and the storyboard extracted from them.

This work on pragmatics of storyboarding contributes to closing a gap in the codesign methodology for WIS design. It links the formalism of storyboarding to the systems requirements, and provides guidelines and means to derive the complex storyboards from informal ideas about a WIS without any technical bias. So, on one hand, this work on pragmatics is a decisive part of the methodology, which does not just consist of a collection of formally integrated models, but also has to state how to use them. It would be rather difficult mapping life cases or user models directly to a conceptual model of a WIS, which resides on a much lower level of abstraction as the storyboard. So on the other hand this work emphasises the need for storyboarding as the decisive tool for high-level WIS engineering. As shown in (Schewe & Thalheim 2007b) this is also the basis for high-level reasoning about WISs addressing such important issues as personalisation of functionality.

Despite the high relevance of pragmatics for the completeness of storyboarding and the codesign methodology as a whole, the work reported in this paper is only part of the story, as it only addresses the context analysis. Together with life cases (Schewe & Thalheim 2007a) and user models (Schewe & Thalheim 2006) they capture usage analysis of WISs, which still does not completely exhaust the problem area associated with pragmatics of storyboarding. We are in the process of writing up a second part of storyboarding pragmatics dealing with WIS portfolios, which combines content and utilisation portfolios that give rise to content and functionality chunks. The content portfolio is used for collecting information requirements. It is based on information needs and demands, and links the storyboard to the lower-level conceptual model of the WIS consisting of a collection of media types. The utilisation portfolio is used for collecting functionality requirements. It describes intentions of users, specific needs and their context.

In addition to this follow-on part on storyboarding pragmatics we are also working on the pragmatism of storyboarding, storyboard refinements, and quality evaluation. All this together plus the ongoing research on logical grounds of storyboarding and their exploitation for reasoning and verification will complete our research on high-level WIS design within the codesign framework.
References


Conallen, J. (2003), Building Web Applications with UML, Addisson-Wesley, Boston.


Schewe, K.-D. & Thalheim, B. (2007a), Life cases: An approach to address pragmatics in the design of web information systems, in ‘Proceedings WebIST’07’.


Web (1991), ‘Webster’s ninth new collegiate dictionary’.

Intention-Driven Screenography

Thomas Moritz\textsuperscript{1}, René Noack\textsuperscript{2},
Klaus-Dieter Schewe\textsuperscript{3}, Bernhard Thalheim\textsuperscript{2}

\textsuperscript{1}Brandenburg University of Technology, Institute of Computer Science
Cottbus, Germany moritz.th@t-online.de
\textsuperscript{2}Christian Albrechts University Kiel, Department of Computer Science
Kiel, Germany noack|thalheim@is.informatik.uni-kiel.de
\textsuperscript{3}Massey University, Information Science Research Centre
Palmerston North, New Zealand k.d.schewe@massey.ac.nz

Abstract: The design and reification of Web Information Systems is a complex task, for which many integrated development methods have been proposed. While all these methods ultimately lead to the construction of web pages, very little attention is paid to the layout of these pages. The work report in this paper amalgamates knowledge from art, cognitive psychology and scenography in an attempt to systematise WIS layout and thus to complement development methods with this respect. We discuss principles and rules for page layout that originate from knowledge of visual perception and communication, and then investigate how layout can support the intentions associated with the WIS. This amounts to guidelines for partitioning pages and using layout objects, colour, light and texture to obtain rhythm, contrast and perspective as the carriers for web page comprehension.

1 Introduction

A Web Information System (WIS) is an information system that is made accessible via the world-wide web. Such systems can be of any size, but the primary focus of research in this area is on large-scale systems that require the support of database technology for the content and are used by an unlimited number of users. Consequently, the number of web pages that are needed to present a WIS to its users is very high, and permanent changes to these pages have to be accounted for. Therefore, the design and reification of such systems cannot be left to “HTML-hackers”. It has become subject to various development methods such as OOHDM [SR98], WebML [CFB+03], WIS co-design [ST05a, ST05b], HERA [HBFV03], WSDM [DL98] and others, and also UML has been adapted to support WISs [Con03, LHSG02]. These methods differ in many respects, and we do not intend to discuss these differences here. However, whatever method is used, it finally boils down to the writing or generation of web pages. So, whatever sophistication has been achieved through the method, a poor layout of the pages can easily destroy it. Nonetheless, astoundingly little effort has been put into WIS layout. It may be argued that layout is the realm of HCI techniques [BGB98, Car91, Joh97, Val82], but then general HCI techniques are not coupled with spe-
Specific development methods, and a lot of the HCI ideas have to be taken into consideration already during WIS design, even at a very high level of abstraction. For instance, the partitioning of pages and colour schemata are linked to the strategic decision on the desired ambience of the WIS as emphasised in [MST05].

Currently, WIS designers are mainly involved in modelling and implementation, while layout is not overly emphasised. However, late consideration of graphical issues may result in inflexibility, and cause problems for extension and change management. Our aim is to support the systematic and early involvement of layout and playout, for which we develop an approach to *screenography*, which adopts and generalises dramaturgy and scenography. Scenography has its roots in theatre, film and television, i.e. outside the web area, and contains the composition of action space, plot and dramaturgy. Dramaturgy controls the sequence of scenes and determines the composition of information. Our claim is to show that WIS layout also requires scenography and dramaturgy to facilitate the understanding and memorisation of content, and to support orientation within the WIS. Screenography aims at providing interfaces of high utility that can be used by any user depending on his/her intentions and tasks. So, we base screenography on a characterisation of tasks, intentions, and specific characteristics of the users, provided by means of user profiles and portfolios. Intentions and tasks determine a user’s expectations and interests; specific characteristics influence the patience in dealing with the WIS. As WISs are to provide needed services, the users’ real needs and life stories have to be specified. We do this on the basis of life cases that generalise, combine, extend and formalise business use cases.

Screenography captures the *layout* and *playout* of WISs. Layout addresses the graphical design of pages in liaison with content to be conveyed and functionality to be provided by the WIS; it supports users depending on their profile. Playout is based on the story specification, the task portfolio of users, and the contexts of the WIS and its users.

The rationale of our work is to complement the WIS codesign method [ST05a] with a systematic approach to WIS layout that is grounded in knowledge from art, cognitive psychology and scenography. In order to do so we start from a general architecture that determines the WIS development needs. On these grounds we refine intentions and use them to specify user profiles and portfolios [ST06a]. In Section 3 the screenography development process is described in detail. We start with the atmosphere that determines the global ambience type. Next we define patterns and refine them to grids that are used to partition the screen and reify the ambience through appropriate colour schemes [MST05]. In Section 4 we discuss cognitive aspects of screenography. As known from art, photography and scenography the attraction of an observer is influenced by rhythm, contrast and perspective, so we have to use the layout elements in a way that supports the desired effects and directs the attention of the users. We concentrate on layout aspects. Cognitive aspects for playout may be considered in a similar way, but are omitted due to space limitations. Finally, in Section 5 we present a brief application case study, in which we discuss improvements of websites originating from screenography.

This paper introduces structural and cognitive concepts in detail. The intention behind structural concepts of screenography has been sketched in [MNST07]. Open research issues concern dimensions of web information systems such as representation advanced functionality and of highly structured content.
2 Development Prerequisites

A prerequisite of screenography is the analysis of a WIS on a high level of abstraction. So, we have to consider the intentions associated with the WIS, and characterise expected users. This is captured by a storyboard that specifies who will use the system, in which way, for which tasks, and for which intentions. The motivation of user for WIS use is explicitly specified through four facets of intentions: purpose (aims or objectives), intents (targets or objects), time (design, end or occasion) and representation (atmosphere or metaphors). Roughly speaking, the first facet represents the ’what’, the second the ’how’ and the third the ’when’ of an intention. Details of these facets were presented in [ST06a].

2.1 Development Architecture.

The Seeheim model for user interface management systems (UIMS) [Gre85] separates the presentation from the application components. Consequently, the development steps regarding these components are separated, too.

The application component distinguishes levels for requirement prescription, specification and implementation. However, for the presentation component only the specification and implementation level are dealt with thoroughly at present. Nevertheless, the requirement level is also important for the presentation component. At this level we define the intention to be supported by the presentation and characterise the WIS users. For instance, we may define the ambience of a decoration to be colour-independent, because the perception and impression of colours depends on capabilities of users as well as cultural and religious views. On the specification level these global definitions acting as guiding conditions.

2.2 User Models.

User modelling is based on the specification of user profiles that address the characterisation of the users, and the specification of user portfolios that describe the users’ tasks and their involvement and collaboration on the basis of the mission of the WIS [ST06b].

To characterise the users of a WIS we distinguish between education, work and personality profiles. The education profile contains properties users can obtain by education or training. Capabilities and application knowledge as a result of educational activities are also suitable for this profile. Properties will assigned to the work profile, if they can be associated with task solving knowledge and skills in the application area, i.e. task expertise and experience as well as system experience. Another part of a work profile is the interaction profile of a user, which is determined by his frequency, intensity and style of utilisation of the WIS. The personality profile characterises the general properties and preferences of a user. General properties are the status in the enterprise, community, etc., and the psychological and sensory properties like hearing, motoric control, information processing and anxiety.
A portfolio is determined by responsibilities and is based on a number of targets. Therefore, the actor portfolio (referring to actors as groups of users with similar behaviour) within an application is based on a set of tasks assigned to or intended by an actor and for which s/he has the authority and control, and a description of involvement within the task solution [ST06b]. A task as a piece of work is characterised by a problem statement, initial and target states, collaboration and presupposed profiles, auxiliary conditions and means for task completion. Tasks may consists of subtasks. Moreover, the task execution model defines what, when, how, by whom and with which data a task can be accomplished. To classify tasks it’s possible to use existing task models, e.g. Concurrent Task Tree (CTT) or Task Object Oriented Description (TOOD). The result of executing a task should present the final state as well as the satisfaction of target conditions.

2.3 Life Cases.

For task completion users need the right kind of data, at the right time, in the right granularity and format, unabridged and within the frame agreed upon in advance. Moreover, users are bound by their ability to verbalise and digest data, and their habits, practices, and cultural environment. To avoid intellectual overburdening of users we observe real applications before the system development leading to life cases. Life cases help closing the pragmatic gap between intentions and storyboarding. Syntax and semantics of life cases have already been well explored in [ST06a].

2.4 Demands for Presentation Development.

Intention, user and life case modelling are very important for screenography. For instance, for developing a satisfactory WIS atmosphere we have to consider the representation facet of the intention, which defines the ambience the presentation should have. It depends on the application area of the WIS and the preferences of the user. The main challenge is the heterogeneity of web clients. Therefore, we specify the atmosphere already on a high level of abstraction. In this area some work has been done and allows the designer to specify device independent the appearance of interaction elements. Examples are UIML (User Interface Modelling Language), AUIML (Abstract UIML), and XUL (Xml User-interface Language). Particularly this approaches developed to realise the functionality aspects.

3 Essential of Screenography

Screenography extends web application engineering by scenographic and dramaturgic aspects and intends to support the interaction between system and user. Screenography aims at an individualised, decorated playout in consideration of intention, user profiles and portfolios, provider aims, context, equipment, functionality and the storyline progress.
3.1 Atmosphere

Defining the atmosphere is the first step of presentation development and a part of intention specification. Due to its definition on a high level of abstraction the atmosphere is independent of equipment features. The reification through page layout and the sufficiency of the available equipment to play out a specific atmosphere will be checked on a lower level. The atmosphere can be described by the ambience of the presentation, which is determined by parameters such as shapes, material, illumination, and colour schema. Further, visual perception is always affected by the current mood and emotions of a viewer.

According to [Mor06] we distinguish the ambience-types powerful, in the sense of dramatic art and vitality, romantic, in the sense of romance and passion, elegant, in the sense of seriousness and dispassion, refreshing, in the sense of ease and transparency, balanced, in the sense of harmony and balance, and energetic, in the sense of phantasm and energy. Figure 1 presents some ambience examples. It shows background facets of the same application, in this case varying only by colour. The colour choice goes back to Goethe [Goe10] and Itten [Itt61]. Each facet consists of 3 colours, a three colour chord.

![Figure 1: Facets of the same application: refreshing, energetic, romantic ambience](image)

3.2 Layout Patterns

After the atmosphere, we have to specify layout patterns considering the prerequisites. Patterns are a powerful conceptual framework for building compelling, effective, and easy-to-use websites [vDLH02]. A pattern consists of visual and functional building blocks. According to [MST05] the functional building blocks realise the access to the presented content and order these. The visual building blocks are important for perception and need to consider the colouring with respect to functionality and aesthetics, the perspective perception of the whole screen, and the visual alignment and partitioning of the screen.

The colouring aspect includes the colour schema development on the basis of the specified atmosphere. Therefore, we have to consider the emotions a user usually associates with colours. The effect of colours can be warm, cool, cold, intensive, hot, light, dark, etc. A user’s perception is also influenced by cultural aspects and age. According to [Mor06] the basis of a colour schema can be a colour chord consisting of \( n \) colours \( (n \in \{2, 3, 4, 6\}) \) that form a regular polygon in the CMY-based colour circle. The colour chords can be complemented to a quality contrast [Itt61] by changing the saturation.

By using any kind of perspective it is possible to realise three dimensional decorations. The perspective aspect of a pattern is mainly determined by the colour schema. In particular,
we have to consider the effect of the colours, because the objects coloured by warm colours appear closer than those in cold colours. Warm colours are assigned to light, while cold colours are assigned to shadow.

The visual alignment is based on a tiling of the screen as a two-dimensional surface. In general, we divide the horizontal and vertical axes using grid points \( x_{\min} = x_0 < x_1 < ... < x_m = x_{\max} \) and \( y_{\min} = y_0 < y_1 < ... < y_n = y_{\max} \). A tile is defined by a rectangular region \([x_i, x_j] \times [y_k, y_\ell]\). Then we partition the whole screen into tiles. A very common tiling is obtained by using just 4 horizontal grid points \( x_0 < x_1 < x_2 < x_3 \), and only 3 vertical grid points \( y_0 < y_1 < y_2 \). Then we can define four tiles:

- \( \text{up} = [x_0, x_3] \times [y_1, y_2] \)
- \( \text{left} = [x_0, x_1] \times [y_0, y_1] \)
- \( \text{middle} = [x_1, x_2] \times [y_0, y_1] \)
- \( \text{right} = [x_2, x_3] \times [y_0, y_1] \)

Usually, the “up” tile is used for some menu bar, the “left” tile for navigation links, the “middle” tile for major content and the “right” tile for side options.

### 3.3 Grid Geometry

Grids were adapted from (conventional) graphic design and used for organising page layouts, e.g. newspapers, magazines and other documents [vDLH02]. The tiling described above is a very common but simple grid that only divides rows and columns, without any other restrictions. More sophisticatedly, the size of visual building blocks can follow a rhythmic structure that can be expressed by a sequence of positive integers. Then an observer perceives larger tiles of a sequence as being more important, in particular, if the sequence shows a monotonic pattern.

Due to its overwhelming use in art and architecture over centuries we are particularly interested in the Fibonacci sequence, which is defined by the recurrence \( f_{n+2} = f_n + f_{n+1} \) with the starting values \( f_1 = f_2 = 1 \). This gives the well-known sequence 1,1,2,3,5,8,13,... It also gives rise to the number of golden section, which played an important role in art and architecture and appears naturally in nature.

Figure 2 illustrates the visual flags model as a simple use of Fibonacci sequence. In fact it dates back to Leonardo da Vinci and orders sections according to some functional criteria. In this example the Fibonacci numbers (multiplied with a scaling constant) are used as horizontal grid points.

![Figure 2: Rhythmic structure of visual building blocks (grid tiles)](image)

Another use of the Fibonacci sequence is to place squares with increasing side length \( f_i \) along a spiral as shown in the lower part of Figure 3. In doing so we obtain a very
nice tiling of the screen (golden square), in which the proportions of the square tiles are determined by the Fibonacci sequence. By combining with the visual flags we realise the so-called Fibonacci grid model (Figure 3).

Figure 3: The Fibonacci grid model

If each tile is associated with a colour of a well-chosen colour schema, this enables desired atmospheric effects as specified in the strategic WIS model [MST05]. The thesis [Mor06] shows this combination of the Fibonacci grid with various colour schemata in different web application projects.

4 Cognitive Aspects of Screenography for Layout

Screenography bases screen and particularly WIS layout on cognitive psychology. The layout of WIS pages contains functional elements such as icons for navigation and functions, and visual elements for presentation of texts based on script and colour and of pictures and structures in different displays. The designed screen is regarded as a structured composition of different elements and is defined as screen layout. Enhanced descriptions how to design the user interface was depicted through Norman [Nor88] and Shneiderman [SP04]. Screenography uses three kinds of principles taken from cognitive psychology: principles of visual communication, of visual cognition, and of visual design.

4.1 Principles of Visual Communication

A clear and well-defined design of a screen layout helps to grasp and to understand the content and enables to select and to access the information. It is a precondition for the successful communication. Visual communication is based on the exchange of visual codes with special meanings. Sender and recipient agree on the meaning of the communication utterances which are typically expressions in a visual language adopted and understood by both partners. A sophisticated specification of visual communication is one precondition for interaction support. It consists of three components:
• **Vision** supports users depending on their physical and physiological properties. Users are used and are limited to certain colouring schemata, to different presentation styles and to different reception styles.

• **Cognition** is based on the physiological and psychological abilities and skills of users. We must take into consideration the approach users take while reading content and using functionality.

• **Processing and memorising characteristics** are based on the psychological ability to read, integrate, and reason about content provided by a page, and to memorise main parts of the content – these vary a lot among users.

Visual communication starts with separating the visual entities on the screen into elementary layout elements. This separation is an analytical process. Visual elements are compared, processed and memorised after recognising them. The scenario for visual communication can be used for developing the layout, i.e. we reverse the order of visual communication. First, visual elements that should be memorised are developed and integrated with the content and functionality necessary for their recognition. Next, these elements are integrated into a draft of the layout. Finally, adornments, presentations, and placements are added.

### 4.2 Principles of Visual Cognition

Principles of visual cognition and visual communication refer to **ordering**, **effect delivery**, and **visualisation**. Ordering is based on an arrangement according to the reading direction and on design according to foreground and background relation. The effect uses background for formation of thematic and optical frame, and schemes of colours and structures. Visualisation is influenced by visual design features such as colour, contrast, composition, overlapping, and cuts [Goe10, Itt61].

We can use a number of principles of visual cognition in screenography. Users are limited by their time, attention, scope, and task portfolio. These limitations must be taken into consideration for layout and playout development. We distinguish four principles that should be preserved:

• The principle of **organisation** requires that screens must be based on a simple, clear and consistent structure. The simplicity, clearness and consistency depends on the cultural background of users.

• The principle of **economy** requires that users should spend minimal effort for recognition and reasoning on visual elements. This principle is supported by minimisation of the set of tools. Users increase their effectiveness of recognition and reasoning.

• The principle of **communication** considers the abilities and the skills of users. This principle is well known from didactics, where content and functionality are adopted to the capacity of users.
Screen design standards allow the user to reuse their previous recognition and reasoning results. These standards define the organisation and design of visual elements.

4.3 Principles of Visual Design

We use a number of principles of visual design in screenography:

- The optical vicinity principle requires that elements are arranged close to each other, if they are related.
- The similarity principle is based on human cognition according to which elements of similar shape are identified as a visual unit.
- Closeness is based on the human visualisation according to which objects need to have a closed shape. Otherwise, missing elements are added.
- The symmetry principle uses units that consist of symmetrically assembled elements. Symmetric and asymmetric structures are visualised in the foreground, whereas all other asymmetric elements are taken to the background.
- The conciseness principle uses visualisation with simplified and consistent organisation of visual screen elements.
- The reading direction principle optically composes static elements (pictures, texts, icons) by the given reading direction. This leads to a shift of the optical focus. We distinguish between geometrical and optical centre. For instance, the thickness of horizontal lines is optically enhanced. Vertical lines seem to be thinner.

These principles help to organise the elements within a page in a way that correspond to human perception. Elements are always recognised within their context. Their value differs based on their syntax, i.e. the formal and aesthetic value, based on semantics, i.e. content and objective value, and based on pragmatics, i.e. ethic and applicability value.

5 Application of Screenography: Case Study

In this chapter we demonstrate the potential of the screenography approach, using a B2C example. This example indicates how we achieve an adaptive playout, considering the intention of the WIS, user profile and portfolio as well as existing lifecases. In general this B2C-WIS offers company and product informations can described by following information pattern (who-what-to_whom-what_activity):

\[
(company)^{information\_on\_products}2(visitor,\_purchaser)^{inform,\_purchase}
\]
In our example acts a small business, producing hand-made collections of wood-figures. The aim of company is to distribute the collections via the web. The content per page shouldn’t be oversized, because the carefully selected product choice and important role of quality. Wood as basic material of all products of the company should be considered. Considering the business philosophy, the presentation style have to be traditional and folksy. The ambience type ‘balanced’ implies that. Before defining the ‘balanced’-colours we have to characterise the user. That’s necessary because the perception and impression of colours depends on capabilities of a user and e.g. cultural as well as religious views. Moreover the age of a user plays an important role. In our example the user is a married european man in his thirties, without special colour preferences. The user prefers a conservative design resp. a adequate composition. In conjunction with the intentions we deduce a suitable colour chord illustrated in fig. 4.

Figure 4: colour chord - ambience type ‘balanced’

Subsequent to the atmosphere determining, we have to choose a pattern. Pattern consist of three parts: visual alignment, colourising and perspective perception (section 3.2). Visual alignment is influenced by lifecases, perspective requirements and the user profile as well as main properties and targets. For instance the content size and the navigation structure and depth influence the tiling. The example user acts deliberate and target oriented. He has no handicaps but don’t prefer very small interaction elements. We consider lifecases to achieve the best possible orientation the user should have during the acting progress. Thus it’s possible to choose the Fibonacci grid for representation, illustrated in figure 3. The way of assigning the colours to the pattern mainly depends on chosen ambience type. The coloured pattern is illustrated in figure 5.

Figure 5: Fibonacci grid model with ambience type ‘balanced’

The last step integrates the content. The representation of content considers user and provider preferences. For instance an adequate picture-size to present details of the hand-made products. User portfolio and lifecases influence the story flow. Furthermore, depending on tasks they have influence in content representation, e.g. the type of progress. The result of the development process is illustrated in figure 6.
6 Conclusion

This paper presented screenography, a novel approach to layout and playout of Web Information Systems (WISs). Screenography transfers the accumulated knowledge in scenography and dramaturgy from its origins in theatre, film and television to the web area thereby exploiting results from creative arts, cognitive psychology, and stage scenography. The rationale behind our work is that layout and playout are not merely activities that start, when the major system design has been completed, but must be treated as integral part of WIS development from the very beginning; a poor layout can nullify all sophisticated modelling work, because the layout is the ultimate carrier of information in a WIS. Screenography is tightly intertwined with storyboarding, the method for WIS usage modelling on a high level of abstraction [ST05b]. It contributes to page partitioning and colour scheme definition in a holistic way.

Screenography is an attempt to turn WIS layout and playout from an art into a craft, i.e. it aims at enabling WIS designers to engineer systems that by virtue of their presentation are conceived as exceeding the expectations of its users. The work presented in this paper is a first step in this direction, yet still much more has to be done to extract knowledge from arts and bring it into a form that can be used by WIS engineers.

References


Quality Assurance in the Design of Web Information Systems

Aleksander Binemann-Zdanowicz¹, Klaus-Dieter Schewe², Bernhard Thalheim¹, Jane Zhao²

¹Christian Albrechts University Kiel, Institute of Computer Science and Applied Mathematics
Olshausenstr. 40, 24098 Kiel, Germany
[binemann|thalheim]@is.informatik.uni-kiel.de
²Massey University, Information Science Research Centre & Department of Information Systems
Private Bag 11222, Palmerston North, New Zealand
[k.d.schewe|j.zhao]@massey.ac.nz

Abstract

Despite the fact that several integrated development methods for web information systems (WISs) have been proposed, quality assurance for such systems has hardly been addressed. In this paper some quality criteria are postulated and investigated by combining semi-formal and formal methods. On a high level of abstractions WISs can be described by abstract locations and transitions between them. These so-called story spaces can be formalised using Abstract State Machines (ASMs), which permit to verify, whether user-tailored versions are compatible with user preferences. Furthermore, ASMs provide a framework for refinement, which permits the integration of story spaces with extended views on databases. This leads to further proof obligations for consistency that can be formalised in the logic associated with ASMs.

Keywords. Abstract State Machines, Web Information Systems, Quality Design, Refinement

Topics. software quality, formal methods, applications

1 Introduction

The field of web information systems (WISs) design has recently attracted a lot of research, and several integrated development methods such as WSDM [10], HERA [14], WebML [8], and Co-Design [18] have arisen from these efforts.

Following [18] there are several levels of abstraction, on which such systems should be specified. On a high level of abstraction a WIS can be described by abstract locations and transitions between them that are initiated by user actions. The technical term used for such a specification is story space, and the abstract locations are called scenes. Initially story spaces can be modelled by layered, edge-labelled directed graphs or equivalently statecharts [12]. Looking more at the flow of actions an assignment-free process algebra can be used, e.g. the language SiteLang from [20]. The algebraic expression resulting from this is called the action scheme or plot of the story space. It can still be visualised using statecharts. An implementation of SiteLang based upon Abstract State Machines (ASM) [7] has been proposed in [1, 4, 3].

On a lower level of abstraction the scenes are specified in details using media types, i.e. views on some underlying database schema that are extended by operations and various options that enable adaptivity and tailored presentations. In particular, the operations refine the actions of the story space including now updates to the database.

In this paper we investigate quality of WISs starting from selected quality criteria. Among them is of course consistency of the specified operations with respect to various static and dynamic integrity constraints. In addition, a specific quality criterion for a WIS is its suitability for users. As users are not known in advance, hence can only be anticipated, it becomes necessary that WISs can customise themselves to users. We formalise this as customisation to the preferences of users. A particular realisation of it is known as content conditioning and is based upon the algebraic description of website storyboards [5, 2]. A third quality criterion concerns the development method as such, which should be scalable to WISs of various types and sizes.

The question is how these quality criteria can be satisfied. As a general idea we propose using first integrated models on various levels of abstraction. This is partly satisfied by the models of story spaces and the media
types, but still we have to verify that both models fit together. In this paper we use the formalism of Abstract State Machines (ASMs) [7] for this purpose. ASMs provide a formal notion of refinement, which will allow us to consider the detailed level of abstraction just as a result of refinement. That is, we remain within the same formal framework. Furthermore, this framework of ASMs is powerful enough to capture all our desiderate, as proven in [6, 11].

On the basis of ASMs we can formalise proof obligations and verify them. With respect to the customisation to users we add user-specific information to the plots. Using also user preferences and goals we can them verify the compatibility with the user-specific plots. For consistency we exploit the dynamic logic of ASMs.

2 Story Spaces

On a high level of abstraction we may think of a WIS as a set of abstract locations, which abstract from actual pages. A user navigates between these locations, and on this navigation path s/he executes a number of actions. We regard a location together with local actions, i.e. actions that do not change the location, as a unit called scene.

Then a WIS can be described by a edge-labelled directed multi-graph, in which the vertices represent the scenes, and the edges represent transitions between scenes. Each such transition may be labelled by an action executed by the user. If such a label is missing, the transition is due to a simple navigation link. The whole multi-graph is then called the story space.

Scenes may be atomic or complex. In the latter case the scene itself represents another set of abstract locations, between which users can navigate. This gives rise to a multi-layer model. Roughly speaking, a story is a path in the story space. It tells what a user of a particular type might do with the system.

The combination of different stories to a subgraph of the story space can be used to describe a “typical” use of the WIS for a particular task. Therefore, we call such a subgraph a scenario. Usually storyboarding starts with modelling scenarios instead of stories, coupled by the integration of scenarios to the story space.

Definition 1 A scenario $\mathcal{E} = (\mathcal{S}, \mathcal{A}, \tau)$ consists of

- a finite set $\mathcal{S}$ of scenes,
- a finite set $\mathcal{A}$ of actions,
- a scene assignment $\mathcal{A} \to \mathcal{S}$, i.e. each action $\alpha$ belongs to exactly one scene,
- and a scene transition relation $\tau \subseteq \mathcal{S} \times (\mathcal{A} \cup \{\text{skip}\})$, i.e. whenever there is a transition from scene $s_1 \in \mathcal{S}$ to scene $s_2 \in \mathcal{S}$, this transition is associated with an action $\alpha \in \mathcal{A}$ with $\alpha = s_1$ or with $\alpha = \text{skip}$, in which case it is a navigation without action, and we have $(s_1, s_2, \alpha) \in \tau$.

We can represent a scenario by an edge-labelled directed graph, in which the vertices correspond to the scenes, i.e. to $\mathcal{S}$ and the edges to the scene transitions, i.e. to $\tau$.

Example 1

Consider an event calendar application, originally developed as part of the project www.cottbus.de [19]. It comprises a series of steps allowing to enter event information and to perform operations upon a created event, e.g. releasing it for deployment, or canceling it. At a general refinement level, the simplified storyboard for the event calendar application looks as follows.

![General Specification of Event Calendar](image)

Figure 1: Refinable Scenario Specification

3 ASM-based Plots

So far our presentation of story spaces was centered around the scenes as abstractions of locations between which users navigate and on which actions are selected and executed. Let us now look closer at the ow of actions, which leads to action schemes or plots giving us a detailed, yet still high-level specification of functionality. For this consider the language SiteLang [20], which defines an algebra that captures the details of plots. We take the set $\mathcal{A}$ of actions and the set $\mathcal{S}$ of scenes from a scenario $\mathcal{E}$ as the basis for defining the set of processes $\mathcal{P} = \mathcal{P}(\mathcal{A}, \mathcal{S})$ determined by $\mathcal{A}$ and $\mathcal{S}$.
De nition 2 Let $E = (S, A, \tau)$ be a scenario. The set of processes $P = P(A, S)$ determined by $A$ and $S$ is inductively defined as follows:

- Each action $\alpha \in A$ is also a process, i.e. $\alpha \in P$.
- $\text{skip}$ is a process.
- If $p_1$ and $p_2$ are processes, then also the sequence $p_1 \cdot p_2$ is a process.
- If $p_1$ and $p_2$ are processes, then also the parallel process $p_1 || p_2$ is a process.
- If $p_1$ and $p_2$ are processes, then also the choice $p_1 \sqcup p_2$ is a process.
- If $p$ is a process, then also the iteration $p^*$ is a process.
- If $p$ is a process and $\varphi$ is a boolean condition, then the guarded process $(\varphi) p$ and the post-guarded process $p(\varphi)$ are processes.

Then the plot of a scenario $E = (S, A, \tau)$ is specified by a process $p \in P(A, S)$. Using ASMs such a plot is nothing more than an update rule, in which there is no assignment [7]. As a consequence there is no need to provide a signature, i.e. updatable functions, at the general abstraction level. Furthermore, all atomic actions $\alpha \in A$ correspond to call-rules. Adding a signature and details to atomic actions is a refinement task.

Example 2 The following is the plot for the scenario in Example 1. The postcondition notYetChosen ensures the same event has not been open for modifications in another session.

Login ; (Add new event ; Enter details) $\sqcup$ skip) ; (View ongoing events ; Choose event(notYetChosen) ; Enter details)* ; (View ongoing events ; Choose event(notYetChosen) ; (Archive past event $\sqcup$ Cancel event $\sqcup$ Release event setup ))

4 Refinement of Scenarios and Plots

Scenes in a scenario or the whole story space can be atomic or complex. In the latter case they represent a detailed dialogue of users with the system. Thus, they can be specified by a scenario, and consequently also by a plot. We may then replace the scene $s$ by its defining scenario $E'(s)$ and redirect transitions that lead into $s$ or start from $s$.

Similarly, whenever the action $\alpha$ in the plot leads to scene $s$, the plot will be refined by adding a SiteLang process following $\alpha$.

Example 3 We first refine the scenario in Example 1. It is a $(1, n)$ refinement [7] with respect to the process Release event setup in Figure 1. Note this is a correct and complete refinement of the general specification. The general process Release event setup is replaced with checking whether the permission is granted and if so, with three actions executed parallelly: publishing a commercial for the event, issuing final orders for necessary supplies, and archiving the released version of the event information. The three latter actions are possibly performed by different actors, but this aspect is not considered on this abstraction level.

![Refinement of Event Calendar at the Release Process](image)

Accordingly, the following plot refines the one in Example 2. The postcondition permissionGranted states the next steps can be accomplished as the relevant part of the refined process Release event setup has been enabled.

Login ; (Add new event ; Enter details) $\sqcup$ skip) ; (View ongoing events ; Choose event(notYetChosen) ; Enter details)* ; (View ongoing events ; Choose event(notYetChosen) ; (Archive past event $\sqcup$ Cancel event $\sqcup$ (Check release permission(permissionGranted) ; (Publish commercial $\sqcup$ Order suppliers $\sqcup$ Archive release ))))

5 Verification of User Preferences

The major quality aspect on the level of story spaces is the suitability of the specified system for its users. In
order to achieve this suitability, we define roles and user types.

A role is associated with certain rights and obligations, which is reflected in the scenario by associating actions with the corresponding roles. That is, only users in these roles may execute the actions. Consequently, if \( r \) is a role, we can obtain a role-specific scenario for \( r \) by first eliminating all actions that are not associated with \( r \) and then eliminating all isolated scenes. Furthermore, the presence of roles permits a simplification of the associated plot.

**Example 4** The following is a role-specific scenario for role **Supplier Contact Person** in the scenario in Example 1. As we can see, the new process *Check whether released* has been introduced. It is thus a correct but not complete refinement [7]. In order to obtain a complete refinement, an appropriate union of refinements for the roles foreseen in the specification is performed accordingly. This procedure is also conform to the well-proven utilisation of the ASM-based SiteLang methodology [3].

![Figure 3: Role-Specific Refinement](image)

Accordingly, the following plot is specific for this role. The precondition *toSupply* determines the respective event is still to be provided with supplies and thus of relevance to this user role.

\[
\text{Login} : \quad (\text{View ongoing events} \land \text{Choose event}) \land \text{Enter details} \\
\text{Choose event} \{\text{notYetChosen} \} : \{\text{toSupply}\} \text{Enter details} \\
\text{Check whether released} : \text{Order suppliers}
\]

Analogously, we may set up user types, for which we may determine customised scenarios and plots in the same way as for roles.

In addition, user types lead to preference rules. These rules define equations on processes. We consider the following types of preference rules:

- An equation of the form \( \{\varphi\} (\alpha \square \beta) = \{\varphi\} \alpha \) expresses that a user conditionally prefers action \( \alpha \) over action \( \beta \), where the condition is expressed by \( \varphi \). The special case \( \varphi = \text{true} \) expresses an unconditional preference.
- An equation of the form \( \{\neg \varphi\} \alpha = \text{fail} \) (or equivalently \( \alpha = \{\varphi\} \alpha \)) expresses that the condition \( \varphi \) is a precondition for the action \( \alpha \).
- An equation of the form \( \alpha \{\neg \varphi\} = \text{fail} \) (or equivalently \( \alpha = \alpha \{\varphi\} \)) expresses that the condition \( \varphi \) is a postcondition for the action \( \alpha \).
- An equation of the form \( \alpha \{\varphi\} = \{\varphi\} \alpha \), which is equivalent to \( \{\neg \varphi\} \alpha = \alpha \{\neg \varphi\} \) and to \( \{\varphi\} \alpha \{\neg \varphi\} = \text{fail} \) expresses that the condition \( \varphi \) (and so its negation \( \neg \varphi \)) is invariant under the action \( \alpha \).
- An equation of the form \( \varphi \land = \text{false} \) expresses that the conditions \( \varphi \) and \( \varphi \) exclude each other.

Here we used an additional process *fail*, which is never defined.

The presence of preference rules allows us to verify the user-type-specific plots. If \( p' \) is such a plot for a user type \( U \), while \( p \) is the general plot, from which \( p' \) was derived, we have to show that \( p = p' \) can be derived from the preference rules, the general rules that govern equalities between processes such as associativity of choice, sequence and parallelism, commutativity of choice and parallelism, neutrality of *skip* for sequence and parallelism and of *fail* for choice, etc., and the general rules for propositional logic.

**Example 5**

The following properties, i.e. user preference rules, can be specified for the role-specific storyboard in Example 4.

Preconditions and postconditions:

\[
\text{Choose event}\{\neg \text{notYetChosen}\} = \text{fail} \\
\langle \neg \text{toSupply}\rangle \text{Enter details}
\]

The verification of the event setup release condition does not affect the event details obtained before (this desired property requires a further refinement of the storyboard):

\[
\{\text{detailsEntered}\} \text{Check whether released} = \text{Check whether released}\{\text{detailsEntered}\}
\]
Once a supplies order has been accomplished, no details can be added to the event specification:

\[ \text{Order suppliers} \{ \neg \text{detailsEditable} \} \]

6 Refinement through Media Types

The specification of WISs through story spaces resides on a propositional level, i.e., assignments are not yet considered. When we proceed to a lower level of abstraction we add media types, i.e., views on some database schema that are extended by operations and other extensions that are not so important for us now.

In terms of ASM refinements the introduction of media types first means to add a signature to the ASM. Each function symbol in this signature corresponds to a type in the database schema. The defining queries of the views give rise to expressions of the form \( L_x \varphi \) denoting the unique value \( x \) determined by formula \( \varphi \).

The operations give rise to more complex ASM rules defining the action in the story space. In general, an operation consists of a signature and a body. The signature consists of an operation name \( O \) (which comes from the story space), a set of input-parameter/type pairs \( \iota_i : t_i \) and a set of output-parameter/type pairs \( o_j : t'_j \). The body is an ASM rule built using the following constructs:

- **skip** and **fail**, denoting operations that do not change anything or are not defined, respectively, are rules.
- An assignment \( f(t_1, \ldots, t_n) := exp \) with an updatable function \( f \) in the signature of arity \( n \) and an expression of the same type as \( x \) is an ASM rule. Here we permit expressions \( L_x \varphi \) corresponding to queries.
- If \( p, p_1 \) and \( p_2 \) are ASM rules, the same holds for the iteration \( p^* \), the choice \( p_1 \sqcup p_2 \), the sequence \( p_1 ; p_2 \), and the parallel rule \( p_1 || p_2 \).
- If \( p \) is an ASM rule and \( \varphi \) is a condition, then the guarded program \( \{ \varphi \} \) and the postguarded program \( p \{ \varphi \} \) are also ASM rules.
- If \( x \) is a variable and \( p \) is an ASM rule, then the selection \( @x \bullet p \) is also an ASM rule.

**Example 6**

Consider the refinement of the process **Release event setup** as shown in Example 3. Its plot looks as follows.

(\begin{align*}
\text{Check release permission[permissionGranted]} \; &\text{; ( Publish commercial } \parallel \text{ Order suppliers } \parallel \text{ Archive release })
\end{align*})

It can be described by the following ASM.

\begin{align*}
\text{do} & \quad \text{Check release permission} \\
& \quad \text{if permissionGranted(e) then} \\
& \quad \text{Publish commercial} \\
& \quad \text{Order suppliers} \\
& \quad \text{Archive release}
\end{align*}

It is then refined by specifying the corresponding views **PermittedRelease**, **CommercialState**, **Order** (assumed updatable), **Archive**, **Supply**, and **Requirements**, as well as the new rule **Generate archive view** as follows.

\begin{align*}
\text{do} & \quad \text{if } \exists x. (\text{PermittedRelease(e)} \neq \emptyset \text{ then } \text{permissionGranted(e)} := \text{true}} \\
& \quad \text{if permissionGranted(e) then} \\
& \quad \text{CommercialState(e) := released} \\
& \quad \text{Order(e) := @x \bullet (Supply(x) \land Requirements(e,x))} \\
& \quad \text{Archive(e) := @x \bullet (Generate archive view)}
\end{align*}

Note that we deviated slightly from the syntax in [7]. So now we obtain a full-edged ASM that specifies the details of the WIS. With respect to the connection to the story space, the propositional conditions \( \varphi \) now have to be refined to conditions on the database schema.

7 Proof Obligations in the Logic of ASMs

With the introduction of media types to support scenes we can now exploit the dynamic logic of ASMs that has been defined in [7] to formalise consistency proof obligations and to verify them. In this logic we look at formulæ of the form \( [p] \varphi \) with an ASM rule \( p \) and a formula \( \varphi \). Informally, \( [p] \varphi \) means that after the successful execution of \( p \) the formula \( \varphi \) necessarily holds. In addition, we use the shortcut \( (p)\varphi \) for a complex ASM rule \( p \):

- \([\text{skip}]\) for a complex ASM rule \( p \):
  \begin{align*}
  & \text{[skip]} \\
  & \text{[fail]} \cdot false \quad [x := t] \cdot \{x/t\} \text{ (substitute all free occurrences of } x \text{ in } t) \quad [p_1 ; p_2] \cdot [p_1] || [p_2] \quad [p^*] \cdot \text{the weakest solution } \varphi \text{ of } \varphi \leftrightarrow \land [p] \varphi \quad [(\varphi) p] \cdot \varphi \rightarrow [p] \\
  & [p(\varphi)] \cdot [p](\varphi \rightarrow ) \quad [@x \bullet p] \cdot \forall x. [p]
  \end{align*}
The equivalence for the iteration operator refers to the implication order, i.e. if \( \varphi \models \varphi \) holds, then is called weaker than \( \varphi \). A more detailed definition including further rules regarding the structure of is given in [7].

In the sequel we will discuss two applications of this dynamic logic:

- We take a look at proof obligations for the operations that result from the specification of the story space.
- We take a look at proof obligations for the operations that arise from static and dynamic integrity constraints on the underlying database schema.

Let \( p \) denote the \texttt{SiteLang} expression that represents the complete story space. If all conditions in \( p \) are replaced by conditions on the database schema and all actions are replaced by the ASM rules defining the realising operations, we obtain an ASM rule, which by abuse of notation shall still be denoted \( p \).

As a WIS has a general purpose, this can be formalised by some post-condition. Thus, \( [p] \) describes the weakest condition, under which the purpose of the system can be achieved. If \( \varphi \) characterises a precondition that should be sufficient for the achievability of the WIS purpose, then we obtain \( \varphi \models [p] \) as a general story space proof obligation. In most cases we should expect \( \varphi \models \text{true} \).

Similarly, we may concentrate on fragments \( p' \) of the story space expression of the form \( \{ \varphi \} p \), which corresponds to a Hoare triplet and thus gives rise to a special story space proof obligation \( \varphi \models [p'] \).

A static constraint on the underlying database schema \( S \) is a condition \( \zeta \), in which the free variables are among the \( R \in S \). Such constraints give rise to the request that whenever an operation is started in a database satisfying \( \zeta \), then the database reached after successfully completing the execution of the operation, must necessarily satisfy \( \zeta \), too.

That is, for all operations \( p \) that are defined on a pre-site and all static constraints \( \zeta \) we obtain a static consistency proof obligation \( \zeta \models [p] \).

A dynamic constraint on the underlying database schema \( S = \{ R_1, \ldots, R_n \} \) is a condition \( \zeta \), in which the free variables are among in \( S \cup S' \) with \( S' = \{ R'_1, \ldots, R'_n \} \) and each \( R'_i \) having the same type as \( R_i \). The additional variables \( R'_1 \) are used to distinguish between \( S \)-databases \( db \), on which an operation \( p \) is started, and \( S' \)-databases \( db' \) resulting after \( p \) has been successfully executed.

Obviously, a dynamic constraint \( \zeta \) has to be interpreted on a pair \((db, db')\) of databases. Following a standard approach to dynamic consistency we associate with \( \xi \) an abstract program

\[
p(\xi) = @R'_1, \ldots, R'_n \bullet \xi \ R_1 := R'_1 \ldots R_n := R'_n.
\]

Then dynamic consistency of an operation \( p \) with respect to \( \xi \) means that \( p \) must “specialise” \( p(\xi) \), i.e. we require that \( [p(\xi)] \rightarrow [p] \) for all conditions on \( S \). Fortunately, this proof obligation can be rephrased using a renaming \( p' \) of \( p(\xi) \) given by

\[
p' = @R'_1, \ldots, R'_n \bullet \xi(R_1/R'_1, \ldots, R_n/R'_n)
\]

\[
R'_0 := R'_1 \ldots R'_n := R'_n.
\]

Then the dynamic consistency proof obligation for \( p \) with respect to \( \xi \) becomes

\[
([p']\langle p \rangle(R_1 = R'_1 \land \ldots \land R_n = R'_n))\{R'_1/R_1, \ldots, R'_n/R_n\}.
\]

Example 7

8 Conclusion

In this paper we combined parts of a semi-formal development method for web information systems (WISs) [18] with the formal method of Abstract State Machines (ASMs) [7]. We showed that the assignment-free flow of actions – called plot – can be easily modelled by ASMs, which leads to three opportunities for quality assurance.

The first opportunity arises from considering user-customised plots, as ASMs permit the verification of consistency of these plots with user preferences. This is in accordance with the personalisation approach in [17], which was based on equational reasoning with Kleene algebras with tests (KATs) [15]. It can also be applied to quality-checking the localisation in [9].

The second opportunity arises from exploiting refinements of ASMs, which permits the formalisation of WISs on a lower level of abstraction, but staying within the same formal framework. In particular, this captures extended views on databases – called media types in [18]. This refinement-based approach guarantees that the lower-level model is consistent with the higher-level one.

The third opportunity arises from the logic associated with ASMs, which permits the verification of further consistency proof obligations for detailed ASMs. This is in accordance with the formalisation of proof obligations for WISs using higher-order dynamic logic [13] in [16].

This captures three important aspects of quality of WISs: the suitability of a system for its users, the consistency of the system with respect to various constraints, and the scalability with respect to various applications. As the approach is based on ASMs, most common theorem provers can be used in accomplishing the verification tasks.
References


Abstract—The development of web information systems (WISs) requires modelling on various layers of abstraction. Based on an abstract abstraction layer model (ALM) the work in this article approaches the modelling on the highest layer dealing with strategic modelling. Strategic modelling addresses a very general characterisation of WISs in terms of its content, functionality, context, usage and presentation, and pragmatic guidelines for achieving these.

The article discusses branding, utilisation space modelling, utilisation portfolio modelling and atmosphere modelling as the major parts of a strategic model. Furthermore, techniques based on linguistic analysis, communication analysis and metaphors make up the informal means to approach WISs in strategic terms.

I. INTRODUCTION

The development of web information systems (WISs) requires modelling on various layers of abstraction. The co-design approach to WIS modelling in [28] emphasis a strategic layer, a business layer, a conceptual layer, a presentation layer, and an implementation layer. Other methods such as HERA [16], WSDM [8], OO-H [13], OOHDM [30], HDM [12], ARANEUS [1], WebML [3] and UML [23] agree on the use of abstraction layers as such, but differ in the concrete use of layers. In particular, higher layers dealing with strategic modelling and business-oriented modelling are often neglected.

The goal of this article is to discuss modelling on the strategic layer of a WIS. Strategic modelling comprises two parts. The first one addresses a very general characterisation of the system in terms of its content, functionality, context, usage and presentation. The second part addresses pragmatic guidelines for strategic modelling.

The strategic characterisation of a WIS starts from the very general question what the WIS is about, i.e. the purpose(s) of the system, and what are criteria for the WIS being successful. The general answer to these questions gives rise to an informal mission statement, and a characterisation of the brand of the WIS. The latter one will follow the general classification scheme for WISs.

Going more into details we first explore the kind of content that is to be presented in the WIS, and the kind of functionality, with which this content can be accessed, customised to the needs of particular WIS users, and updated. This defines the utilisation space of the WIS.

We then explore the utilisation portfolio to gain even more details. This means to model the users (or actors) who will use WIS, their goals, i.e. why they are supposed to use the system, and the tasks that have to be performed to reach these goals.

The fourth and last part of a strategic WIS model are general rules for the formation of the WIS presentation. These address the gestalt, atmosphere and progression of the system based on knowledge about the cognitive perception of form, colour and other style elements.

Taking these parts together we should keep in mind that the role of strategic modelling is to lay out the plan for the whole WIS without drifting into technical details. As a consequence, the techniques applied on this level will be rather informal, whereas formalisation will be achieved on lower levels of abstraction. We will discuss linguistic analysis using word fields [10], the use of metaphors [32] and communication analysis [26] as the major methodological means for strategic modelling.

The rationale behind our approach is an observation made in theoretical linguistics [4]: whenever a complex construction has to be explained, humans first think in terms of concepts, which are then mapped to a linguistic construct and only finally translated into sentences. Carrying this idea over to WIS development means to first lay out the fundamental concepts that are to be captured by the system, then map them onto a conceptual model, before finally approaching an implementation using common available technology.

We will not go into the details of the conceptual modelling of WISs, as [28], [27] contain a detailed account of storyboarding, and content and functionality modelling. We will, however, show how the strategic model impacts on the formation of a WIS in terms of its layout and playout. For this we start from the means that are available for visual communication such as visual ordering, partitioning, colouring and perspective. We then go into more details discussing grid models and colour selection in accordance with the layout, atmosphere and progression patterns identified on the strategic layer.

In Section II we describe the abstraction layers of the co-design method to put this article into context. In Section III we then address the first part of strategic modelling, i.e. the general characterisation of WISs in terms of content, functionality, context, usage and presentation. The second part concerning pragmatic guidelines for strategic modelling will
be handled in Section IV. Finally, in Section V we discuss the impact of strategic modelling on visual design patterns.

II. ABSTRACTION LAYERS IN WIS MODELLING AND DESIGN

The methodology for the development of web information systems is based on an Abstraction Layer Model [28], which is illustrated in Figure 1. The general ideas of this model will be explained in this section.

We identify several layers of abstraction. The top layer is called the strategic layer. It is used to describe the system in a general way: What are the intentions? Who are the expected users?

The next lower layer is called the business layer, which is used to concretise the ideas gathered on the strategic layer. This means to get a clearer picture of the different kinds of users and their profiles. This may also include the different roles of users and tasks associated with these roles. The major part of this layer, however, deals with the description of the story board. Stories identify possible paths through the system and the information that is requested to enable such paths. So the general purpose of the business layer is to anticipate the behaviour of the system’s users in order to set up the system in a way that supports the users as much as possible.

The central layer is the conceptual layer. Whilst the business layer did not pay much attention to technical issues, they came into play on the conceptual layer. The various scenes appearing in the story board have to be analysed and integrated, so that each scene can be supported by a unit combining some site content with some functionality. This will lead to designing abstract media types. The information content of the media types must be combined to design the structure of an underlying database.

The next lower layer is the presentation layer which is devoted to the problem of associating presentation options to the media types. This can be seen as a step towards implementing the system.

Finally, the lowest layer is the implementation layer. All the aspects of the physical implementation have to be addressed on this layer. This includes setting up the logical and physical database schemata, the page layout, the realisation of functionality using scripting languages, etc. As far as possible, components on the implementation layer, especially web-pages, should be generated from the description on the higher layers.

On each layer except the strategic layer, we identify two dimensions for the description of the system: focus and modus. The focus dimension distinguishes between local and global components; the modus dimension distinguishes between static and dynamic components. As this leads to four combinations, we distinguish between the following components:

- global and static components, which are addressed by a data specification,
- global and dynamic components, which are addressed by a function specification,
- local and static components, which are addressed by a view specification, and
- local and dynamic components, which are addressed by a dialogue specification.

Each layer is associated with layer specific modelling tasks. The transition from the strategic to the business layer is associated with the activities of story boarding and user profiling. The transition from the business layer to the conceptual layer is associated with conceptual modelling, which addresses database modelling, operations modelling, view modelling, and media type modelling. The transition to the presentation layer is associated with the definition of presentation styles. Finally, the transition to the implementation layer is associated with all implementation tasks.

The fact that these layers exist in the model and that the methodology is based on transitions between these layers does not imply that a development project must first finish the work associated with one layer before it can proceed to the next lower one.

In the following subsections we will discuss further several details of the abstraction layers.

A. Strategic Layer

As already stated the purpose of the strategic layer is to define the intentions of the web-based system and to identify the target users. Intentions should be grouped into major and minor intentions. For each identified intention a time-scale should be determined as well.

For instance, a group system for organising a conference intends to provide

- general information about the conference, its history, its scope, its location and dates, etc.,
- specific information for authors how to submit papers, uploading and downloading facilities, the reviewing process, etc., and

Fig. 1. Abstraction Layers in Web Information Systems
The identification of target users does not yet aim at classifying them according to their profiles, i.e., the questions how users access the system will not yet be addressed. However, distinguishing between internal and external users, users with specific access rights and obligations, thus playing a specific role, and identifying the tasks assigned to such users is considered as an activity on the strategic layer.

For instance, a group system for organising a conference may identify different user roles:

- The normal users are potential participants and authors of papers. They should access the general conference information and have an interface for uploading abstracts and papers.
- The programme committee members and especially the co-chairs need a specific access to the reviewing system as a subsystem, which allows papers to be downloaded, reviews to be uploaded and discussed, etc.
- The organisation committee members need a specific access to upload specific information of organisational nature such as registration and travel information.
- The administrator is the one who can grant and revoke access rights for the other users.

A common technique applied to these tasks is to have a brainstorming session. On this level, intentions, target users, roles and tasks should be described by informal documents using just text or tabular representations. In the following sections we will investigate the strategic layer in more detail.

B. Business Layer

Based on the findings of the strategic layer the development process will proceed with the tasks of story boarding and user profiling. The emphasis is on anticipating the behaviour of the target users, i.e., how they will navigate through the system, which information they will need on their path(s) through the system, and which actions they will carry out. Therefore, the activities on the business layer are the following:

- Analyse roles and tasks of the intended users;
- Classify users according to user profiles;
- For the different roles and profiles describe stories, i.e., navigation paths through the system;
- Integrate stories into scenarios, i.e., networks of scenes and transitions between them;
- For each scene describe the information the users consume and produce, i.e., information taken from the system and entered into the system, respectively;
- Describe the actions of the users for the different roles, tasks and profiles within the identified scenarios;
- Identify metaphors that may help a user to achieve familiarity with the system.

The user roles that were identified on the strategic layer already provide a first classification of users. This classification refers to the rights and the obligations of the users. For instance, certain parts of the system may be reserved to specific roles. In the conference organisation example that we used in the previous subsection, the paper reviewing subsystem is exclusive to the members of the programme committee (and the administrator), whereas the general information about the conference is open to all users.

Certain tasks are associated with each role. Identifying these tasks already gives an indication of different stories, as different roles should lead to different navigation paths. It is also possible that an individual user can have more than one role. For instance, in the conference organisation example, each programme committee member is of course also a "normal user".

Defining roles and tasks is again a brainstorming activity, which will result in non-formalised documents. These documents mainly identify the name of the role, a short textual description and a list of associated tasks. Each task should also be given a short description, but it must be described in more detail further on. This will lead to stories.

For each role the users have to be further classified leading to user profiles. User classification can be done by identifying several dimensions that are useful for the description of the users’ behaviour and using scales for each of these dimensions. Each possible combination of values taken from these scales could define a user profile, but it may be advisable to combine some of them. Thus, user profiles could be described by a matrix-document.

For instance, in the conference organisation example the users could be classified according to their interest in different parts of the conference including the social programme. In an information service system dimensions for user profiling could be the familiarity with technical systems, the educational background, the preferences with respect to information density, etc.

The roles and user profiles tell us who will use the system. Story boarding will analyse how the different users will navigate through the system. In practice, we may think of a story as a sequence of web-pages visited by a user. However, story boarding does not start on the level of realised pages. It abstracts not only from presentational issues, but also to a large extent from technical issues. The business layer is still dealing with setting up the system in a way that non-technical stakeholders will be able to assess, whether their goals can be met.

Thus, instead of talking about pages we use the abstract term of a scene. Story boarding identifies useful scenes and transitions between them. This can be described by graphs or transition matrices. Such graphs can be easily sketched in brainstorming sessions and interviews with stakeholders. Graphs, however, do not exhaust story boarding. For each scene we may ask

- which information has to be provided to the user (this will be called information consumption),
- which information is to be requested from the user (this will be called information production),
• which actions can be carried out by the user to navigate to a successor scene including the possibility that the successor scene is the same scene again, and
• which metaphors are useful to support the actions associated with a scene.

Providing information consumption, information production and actions leads to enriched graphs or matrix representations. Furthermore, each transition can be coupled with information that is communicated between the two scenes. For the target scene this is context information.

The most difficult activity in story boarding concerns the integration of individual stories. Scenes that have nearly the same information consumption but appear in different stories could be combined into one scene. This leads to further enriched graphs (or matrices), which are called scenarios.

Finally, scenarios can be described with various levels of detail. Usually, we start with (non-integrated) scenarios, in which a lot of information is left out. Details are added during a process of information gathering. This may lead to combining scenarios, introducing branches, replacing scenes by complete sub-scenes, etc. These activities are subsumed under the notion of story tuning.

Story tuning has achieved its objectives, when all the scenes in the scenarios can be transformed into media types, which will be discussed in the context of the conceptual layer (see the next subsection for further explanation). In fact, the media types are sufficient to describe the system. However, in order to determine them story boarding and tuning are indispensable.

How is story boarding related to the dimensions that we discussed in the previous subsection? Locally, we have to consider the scenes appearing in the scenarios. The static part (view specification) of the scene, is determined by the consumed and produced information, whilst the dynamic part (dialogue specification) is determined by the actions associated with a scene. Globally, we consider complete scenarios. The dynamic part (function specification) is determined by the stories, whereas the static part (data specification) should be a description of all the information used in a scenario, which is left implicit on this layer.

The results of these activities make up the storyboard, i.e., the description of the business layer. Details on how to describe the storyboard are handled in [28].

C. Conceptual Layer

Based on the scenarios that are defined on the business layer the development process continues with a conceptual modelling activity aiming at the definition of media types. However, media types can be defined already during story tuning. The emphasis of the conceptual layer is on analysing the data processed in the scenarios and the actions on these data in order to describe the content and functionality aspects in a formalised way, and to develop database support. Therefore, the activities on this layer are the following:

• Group suitable data and specify operations for the actions, i.e., specify content and functionality.
• Specifying content and functionality leads to defining media types.
• Restructure the data in order to define connections to databases.
• As database design follows different objectives (no redundancy, optimized access), the content specification should lead to views, i.e., transformations which turn the content of a database into the content of a media type.
• Extend media types in a way that they can be tailored to different users, different end-devices and different communication channels without designing multiple sites.
• Link the media types with the scenarios and ensure that all scenarios are adequately supported.

With respect to the static component, i.e., the data, the various scenes will be analysed and the information consumption will be formalised using data types. This gives a local view on the data that is present in the system. For example, a scene in our conference organisation example (associated with the role of a programme committee member) might be devoted to look at a completed review. So the information consumption is just the description of reviews. The analysis may lead to describing the parts of the review such as paper title, category, subject area, confidential comments, positive aspects, negative aspects, assessment of quality, assessment of readability, etc. This can be described in an abstract way using data types such that all possible reviews would become instances of that data type. Later on, such a data type will be called a content type.

Globally, however, we may need to organise the data in a different way. For instance, in the database of our conference organisation system we might find additional information on the programme committee member and additional reviewer(s) of a paper that are not visible to all programme committee members. However, it must be possible to map the database to the content type. Such a mapping is called a view. In our example we would simply have to extract just the information on the review.

With respect to the dynamic component, we would have to add operations. Locally, these operations are determined by the user actions associated with a scene. Globally, we would need operations on the database — to be precise, these operations will be transactions — which realise the operations on the media type. Of course, the information production, i.e., the data requested from the user, must become the input of the operations.

How is conceptual modelling related to the dimensions that we discussed in connection with the abstraction layer model? Globally, we now provide a database schema for the static part (data specification) and database transactions (function specification). Locally, we have the analog of media types. The static part (view specification) is defined by views on the database schema; the dynamic part (dialogue specification) is given by the operations associated with a media type.

The result of this layer will be a media schema, i.e. a collection of media types, which adequately represent the scenarios. Details on media types are handled in [28].
D. Presentation Layer

The conceptual layer has produced the collection of media types. The content and functionality of a system is already fully determined by the media types. It has also been verified that the intention and usage of the system is adequately supported by the media types. So, the remaining problem is the presentation, which is addressed on the presentation layer.

In general, a media type defines a set of media objects. The actual media objects depend on the state of the database. Each of these media objects is an abstract representation of a web-page. So, the major activity on the presentation layer is to specify style options for the pages. This will be done by enhancing the media types with the style options.

E. Implementation Layer

The implementation layer deals with all activities that are left to transform the presentation enhanced media schema into an executable and accessible web-based system. As a result of the higher layers we obtain a media schema, which consists of a database schema and a collection of media types. Each media type is defined by a view on the database schema and operations. Furthermore, the media type is associated with presentation style options. Therefore, the activities on the implementation layer are the following:

- Define a logical database schema and implement it by a database management system.
- Define the views of the media types using the database query language.
- Generate web pages (HTML-documents) from the media types and the style options as far as possible.
- Implement the operations on media types using scripting languages.

III. STRATEGIC CHARACTERISATION OF A WIS

The strategic layer of a WIS is meant to set the target for the models that are to be developed at the lower layers. We will characterise a WIS by

- a mission statement describing in general terms what the WIS is about,
- a utilisation space describing content, functionality and context,
- a utilisation portfolio describing actors, goals and tasks, and
- general principles describing the ambiente and desired atmosphere of the WIS.

The last part will guide the formation of the WIS.

A. Mission Statement and Brand

The brand of a WIS is based on a rough classification scheme for WISs. This classification scheme has the form \( \mathcal{P}^{W}2\mathcal{U}^A \) and represents in an extremely terse form the following very general information:

- \( \mathcal{P} \) stands for “provider”, and thus indicates which role the system plays. This specifies very roughly what kind of content can be expected from the system. For instance, if the provider is a bank, the provided services will most likely center around accounts, investments, savings and loans.
- \( W \) stands for “what”, and thus adds more detail to the kind of content offered by the WIS. For instance, if the provider is a bank, the provided content may just be accounts, investments, savings, loans, and mortgages.
- \( U \) stands for “user”, and thus indicates to whom the services offered by the WIS are directed. For instance, if the provider is a bank, the users are probably just the customers, enterprises or other banks.
- \( A \) stands for “actions”; and thus indicates the functionality of the WIS offered to its users. For our example of a bank offering accounts, investments, savings, loans, and mortgages possible actions can be apply_for_loan, apply_for_mortgage, set_up_account, buy_stock, etc.

The brand is usually the result of a brainstorming activity discussing the what, whom and for whom of the WIS. The aim is to fill these general placeholders \( \mathcal{P}, W, U \) and \( A \) with meaningful terms that describe the WIS in very general, terse terms. We shall look at this brainstorming activity in more detail in Section IV.

The brand gives a rough picture of the content and functionality of the WIS and its users using only descriptive keywords. This is, however, a valuable source of information for refinement using linguistic methods. We will look at these pragmatic methods for strategic analysis also in Section IV.

The mission statement complements the brand by an informal, textual description. The importance of having it was emphasised in [7]. Each of the actions in the brand are taken as the major tasks. For each of them the mission statement describes, which types of users are involved, which activities they are supposed to execute, which content will be provided for them and requested from them, and what will be the results of these activities. However, no attempt is made to decompose the tasks or to refine them, as this is left to storyboarding [28].

Furthermore, the mission statement contains metaphors that turn out to be adequate for describing the activities associated with the WIS. These metaphors refer to the content and functionality keywords used in the brand.

In addition to its descriptive character the mission statement also has an explanatory character in the sense that it contains the reasons for setting up the WIS. That is, the mission statement will describe what the major and minor purpose of the system is, how each task will contribute to these purposes, and what the benefits of the system for the provider and the users will be.

Example 1: Let us consider the example of a WIS that deals with loan applications. In the case the provider is a bank, and the content will be centered around (personal) loans and mortgages. The only users we think of are customers, and the tasks they execute are applications for loans and mortgages, respectively. This leads to the following brand:

bank\_loan, mortgage\_loan, apply\_for\_loan, apply\_for\_mortgage

The mission statement is the following informal explanation for the brand:
The mission of the system is to provide on-line access for customers to personal loan and mortgage applications. The system will provide information about the available types of loans including conditions for repayment, i.e. principal and interest, conditions for creditworthiness, and intended purposes of the loans. For mortgages this further includes securities and conditions for bailsmen. The information about loans will be complemented by easy loan examples. Then the system will allow customers to enter their personal data, select the appropriate loan, and check, whether their personal finances are in accordance with the rules for repayment.

The major purpose of the system is to open an additional sales channel, which is addressing mainly those customers who are capable of dealing with electronic systems, do not need intensive advice, and therefore prefer the convenience of avoiding personal contact at a branch office. A secondary purpose is to improve the efficiency of banking in the loan sector.

The expected benefits of installing the system for the bank are a closer binding of customers to the bank, the possible attraction of new customers, and an improvement of cost efficiency, while at the same customers benefit from increased availability of bank services.

In general, it is sufficient to formulate the mission statement using free-form text as in Example 1, but it is also possible to use semi-formal structured text. In doing so the brand and mission statement take the following form:

| Content: | loan, mortgage |
| Users: | customer |
| Tasks: | apply_for_loan, apply_for_mortgage |
| Major Purpose: | open an additional sales channel address technology-experienced, informed, goal-oriented customers |
| Minor Purpose: | improve banking efficiency in loan sector closer binding of customers attraction of new customers improvement of cost efficiency increased availability of bank services |
| Benefits: | |

Furthermore, we obtain the following informal descriptions for the tasks apply_for_loan:

| Task: | apply_for_loan |
| Description: | The system will provide information about the available types of loans including conditions for repayment, i.e. principal and interest, conditions for creditworthiness, and intended purposes of the loans. The information about loans will be complemented by easy loan examples. Then the system will allow customers to enter their personal data, select the appropriate loan, and check, whether their personal finances are in accordance with the rules for repayment. |
| Participants: | customer |
| Required Content: | list_of_loans, loan_conditions, loan_purpose |
| Produced Content: | loan_application, customer_data |
| Result: | confirmed_loan_application |

The description of the task apply_for_mortgage will look similar.

B. Utilisation Space

The term “utilisation space” is used as a metaphor to characterise the WIS as a space, through which a human user can navigate. As such it has to cover mainly the content, functionality and context in general terms. The goal is to enable optimal orientation in the utilisation space, such that searching and finding information needed for certain tasks will be facilitated.

The type of content is already characterised by the brand, to be precise by its what-part. This gives a set of nouns describing the content in coarse terms. Similarly, the what_for-part of the brand gives verbs describing the functionality, i.e. what to do with the content.

The utilisation space will now add details to content and functionality, set the nouns and verbs used in the brand into relation, and place both into a utilisation context. This will be done in the following way:

- Refine the content keywords and place them in semantic relationships. These relationships can capture speciali-
sation, part-of relationships, or associations of global context with details. They indicate navigation facilities and order principles among the content. Word fields are a valuable tool for the refinement (see Section IV).

- Refine the functionality keywords to discover various facets that can be placed in semantic relationships in the same way as the content. Again, word fields are a valuable tool for the refinement.
- Relate the functionality with the content, i.e. specify in which context a particular content is needed, i.e. which content is needed by which activity, which content is produced by which activity, in which order (if any) the content will be used by an activity. In doing so, we obtain a progression model for the functionality.

Semantic relationships for content – and similarly for functionality – can be represented by rooted trees, where the root is defined by a keyword taken from the brand. Thus, we can obtain the following more detailed description of content items:

<table>
<thead>
<tr>
<th>Content Item</th>
<th>Derived From</th>
<th>Relationship</th>
<th>Usage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(name)</td>
<td>(content item)</td>
<td>(description)</td>
<td>(list of tasks)</td>
<td>(textual description)</td>
</tr>
</tbody>
</table>

In the same way we obtain more detailed description of tasks:

<table>
<thead>
<tr>
<th>Task</th>
<th>Derived From</th>
<th>Relationship</th>
<th>Description</th>
<th>Participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>(task name)</td>
<td>(task name)</td>
<td>(description)</td>
<td>(textual description)</td>
<td>(list of users)</td>
</tr>
</tbody>
</table>

Example 3: Figure 2 represents a tree of content items with root loan, which is specialised by personal loan and mortgage. Details for mortgages have been omitted, but for personal loans the three decisive facets – conditions for obtaining the loan, i.e. creditworthiness, conditions for loan repayment, i.e. life span, principal and interest, and loan purpose have been indicated. Similarly, Figure 3 represents a task tree with root apply_for_loan, which again is specialised by apply_for_personal_loan and apply_for_mortgage. For the latter one further details have been omitted. For apply_for_personal_loan the components contributing to selecting terms and conditions, outlining personal finances, declaring the purpose of the loan, and entering customer details are shown in the tree.

Fig. 2. Semantic tree of content items in loan application

Fig. 3. Semantic tree of tasks in loan application

The description of tasks may be extended by

<table>
<thead>
<tr>
<th>Required Content</th>
<th>Produced Content</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(list of content items)</td>
<td>(list of content items)</td>
<td>(textual description)</td>
</tr>
</tbody>
</table>

Typical situation parameters are preconditions and triggering conditions for task enactment, priorities among tasks, frequency of tasks, and repetition patterns. Acceptance constraints are used for the internal control, whether a task can be considered to be completed. The execution context is used for the description of duration restrictions, foreign interaction, scope, and robustness.

The decisive criterion for orientation in the utilisation space is the preservation of context. This can be achieved, if for each task a mental model can be constructed. This model captures the progression of content perception over a time axis. It represents a system map for the user showing information that has already been used and processed.

For strategic modelling it is important to reflect, whether the stages that represent perception of content are logically
connected. It is also important to see that later stages enable the possibility to be redirected to information that was already processed earlier.

If available, metaphors may help to set up this mental model. For instance, the “desktop” metaphor is a well-established tool that has significantly contributed to ease the use of computer technology by technical laypersons. Similarly, the “shop” metaphor is frequently used in commerce applications. We will discuss metaphors in Section IV.

Example 4: We may regard the task of personal loan application in the loan application system from Example 1 as a matching problem. We have to match the needs of a customer with the purposes of the available loans, and the payments arising from loan conditions with the payment latitude of the customer, which is determined by creditworthiness conditions set by the bank and the personal finances, i.e. income and obligations, of the customer.

Thus, a suitable mental model consists of the set of loan options, i.e. loan type, conditions and purpose, and the implications of each option with respect to the payments. This model develops over time in a way that non-suitable options are deleted first, then a selection is made, and finally, organisational data are added to turn the selected loan option into a full loan application. This also indicates the required functionality, which must comprise selecting information about loan options, tentative and affirmative selection of such options and decision aids.

Each characteristic aspect of a WIS can be associated with a specific context. In general, we distinguish between the following different kinds of context:

- The general context associates certain characteristics of mission, utilisation space, utilisation portfolio, and general principles to other characteristics of the WIS.
- The usage context associates scenarios that describe the behaviour of users with other such scenarios.
- The website context specialises scenarios based on the supported environment, the provider and software and hardware environments.
- The runtime context permits the incorporation of the current situation and the information on the current usage of the WIS into the offered content and functionality.

The different kinds of context permit a layering and stepwise incorporation of context information into the WIS based on the abstraction layer model. The general context is specified by content items, functionality, actors, goals and tasks as shown in Figure 4, thus can take the following form:

| Content context: | (list of related content items) |
| Functionality context: | (list of functionality) |
| Actor context: | (list of cooperating actors) |
| Goals context: | (list of related goals) |
| Task context: | (list of related tasks) |

C. Utilisation Portfolio

The utilisation portfolio complements the utilisation space emphasising the whom-part of the brand. Thus, it is mainly concerned with the WIS users, for which we will later adopt the term “actor”, their goals and the tasks that have to be executed to achieve these goals.

Tasks correspond to the actions in the brand and their refinement in the utilisation space. So we can assume a task hierarchy emphasising specialisation between tasks and decomposition of tasks into subtasks, as long as these can be described in a simple way.

The users used in the brand and mission statement will be roughly classified according to roles they have with respect to the WIS. Each role has particular goals, and each of these goals corresponds to a task that is meant to achieve this goal. This does not mean that the task has to be executed by the user in this role; it may well refer to tasks executed by users in other roles. Tasks are broken down into subtasks to a level that elementary tasks can be associated with a single role. In addition, subtasks should refer to subgoals.

Furthermore, we obtain dependencies between goals, e.g. being a subgoal, a specialisation, or any other kind of dependency. Thus, we complement the informal description of the system by adding goals:

| Goal: | (goal name) |
| Derived From: | (goal name) |
| Relationship: | (description) |
| User: | (role name) |
| Description: | (textual description) |

The relationship between tasks, roles and goals can be represented in a graph, which we call a task-goal graph. In these graphs we have three different types of vertices for actors, tasks and goals, respectively. Furthermore, we have five different kinds of edges for task-goal relationships, involvement of an actor in a task, goal-goal-relationships, as well as for subtasks and task specialisation. Figure 5 shows the legend for such graphs.

Example 5: The task-goal graph in Figure 6 illustrates goals, tasks and actors in a loan application. In this case the goal buy_new_car depends on obtaining a personal loan, i.e. on the goal personal_loan. Sufficient for achieving this
goal is the successful execution of task approve_personnel_loan, which has to be done by a bank clerk. However, this task will only be triggered by a task apply_for_personnel_loan to be executed by the customer. This task is thus necessary for the goal. Furthermore, the task decomposes into four subtasks select_conditions, enter_customer_details, declare_purpose and set_up_budget.

A valuable tool for setting up the utilisation portfolio is communication analysis, which addresses how a user will communicate with the WIS and why this communication is the best for the provider and the user. We shall discuss this in more detail in Section IV.

D. The Atmosphere of a WIS

While the brand, mission statement, utilisation space and utilisation portfolio aim at the characterisation of content, functionality and usage of the WIS in strategic terms, the atmosphere addresses the gestalt of the WIS, i.e. how the WIS should be configured. At the end the WIS will be implemented by and presented through web pages, which should convey a uniform impression to the WIS users.

Categories characterising the impression of pictures can be used such as energetic, romantic, elegant, refreshing, harmonic, or stimulating. Each of these categories will have implications on the choice of form and colour [24]. On the strategic level the choice of one of these categories corresponds to the question which impression the WIS shall convey. The question is, which atmosphere is best suited for the envisioned content and functionality.

Example 6: In Example 4 we characterised the application for loans as a matching problem. Choosing a harmonic presentation for the WIS would suggest the intention to find an “optimal” solution for the customer and the bank. Choosing a stimulating atmosphere might suggest to encourage the customer in his/her application. An elegant atmosphere of the WIS may be chosen to convey confidence to the customer, i.e. that his/her financial affairs will be dealt with in the best way.

In addition, the atmosphere of a WIS is concerned with the progression patterns for the tasks. These patterns reflect the logical connection of information revealed to the user during the execution of a task. We distinguish between the following progression patterns:

- A circular progression pattern is centered around a particular content item, the phases of which form the core of the content to be delivered. At each stage details are added.
- A loop progression pattern emphasises the iteration of content, each time taking a different perspective. This is similar to a circular pattern, however puts more emphasis to changes.
- An incremental progression pattern emphasises the development of several content items over time. At each stage some of the items may be completed.
- An evolutionary progression pattern emphasises the stages of the content items, in particular those that are used in the result of tasks. At each stage content item
may still be incomplete.

- A network progression pattern emphasises the flexible treatment of content items during the development of tasks and the logical connection between various such objects.

**Example 7:** If we choose a circular progression pattern for a personal loan application, the presentation of information will be centered around the loan, each time gaining a clearer picture of the result. In Example 4 we explained that it would be a good idea to successively discard possible loan options. This is in accordance with circular progression.

An incremental progression pattern would emphasise the various components of a loan application, i.e. the customer data, the conditions, and the budget. Each of these components would be treated as a separate item. While this is common in many WISs that offer form-oriented access, it may not be the best choices, because it does not support well the interaction between a tentative choice of conditions, the calculation of repayment costs and the personal financial situation of the customer.

An evolutionary progression would be very similar to an incomplete one. However, each component could be left incomplete. This would help with the problem of changing conditions.

A loop progression would be similar to a circular progression. The difference is that the circular progression emphasises more the narrowing of options, while a loop progression would permit returning to options that have already been discarded.

Finally, a network progression is again similar to an incremental one, but leaves much more flexibility, as to fixed order of the components is presumed.

Progression patterns have a direct impact on the placement of content on web pages, thus on the tiling of pages and the mapping of content to the tiles. Such a mapping of patterns to web page grids is discussed in detail in [24].

**IV. STRATEGIC ANALYSIS**

Let us now address guidelines for strategic modelling. As already indicated in Section III, we will discuss linguistic analysis, communication analysis, and metaphors.

**A. Linguistic Analysis**

Linguistic analysis addresses the logical connections between the tasks, the content and the goals of users using techniques that are applied to analyse natural language descriptions of users’ activities. The major source of information used for developing information systems is a description of processes, which may be delivered verbally or in the form of texts. Linguistic analysis considers such natural language descriptions and uses them for analysing the activities of the users: What are these activities? Does the description indicate any sequencing or continuation? What data is needed for or used by the activities? What are the relationships between these data? As user activities can be described by verbs, we suggest analysing the corresponding word fields.

According to [14] a word is an abstract concept, which in order to become concrete needs a word form carrying the grammatical variants. Word fields combine different aspects:

- morphology, i.e. the forms of the written or spoken word,
- phonology, i.e. the sound of the spoken word,
- syntax, i.e. the construction of sentences using the word forms,
- semantics, i.e. the meaning(s) of the word, and
- pragmatics, i.e. the usage of the word in written or spoken language.

If we restrict ourselves to written language communication considering the phonology becomes obsolete. Therefore, we tacitly assume that verbally communicated information in a formalised language usage context such as business communication can also be delivered in textual form without loss of semantics. If we had to analyse everyday language, this would no longer be true.

A word field [19], [29] is a linguistic system in which similar words that describe the “same” basic semen and are used in the same context are combined to a common structure and data set.

In contrast to common synonym dictionary, word fields define the possible and necessary actors, the actions and the context. Word fields can be used for verbs, nouns and adjectives. We focus on verb fields and extend the implementations of the WordNet dictionary for verbs. Word fields are well-specified for a number of verbs. For each verb field we can at least define the following abstract information:

- basic semen,
- context of usage,
- possible/necessary actors, actions and context, and
- the star or snowflake specification, which comprises
  - the logical structure specifying possible arguments,
  - the semantic description of relevant and irrelevant valences, and the association with semantic type (colour, space, time etc.),
  - the kernel semantics describing the word field with semen, and
  - example sentences providing additional information usable for help desk systems.

**Example 8:** A typical example is the verb “inform”, for which we can differentiate between four major facets of the word field:

1) To bring/advice something to somebody’s attention something: The word field is semantically characterized by: objectivity; functional information; official information; explanation; to be associated with something in future; using different representational media and presenters; with a degree of extraction from open or hidden; variety of styles such as short content description, long pertinent explanation, long event-based description. Typical related verbs are: announce, notify, publicize, publish, explain, present, herald, report, describe, outline, portray, passing a message, and scout.
2) To bring something that is unknown or hidden to somebody’s attention: The word field has a variety of presentation varying from proving a statement, arguing on latent facts, or unintentionally stating away. The association to the senders way of thinking and acting is more strict than within the first variant. It is more personalized. Typical synonyms or associated verbs are: explain, report, clarify, elucidate, prove, admit, confess, scout, and betray.

3) To react on events or statements: The reaction may be positive or negative. Typical application pattern are negotiations and adjustments. Typical associated other word fields are: confirm, approve, dispute, negate, and prohibit. notify, state, announce, promise.

4) To make suggestions for future activities: A business site or an advertisement site does not display information without intention to generate surplus value. The intention of displaying information within an entertainment site is often to attract readers to become involved. Typical related actions are described by words such as propose, register, and offer.

We can generalise the theory of word fields as shown in Figure 7. Generalised word fields are significantly different from word forms that carry the grammatical variants [14].

Under this premise we understand a word field to consist of its intext, its context, its semantics and its pragmatics.

- The intext combines morphology and syntax. We are mainly interested in the latter one, especially for verbs. In this case we may ask for the basic syntactical form, various mandatory and optional arguments, variants of the arguments, extensions to the basic syntax, and relationships and constraints between the arguments of the word when it is used in sentences. For instance, the verb “to give” uses a mandatory argument “something” and an optional argument “to somebody”. The basic syntactic form “Someone gives something to somebody” expresses an action, specifies the actor and the receiver, and an object used in the action.

- The context refers to application areas. According to the context we may identify related words, concepts related to the arguments, categories of actors and actions, and further constraints. For instance, in a banking context the verb “to give” may be used in a sentence such as “The bank gives a home loan to the user”. In this case we conclude that the object of the action is subject to contracted conditions regarding the use of the money for the claimed purpose, payment of principal and interest. A more precise statement would replace the verb “to give” by “to grant” or would replace the whole sentence by “The bank offers a home loan contract to the user”, in which case it would become clearer that a follow-on action, i.e. the acceptance of the offer, is expected.

- The semantics refers to the meaning of a word in a particular context. For example, the already used sentence “The bank gives a home loan to the user” includes the meaning of setting up a contract.

- The pragmatics refers to how the word is used in the application context.

Word fields have been investigated in more detail in [10] extending work in [11].

Example 9: In electronic commerce verbs describing activities around contracts can to a large extent be considered as specialisations of the verbs “to give”, “to take” and “to negotiate”. We will concentrate only on the first two of these, as the treatment of negotiations leads to a field in its own right.

As to the intext, the basic syntactic form associated with the verb “to give” is to give something to somebody. This indicates the existence of two roles, the giver and the taker, the involvement of an object, and an activity of the giver handing over the object to the taker. Of course, this also indicates a follow-on action with interchanged roles, which can be described by using the verb “take”.

The verb “to give” is quite abstract leaving enough space for interpretation, i.e. variants and constraints. For instance, the activity may be unconditional or conditional. In our example sentence “The bank gives a home loan to the user” the action definitely requires the follow-on acknowledgement to happen, whereas a sentence such as “The company delivers the ordered goods within 10 days.” — note that “to deliver” may be considered as a specialisation of “to give” — indicates an unconditional action. Further variants of “to give” are connected with specification of how the giving-action is to be executed. Unless the respective conditions are explicitly stated or generic business conditions apply, all variants are admissible. The verb itself does not express them.

The semantics of the verb “to give” depends on the context in which it is used. In a banking context it refers to the objects a bank can give to its users and vice versa, e.g. money, information, stock options, accounts, securities etc. In many cases the giving-action is coupled with choices and offers, which are not self-contained, but require follow-on actions. According to the more specific objects involved in the actions, more specialised verbs such as “to offer”, “to inform”, “to grant”, “to propose”, “to present”, “to acknowledge”, “to confirm” etc. are used. The related word fields for these verbs give more details on the semantics of a sentence in the give context.

As to the pragmatics, the use of the verb “to give” usually indicates that a follow-on take-action is assumed. Depending on the used specialisation, the continuation will also be specialised. For instance — staying within the banking context — the use of the verb “to offer” can be followed by “to select”, “to choose”, “to accept”, but also negated forms such as “to reject” or questioning conditions for taking are possible.

The verb “to take” is almost dual to the verb “to give”. Therefore, we dispense with a detailed discussion of its word field.

Example 10: Let us look again at loan applications focusing on the task of setting up a budget. This scene arises from a sentence such as “The user and the bank set up a budget for the requested loan”. Several word fields will be used in the
analysis of this sentence. Central will be the verb “to set up”, which in general means to collect all the details describing the object argument, i.e. in this case “budget”. In the banking context this refers to the collection of information about debits and credits of the user including the payments for the loan.

This is determined by the word field of “budget”, which immediately leads to two components associated with the debits and the credits, respectively. Furthermore, the word fields for “credit” and “debit” will indicate further specialisations, leading to the collection of information about salaries, rents, interests, rates, living expenses, etc.

B. Communication Analysis

Communication analysis addresses how a user will communicate with the WIS and why this communication is the best for the provider and the user. This helps to understand user goal and set up the utilisation portfolio accordingly [26]. Communication analysis is based on results in communication theory [5].

If a user can get the impression that guidance and information are as good as would be expected from a human, the system will more likely be accepted and thus be successful. From this we draw the conclusion that the WIS has to be defined in a way that best reflects the user-business communication in case there were no system support. Therefore, it appears to be a good idea to start with an analysis of this communication.

Communication analysis summarises the activities and techniques that are applied to understand the typical human-to-human interaction in the application domain, and classifies typical communication barriers [15]. We use certain dimensions of understanding messages as a framework for understanding communication. These are used to identify communication flaws. We proceed with identifying different types of communication barriers, which affect the communication. Then we argue that using localisation abstraction and metaphors can help to overcome communication barriers, implying that their use may enhance the user’s understanding and successful navigation through a WIS.

We may understand communication as an exchange of messages. Here we deal with semantic and pragmatic aspects of these messages. In particular, we are concerned with successful communication, i.e. communication by means of which both partners do meet their respective goals. We can distinguish four main dimensions of understanding messages depicted in Figure 8:

- **Content**, i.e. what the message is about;
- **Presentation**, i.e. what the sender tells about the message and the sender him/herself;
- **Relationship**, i.e. what the sender thinks about the receiver and their relationship, and what the receiver thinks about the sender;
- **Appeal**, i.e. what the sender wants the receiver to do.

![Fig. 8. Dimensions of understanding messages](image-url)

In WISs we mainly have to deal with written messages. These dimensions, however, apply to spoken messages, but can also be applied to written messages. Assuming that a necessary condition of non-successful communication is that at least one message is flawed with respect to one of the dimensions, we can distinguish the following message flaws:

- **Content flaw**, i.e. the sender does not say what s/he wanted to say, or the receiver understands the utterance...
differently:

- **Presentation flaw**, i.e. what the sender tells about him/herself is inaccurate, or the receiver understands it incorrectly;
- **Relationship flaw**, i.e. what the sender thinks about his/her relationship to the receiver is incorrect, or the receiver understands it incorrectly;
- **Appeal flaw**, i.e. what the sender wants the receiver to do is not persuasive, or the receiver does not correctly understand the (valid) appeal issued by the sender.

Clearly the above mentioned dimensions of message understanding are not really independent in a formal sense and neither are the flaws. But as can be seen below, they help to formulate and classify communication barriers that appear to be realistic.

Localisation abstraction addresses problem arising from content, presentation and relationship flaws. As the communication we are dealing with here is a mediated human-to-human interaction, various kinds of knowledge and ability have been represented inside the web information system as data or program. The information system in response to a user’s inquiry chooses the data or program best suited. This requires having a model of the user (type) represented inside the web information system as data or program. The information system further obtains an answer to the inquiry that can be just the identified data as it is stored in its database. However, it can also be the result of applying business functionality to these data. Finally, the application actually delivers the answer to the user. Since the functionality required by the user is distributed over the information system’s information space, the user might not be able to access it immediately, but might first need to position him-/herself on the most suitable location. Thus communication gets mixed up with navigation. Therefore, simplifying navigation can result in significantly reducing communication barriers. The simplification of choice is realised by supporting tools.

At least the following navigation functions appear worth supporting with tools:

- **Position signalling**, a function pointing out to the user his or her actual position in application space.
- **Heading determination**, a function determining the direction and also maybe the means best suited to approach a certain position.
- **Short distance environment exploration**, a function used to explore the immediate neighbourhood of a given location.
- **Long distance environment exploration**, a function used to explore the regions of the application space that are far away from a given location.

In a web information system these functions can be supported by the use of **localisation abstraction**. A particular view on the system’s information space model is associated with a user and may be displayed to him or her. The user’s location in the application space and his/her favourite locations may be part of this model. The model may be presented as a graph. Its vertices could represent locations and the edges the connections between them. The vertices could be labelled by the functionality, i.e. knowledge or ability that is available at the specific location. After each navigation step the model must be updated. Algorithms are then needed for obtaining, updating and displaying the models. It should be possible to obtain the needed results on the fly if the intended model is not too sophisticated.

Using the localisation abstraction properly can thus target navigation type barriers, as it reduces the perceived complexity of the information space and thus increases the users’ ability to aim for his/her purpose. Reasonably chosen metaphors may help the users to apply their knowledge and abilities in a situation (i.e. while working on the information space) they are unfamiliar with. Using metaphors may cause users to be much more efficient in identifying the functionality required for their purpose, move to there, invoke the functionality and thus meet their goals more efficiently.

**Example 11**: Let us look at communication flaws in a system dealing with loan applications. Content flaws can be the following:

- Users have a variety of knowledge levels concerning loan types offered, their application areas, the terms and conditions, the assessment criteria, etc.
- Users may or may not have a clear picture about what the loan is needed or going to be used for. Some may have only a vague idea of the purpose.
- Users may – depending on their level of education – easily understand the differences between loan types and their respective suitability or not.

Presentation flaws in a loan application system can be as follows:

- Users may have a clear mind about their financial capability, and thus be able to set up a reasonable budget. Others may over- or under-estimate their capability to afford the payments that will be required in case of application approval.
- Users may have sufficient computer literacy to handle an electronic system without difficulties or need a lot of help in doing so.
- Users might – due to their insufficient language fluency – need specific support while filling in the application form.
- Users might – due to their cultural or social background – not like to outline their financial details to the bank (or a computerised system).

Relationship and appeal flaws can take the following form:

- Users might – due to their personal background – tend to uncritically follow bank staff’s advice.
- Users might – due to their cultural background, or family related reasons – not dare apply for a loan.
- Users might – due to their insufficient understanding of banking business or their difficult state of affairs – just
appeal for help and overlook the fact that the bank only in rare cases can afford not to benefit from accepting a loan application.

C. Metaphors

Metaphorical structures, i.e. metaphors, allegories, metonymies or synecdoches, can be used to help the scanning or zapping user to understand a page or a website. In general, metaphorical structures have a communicative or cognitive function [6], [9]. For example, many users do not realise that within a computer context a trash often stands as a metaphor for the action of deleting files. The interpretability of metaphorical structures depends on the common experience of the sender and the receiver. Metaphors such as folder, files, trash can, and recycle bin are common to computer users but not so common to naive users. However, metaphors can also mislead users, e.g. the use of a trash box for ejecting disks in an Apple environment or the use of a clipboard for singleton messages in a Windows environment. Naive users are not acquainted to these metaphors. Thus, metaphorical structures, if they are to succeed, must be familiar to users. The users can understand content, functionality, and intention intuitively via using such metaphorical structures. WIS development can profit from well-integrated metaphorical structures that aim at focusing on a deeper context explanation as well as on a more specific navigation and selection.

Metaphorical structures can be used in the context of WIS design for supporting the scanning or zapping user and for the set-up of a story for the WIS. They can help introducing the content, style and “the feeling inside” of a complex web presentation. Mental models are also used in [31] to define persistent internet-metaphors: digital library, electronic mail, electronic marketplace, and digital world. These metaphors are based on ancient myths and archetypes that have influenced human thinking for thousands of years: keeper of knowledge (the digital library), communicator (electronic mail), the trader (electronic marketplace), and the adventurer (digital world). The internet representation of these metaphors is a matter of common knowledge, but the fact of using metaphors is mostly not known to users.

Metaphorical structures lead to more sophisticated user interfaces [17], [22] which are intuitively understandable, which can be used intuitively, whose “philosophy” or “mission” can be captured within seconds by novel users and need not be learned – a requirement sometimes called “grandma-safe”.

Metaphors should meet requirements such as simple interpretation, common sense of the meaning, abstraction in the right (user-centered) way and integrability into the dialogue [25]. In our daily language, metaphorical structures have great power, so why should we not use them within WISs? In order to do that, we have to integrate the metaphorical structures into the complete WIS development.

The summary of [21] in [18] states that metaphors refer to properties of concepts rather than words. Consequently their function is to help understand concepts rather than being just aesthetic or artistic expressions. Metaphors are an intrinsic and indispensable part of everyday language; they are easily used by “ordinary” people, and they are inevitable in human thought and reasoning. This is rephrased in [20] saying that the new view of metaphors takes imaginative aspects of reason – metaphor, metonymy, and mental imagery – as central to reason rather than as a peripheral and inconsequential adjunct to the literal. Thus, we may say that human communication is essentially metaphorical.

In general, a metaphorical structure [32] is the unusual usage of a language expression, i.e. using a language expression in a meaning that is not expected in the application context. The language expression is used as a language pictorial which works on the basis of a similarity of two objects/words. These objects/words are described by some dominant properties [2].

For metaphors there are two different definitions. According to the traditional definition a metaphor is characterised by a substitution of names. For instance, the notion “virus” stands for a “damaging computer program”. A second, more modern definition characterises a metaphor as generating a new coherence of meanings. That is, two word fields are put into a similarity relation.

Example 12: Consider the sentence “URLSnoop is a Trojan Horse that gathers email addresses secretly”. In this case the characteristics of an event in ancient Greek mythology is projected onto the computer. In doing this the characteristics of the computer update the ancient myth leading to a modern serial of the Troy story.

An allegory is an extended metaphor that stands for a complex idea.

Example 13: “Orchestra” is used as an allegory for a group of people working together, e.g. a company. Related metaphors are the “conductor” for the executive manager of the company, the “first violin” for the star designer or the chief programmer, and “playing together” for working together. Thus, the allegory “orchestra” generates a complex field of several related metaphors.

A metonymy is a replacement of a term by something that is related to this term. The relation can be real, mental, factual, or causal.

Example 14: In the sentence “This Kafka is interesting” the author Kafka stands for one of his books. In “The Kielers will win the German League” the citizens of Kiel are standing for the Kiel handball team.

Synecdoches are closely related to metonymies. In contrast to a metonymy a synecdoche stands for a part-whole-relation. For instance, in the phrase “to start a computer” the notion “computer” stands for its operating system.

In addition to these four further metaphorical and rhetorical structures are known such as synonyms, paraphrases, analogies, oxymorons, emphases, or ironies.

The generation of metaphors could be supported by logically decomposing the information space into a small number of domains that appear homogeneous with respect to the offered functionality. This allows one to relate the domains to user types and expected user actions. Generation of metaphors appears then as being connected to finding characterizing
names for the three valued relationships mentioned between user types, expected user action and domain. It might be useful for this to omit user related information from the user type that appears not to be relevant for the expected behavior. Further, methods from requirements engineering such as role-playing or brainstorming can be applied to generate first ideas on the metaphors to use. Clearly one must avoid, for a given user type, having too many, conflicting or contradicting metaphors built into the web information system.

In order to use an appropriate linguistic representation for a metaphorical structure the following questions has to be considered:

- Who will the user be (experiences, age, knowledge etc.)?
- What kind of communication is used (verbal communications vs. written communication vs. internet communication etc.)?
- What are the possible communication objects (e.g. verbal communications have the gestures; the internet communication has the colours and icons)?
- Which connotations (i.e. social and emotional side conditions) should not be used?
- Which cultural background does the user have?

In general, metaphorical structures are used with different intentions. They can be predicative or persuading. Attributive, genitive, and apposition metaphorical structures add some information. Icons or colors illustrates properties. Thus the choice of a representation form is restricted. It is not possible that all cognitive characteristics can have all representation forms.

Metaphorical structures can be classified as follows:

- **conventional** metaphorical structures on the basis of stereotypes such as a phone icon;
- **ex-metaphorical structures** in technical terminology;
- **creative** metaphorical structures which require greater reasoning abilities from the user;
- **re-metaphorical structures** through activation of images with modification.

**Example 15:** Let us continue Example 11 and explore metaphors that could be used in loan applications. These metaphors correspond to message flaws to be dealt with below. These flaws are consequences of specific perceptions of the applicant itself, the bank and its staff. Since we are dealing with a quite formalized business we think these perceptions should be based on a conceptualization of this business. We identify three main roles (besides the decision-making) played by bank staff. The first role is that of an information provider, which leads to the dictionary metaphor. The second role is that of a helpful assistant who answers questions on how to fill in the form. This gives an operations manual metaphor. The third role is that of an adviser who provides advice about the most suitable loan type or the acceptability of the user for a particular loan. This gives an adviser metaphor.

The communication between a user applying for a loan and a bank employee dealing with loan applications is mainly determined by the metaphors described above, and some communication barriers. Recall that the identified metaphors mainly relate to the role of bank staff as seen by the user. To some degree this is a consequence of the fact that the loan application processing is highly rule based and only leaves a little latitude to bank staff.

Let us finally briefly consider communication barriers impacting the traditional loan application process. The level of education may be a barrier to successful communication between a user and bank staff. This is reflected in the user dimensions and should lead to further explanatory system components. Another severe communication barrier is the user’s fear or bluff as its counterpart. A user might fear the loan application not being accepted, especially if the loan is essential and the financial situation is tough. Communication barriers may also arise due to gender, age, ethnicity, cultural or family background.

Dealing with bluff and fear requires the system to be as transparent as possible and to emphasize the identified advice component. Inabilities with respect to language can be dealt with based on a multi language feasible design. Cultural background and ethnicity up to some degree can be reflected by the system supplying the applicable general legal regulations. Age and gender may be addressable by design and layout issues as well as focussing on the most required loan application areas and amounts.

## V. Visual Design Patterns

We will now discuss how the decisions made on the strategic level impact on the formation of the WIS presentation. We may assume that the result of conceptual WIS modelling is available, i.e. there is a clear specification, which content is to be presented, and how the different content units are logically linked together. This is captured by the media schema in [28]. So our focus now in on how the WIS content and functionality is to be presented to its users. We concentrate on the visual design, though audio design could be tackled as well.

### A. Basic Principles

Visual design patterns are composed of visual and functional building blocks. The visual building blocks correspond to the geometric partitioning of the screen, whereas the functional building blocks realise the access to the presented content. Thus, the functional building blocks correspond directly to the represented content and its organisation along the strategic progression patterns. They order the content, whereas the visual building blocks place the content on the screen using a flexible graphical structure with constant colour coding and repeating elements that reflect the functional order.

Within the limitations of this paper we concentrate on the visual building blocks. In doing so, we have to consider the following three aspects:

- the visual alignment and partitioning of the screen;
- the colouring with respect to functionality and aesthetics;
- and the perspective perception of the whole screen.
The visual alignment is based on a tiling of the screen as a two-dimensional surface. In general, we can divide the horizontal and vertical axes using grid points \( x_{\text{min}} = x_0 < x_1 < \ldots < x_k = x_{\text{max}} \) and \( y_{\text{min}} = y_0 < y_1 < \ldots < y_k = y_{\text{max}} \). A tile is defined by a rectangular region \([x_i, x_j] \times [y_i, y_j]\). Then use a partition of the whole screen into tiles.

Example 16: A very common tiling is obtained by using just 4 horizontal grid points \( x_0 < x_1 < x_2 < x_3 \), and only 3 vertical grid points \( y_0 < y_1 < y_2 \). Then define four tiles

\[
\text{up} = [x_0, x_3] \times [y_1, y_2] \quad \text{left} = [x_0, x_1] \times [y_0, y_1] \\
\text{middle} = [x_1, x_2] \times [y_0, y_1] \quad \text{right} = [x_2, x_3] \times [y_0, y_1]
\]

Usually, the “up” tile is used for some menu bar, the “left” tile for navigation links, the “middle” tile for the major content, and the “right” tile for side options.

The colouring scheme will be the major instrument to achieve the desired atmosphere as specified in the strategic model. For the perception of the presentation by a user the interaction of colours is decisive. The basis can be a colour chord consisting of \( n \) colours \((n \in \{2, 3, 4, 6\})\) that form a regular polygon in the colour circle. These are complemented by adding grey tones enabling to achieve contrast between light and dark colours. A quality contrast results from brightening the coulours of a chosen ground chord in the same way.

From the theory of colour harmonics it is known that the choice of colour chord impacts directly on the perception leading to the sensations such as warm, cool, cold, intensive, hot, light, dark, etc. Conversely, the desired sensation indicates guidelines for the choice of the colour scheme.

The colouring scheme also impacts on achieving a three-dimensional impression (if desired) or not. The technical means for achieving a perspective perception are contrast, colour perspective, depth of sharpness, and the differentiation of motif [24].

**B. Grid Geometry**

Following the general principles the grid geometry addresses the visual alignment. We now leave the simple tiling into several columns and rows aside and concentrate on grids, in which the visual building blocks have sizes following a rhythmic structure that can be expressed by a sequence of positive integers. Such grid models can then be combined with colour schemes in a way that the rhythmic proportions of the colours conform to the desired atmosphere.

We are particularly interested in the Fibonacci sequence, which is defined by the recurrence \( f_{n+2} = f_n + f_{n+1} \) with the starting values \( f_1 = f_2 = 1 \). This gives the well-known sequence \( 1, 1, 2, 3, 5, 8, 13, \ldots \). Solving the recurrence equation gives

\[
f_n = \frac{1}{\sqrt{5}} \left( \frac{1 + \sqrt{5}}{2} \right)^n - \frac{1}{\sqrt{5}} \left( \frac{1 - \sqrt{5}}{2} \right)^n
\]

involving the two roots of the equation \( x^2 - x - 1 = 0 \). The positive root is known as the number of the golden section, which played an important role in art and architecture, and appears naturally in nature.

A simple use of the sequence leads to the model of visual flags, which in fact dates back to Leonardo da Vinci. This is illustrated in Figure 9. The flags enable the selection of sections that are ordered according to some functional criteria.

In terms of tiling we simply use the Fibonacci numbers as the horizontal grid points, and do not bother about vertical grid points at all. Multiplying the Fibonacci numbers with a constant can be used to define a partition of the screen width.

![Fig. 9. The grid model of visual flags](image)

Another use of the Fibonacci sequence is to place squares with increasing side length \( f_i \) along a spiral as shown in the lower part of Figure 10. It is therefore called the Fibonacci grid model. In doing so we obtain a very nice tiling of the screen, in which the proportions of the square tiles are determined by the Fibonacci sequence. If each tile is associated with a colour of a well chosen colour scheme this enables to achieve the desired atmospheric effects as specified in the strategic model. The thesis [24] shows this combination of the Fibonacci grid with various colour schemes in different web application projects.

The Fibonacci grid can be combined with visual flags as shown in Figure 10. In doing so we obtain a visual alignment with the following characteristics:

- It can be combined with a harmonic colouring due to the proportions of the tiles that are defined by the Fibonacci sequence. This has been often exploited in art.

![Fig. 10. The Fibonacci grid model with flags](image)
• Its visual flags can be exploited for global navigation according to the functional specification.
• The implicitly given Fibonacci spiral can be used as a line along which the content progresses.

A romantic atmosphere can be achieved using a three-colour chord with pastell colours such as (light) blue, pink and yellow. Similarly to the energetic atmosphere yellow and blue in increasing brightness would be used for the flags and the initial tiles along the Fibonacci spiral, where pink would be reserved for the largest tile.

An elegant atmosphere can be achieved using a two-colour chord, e.g. red and green, in combination with grey. As before, grey would be reserved for the largest tile in the Fibonacci grid, while the other colours in increasing brightness would be used for the initial tiles and the flags, respectively.

A refreshing atmosphere can be achieved using a three-colour chord with light colours in a “temperature contrast”, i.e. the individual colours show opposite effects with respect to associated temperature. For instance, we might choose (light) purple, yellow-orange, and a blueish green such as cyan. The use on the Fibonacci is as before with yellow-orange for the largest tile.

A harmonic or balanced atmosphere can be achieved using a two-colour chord, e.g. dark blue and ochre, in combination with grey with grey being used for the largest tile. Similarly, a stimulating atmosphere can also be achieved using a two-colour chord, e.g. yellow-green with red-purple, in combination with grey. Figures 11 and 12 show sample pages for these atmospheric effects.

Fig. 11. Sample Page for Stimulating Atmosphere

Fig. 12. Sample Page for Harmonic Atmosphere

C. Atmospheric Effect of Colour Schemes

On the strategic level we specified the desired atmosphere of a WIS. For this we used categories such as energetic, romantic, elegant, refreshing, harmonic and stimulating. All these categories refer to a desired sensation that the WIS presentation should convey to the users. The question is, which colour schemes can be identified to support these desired effects.

For each category we first need a ground colour chord, which is extended to a quality contrast. In particular, colours will be brightened uniformly. Based on these base decisions we develop an ordering of colours. The colour ordering in combination with the association to grid tiles determines the overall effect of the colour scheme. The colour ordering is based on formal and functional criteria:

• Formally, the order is determined by the placement of the colours in the colour circle. It is known that equi-distant colours harmonise better.
• Functionally, the order of colours is determined by the subjective sensation and associations of a viewer. This is taken from psychological studies and centuries of experience in art.

An energetic or powerful atmosphere can be achieved using a three-colour chord with bright and high-croma colours such as a blueish purple with green and orange. The blueish purple colour with increasing brightness can be used for the visual flags. Orange with increasing brightness can be applied to the initial squares on the Fibonacci spiral, where green is reserved for the largest tile. However, a tile will always inherit the colour of its corresponding flag.

VI. CONCLUSION

In this article we discussed the strategic modelling of web information systems on the basis of the abstraction layer model from [28]. The starting point is an informal mission statement and a characterisation of the brand of the WIS, the latter one following a general classification scheme. Both together describe what the WIS is about, i.e. the purpose(s) of the system, and both result from brainstorming.

Going more into details we presented models of utilisation space and utilisation portfolio, which describe in very general
terms the content that is to be presented in the WIS, the functionality, with which this content can be accessed, as well as the users of the WIS, their goals, and the tasks that have to be performed to reach these goals. The fourth and last part of a strategic WIS model are general rules for the gestalt, atmosphere and progression of the system based on knowledge about the cognitive perception of form, colour and other style elements.

The strategic model complements the conceptual model of WISs that was presented in [28]. Furthermore, it shifts interface design to a higher-level of abstraction in a way that permits intentions, metaphors and context information to be exploited from the very beginning, whereas it is usually left only to the generation of pages using style patterns. This enables a much tighter coupling of the abstraction layers in WIS development.

REFERENCES


Thomas Moritz studied graphic design and scenography for television at the University of Arts in Berlin (Germany). During 1978 and 1994 he worked as scenographic designer, computer artist and as supervisor of animation at different television channels. Between 1995 and 1997 he was visiting professor for scenography at the University of Film and Television in Potsdam (Germany). In 1998 he was representing professor for media design and media psychology at Anhalt University of Applied Sciences. Since 2000 as visiting professor for media design at Brandenburgian Technical University at Cottbus (Germany). Major fields of interest are web design and graphical user interfaces.

Klaus-Dieter Schewe studied mathematics and computer science at Bonn University (Germany). In 1985 he received his Ph.D. in Mathematics from Bonn University. During 1985 and 1990 he worked with large industrial companies in the fields of Artificial Intelligence, Software Engineering and Office Information Systems. Returning to Hamburg University in 1990 he worked on Formal specifications and semantics and Database Theory. In 1995 he received the Habilitation (= D.Sc.) in Theoretical Computer Science from the Brandenburgian Technical University at Cottbus (Germany). From 1994 to 1999 he worked at the Computer Science Department of the Technical University Clausthal. Since 2000 he is Chair of Information Systems at Massey University in New Zealand. Since 2003 he is also the Director of the Information Science Research Centre at Massey University. Since his habilitation his major fields of interest are Formal specifications and semantics, Logic in Computer
Bernhard Thalheim studied mathematics and computer science at the Technical University Dresden and then at Moscow State University Lomonossov, from which he received a Ph.D. degree in 1979. Back in Germany he completed his Habilitation (= D.Sc.) in Theoretical Computer Science at the Technical University Dresden in 1986. After that he held full professor positions at Rostock University and the Brandenburgian Technical University at Cottbus. Since 2003 he is Chair for Databases and Information Systems at Christian Albrechts University Kiel. His major research interests are Database Theory, Logic in Computer Science, Design Methodologies for Integrated Information Systems, in particular Web Information Systems, and Database Component Ware.
Databases of Personal Identifiable Information

Sabah S. Al-Fedaghi
Kuwait University - Kuwait
sabah@eng.kuniv.edu.kw

Bernhard Thalheim
Kiel University - Germany
thalheim@is.informatik.uni-kiel.de

Abstract

This paper explores the difference between two types of information: personal identifiable information (PII), and non-identifiable information (NII) to argue that security, policy, and technical requirements set PII apart from NII. The paper describes databases of personal identifiable information that are built exclusively for this type of information with their own conceptual scheme, system management, and physical structure.

1. Introduction

This paper explores the privacy-related differences between types of information to argue that security, policy, and technical requirements set personal identifiable information apart from other types of information, which involves establishing a PII database with its own conceptual scheme, system management, and physical structure.

Different types of information of interest in this paper are shown in Figure 1. We will use the term infon to refer to "a piece of information" [4]. The parameters are objects, and the so-called anchors assign these objects such as agents to parameters. Infons can have subinfons that are also infons. Let INF the set of infons in the system.

Four types of infons are defined:
1. So-called “private” information is a subset of INF. “Private” information is partitioned into two types of information: PII and PNI.
2. PII = the set of pieces of personal identifiable information. We use the term pinfon to refer to this special type of infon.
3. PNI = the set of pieces of non-identifiable information.
4. NII = (INF – PII). We use the term ninfon to refer to this special type of infon.

NII is the set of pieces of non-identifiable information and includes all pieces of information except personal identifiable information.

2. Related Works

Separating “private” data from “public” data has already been adopted in privacy-preserving systems. However, these systems do not distinguish explicitly personal identifiable information. The Platform for Privacy Preferences (P3P) is one such system that provides a means for privacy policy specification and exchange but “does not provide any mechanism to ensure that these promises are consistent with the internal data processing” [3]. It is our judgment that “internal data processing” requires recognizing explicitly that “private data” is of two types: personal identifiable information and personal non-identifiable information, and this difficulty is caused by the heterogeneity of data. Hippocratic databases have been introduced as systems that integrate privacy protection within relational database systems [1]. Nevertheless, in principle, a Hippocratic database is a general DBMS with a purpose mechanism. Purposes can be declared for any data item that is not necessarily personal identifiable information.

Information (INF)  

Private/personal information

PNI  

PII

Figure 1. Types of information.
3. Infons

This section reviews the theory of infons, and simultaneously presents it in the PII context in anticipation of materials (e.g., examples) developed in the remaining part of the paper that introduce a theory of PII infons.

The theory of infons provides a rich algebra of construction operations that can be applied to PII. An infon is a discrete item of information and may be parametric. The parameters represent objects. Anchors assign objects to parameters. PII infons are distinguished by the mandatory presence of at least one proprietor, an object of type identifiable person.

Infons in an application domain such as personal identifiable information are typically interrelated with each other; they partially depend on each other, partially exclude each other, and may be (hierarchically) ordered. Thus we need a theory that allows constructing a “lattice” of infons (and PII infons) that includes basic and complex infons while taking into consideration their structures and relationships. In such a theory, we identify basic infons that cannot be decomposed into more basic infons. This construction mechanism of infons from infons should be supported by an algebra of construction operations. We generally may assume that each infon consists of a number of components. The construction is applied in performing combination, replacement, or removal of some of these components; some of them may be essential (not removable) or auxiliary (optional).

Infons may also be related to each other. We use predicates that relate infons with each other. Since infons may consist of infons, then binary predicates are sufficient to describe relationships among them. Thus, the world of infons can be specified as the triple: (A; O; P) as follows.

- Atomic infons A
- Algebraic operations O for computing complex infons such as combination ⊕ of infons, abstraction □ of infons by projections, quotient □ of infons, renaming of infons, union △ of infons, intersection ▽ of infons, full negation ¬ of infons, and minimal negation ¬ of infons within a given context.
- Predicates P stating associations among infons such as the subinfon relation, a statement whether infons can be potentially associated with each other, a statement whether infons cannot be potentially associated with each other, a statement whether infons are potentially compatible with each other, and a statement whether infons are incompatible with each other.

The combination of two infons results in an infon that has all components of the two infons. The abstraction is used for a reduction of components of an infon. The quotient allows concentrating on those components of the first life that do not appear in the second infon. The union takes all components of the two infons and does not combine common components into one component. The full negation allows generating all those components that do not appear in the infon. The minimal negation restricts this negation to some given context.

We require that the subinfon relation is not transitively reflexive. The compatibility and incompatibility predicates are not contradictory. The potential association and its negation must not conflict. The predicates should not span all possible associations among the infons but only those that are meaningful in a given application area. We may assume that two infons are either potentially associated or cannot be associated with each other. The same restriction can be made for compatibility.

This infon world is very general and allows deriving more advanced operations and predicates. If we assume the completeness of compatibility and association predicates we may use expressions defined by the operations and derived predicates. The extraction of application-relevant infons from infons is supported by five operations.

1. Infon projection narrows the infon to those parts (objects or concepts, axioms or invariants relating entities, functions, events, and behaviors) that are of concern for the application relevant infons. For example, a projection operation may produce the set of proprietors from a given infon, e.g., {Mary, John} from John loves Mary.

2. Infon instantiation lifts the general infons to those that are of interest within the solution and instantiates variables by values that are fixed for the given system. For example a PII infon may be instantiated from its anonymized version, e.g., John is sick from Someone is sick.

3. Infon determination is used for selecting those traces or solutions to the problem under inspection that are the most perspective or best fitting for the system envisioned. The determination typically results in a small number of scenarios for the infons that are going to be supported. For example, infon determination to decide whether an infon belongs to a certain piiSphere (PII of a proprietor – to be discussed later).

4. Infon extension is used for adding those facets that are not given by the infon but by the environment or the platforms that might be chosen or that might be used for simplification or support of the infon (e.g., additional data, auxiliary functionality). For example,
infon extension to related non-identifiable information (to be discussed later).

5. Infons are often associated, adjacent, interact, or fit with each other. Infon join is used to combine infons into more complex and combined infons that describe a complex solution. For example, joining atomic PII to form compound PII and a collection of related PII information.

The application of these operations allows extraction of which subinfons, which functionality, which events, and which behavior (e.g., the action/verb in PII) is shared among information spheres (e.g., of proprietors). These shared facilities provide cross-cutting concerns among all information spheres of relevant infons. They also hint at possible architectures of information and database systems and on separation into candidate components. For instance, entity (say, non-person entity) sharing describes which information flow and development can be observed in the information spheres.

The theory of PII infons can be applied in several areas such as the technical and legal aspects of information privacy and security.

4. Personal Identifiable Information

It is typically claimed that what makes the data “private” or “personal” is either specific legislation, e.g., a company must not disclose information about its employees, or individual agreements, e.g., a customer has agreed to an electronic retailer's privacy policy. However, this line of thought blurs the difference between personal identifiable information and other “private” or “personal” information. Personal identifiable information has an “objective” definition in the sense that it is independent of such authorities as legislation or agreement.

PII infons involve relationships (e.g., possession) with their proprietors, persons but non-proprieters, and non-persons such as institutions, agencies, or companies: For example, a person may possess PII of another person, or a company may have the PII of someone in its database. However, proprietorship of PII is reserved only to its proprietor regardless of who possesses copy of it.

To base personal identifiable information on firmer ground, we turn to stating some principles related to such information. For us, personal identifiable information (pinfon) is any information that has referent(s) to uniquely identifiable persons. In logic, reference is the relation of a word (logical name) to a thing.

Following Devin’s formalism [4] an infon has the form <<R, a1, ..., an, 1>> and <<R, a1, ..., an, 0>>. R is an n-place relation and a1, ..., an are objects appropriate for R. 0 and 1 indicate these may be thought objects do, respectively, not, stand in relation R. For simplicity sake, we may write an infon <<R, a1, ..., an, 1/0>> as <<a1, ..., an>> when R is known or immaterial.

A PII infon, pinfon, is an infon such that at least one of the objects is a singly identifiable person. Any singly identifiable person in the pinfon is called proprietor. The proprietor is the person about whom the pinfon communicated information. If there is exactly one object of this type, the pinfon is an atomic pinfon; if there is more than one singly identifiable person, it is a compound pinfon. An atomic pinfon is a discrete piece of information about a singly identifiable person. A compound pinfon is a discrete piece of information about several singly identifiable persons. If the infon does not include a singly identifiable person then it is called a uninfon.

The following is a series of propositions that establish the foundation of the theory of personal identifiable information. The symbol “→” denotes implication.

1. Inclusivity of INF
σ ∈ INF ↔ σ ∈ PII ∨ σ ∈ NII
2. Exclusivity of PII and NII
σ ∈ INF ∧ σ ∈ PII → σ ∈ N
σ ∈ INF ∧ σ ∈ N → σ ∈ PII
3. Identifiability
Let ID denote the set of (natural) persons.

Then

β ⊤ (σ, 1) → β ⊥ (σ, 1) ∈ INF
4. Inclusivity of PII
Let n denote the number of uniquely identified persons in the infon σ, then σ ∈ INF ∧ n > 0 ⇔ σ ∈ PII
5. Proprietary
For σ ∈ PII, let PROP(σ) be the set of proprietors of σ.

Let PERSONS denote the set of (natural) persons. Then,

σ ∈ PII → PROP(σ) ∈ PERSONS
6. Inclusivity of NII
σ ∈ INF ∧ n = 0 ⇔ σ ∈ NII
7. Combination of non-identifiability with identity
Let ID denote the set of (natural) persons:

β ⊥ (σ, 1) Then,

σ1 ∈ PII ⊥ β ⊥ (σ2 ∈ NII ⊥ σ3 ∈ ID) Then
assuming σ1 ∉ ID. “#” here denotes the “merging” of two subinfons.
8. Closability of PII
σ1 ∈ PII ⊥ σ2 ∈ PII → (σ1 ⊥ σ2) ∈ PII
9. Combination with non-identifiability
\( \sigma_1 \in \text{NII} \not\exists \sigma_2 \in \text{PII} \rightarrow (\sigma_1, \sigma_2) \in \text{PII} \)

10. Reducibility to non-identifiability
\( \sigma_1 \in \text{PII} \not\exists \sigma_2 \in \text{ID} \leftarrow \sigma_1 \in \text{NII} \)

where \( \sigma_2 \) is a subinfon of \( \sigma_1 \). “\( \not\exists \)” denotes removing \( \sigma_2 \).

11. Atomicity
Let \( \text{APII} \) is the set of atomic personal identifiable information. Then, \( \sigma \in \text{PII} \land n_\sigma = 1 \leftarrow \sigma \in \text{APII} \)

12. Non-atomicity
Let \( \text{CPII} \) = the set of compound personal identifiable information. Then, \( \sigma \in \text{PII} \land n_\sigma > 1 \leftarrow \sigma \in \text{CPII} \)

13. Reducibility to atomicity
\( \sigma \in \text{CPII} \leftrightarrow \langle \langle \sigma_1, \sigma_2, \ldots, \sigma_m \rangle \rangle, \sigma_i \in \text{APII}, m = n_\sigma, \text{and } 1 \leq i \leq m, \text{and } \{\text{PROP}(\sigma_1), \text{PROP}(\sigma_2), \ldots, \text{PROP}(\sigma_3)\} = \text{PROP}(\sigma) \)

Next we discuss each of these propositions.

Inclusivity of INF
\( \sigma \in \text{INF} \leftrightarrow \sigma \in \text{PII} \lor \sigma \in \text{NII} \)
That is, infons are the union of pinfons and ninfons. PII is the set of pinfons (pieces of personal identifiable information), and NII is the set of ninfons (pieces of non-identifiable information).

Exclusivity of PII and NII
\( \sigma \in \text{INF} \land \sigma \notin \text{PII} \rightarrow \sigma \in \text{N} \)
\( \sigma \in \text{INF} \land \sigma \notin \text{N} \rightarrow \sigma \in \text{PII} \)
That is, every infon is exclusively either pinfon or nfon.

Identifiability
Let \( \# \) be a parameter for a singly identifiable person, i.e., a specific person, defined as
\( \# = \text{IND}_1 \langle \langle \text{singly identifiable}, \text{IND}_1, 1 \rangle \rangle \)
where IND indicates the basic type: an individual [4]. That is, \( \# \) is a (restricted) parameter with an anchor for an object of type singly identifiable person.

The individual \( \text{IND}_1 \) is of type person defined as
\( \langle \langle \text{person}, \text{IND}_1, 1 \rangle \rangle \)
Put simply, \( \# \) is a reference to a singly identifiable person. We now elaborate on the meaning of “identifiable.”

Consider the set of unique identifiers of persons. Ontologically, the Aristotelian entity/object is a single, specific existence (a particularity) in the world. For us, the identity of an entity is its natural descriptors (e.g., tall, black eyes, male, blood type A, etc.). These descriptors exist in the entity/object. Tallness, whiteness, location, etc. exist as aspects of the existence of the entity. We recognize the human entity from its natural descriptors. Some descriptors form identifiers. A natural identifier is a set of natural descriptors that facilitates recognizing a person uniquely. Examples of identifiers include fingerprints, faces, and DNA. No two persons have identical natural identifiers. An artificial descriptor is a descriptor that is mapped to a natural identifier. Attaching the number 123456 to a particular person is an example of artificial descriptors in the sense that they are not recognizable in the (natural) person. An artificial identifier is a set of descriptors that is mapped to a natural identifier of a person. By implication, no two persons have identical artificial identifiers. If two persons somehow have the same Social Security number, then this Social Security number is not an artificial identifier because it is not mapped uniquely to a natural identifier.

We define identifiers of proprietors as infons. Such definition is reasonable since the mere act of identifying a proprietor is a reference to a unique entity in the information sphere. Hence,
\( \langle \langle \text{is}, \#, 1 \rangle \rangle \rightarrow \langle \langle \text{is}, \#, 1 \rangle \rangle \in \text{INF} \)
That is, every unique identifier of a person is an infon. These infons cannot be decomposed into more basic infons.

Inclusivity of PII
Next we position identifiers as the basic infons in the sphere of PII. The symbol \( n_\sigma \) denotes the number of uniquely identified persons in the infon \( \sigma \). Then we can define PII and NII accordingly:
\( \sigma \in \text{INF} \land n_\sigma > 0 \leftarrow \sigma \in \text{PII} \)
That is, an infon that includes unique identifiers of persons is personal identifiable information. From (3) and (4), any unique personal identifier or piece of information that embeds identifiers is personal identifiable information. Thus, identifiers are the basic PII infons (pinfons) that cannot be decomposed into more basic infons. Furthermore, every complex pinfon includes in its structure at least one basic infon, i.e., identifier. The structure of a complex pinfon is constructed from several components:
- Basic pinfons and ninfons, i.e., the pinfon John S. Smith and the ninfon Someone is sick form the atomic PII (i.e., PII with one proprietor) John S. Smith is sick. This pinfon is produced by an instantiation operation that lifts the general infons to pinfons and instantiates the variable (Someone) by a value (John S. Smith).
- Complex pinfons form more complex infons, e.g., John S. Smith and Mary F. Fox are sick

We notice that the operation of projection is not PII-closed since we can define projecting of ninfon from pinfon (removing all identifiers). This operation is typically called anonymization.

Every pinfon refers to its proprietor(s) in the sense that it “leads” to him/her/them as distinguishable entities in the world. This reference is based on his/her/their unique identifier(s). The relationship between persons and their own pinfon is called proprietorship [1]. A pinfon is proprietary PII of its proprietor(s).
Defining pinfon as “information identifiable to the individual” does not mean that the information is “especially sensitive, private, or embarrassing. Rather, it describes a relationship between the information and a person, namely that the information—whether sensitive or trivial—is somehow identifiable to an individual” [5]. However, personal identifiable information (pinfon) is more “valuable” than personal non-identifiable information (ninfon) because it has an intrinsic value as “a human matter,” just as privacy is a human trait.

To exclude such notions as confidentiality that are applicable to the informational privacy of non-natural persons (e.g., companies), the next proposition formalizes that pinfon is applied only to (natural) persons.

For \( \sigma \in \text{PII} \), we define \( \text{PROP}(\sigma) \) to be the set of proprietors of \( \sigma \). Notice that \( |\text{PROP}(\sigma \in \text{PII})| = n_o \). Multiple occurrences of identifiers of the same proprietor are counted as a single reference to the proprietor. In our ontology, we categorize things (in the world) as objects (denoted by the set \( \text{OBJECTS} \)) and non-objects. Objects are divided into (natural) persons (denoted by the set \( \text{PERSONS} \)) and non-persons. A fundamental proposition in our system is that proprietors are (natural) persons.

**Proprietary**
\[ \text{PROP}(\sigma \in \text{PII}) \rightarrow \text{PROP}(\sigma) \in \text{PERSONS} \]
That is, pinfons are pieces of information about persons.

**Inclusivity of NII**
\[ \sigma \in \text{INF} \wedge n_o = 0 \leftrightarrow \sigma \in \text{NII} \]
That is, non-identifiable information (ninfon) does not embed any unique identifiers of persons.

**Combination of non-identifiability with identity**
Next we can specify several transformation rules that convert from one type of information to another. These (privacy) rules are important to decide what type of information applies to what operations (e.g., information disclosure rules).

Let \( \text{ID} \) denote the set of (basic) pinfons of type \( << \) is, \( b \), 1\(>> \). That is, ID is the set of identifiers of persons (in the world). We now define construction of complex infons from basic pinfons and non-identifying information. The definition also applies to projecting pinfons from more complex pinfons by removing all or some non-identifying information.

\[ \sigma_1 \in \text{PII} \leftrightarrow \langle \langle \sigma_2 \in \text{NII} \otimes \sigma_3 \in \text{ID} \rangle \rangle \]
assuming \( \sigma_1 \notin \text{ID} \).
That is, non-identifiable information plus a unique personal identifier is personal identifiable information and vice versa. Thus the set of pinfons is closed under operations that remove or add non-identifying information. We assume the empty information \( \emptyset \) is in NII. “\( \otimes \)” here denotes “merging” two subinfons. We also assume that there is only a single \( \sigma_3 \in \text{ID} \) added to \( \sigma_2 \in \text{NII} \); however, the proposition can be generalized to apply to multiple identifiers. An example of proposition 7 is

\[ \sigma_1 = \langle \langle \text{John loves apples} \rangle \rangle \leftrightarrow \langle \langle \sigma_2 = \text{Someone loves apples} \otimes \sigma_3 = \text{John} \rangle \rangle \]
Or, in a simpler description:
\[ \sigma_1 = \text{John loves apples} \leftrightarrow \{ \sigma_2 = \text{Someone loves apples} \otimes \sigma_3 = \text{John} \} \]

Proposition can also be applied to the union \( \Psi \) of pinfons.

**Closability of PII**
\( \text{PII} \) is a closed set under different operations (e.g., merge, concatenate, submerge, etc.) that construct complex pinfons from more basic pinfons. Hence,
\[ \sigma_1 \in \text{PII} \otimes \sigma_2 \in \text{PII} \rightarrow (\sigma_1 \otimes \sigma_2) \in \text{PII} \]
That is, merging personal identifiable information with personal identifiable information produces personal identifiable information. Also, PII is a closed set under different operations (e.g., merge, concatenate, submerge, etc.) that construct complex pinfons by mixing pinfons with non-identifying information.

**Combination with non-identifiability**
\[ \sigma_1 \in \text{NII} \otimes \sigma_2 \in \text{PII} \rightarrow (\sigma_1 \otimes \sigma_2) \in \text{PII} \]
That is, non-identifying information plus personal identifiable information is personal identifiable information.

**Reducibility to non-identifiability**
Identifiers are the basic pinfons. Removing all identifiers from a pinfon converts it to non-identifying information. Adding identifiers to any piece of non-identifying information converts it to a pinfon,
\[ \sigma_1 \in \text{PII} \otimes \sigma_2 \in \text{ID} \leftrightarrow \sigma_3 \in \text{NII} \]
where \( \sigma_2 \) is a subinfon of \( \sigma_1 \). Proposition 10 states that personal identifiable information minus a unique personal identifier is non-identifying information and vice versa. “\( \otimes \)” here denotes removing \( \sigma_2 \). We assume that there is a single \( \sigma_2 \in \text{ID} \) embedded in \( \sigma_1 \); however, the opposition can be generalized to apply to multiple identifiers such that removing all identifiers produces \( \sigma_1 \in \text{NII} \).

**Atomicity**
Furthermore, we define atomic and non-atomic (compound) types of pinfons. Let
\[ \text{APII} = \{ \text{atomic personal identifiable information} \} \]
Each piece of atomic personal identifiable information is a special type of pinfon called apinfon.
As we will see later, cpinfons can be reduced to apinfons, thus simplifying the analysis of PII.
Formally, the set APII is defined as follows. \( \sigma \in PII \land n_\sigma = 1 \iff \sigma \in APII \)
That is, an apinfon is a pinfon that has a single human referent. Notice that \( \sigma \) may embed several identifiers of the same person, yet the referent is still one. Notice that apinfons can be basic (a single identifier) or complex (a single identifier plus non-identifiable information).

**Non-atomicity**
Let CPII = a set of compound personal identifiable information. Each piece of compound personal identifiable information is a special type of pinfon called cpinfon. Formally, the set CPII is defined as follows.
\[ \sigma \in PII \land n_\sigma > 1 \iff \sigma \in CPII \]
That is, a cpinfon is a pinfon that has more than one human referent. Notice that cpinfons are always complex since they must have at least two apinfons (two identifiers).

The apinfon (atomic personal identifiable information) is the “unit” of personal identifiable information. It includes one identifier and non-identifiable information. We assume that at least some of the non-identifiable information is about the proprietor. In theory this is not necessary. Suppose that an identifier is amended to a random piece of non-identifiable information (noise). In the PII theory the result is (complex) atomic PII. In general, mixing noise with information preserves information.

**Reducibility to atomicity**
Any cpinfon is privacy-reducible to a set of apinfons (compound personal identifiable information). For example, *John and Mary are in love* can be privacy-reducible to the apinfons (apinfon) *John and someone are in love* and *Someone and Mary are in love*. Notice that our PII theory is a syntax (structural) based theory. It is obvious that the privacy-reducibility of compound personal identifiable information causes a loss of “semantic equivalance,” since the identities of the referents in the original information are separated. Semantic equivalency here means preserving the totality of information, the pieces of atomic information, and their link.

Privacy reducibility is expressed by the following proposition:
\[ \sigma \in CPII \iff \langle \sigma_1, \sigma_2, \ldots, \sigma_m \rangle, \sigma_i \in APII, m = n_\sigma, \text{ and } 1 \leq i \leq m, \text{ and } \{\text{PROP}(\sigma_1), \text{PROP}(\sigma_2), \ldots, \text{PROP}(\sigma_m)\} = \text{PROP}(\sigma) \]
The reduction process produces \( m \) atomic personal identifiable information with \( m \) different proprietors. Notice that the set of resultant apinfons is a compound pinfon. This preserves the totality of the original cpinfon through linking its apinfons together as members of the same set.

**5. Categorization of atomic personal identifiable information**

In this section, we identify categories of apinfons. Atomic personal identifiable information provides a foundation for structuring pinfons since compound personal identifiable information can be reduced to a set of apinfons. We concentrate on reducing all given personal identifiable information to sets of apinfons. Justification for this will be discussed later.

**5.1. Eliminating ninfons embedded in an apinfon**

Organizing a database of personal identifiable information requires filtering and simplifying apinfons to more basic apinfons in order to make the structuring of pinfons easier. Proposition (9) tells us that pinfons may carry non-identifiable information, ninfons. This non-identifiable information may be a random noise or information not about the proprietor directly. Removing random noise is certainly an advantage in designing a database. Identifying information that is not about the proprietor clarifies the boundary between PII and NII.

A first concern when analyzing an apinfon is projecting (isolating, factoring) information about any other entities besides the proprietor. Consider the apinfon *John’s car is fast*. This is information about John and about a car of his. This apinfon can be projected as:
\[ (\text{John’s car is fast}) \Rightarrow \{\text{The car is fast, John has a car}\} \]
where \( \Rightarrow \) is a production operator.

*John’s car is fast* information embeds the “pure” apinfon *John has a car* and the ninfon *The car is fast*. *John has a car* is information about a relationship that John has with another object in the world. This last information is an example of what we call self information. Self information (sapinfon = self atomic pinfon) is information about a proprietor, his/her aspects (e.g., tall, short), or his/her relationship with non-human objects in the world. So, it is useful to further reduce apinfons (atomic) to sapinfon (self).

Sapinfon is related to the concept of “what the piece of apinfon is about.” In the theory of aboutness, this question is answered by studying the text structure and assumptions by the source about the receiver (e.g., reader). We formalize aboutness in terms of the procedure \( \text{ABOUT}(\sigma) \), which produces the set of entities/objects that \( \sigma \) is “talking” about. In our case, we aim to reduce any self infon \( \sigma \) to \( \sigma’ \) such that \( \text{ABOUT}(\sigma) = \text{PROP}(\sigma’) \).
Self atomic information represents information about:
- Aspects of proprietor (identification, character, acts, etc.)
- His or her association with non-person “things” (e.g., house, dog, organization, etc.)
- His or her relationships with other persons (e.g., wife, friend, employee, etc.)

With regard to non-objects, of special importance for privacy analysis are aspects of persons that are expressed by sapinfon. Aspects of a person are his (physical) parts, his character, his acts, his conditions, his name, his health, his color, his handwriting, his blood type, his manner, his intelligence, etc. Their existence depends on the person, in contrast to (physical or social) objects associated with him/her such as his/her house, dog, spouse, job, professional associations, etc.

Let SAPII denote the set of sapinfons (self personal identifiable information).

14. Aboutness proposition
\[ \sigma \in \text{SAPII} \iff \text{ABOUT}(\sigma) = \text{PROP}(\sigma) \]
That is, atomic personal identifiable information \( \sigma \) is said to be self personal identifiable information (sapinfon) if its subject is its proprietor. The term “subject” here means what the entity is about when the information is communicated. The mechanism (e.g., manually) that converts APII to SAPII has yet to be investigated.

5.2. Sapinfons involving aspects of proprietor or relationship with non-person

We further simplify sapinfons. Let \( \text{OPJ}(\sigma \in \text{SAPII}) \) be the set of objects in \( \sigma \). SAPII is of two types depending on the number of objects embedded in it: singleton, ssapinfon and multitude, msapinfon. The set ssapinfons, SSAPII, is defined as:

15. Singleton proposition
\[ \sigma \in \text{SSAPII} \iff \sigma \in \text{SAPII} \land (\text{PROP}(\sigma) = \text{OPJ}(\sigma)) \]
That is, the proprietor of \( \sigma \) is its only object in \( \sigma \). The set msapinfons, MSAPII, is defined as:

16. Multitude proposition
\[ \sigma \in \text{MSAPII} \iff \sigma \in \text{SAPII} \land (|\text{OPJ}(\sigma)| > 1) \]
That is, \( \sigma \) embeds other objects beside its proprietor. We also assume logical simplification that eliminates conjunctions and disjunctions of PIIS.

Now we can declare that the sphere of personal identifiable information (piiSphere) for a given proprietor is the database that contains:
- All ssapinfons and msapinfons of the proprietor, including their arrangement in super-infons (e.g., to preserve compound personal identifiable information).
- Related non-identifiable information to the piiSphere of the proprietor as discussed.

5.3. What is related non-identifiable information?

Consider the msapinfons Alice visited clinic Y. It is msapinfons because it represents a relationship (not aspect of) the proprietor Alice had with an object, the clinic. Information about the clinic may or may not be privacy related information. For example, opening, number of beds, and other information about the clinic are not privacy related information. Thus, such information ought not to be included in Alice’s piiSphere. However, when the information is that the clinic is an abortion clinic, then Alice’s piiSphere ought to include this non-identifiable information about the clinic.

6. Justifications for PII databases

We concentrate on what we call PII database, PIIDB, that contains personal identifiable information and information related to it.

Security requirement: We can distinguish two types of information security:
- (1) Personal identifiable information security, and
- (2) Non-identifiable information security.

While the security requirements of NII are concerned with the traditional system characteristics of confidentiality, integrity, and availability, PII security lends itself to unique techniques that pertain only to PII.

The process of protecting PII involves: (1) Protection of the identities of the proprietor, (2) Protection of the non-identity portion of the PII. Of course, all information security tools such as encryption can be applied in this context, yet other methods (e.g., anonymization) utilizing the unique structure of PII as a combination of identities and other information can also be used. Data-mining attacks on PII aim at determining the identity of the proprietor(s) from non-identifiable information; for example, determining the identity of the patient from anonymized information that gives age, sex, and zip code in health records (k-anonymization). Thus, PII lends itself to unique techniques that can be applied in protection of this information.

Another important issue that motivates organizing PII separately is that any intrusion on PII involves information besides the owner’s information (e.g., a company, proprietors, and other third parties, e.g., privacy commissioner). For example, a PII security
system may require immediate alerting of the proprietor of intrusion on his/her PII.

An additional point is that the sensitivity of PII is in general valued more highly than the sensitivity of other types of information. PII is more “valuable” than non-PII because of its privacy aspect, as discussed previously. Such considerations imply a special security status for PII.

**Policy requirements.** Policies that are applied for PII are not applicable to NII (e.g., consent, opt-in/out, proprietor’s identity management, trust, privacy mining). While the NII security requirements are concerned with the traditional system characteristics of confidentiality, integrity, and availability, PII privacy requirements are concerned also with such issues as purpose, privacy compliance, transborder flow of data, third party disclosure, etc. Separating PII from NII can reduce the complex policies required to safeguard sensitive information where multiple rules are applied, depending upon who is accessing the data and what the function is.

In general, PIIDB goes beyond mere protection of data:
1. PIIDB identifies proprietor’s piiSphere and provides security, policy, and tools to the piiSphere.
2. PIIDB provides security, policy, and tools only to proprietor’s piiSphere, thus conserving privacy efforts.
3. PIIDB identifies inter-piiSphere relationships (proprietors’ relationships with each other) and provides security, policy, and tools to protect the privacy of these relationships.

**7. Personal Identifiable Information Database (PIIDB)**

The central mechanism in PIIDB is an explicit declaration of proprietors in a table that includes unique identifiers of all proprietors in the PIIDB. The principle of uniqueness of proprietor’s identifiers requires that the internal key is mapped one-to-one to the individual’s legal identity or physical location.”

This is an important feature in PIIDB to guarantee consistency of information about persons. This “identity” uniquely identifies the piiSphere and distinguishes one piiSphere from another.

PIIDB obeys all propositions defined previously. Some of these propositions can be utilized as privacy rules. As an illustration of the applications of these propositions, consider the case of privacy constraint that prohibits disclosing $\sigma \in$ PII. By proposition (9) above, mixing (e.g., amending, inserting, etc.) $\sigma$ with any other piece of information makes the disclosure constraint apply to the combined piece of information. In this case a general policy is: Applying a protection rule to $\sigma_1 \in$ PII implies applying the same protection to $(\sigma_1 \land \sigma_2)$ where $\sigma_2 \notin$ PII.

**8. Conclusion**

The theory of PII infons can provide a theoretical foundation for technical solutions for protection of personal identifiable information. In such approach, privacy rules form an integral part of the design of the system. PII can be identified (hence becomes an object to privacy rules) during processing of information that may mix it with other types of information. We proposed to analyze and process PII as a separate database with clear boundary lines with non-identifiable information. This facilitates meeting the unique requirements of PII.

**9. References**

Privacy Enhanced Information Systems

Sabah S. Al-Fedaghi\textsuperscript{1}, Gunar Fiedler\textsuperscript{2}, and Bernhard Thalheim\textsuperscript{2}

\textsuperscript{1} Computer Engineering Department, Kuwait University, Safat 13060, Kuwait
sabah@eng.kuniv.edu.kw

\textsuperscript{2} Kiel University, Computer Science Institute, 24098 Kiel, Germany
fiedler | thalheim@is.informatik.uni-kiel.de

Abstract. Privacy is becoming a major issue of social, ethical and legal concern on the Internet. The development of information technology and the Internet has major implications for the privacy of individuals. Studies of private or personal information have related it to such concepts as identifiably, secrecy, anonymity, control, etc. Few studies have examined the basic features of this type of information. Also, database models (e.g., ‘Hippocratic’ [AKX02] database) for this type of information have been proposed. This paper studies the nature of private information and develops a new conceptual model for databases that contain exclusively private information. The model utilizes the theory of infons to define “private infons”, and develops taxonomy of these private infons based on the notions of proprietary and possession. The proposed model also specifies different privacy rules and principles, derives their enforcement, and develops and tests architecture for this type of databases.

1 Introduction

1.1 Information Privacy

The notion of privacy is becoming an important feature in all aspects of life of modern society. “[P]rivacy will be to the information economy of the next century what consumer protection and environmental concerns have been to the industrial society of the 20th century.” [Gle96] One of the most important factors that contributed to this increasing significance of the concept of privacy has been the appearance of global computer networks. They have allowed information to be moved around the world and made national boundaries irrelevant to most types of communication. This has raised concerns about privacy in the on-line environment and has affected different aspects of dealing with private information in the areas of security, commerce, government, etc.

We distinguish today between media privacy, territorial privacy, communication privacy, bodily privacy, and information privacy. The first four forms of privacy are now well supported by laws, constitutional rights and other legal frameworks. Although a number of laws protecting information privacy have been introduced in most developed countries, information privacy is challenged and not well protected due to the technological development, e.g. the commercial success, misbehavior, tools for the “glass box customer” and WWW misuse. Already [Wes67] coined principles of privacy that must be preserved by any system: openness and transparency, individual participation, collection limitation, data quality, use limitation, reasonable security, and accountability.
1.2 Database Privacy

Protecting sensitive and classified data stored in database management systems requires measures beyond controlling access to files managed by operating systems. An example in this context is the problem of inference, which is the deduction of unauthorized information from the observation of authorized information - where unauthorized information is deduced from the legitimate responses. There is a lot of research work in the area of applying privacy protection in the context of databases. In the medical field, a great deal of interest is in anonymizing textual information. Sweeney’s pioneering work [Swe96] is based on removing the personally identifying information from the text so that the integrity of the information remains intact even though the identity of the persons remains confidential. The problem in this case is identification detection problem where identifiers of persons are detected and anonymized. Sweeney’s recognition methodology aims at detecting information that can personally identify any person. One important issue is the definition of “personal information.” Is it the whole text, the paragraph, the sentence, the phrase or only the word that denotes the identity? For tabular data, several techniques are proposed at the relation level. For example, the k-anonymization technique [Sam01] assumes that a relational table with a prime key that refers to a person is “personal information”. Its main concern is anonymizing entries in the table in order to block any attempt to reach “identifiability” that stems from these entries. Systems that use such techniques aim at protecting individual identifiable information and simultaneously maintaining the entity relationship in original data. Still, the definition in these works of “personal information” is not clear. Implicitly, it is understood that the privacy aspect comes from associating the attribute name with the identifying key of the relation. Statistical Databases aims at providing statistical information without compromising sensitive information about individuals. However, statistical databases goal is to prevent disclosure of private information and is limited to implied information. Agrawal et al. [AKX02] is first in proposing ‘Hippocratic’ database systems that put privacy as central concern. They articulate what it means for a database system to responsibly manage private information under its control. Nevertheless, their proposed model can be considered as an extension of the relational database model. It lacks modeling features that put privacy as a central concern of the conceptual schema.

Generally, in database design the conceptual schema encompass the enterprise’s view that encompasses all types of information. We concentrate on building conceptual schema exclusively for private information. We envision a system where not only enterprises have ‘private information’ databases but every individual has his/her own ‘private information’ database that includes his/her own private information and other’s private information possessed by him/her. Our approach is comprehensive in terms of analyzing the overlapping between different conceptual schemas. Suppose that we have one enterprise and 2000 individuals. Our model is concerned with developing a framework of 2001 conceptual schemas and their sub-schemas. In the e-environment the corresponding (private) databases interact with each other utilizing the knowledge in their schemas. Private information agents in all databases work on behalf of their owners to facilitate private information transactions. For example, the agent of individual x accesses the database of the enterprise to update its owner’s private information, then the agent record in the individual’s database that such information is consented to the enterprise’s database. Or, the agent of the enterprise communicates with the individual’s database to verify that the private information it has about the individual, is consented.
1.3 Our Contribution

Our approach aims at developing a systematic methodology that proceeds from a working definition of “private information” to a complete set of conceptual schemas that includes privacy-based decomposition and constraints. We achieve this goal through:

**Framework:** This paper introduces a formalism to specify informational privacy assertions based on a theoretical foundation. We introduce a new definition of private information and categories of this type of information to entangle issues of compound private information (*John and Mary use contraceptives* is ‘private’ information of both *John* and *Mary*). We also introduce a classification of relationships of “private information pieces” to its possessors and proprietors. The proprietor here refers to the person identified in information pieces.

**Model:** Our theoretical framework is applied to the restrictions and conditions of possession. Thus, the private information in possession of *z* can be classified according to whether this possession is according to: consent/no-consent of the proprietor, legal/illegal possession of private information, and awareness/no-awareness of the proprietor of private information of this possession.

1.4 Related Work

The comprehensiveness of our approach extends from the system’s level (e.g., conceptual schema design) to the specification level that includes functionalities of subsystems such as constrains and policies. At the system level, J. Biskup introduced in his recent script [Bis05] a general approach for development of security preserving systems. According to his model [*participants* in the information society include nearly every individual, group, institution, association or private company. Informational activities among participants must be in some way based on *trust*. Orthogonally we distinguished between known and unknown services provided[Gib05]. An individual is seen as an actor being involved in a large variety of different social roles including determining which personal information he/she is willing to share. Since there are potentially conflicting interests, the design of a security system takes into consideration the specific needs and wishes of communities of participants. We visualize the general architecture of this system as shown in Figure 1. Participants are threatened by certain environments and explicitly forbid services in other environments. Services are provided by a system in a selected environment. Participants may trust service provider(s) and require a service. We may add other restrictions such as acquiring only trusted services.

The correctness properties of a secure system may be specified by three path constraints in the security preserving system:

- A security system is *reliable correct* if all the participants’ requests can be handled by services in which the participant has trusts.
- The system is a *confine* system if no illicit services pop up with services used.
- The system is a *trusted* system if whenever a service pops up with the requested services, it is whitelisted.

Privacy system is not a subsystem of the security system. Some security measures may compromise privacy protection as in network monitoring. Nevertheless, the security tools used by a security system are very important for any privacy system since they can provide
mechanisms to facilitate the privacy aspects of a system. Generally, data security involves protecting the information itself against various risks, such as the risks of being accessed or modified by unauthorized persons. ‘Data security’ is not an aspect of ‘privacy protection’ unless the involved data represents ‘private information’. Consequently, in our model participants in the ‘private information society’ include every individual and enterprise. This ‘society’ could be a hospital and its patients or it could be the whole society with all individuals and enterprises participating in private information exchanges.

At the technicalities level, there is a great deal of work in the area of protecting private information. In authentication systems (e.g., Microsoft Passport) the private information is generally limited to identification/contact information. Some published works propose agents that are trusted by a user to store and make available private information such as user profiles, other published works propose vocabularies for composing policies to allow or deny access to the private information [Lee02]. The Platform for Privacy Preferences (P3P) has provided tools and services that improve trust and offer an automated way for users to gain more control over the use of their private information. APPEL (A P3P Preference Exchange Language) can be used by the user agent to make automated and semi-automated decisions regarding the acceptance of privacy policies from P3P enabled Web sites. P3P enhancements also allow negotiating for a different privacy policy that can be the used in any subsequent release of private information. Recent works in this field address semantic issues for privacy management. It is pointed out that a standard method of exchanging privacy policies (a privacy ontology) is needed for the Semantic Web [KHM02, TDT04]. P3P Project extensions allow data and conditions to be expressed in an RDF privacy policy [W3C02]. There are also several projects that exploit the development of privacy protection agents using semantic web technology and the W3C’s Platform for Privacy Preference protocol [Gre02].

These works lack an encompassing model for private information databases. This situation is similar to if we had relational constraints, operations, policies, etc. without having the underlying relational database model. A basic step in developing a private information model is building ontology to identify general types of private information. Our proposed ontology provides a deeper level of meaning of sub-categorization of private information. For example, these general types of private information are used in analyzing the relationship between privacy and ethics [AF05b] and in refining the notion of anonymity in the context of private

![Figure 1: The structure of a security system model.](image-url)
1.5 Problems Found In A Privacy Project

Private information includes sensitive information that many people consider it deserving the strongest protection. We give here one case related to health information. The collection and processing of medical information have witnessed great changes since the early 90’s. The setting for information began to shift quickly from manual forms to electronic medium. This has raised concerns about protecting the privacy of medical records.

The German government issued in 2002 the health card project as a competitive project between the German states. The Schleswig-Holstein health card project is considered the most advanced [LSH]. All relevant patient data are stored on the SIM\(^1\) of a health identity card. The data must be protected by one pin and, additionally, might be centrally stored. Data to be stored on the card consist of personal health condition, all data associated with medical treatment, physician visits, and specific disease and disorder information. Moreover, access keys are provided to read the full medical record prepared by doctors.

The German health card project is currently based on a simple access model that consists of:

- **Assigning views to each role** such as the pharmacist view, and
- **Securing and encrypting information based on pin codes** managed by patients.

The storage model used in the Schleswig-Holstein is based on:

- **Central storage** of all relevant information that is accessible by everybody who has the access portfolio, and
- **Local partial storage** of any recent information on the personal health card of each patient.

The patient information is accessible by those who are permitted to access it either by using the health card of the patient directly or by a pin number provided by the patient and pointer to the central storage.

We regard these models to be not appropriately designed to handle private information. They are not ‘private information models’ but general database models designed for any type of data such as inventory data, maintenance data, budget data etc. This generality may cause difficulties in controlling the flow of private information. For example, once information has been retrieved, the patient has no control on, or even knowledge of, its distribution. In our case we envision cooperating among private information databases with their software agents that check each other and report abnormal situations. A person can examine his/her private information in an enterprise just as he/she can check his/her account in a bank.

2 Infons Owned by Agents

2.1 Infons of Agents

Infons have been introduced by [Dev91], to represent possible facts and are extensively used in situation theory [She96]. The theory of infons can be understood as a generalized theory of

\(^{1}\text{SIM}=\text{Subscriber Identity Module, a chip with a processor and some memory. SIMs are typically used for user identification, encryption and data storage.} \)
notions introduced in [Kau67]. They are less rigid than concepts considered in [GW98]. An example of an infon might be the predicate Patient(Bernhard, Thalheim, ..., 100352422728) or the dual (negated) predicate \( \neg \text{Village}(Dresden, Saxony) \). Infons can be composed of infons, e.g., the infon \( \text{Possesses}(A_{100352422728}, \neg \text{Village}(Dresden, Saxony)) \) which says that the agent \( A_{100352422728} \) possesses the infon \( \neg \text{Village}(Dresden, Saxony) \).

An infon is a discrete item of information and may be parametric. The parameters are objects, and the so-called anchors assign these objects such as agents to parameters. An action results in changing many infons in the world, and the change in an infon is essentially dependent on the situation where the action occurs. **Primary infon sets** \( \mathcal{I} \) are (temporary, epistemic) subsets of the set of all infons with the restriction that all dual infons (same infons with different polarity) also belong to \( \mathcal{I} \) and that all instantiations of parameters of an infon \( \mathcal{I} \) are in \( \mathcal{I} \). More formally the second condition says that for \( A \in \mathcal{I} \) and an anchor \( f \) the infon \( A(f) \) is obtained by replacing each parameter \( \dot{x} \in (\text{para}(A) \cap \text{dom}(f)) \) by the object \( f(\dot{x}) \) in \( A \).

Infons not belonging to a primary infon set are called secondary. An infon constraint relation \( \rightsquigarrow \) is a subset of the Cartesian product of the set of all infons and the set of all secondary infons. We may use [Kau67] to develop a general logic of infons. Infons can be ordered by the generalization/specialization order \( A \preceq B \) specifying that \( A \) is at least as specific as \( B \). Similarly, \( \succeq \) is the opposite relation. The relation \( \equiv \) specifies that two infons are similar. We can now derive operations:

**Homogeneity of two infons** defines the relation \( A \equiv B := \exists X (A \equiv X \wedge B \equiv X) \), i.e. two infons can be homogenized if a more specific infon \( X \) exists.

**Inhomogeneity of two infons** is the opposite, i.e., \( A \not\equiv B := \neg \exists X (A \equiv X \wedge B \equiv X) \).

**Compatibility of two infons** requires the existence of a more general infon, i.e., \( A \equiv B := \exists X (X \equiv A \wedge X \equiv B) \).

**Compatibility of two infons** is given through \( A \equiv B := \neg \exists X (X \equiv A \wedge X \equiv B) \).

**Derived predicates** are

- **divergency** \( \equiv := \equiv \wedge \equiv \)
- **isolation** \( \equiv := \equiv \wedge \equiv \)
- **potential homogenizability** \( \equiv := \equiv \wedge \equiv \wedge \equiv \wedge \equiv \) and
- **heterogeneity** \( \equiv := \equiv \wedge \equiv \).

We may directly derive a number of general laws and other operators:

- \( \vdash A \equiv B \iff \exists x \forall y (x \equiv y \rightarrow (A \equiv y \wedge B \equiv y)) \)
- \( \vdash A \equiv B \iff \exists x \forall y (x \equiv y \rightarrow (A \equiv y \wedge B \equiv y)) \)
- **Product:** \( C = A \square B := \forall Y (C \equiv Y \leftrightarrow (A \equiv Y \wedge B \equiv Y)) \)
- **Sum:** \( C = A \oplus B := \forall Y (C \equiv Y \leftrightarrow (A \equiv Y \wedge B \equiv Y)) \)
- The **negation** specifies the negation of all properties of the infon but not the infon itself:
  \( B = \bar{A} := \forall x (x \equiv B \leftarrow x \not\equiv A) \).
The difference between two infons is given through all those properties that are contained in the first infons but form an infon that is incompatible with the second infon:

\[ C = A \sqcap B := \forall x (C \supseteq x \iff (A \supseteq x \iff B \not\supseteq x)) \]

This general logic and general computation theory can be used to infer new infons based on the set of primary infons.

Agents \( A \) from a general agent set \( \mathfrak{A} \) are divided into two disjoint sets: individuals \( \mathfrak{V} \) and non-individuals \( \mathfrak{N} \). An individual represents a natural person while a non-individual represents an artificial person like a company, a government agency, etc.

We may, furthermore, model that an infon belongs to an individual or agent by using the general predicates. The relationship between infon and the sets \( \mathfrak{V} \) and \( \mathfrak{N} \) is categorized as follows:

- **Possesses** is a relation between members in \( \mathfrak{A} \) and primary infons. Possesses \(( z \in \mathfrak{A}, f \)) means that infon \( f \) is possessed by \( z \). For example, an individual may possess private information of another individual or, a company may have in its database, private information of someone.

- **Knows** is a relation between individuals in \( \mathfrak{A} \) and primary infons. Knows \(( v \in \mathfrak{V}, f \)) means that \( v \) has knowledge that \( f \) is in possession of \( z \in \mathfrak{A} \). The predicate can be combined with itself, for instance, we can state \( \text{Knows}(A_{100352422728}, \text{Possesses}(A_{100352422729}, \neg \text{Village}(Dresden,Saxony))) \) that is, \( A_{100352422728} \text{ Knows that } A_{100352422729} \text{ possesses the misinformation Village(Dresden,Saxony)} \). Also,
  \[
  \text{Knows}(A_{180753}, \text{Knows}(A_{100352422728}, \text{Possesses}(A_{100352422729}, \neg \text{Village}(Dresden,Saxony))))
  \]

- **Belongs** is a relation between entities in \( \mathfrak{A} \) and primary infons. Belongs \(( v \in \mathfrak{V}, f \)) means that \( v \) is a referent in \( f \). For example, if \( f \) is \textit{John is an alcoholic}, then \( f \) “belongs” to John. The infon \( f \) is a private information assertion that identifies John, hence, Belongs(John, f).

Infons are similar to memes discussed in [Bla99]. Memes are the units of cultural evolution and selection. They can be folded and be used for derivations. We will use situation theory for development of a theory of information flow. This paper focuses on infons that represent private information.

### 2.2 Private Infons

We propose an **Information Privacy Model** (IPM) that focuses on linguistic forms of private information taking infons as its basic components. A ‘model’ here refers to abstractions representing the knowledge, organization, expressions, and constraints acquired from the world around us. The realization of this model means introducing a set of rules and data structures for which the rules are true. The model in this case, is formed from certain graphs, called schemata, that serve as patterns or frames for the model. The schema is ‘conceptual’ in the sense that, it represents a ‘mental’ model that is built from concepts and conceptual relations. ‘Language’ is the main vehicle that describes entities, actions and states in the domain of knowledge, which in our case is informational privacy. Linguistic elements will be our starting point in identifying the meaning of ‘private’ information.

Infons are categorized according to the number of their referents as follows:
Zero infon is an infon that has no referent that signifies a single individual (person) $A \in \mathcal{V}$.

For example, *Spare part number 54321 is in slot a-2* is a zero infon because it does not refer to a person. The zero infon is not private information because it refers to non-individual object in $\mathcal{W}$. Some zero infons may be embedded into private information. For example, *This dog with collar belongs to John* embeds the zero infon *The dog has a collar*, and the non-zero assertion *John has a dog*.

Atomic infon is an infon that has a single referent that signifies a single individual $A \in \mathcal{V}$.

Compound infon is an infon that has more than one referent that signifies individuals in $\mathcal{V}$.

Let $A$ be an infon. If $A$ is true then $A$ is said to be information. Consequently, there are zero information, atomic information, and compound information according to the number of the referents. Atomic (compound) information becomes private if it refers to identifiable individual(s).

Private information is also related to who possesses it. A single piece of atomic private information may have many possessors; where its proprietor (i.e., the referent in the private information assertion) may or may not be among them. Atomic infons can be possessed by any agent in $\mathfrak{Z}$. Individuals can have private information of other individuals. Companies and government agencies can possess a great deal of private information about individuals. “Ownership” of atomic private information is limited in IPM to the subject of the assertion, i.e., the individual about whom the atomic private information refers. If $A$ is atomic private information of $A \in \mathcal{V}$; i.e., $A$ is its subject, then $A$ is proprietary private information of $A$ and $A$ is its proprietor.

Every piece of private information has its proprietor. A proprietor of private information may or may not be its possessor and vice versa. Individuals can be proprietors or possessors of private information; however, non-individuals can only be possessors of private information. The notion of proprietorship here is different from the legal concept of ownership. The ‘legal owning’ of a thing is equated with the exclusive possession of this thing with the right to transfer this ownership of the thing to others. “Proprietorship” of private information is non-transferable in the absolute sense. Others may possess or (legally) own it but they are never its proprietors (i.e., it cannot become their proprietary data).

Two or more individuals “have” the same piece of compound assertion when each individual has at least one atomic assertion embedded into the compound assertion. For example, *John, Jim and Alice hate each other* is a compound assertion of *John*, *Jim*, and *Alice*. It embeds the atomic assertions *Someone hates someone*, where *someone* can be *John*, *Jim*, or *Alice*. These compound assertions can be represented by the three atomic assertions and the meta-assertion that states that they are components of a single compound assertion. In logic, we have the predicates $\text{hates}(John, y)$, $\text{hates}(Jim, x)$, and $\text{hates}(Alice, z)$ that are mapped to the predicate $\text{hates}(x, y, z)$ with objects $\{John, Jim, Alice\}$. It is not possible that two individuals have identical atomic private information because simply they have different identities. Atomic private information is the “source” of privacy. Compound private information is “private” because it embeds atomic private information. Also, the concept of proprietorship is applied to compound private information, which represents ‘shared proprietorship’ but not necessarily shared possession or ‘knowing’
2.3 Reducing Compound Infons to Atomic Infons

A piece of compound private information is not a collection of atomic private information. Compound private infons are fundamental structures obtainable from relations and individuals. The infon \( \text{InLove}(A_1, A_2) \) does not have the same ‘collectivity’ or compositionality as the infons \( \text{In}(A_1, \text{London}) \) and \( \text{In}(A_2, \text{London}) \). The latter is pseudo (not real) compound private information. It is a collection of the atomic private information. The structure of compound private information is important if we are interested, not only in breaking up the compound assertion into its embedded atomic assertions, but also in reconstructing the original compound assertion. For example, \( \text{John, Jim and Alice hate each other} \) can be reconstructed uniquely from its atomic assertions, while the pseudo compound assertion \( \text{John and Jim hates Alice} \) is not. Notice that the later embeds the atomic assertion \( x \) and \( \text{someone hates someone} \).

A compound infon is reducible to a set of atomic infons but it has more than that. The ‘compound’ characterization refers to compound identities. If the atomic private information \( \text{InLove}(A_1, x) \) is composed in the compound private information \( \text{InLove}(A_1, A_2) \) then the possessor of the remaining part, \( \text{InLove}(y, A_2) \), has no private information of \( A_1 \). If the infon \( \text{John and Alice are in love} \) is partially anonymized and released as \( \text{Someone and Alice are in love} \), then no private information of John is released. That is the reason that we have concentrated on atomic private information. Atomic infons are “pure” private information while compound information is not proprietary information of a single individual. It is shared privacy, thus its control is shared among its proprietors. Nevertheless, “pure” compound information (the part that connects the resultant atomic infons) is an essential part of privacy-based databases.

Suppose that we have the compound private infon \( A(A_1, A_2, \ldots, A_n) \) where \( A_1, A_2, \ldots, A_n \) refer to different identifiable individuals (e.g., \( \text{hates(John, Jim, Alice)} \)). Privacy-reducibility of \( A(A_1, A_2, \ldots, A_n) \) refers to producing \( n \) atomic private infons and the zero infon: \( \text{Infon}_1 \boxplus \text{Infon}_2 \boxplus \ldots \boxplus \text{Infon}_n \).

For example, \( \text{John, Jim and Alice hate each other} \) embeds the atomic assertions \( x \text{ hates someone} \), where \( x \) can be John, Jim, or Alice and the meta-assertion that states that they are components of a single compound assertion. The process of breaking up the compound infon to its atomic infons is a straightforward procedure. It can be described as follows: For each \( A_j \), produce its atomic private infon by replacing all \( A_i (i \neq j) \) in \( A(A_1, A_2, \ldots, A_n) \) by the non-identifiable description of \( A_i \). Additionally, we introduce the zero-information meta-infon described previously. Notice that the zero-information meta-infon is a zero infon because it refers to assertions not individuals.

Thus, we have isolated the private information of each individual and at the same time have preserved the “possibility” of recovering any compound private information. In the restrictive structure of databases, reconstructing the original compound private information requires access to the \( n + 1 \) databases (\( n \) atomic infons plus one zero infon). For example, consider the relational schema MARRIED(HUSBAND, WIFE), WIFE-INFORM(NAME, AGE), and HUSBAND-INFORM(NAME, AGE). The first table contains zero privacy information that links tuples in the other two ‘private information’ relations. So the original compound infon \( \text{John whose age is 40 is married to Alice whose age is 30} \) can be represented by the atomic infons: (1) \( \text{(John whose age is 40) is married to (someone whose age is 30)} \) and (2) \( \text{(Someone whose age is 40) is married to (Alice whose age is 30)} \). MARRIED(HUSBAND, WIFE) includes exclusively pointers and releasing it does not compromise the privacy of any husband or wife in the database. For example, if John is the 4th tuple in HUSBAND-INFORM
and Alice is the 9th tuple in WIFE-INFORM, the tuple (4, 9) can be one way of implementing the zero privacy information that links John and Alice.

The methodology of syntactical construction of the original assertion is not of central concern here. Whether the resultant set of atomic assertions is semantically equivalent to the original one is an interesting problem. The reduction process of the original compound private infon is utilized here to identify “privacy centers” and use these ‘centers’ in different privacy related applications.

2.4 Possession and Property of Infons

Every individual has his/her stock of private information. Some of this information is his/her own and some is in his/her possession. In this section we study different types of atomic assertions associated with a single individual in order to identify their relationships to the proprietor. We divide the set into two disjoint sets of atomic assertions Individual.Proprietary and Individual.Possession as shown in Figure 2. Each set is defined as follows:

![Figure 2: The taxonomy of infons of an individual.](image)

1. **Possession**: This is the set of pieces of atomic private information of the others that are in the possession of an individual. For example if individual A knows that individual A' has cancer then this information is in the set Individual.Possession. A of A'.

2. **Proprietary**: This is the set of atomic private assertions (atomic infons) of the individual him/herself. It has two categories:
   
   (a) **Not Know (NKnow)**: This is the set of atomic private information, which the proprietor does not know about it, e.g. results of medical analysis. This set is included in the tree for completeness sake. We often assume that if the object Individual (as a data structure) represents an entity of type individual then this set should be empty. However Individual can be part of another schema, in this case NKnow may not be empty.

   (b) **Known**: This is the set of atomic private information assertions (infons) that the proprietor does know about it. It includes two types of atomic private information assertions as follows.

   (b1) **Not Known by other individuals**: This is the set of atomic private information assertions that are known only by their proprietor and not known by others.
(b2) Known by other individuals: This is the set of private information assertions that the individual knows that it is known by others (in possession of others). It is divided into many sets of pieces of atomic information that are in possession of subsets of others:

(b21) Infons that are shared with others by contracts. The contracting schema depends on the individuals.

(b22) Infons that are shared with others: (1) with the consent of the individual or, (2) against his/her will. An example of the latter type is private information (e.g., names) of victims obtained from a publicly released police report.

(b23) Infons that are shared with everybody and are thus public.

Contracts are specified through the “who-how-whom-what-on_what_basis” frame of contracting. Contracts may be built using parametric secondary infons. The basis for contracts of information sharing is usually another contract. We may envision that contracts use cases of information sharing in the style of UK jurisprudence. The ‘who-whom’ specification might use a specification of collaboration between individuals or non-individuals. The style of information sharing is based on a specification of workflows which might include message passing activities whenever an infon is used by an agent who is not its proprietor.

2.5 States of Possession of Infons

We have developed a general theory of infons, agents, possessions and property. Let us sharpen the requirements for a privacy enhanced information system. The general state of possession of infons is displayed in Figure 3.

![Figure 3: States of possession of infons monitored by the privacy enhanced system.](image)

We distinguish three perspectives of possession:

**Legality of possession** that ranges in the hierarchy of private information from proprietary through Known by contract to Known illegal.

**Awareness of possession** that can be recorded for each individual.

**Consent of possession** that may be based on contracts, on consent or be none.

We will concern with non-individual entities or agents (companies, government agency, etc.) that possess private information about individuals. Each non-individual agent as ‘an enterprise’. By our definition of private information an ‘enterprise’ cannot be a proprietor of private information. We will use paths within the schemas to identify sets of atomic infons. For
instance, $A_z$.Possession.$A_x$ produces all private infons of $A_x$ in possession of $A_z$.

The states of possession of private information by enterprises may be restricted as follows:

**Awareness of the individual:** This constraint indicates that the individual should be aware of any private infons in the possession of the enterprise. This fact can be embedded into the conceptual schemas structures.

**The consent of the individual:** This constraint indicates that the individual should give his/her consent of any private information in the possession of an enterprise. The consent implies awareness, however, awareness may not indicate consent. For example, private information about an ex-patient may be in the possession of the hospital without his/her consent.

**Greedy deletion of information:** This constraint indicates that any information, which is stored without the consent or without awareness of the individual must be deleted from the system.

**Restrictions on visibility of information impact:** This constraint concern implied information. An enterprise agent is, in general, able to draw an inference of what certain information holds if the agent knows the bearing of other information to the information the agent seeks. For instance, a conclusion could be drawn by a health insurance company if a patient requests medicine for sexually transmitted diseases.

A typical situation of possession is displayed in Figure 4.

![Figure 4: Conceptual structure of private information with the notions of consent and awareness. $A_V$ denotes a possessor and t denotes an infon.](image)

Additionally, this model allows representing other states of possession, e.g. illegal possession. Consider the case of a police detective, say $A_Z$, who has authority to watch a suspect, say $A_V$, secretly. The collected private information of the suspect $x$ would be stored in

$A_Z$.Possession.$A_V$.NConsent.NKnow

where NConsent denotes the set of non-consented infons and NKnow denotes its subset of infons that the proprietor does not know that the detective possesses them.

Now suppose that the detective forced the suspect to surrender his ID card. All information in the ID card will be stored in

$A_Z$.Possession.$x$.NConsent.Know
because the suspect, in this case, knows this fact. $A_i$ denotes a possessor and $t$ denotes an infon.

We notice here that this discussion assumes that we are modeling the state of private information in the universe under consideration. Each entity in this universe has its own private information corpse of data classified according to our model. For example, if the universe contains two individuals and one enterprise, then the “universal” conceptual schema has three sub-schemas. One of the individuals may not keep track of his/her private information state. Still, he/she has in his/her possession private information of others, has his/her proprietary information which includes information he/she knows that it is in possession of the enterprise, information in possession of the other individual illegally, etc.

2.6 The Model of Possession Management

Possession management can be supported by an information system that encapsulates the information obtained from other agents. We may envision distributed privacy enhanced information systems and systems with a centralized storage engine. Since there is no conceptual difference between the two approaches we concentrate in this paper on the centralized engine. The structure of the infon management schema\(^2\) is displayed in Figure 5.

The database restricts the infons known by others to those which the individual knows. An infon belongs (signifies in the figure) to one individual (atomic infon), to a number of individuals (compound infon), or none (zero infon). True infons are information (i.e., the information type is a subtype of the infon type). Derivable information is considered to be

---

\(^2\)The development of the database schema is based on the higher-order entity-relationship model [Tha00a] that supports co-design of structuring and functionality of information systems by a high-level specification language.
trivial depending on the derivation system. Individuals and non-individuals may possess in-
fon which belong to an individual forming thus an association between the signifies relationship types and either non-individuals or individuals. We may assume that each owned infon is also possessed by its proprietor (it may however not be the case in applications such as the health card since some infons (e.g., having cancer) may be withheld from individuals). The possession may be known (recorded through the property_Known class) or unknown. Known possessed infons may be known by other individuals or non-individuals depending on whether the sharing of information is consented or contracted sharing. Additionally infons may be classified within a privacy classification schema.

3 Management of Infons of Individuals

Infon management is based on the separation of matters of concern that include the following:

Data are managed by the database systems of individuals and non-individuals. The database systems of the individuals is equipped with an export facility for data that might be provided to other agents. The export is based on predefined private information schemas or views. The result of an export enactment is small units of content that includes data together with meta-data describing the data and its possible usage.

Infons are small units of content and are classified according to their truth and the applicable classification schema. Compound private information extends the infons of an agent.

Agents have their own profile [Tha00b] (consisting of private data (if applicable), access restrictions, and privacy and security profiles) and their own portfolio [Tha01] (consisting of permitted tasks, forbidden services, access rights, applicable contracts and consent).

Deliverables are generated by the privacy enhanced information system upon requests by agents.

3.1 The Model of Infon Possession Management

The playout of content is based on a view management system. This system binds the infons to individuals depending on the kind of possession, proprietorship, and the strategy for orchestration in the system. The general structure is shown in Figure 6.

The strategy for privacy orchestration is based on a workflow specification. Each workflow step uses views that are defined in terms of infons. These views compile compound private information that consists of two spheres of private information:

1. Atomic private information embedded into compound private information $C$. So in general: 
   \[ C(\mathcal{A}_{V_1}, \mathcal{A}_{V_2}, ..., \mathcal{A}_{V_n}) \Rightarrow \text{subset of atom}(\mathcal{A}_{V_1}) \land \text{subset of atom}(\mathcal{A}_{V_2}) \land ... \land \text{subset of atom}(\mathcal{A}_{V_n}). \]

   For example, in the view of a husband we can find the atomic private information that he is married, date of marriage, marriage certificate number, etc. but the identity of the wife is excluded.

2. Zero-information that links two or more atomic information in atomic databases.

Thus, in general an individual has:

(A) Proprietary database: This database includes proprietary private information as described previously.
(B) Possession database: This database includes others’ private information in possession of the individual as described previously.

(C) Compound database: This zero-information database facilitates linking shared private information as described previously.

Further, a non-individual has:

(B) Possession database: This database includes others’ private information in possession of the individual as described previously.

(C) Compound database: This zero-information database facilitates linking shared private information as described previously.

3.2 System Support

The architecture of the envisioned enhanced privacy information system is displayed in Figure 7.

Although any database management system could be used as a basis for an enhanced privacy information system we chose the content management system CoreMedia SCT (Smart Content Technology) [Cor04] for the implementation of our experimental prototype. CoreMedia SCT supports generation, maintenance and controlled delivery of complex content structures. The system components of CoreMedia SCT like the storage engine or the delivery server are designed to support scaleable, highly performant, highly available, and secure information systems for a distributed collaboration of its users.

CoreMedia SCT repositories are used for storing the content. Due to the possibilities of the SCT document model and programming interface, state of the art, encryption technology can be seamlessly integrated. Additionally, we propose an infon possession tracking system.

Content managed by the CoreMedia SCT might be the suite of encrypted infons, the workflow content that manages agents, atomic information, the signification, the possession, the known possession, or the sharing of infons. CoreMedias playout facilities deliver this content to individuals and non-individuals depending on the profile and portfolio of agents.
The preprocessor controls the legitimacy of requests issued by agents. It adds the portfolio and profile of the issuer to the request. Depending on the strategy for privacy orchestration, the extended request is issued to the content management system. The filter is based on the view depicted in Figure 7. The delivery to an agent is recorded in the content management system.

3.3 Recording Possession of Infons

Within an enhanced health card project we separate six private information schemas: patient, hospital, doctor, pharmacist, insurance and information desk schemas. These schemas are limited to private information and embeds in their structure information about proprietary of, possession of, and knowledge of private information. The actual complete database may exist in one system (e.g., hospital database) or may be scattered on different computers (e.g., patient’s laptop, doctor’s laptop) or in-between situation.

All parties participate in this comprehensive picture of the patient’s private information. It is a portion of a global picture of any individual’s private information. When a person introduces new data that requires the possession of his/her private information, the corresponding infons, views and usage data are generated. The patient’s computer and the hospital’s system can exchange information and check each other according to their schemas.

This multi-view of private information that is designed according to possession, knowledge, etc. is not limited to enterprises but -in a comprehensive database- applies also to individuals. Thus, the system, upon receiving a request to add new private information $A_{V.t}$ to its database, would activate certain procedures to ensure that $A_{V}$ is aware that $A_{V.t}$ is now in the possession of $A_{Z}$.

On his/her part, the individual $A_{V}$ can be permitted to access $A_{Z}. Possession. A_{V}$ to look at all private information about him/her in possession of $A_{Z}$. So the relationship between $A_{V}$ and $A_{Z}$ is reciprocal with regard to possession of private information. Various scenarios can be incorporated in this relationship with regard to updating the private data, correcting it, etc. In ideal situation $A_{Z}. Possession. A_{V}$ should always be equal to $A_{V}. Known. A_{Z}$. Thus, the individual and the enterprise have identical copy of the private information of the individual that is in possession of the enterprise. This can also be applied for sub-schemas of the enterprise that contain portions of this private information. Several routines can be performed automatically on both sides, the proprietor and possessor, and appropriate reports of discrepancies can be produced.

The enterprise schema of $A_{Z}$ may have “hidden data” in $A_{Z}. Possession. N$Know that
includes private information that is legally hidden from the proprietor of that information. For example, in a hospital DB system, the patient may be aware of $A_Z$.Possession.Know.$A_V$ but is not aware of $A_Z$.Possession.NKnow.$A_V$.

4 Conclusion

This paper introduces a formalism to specify information privacy infons based on a theoretical foundation. The resultant formalism is then applied to the notions of awareness of possession, consent, and certain known privacy rules. Current privacy research lacks such formalism. The new formalism can benefit in two areas. First in introducing precise definitions of informational privacy notions. It also, can be used as a base to develop a formal and informal specification language. Informal specification language can be used as a vehicle in specifying different privacy constrains and rules. Further work can develop a full formal language that can be used in privacy systems.

In this paper we report the findings, the constructs and the implementation for an enhanced health card project that is currently finalized [Fie04] in a project of the Kiel information systems engineering group. The experience we gained has shown that further development on the basis of security meta-profiles, on patient information facilities, infon tracking functions, and encryption mechanisms is necessary. We have shown that in a protected environment these requirements can be satisfied. We are in doubt that a completely satisfying solution can be provided in open environments although most systems that can be envisioned or are currently used will be open systems.

Our current investigation are centered around development of reliable correct systems similar to [WLT05] by providing a framework that traces and protects users against ‘bad’ services displayed in Figure 1. We preferred closed privacy supporting systems. Alternatively, open privacy supporting systems may be developed using enforcement techniques [Tha00a] or information wallets [AF05b]. The change management for infons has been extended in order to cope with changes from zero to private, from private to zero infons. The logics of known infons, possessed infons and not known infons can be extended to trusted, whitelisted, known, and forbidden services using modal epistemic intuitionistic logics. We plan to investigate privacy keeping through noising.

References


Acknowledgement. The authors thank the reviewers for their comments and Joachim Biskup for his advices that led to improvement of the paper.
LIFE CASES
An Approach to Address Pragmatics in the Design of Web Information Systems

Klaus-Dieter Schewe
Massey University, Department of Information Systems & Information Science Research Centre, Palmerston North, New Zealand
k.d.schewe@massey.ac.nz

Bernhard Thalheim
Christian Albrechts University Kiel, Department of Computer Science, 24098 Kiel, Germany
thalheim@is.informatik.uni-kiel.de

Keywords: storyboarding, pragmatics, life case, application domain description

Abstract: On a high level of abstraction a Web Information System (WIS) can be described by a storyboard, which in an abstract way specifies who will be using the system, in which way and for which goals. While syntax and semantics of storyboarding has been well explored, its pragmatics has not. This paper contributes the first step towards closing this gap. For this we present life cases, which capture observations of user behaviour in reality. We discuss the facets of life cases and present a semi-formal way for their documentation. Life cases can be used in a pragmatic way to specify a story space, which is an important component of a storyboard.

1 Introduction

A Web Information System (WIS) is an information system that can be accessed through the world-wide-web. On a high level of abstraction a WIS can be described by a storyboard, which in an abstract way specifies who will be using the system, in which way and for which goals. In a nutshell, a storyboard consists of three parts:

- a story space, which itself consists of a hierarchy of labelled directed graphs called scenarios, one of which is the main scenario, whereas the others define the details of scenes, i.e. nodes in a higher scenario, and a plot that is specified by an assignment-free process, in which the basic actions correspond to the labels of edges in the scenarios,
- a set of actors, i.e. abstractions of user groups that are defined by roles, which determine obligations and rights, and user profiles, which determine user preferences,
- and a set of tasks that are associated with goals the users may have.

In addition, there are many constraints comprising static, dynamic and deontic constraints for pre- and postconditions, triggering and enabling events, rights and obligations of roles, preference rules for user types, and other dependencies on the plot.

While syntax and semantics of storyboarding has been well explored, its pragmatics apart from the use of metaphors (Thalheim and Düsterhöft, 2000) has not. Pragmatics is part of semiotics, which is concerned with the relationship between signs, semantic concepts and things of reality. Main branches of semiotics are syntax, i.e. the construction of the language, semantics, which is concerned with the interpretation of the words of the language, and pragmatics, which is concerned with the current use of utterances by the user and context of words for the user. Pragmatics permits the use of a variety of semantics depending on the user, the application and the technical environment.

This paper contributes the first step towards pragmatics of storyboarding. For this we present life cases, which capture observations of user behaviour in reality. The idea is that these observations give rise to both a concrete scenario for the usage of the WIS including roles and user profiles. We then derive the candidate scenarios by abstraction. Further integration and refinement of these scenarios gives rise to the story space. We deal with the problem of extracting user models in a separate article. The third component of the storyboard, the tasks, has already been subject of strategic modelling. The life cases permit the refinement and further decomposition of these tasks.
2 Related Work

Most methods for web application engineering such as OOHDM (Schwabe and Rossi, 1998), WebML (Ceri et al., 2003), HERA (Houben et al., 2003), WSDM (De Troyer and Leune, 1998) and variants of UML (Conallen, 2003; Lowe et al., 2002) pay little or even no attention to high-level modelling. The rationale underlying our work is that this is far too little to capture strategic and usage issues of WISs. The integration of goals and soft goals into the information systems development process has been proposed by (Mylopoulos et al., 2000; Giorgini et al., 2002). The detailed distinction of intentions as targets, objects, objectives and aims is based on linguistics. The integration of the temporal dimension is necessary because of the information systems context. The extension by the representational dimensions has been made in order to specify aspects of WISs.

Life cases extend and formalise scenarios used in software engineering and user modelling (Courage and Baxter, 2005). Early work like already discovered that human computer interfaces can be developed in techniques of movie writing or play development (Vale, 1982). Life cases have already been envisioned in (Carroll, 1991). Scenarios as defined in software engineering are examples of interaction sessions. This kind of scenario starts with an outline of the interaction, and during requirements elicitation, details are added to create a complete description of interaction. They are specified by a general description of the start and termination states of the system, a description of the normal flow of events, and a description of concurrent events (Sommerville, 2004). Later scenario research has been integrated into agile programming. Customer involvement is one of the requirement of agile methods. Customers should be integrated as early as possible into the development process (Ambler, 2002). Both approaches are too much oriented towards systems and do not consider life cases as we do. Scenario development has been applied to development of websites in (Rosenfeld and Morville, 1998).

In (Harel and Marelly, 2003) life cases were integrated into the entire software engineering process. Recently, software engineering research developed an engineering approach to simple life case. Business use cases (Robertson and Robertson, 2006) generalise the requirements shell of (Robertson and Robertson, 1999) that is based on verbal description of event/use cases, requirement types, the rationale, originator, fit criterion, customer (dis)satisfaction, supporting materials, and the history. Our life case specification is not as complex as these business use cases. They combine intention, users, actors, constraints/assumptions/scope or context, tasks, migration, risks, costs, documentation, testing, and requirements for functions, data, look and feel, usability, humanity, operating, maintenance, support, security, culture, politics, legal and performance. We concentrate on the requirements the user has as such.

3 Facets of Intention

The description of intention is based on a clear understanding of aims and targets of the WIS including a specification of long-range and short-term targets, expected visitors, characteristics of this audience, tasks performed by the users, necessary and unnecessary content, and finally an understanding of restrictions of usage.

Utilisation scenarios are developed on the basis of intentions. An intention specifies what one purposes to accomplish or do, i.e. one has in mind to do or bring about. It has four facets that are based on the general characteristics of WISs:

**Purpose facet:** The purpose of stakeholders specifies an anticipated outcome that is intended or that guides the planned actions. It may be based on two additional pieces of information: The aims specify what is intended to be attained and what is believed to be attainable. The objectives are more general than the aims. They specify something toward which an effort is directed, e.g. goals or ends of an action.

The purpose is already specified at the strategic layer. It depends on the audience intended for the WIS, and is influenced by the mission that is determined by the WIS provider.

**Intent facet:** The intent suggests a clearer formulation or greater deliberateness. It distinguishes between the following two aspects: The targets of stakeholders specify the steps of a plan and its termination or satisfaction conditions. The object of stakeholders is related to wishes or needs of users, and specifies an effort or activity of a user in order to satisfy the need.

The intent facet is related to tasks the user wants to accomplish. The intent may be ordered into major intents specifying the main uses of the system and minor intents that may be only partially of interest. Intents may be generalized to themes that represent classes of intents.

**Time facet:** The time facet is used for specification of the general time restrictions such as the design, which implies a more carefully calculated plan,
and the end, which stresses the intended effect of an action, often in distinction or contrast to the action or means as such. Time facets may be very general. In this case, we use occasions to represent an entire class of time frames.

**Representation facet:** Intentions are described or annotated through utterances, words or pictures. The word representation is related to word fields used in the strategic model. The icon representation is based on metaphors. Word fields may be specialised to concept fields discussed later in this chapter. Representation is deeply dependent on the cultural environment and the community that uses the WIS. Intentions can be supported by providing stimuli that rouse or incite activity.

Intentions may be restricted by a **scope**. The scope allows to concentrate on the main aspects. Typical restrictions are the cultural environment, education or other profile properties of potential users, or specific time facets for utilisation of the WIS.

The first two facets of intention have a general form (objective, object) and a more concrete form (aim, target). The purpose facet depends on the mission and the audience, whereas the intent facet depends on the tasks. Therefore, the first facet specifies the 'what', the second the 'how', and the third facet the 'when' of an intention. We may either concentrate on a more general specification or a more concrete one.

The different facets may be considered separately or altogether in a condensed form. The detailed consideration is necessary, whenever a fastidious audience requires sophisticated content. High demands come into consideration due to the content provided, the community that must be satisfied, the sophisticated functionality required for the WIS, or the high attention the WIS gains.

**Example 3.1** The purpose facet is often considered to cover 'soft intentions' or by 'business rules'. In an information service a user may be interested or becoming more interested in some content. The intent facet is different and mainly driven by tasks the user tries to solve.

Similarly, we may derive a number of intentions a user has in mind whenever s/he visits an edutainment site:

**Aim:** An aim of an edutainment user may be to obtain a certificate that proofs the success of learning. Aims of providers of edutainment sites might also be binding the learner to some products such as the software of a supporter of the website. Typical general aims within an edutainment site are to grant equal rights to everybody; nobody should have higher rights than other learners.

**Objectives:** Typical objective of using an edutainment site are greater ease of learning, greater pleasure or satisfaction during learning, and more fun. Another general objective is security and privacy. Edutainment applications also host confidential information, e.g. information on progress and errors. The learners need to be sure of the security and privacy of their data. They should be informed of the security precautions.

The audience of an edutainment site may be rather unspecific or more specific such as students of a college or analysts of a bank.

The mission of an edutainment site is to support learning by providing easy-to-grasp knowledge. At the same time, the provider may be interested in increased visitor-to-customer conversion rate, increased number of returning customers, and increased revenue.

The intent is related to the more concrete tasks a user has in mind while visiting a WIS.

**Example 3.2** Intents come directly from analysing the needs and the demands of users. Some possible tasks a user may want to accomplish in an edutainment site are the following:

- A user realises that certain knowledge or information is necessary for solving a task. For instance, an analyst of a bank wants to know what the changes are in the customer community that led to a rise of faulty credits.
- A student in a college needs to know methods for analysing data. The student is interested in a data mining course that provides material on his/her educational level, permits learning the content interactively, and is comparable to the material the student is currently working with in the college.

Possible intents in an edutainment site are the following:

**Objects:** There are a number of objects such as faster task completion, successful completion of more tasks, or commission of fewer errors.

**Targets:** In the bank analyst case, the analyst needs to know data mining methods, to capture the achievements and the disadvantages and pitfalls of such methods.

The themes in the bank analyst case could be the interest in learning association methods, preprocessing of data, and prediction methods.

The time facet represents general time restrictions such as the time or the interval for visits of the WIS, the time and the interval of repeated visits, or the temporality that may force a user to leave the site. The
representation facet represents the flavour or atmosphere of a site, which is largely determined by the other three facets.

The set of intentions we should consider may be rather large. Moreover, the facetized representation may become too difficult to manage and to satisfy. The order we may use depends on the audience and on the provider. The audience is oriented towards solving tasks. The provider has a ‘mission’. We can use these two restrictions to figure out which kind of website supports this profile, to harmonize our understanding with the corporate identity, to order the target website supports this profile, to harmonize our understanding how the site will look like in realisation (short-term, long-term), and finally to develop an understanding how the site will look like in two years from now on.

Example 3.3 The intention prepare for a visit is based on a number of intents such as addressing visitors beforehand for some purpose, use, or activity. It includes the aim to put the visitor of the WIS in a proper state of mind. A task associated with preparation may be to work out the details of a plan in advance. The task may be extended by putting pieces of information together into a compound form or preparing a report. The time facet may range from ‘now’ until ‘next opportunity’. Typical metaphors supporting this intention are baskets, descriptions, and cultural journals.

The intention prepare for a cinema visit extends the intention of prepare for a visit by a certain content, a refinement of the time facet to the next possible visit, and a refinement of the purpose facet now targeting at entertainment.

We can combine the description of intentions in a semi-formal way as follows. The items in the list are optional except the first one.

- **Intention space**: (intention name)
- **Purpose**: (outcome description)
- **Aims**: (list of aims)
- **Objectives**: (list of objectives)
- **Intents**: (outcome description)
- **Targets**: (list of weighted targets)
- **Objects**: (list of weighted objects)
- **Themes**: (class of intents)
- **Time**: (outcome description)
- **Design**: (general flow)
- **End**: (effects, termination conditions)
- **Occasion**: (list of objectives)
- **Presentation**: (general style guide)
- **Atmosphere**: (general description of atmosphere)
- **Metaphors**: (list of metaphors)
- **Based On**: (tasks, audience, mission, goal)

4 Life Cases

Life cases allow to overcome the information overload and lost in the hyperspace syndromes typically observed for WISs. For completion of tasks users need the right kind of data at the right moment of time, in the right dose, of the right form, in the complete extent, and within the restrictions agreed upon in advance. Moreover, users are limited in their abilities for verbalisation and digestion of data, and by their habits, practices, and cultural environment.

These limitations may cause intellectual overburdening of users. Most systems that require sophisticated learning courses for their exploration and utilization did not consider these limitations and did not cope with real life situations. The approach we use for avoiding overload is based on observation of real applications before developing the system.

We may extract life cases from observations in reality, which are very useful source, whenever a WIS is going to be developed from scratch or is required to provide a ‘natural’ behaviour. In the latter case users are not required to learn the behaviour of the WIS. Instead, the user can continue using a ‘classical’ behavioural pattern.

Example 4.1 As a motivating example let us consider the life case relocation of a person, which consists of

- the change of basic relocation data including the possible removal of data on the old location,
- the change of official documents such as the passport,
- the optional change of relation enhancements such as the registration of pets, relocation of cars,
- the change of personal specific data such as family enhancements, or relationships to religious bodies,
- the change of data for additional relocation announcements such as tax, insurance changes, and
- specific additional tasks such as applications for housing allowances.

The person acts in the role of an issuer. We observe that relocation is enhanced by the profile of the issuer, by the specific tasks related to the relocation of the issuer, by specific laws and regulations, and by advanced functionality required for associating the life case with other life cases.

The life case relocation consists of steps such as change of address data, change of data for associated people, change of registration data for cars, pets, etc., change of specific data, e.g. data for public authority responsible for aliens, change of data for social
aid, etc. These steps are bundled together due to their relationship to one person and to one life case. The associations may be represented by adhesion of different steps, e.g. representing the association of steps by a hypergraph.

4.1 The Concept of Life Case

Life cases are characterized by:

Observations: We are interested in the collection and assessment of behaviour relating to the specific application. This would typically involve an observation of behaviour of users in real environments, including a background check that relates usage to intentions, goals or tasks.

Processes: This involves arranging all the actions observed in an application into a main logical and coherent pattern. In some cases, deviations of the main pattern must be modelled through exceptions. In most cases, we can use parallel execution of independent actions.

Assessment: This involves the reconstruction of the sequence of actions and specific behaviour of users. This will aid in understanding the role each individual has within the story. It assists in developing the user profile.

Individual profiles: A list of background including physical, and behavioural characteristics of individuals is conducted. This list can also be used for deriving the most appropriate interview technique we discuss below.

Interpersonal coherence: A variation in the activity will relate to variations of other life cases.

Significance of time and place: The choices made also depend on mobility, surrounding, and schedules of events of interest.

Characteristics of the life case: Individuals using a service may be grouped by characteristics. Based on this grouping a similar behaviour is observed.

Experience and skills: Individuals may have their own experience with services provided in real life and thus use different behavioural pattern of service employment.

In general, life case studies are designed to produce a picture of service employment that is as accurate as possible. Determining why, how, where, when and why a service is called using what data provides a better picture for utilisation scenario. As life cases are used for quality control, we must carefully record our observations. We either use a natural language specification or a semi-formal one as described later.

Example 4.2 Let us consider the support of hotel search within an information service. In this case we may observe the behaviour of individuals in travel agencies while seeking for hotels. We observe that in most cases search based on associations is preferred over search by main properties such as name, address or facilities. Associations may be of a large variety, e.g. convenience to reach a hotel, location in certain maps, places of interest, or events that have caused the search. Other search criteria may be bargain or bundled offers. At the same time we observe that hotel search is combined with other intentions of users such as visiting cultural institutions. It may depend on results for search of other individuals.

Another example of a life case study is the booking of train tickets depending on individuals, offers of railway companies, circumstances of individuals are using trains, etc.

4.2 Life Case Development

Life cases may be developed and interpreted step by step:

1. The first step during life case collection is the survey of possible application cases we might consider. The observations could have more than one meaning and may follow a variety of user-related goals. In this case we consider the most likely meaning.

2. The second step involves a deep assessment of the life cases. We extract the different execution orders, the relationship among actions, and the roles individuals play these actions.

3. The third step extracts those life case characteristics that are distinguishing features and are relevant for time and location. At the same time we search for similarities within these life cases.

4. The final step is concerned with the characterization of potential users, their behavioural patterns, and their roles in various stages of the life case.

Collectively, this information will produce a picture of the life case we are intending to support by a WIS. This may produce further life cases, or may aid in reducing the amount of development. It may result in a prioritisation of life cases under consideration, assist in the linkage of potentially related life cases, assist in assessing the potential of the WIS development, provide the WIS developers with relevant leads and strategies, and keep the WIS development on track and undistracted. These life case are mapped to scenarios in the sequel. The integration of scenarios can also be based on life cases.
**Example 4.3** Let us consider life cases of an information service for a city or a region:

- **Attracting visitors:** The information we are providing is based on some information need visitors may have. The life case follows those that we observe during marketing activities.
- **Inhabitants information:** The life case is based on selected information chunks that may be of interest to individuals, an ordering of the corresponding available information, and a derivable personal newspaper.
- **Informing tourists:** The life case is similar to those observed in city information centres.
- **Providing official city information to inhabitants:** This life case follows the message board metaphor that is used for newspaper information.

During the development of city information services such as the service www.cottbus.de, we have made a life case analysis for city information services and detected around 30 different life cases. We can categorize these life cases by the content and information demand of individuals. We distinguish:

- **Life cases related to tourist content for inhabitants permanently living in the city, for inhabitants temporarily living in the city, for short-term tourists and business people on short-term trip, for vacationists planning a trip for more than 5 days, for teenagers with uncertain wishes, for seniors with particular interests, for week-end visitors, for visitors of highlights, festivals, etc.;**
- **Life cases related to official content for inhabitants on search through directory of public authority, for inhabitants on search for emergency cases, for inhabitants orienting themselves before applying at public offices, for inhabitants interested in problems of city management, etc.;**
- **Life cases related to business content, e.g. for investors considering engagement in the area.**

For the collection and development of life cases it is normally a good idea to interview the personnel currently providing the service.

Life case should be checked against sufficiency. As they may have a variety of traces, we add two stages to life case analysis:

**Exploration:** The actual life case is provided to WIS customers and incorporated into their processes. Life case exploration can be conducted in normal environments of the user such as home or work. Then users can show the actions while they are explaining their aims and targets.

**Apprehension:** Finally, cross checking of life cases and utilisation scenario is used for correction, extension and affirmation of the developed utilisation scenarios.

During life case elicitation and specification users are confronted with sketches of scenarios. We ask what users think about these conceptualisations. The feedback is used for the refinement of life cases. We may use affinity diagramming, in which case we arrange the individual conceptions we discover on a blackboard, associate them, and discuss their relationship to the general characteristics of WISs. Another technique is based on card sorting similar to the model-view-control cards used in software engineering. In this case cards represent simple life situations. Their associations are then used for organising the conceptions. These sketches can also be used for discussing deviations from the normal case and for search of exceptional life cases. Instead of directing users to specific life cases we can show them two or more alternatives for the problem solution.

Life case analysis goes beyond task analysis. It is simple and easy to carry out, lightweight and straightforward to understand, and yet quite accurate in identifying the real demands users of a WIS might have. While observing real life cases and mapping them to life cases that must be supported by the WIS we carry out a user-centered development. Additionally, we may identify and resolve conflicting or competing intentions. These conflicts can be resolved on the basis of life cases by accommodating both intentions, i.e. fulfilling the stakeholder intentions and supporting the demand of users.

Besides observation of real life situations life case detection can also been based on interview techniques. Research on artificial intelligence has also resulted in a number of interview techniques:

- Unstructured interviews can be considered as an informal and explorative conversation on goals a user is following and on tasks a user has to complete. Unstructured interviews observe the rules that are applied to brainstorming. All interruptions should be avoided. The questions asked must be short and simple to answer and should be open-ended questions. Interview partners should share their thoughts and experiences. There is no judgement, confrontation or condescension. Users should not be lead toward a certain scenario. Unstructured interviews should only be interrupted if clarification is required. They need consecutive feedback in order to give the interviewees the impression that the interviewer is listening. While the interviewees are talking, everything that seems to be important is written down and used for later questions.
- Structured interviews are based on query plans.
Life cases can also be detected by observing and critically analysing products of competitors. Elicitation of life cases from existing solutions is based on specifications, results from interviews with business users, excerpts from documents and spreadsheets, analyzed messaging and transactions, knowledge on the solutions that are currently used, meta-data and context information on the utilization within the current framework, and third party information. As reasons for restrictions cannot be captured, the copying of already existing solutions can only be used in exceptional situations.

Users may also use protocols of “loud thinking”, in which case they are provided with a real-life problem of a kind that they deal with during their working life, and asked to solve it. They imagine that they are solving the problem presented to them. As they do so, they are required to describe each step and the reasons for doing what they do. The transcript of their verbal account is the protocol. In this case, they work and explain their current work. These protocols can be the basis for capturing a scenario. This interview technique should be combined with other interview techniques.

There are other interview techniques that might be useful for life case elicitation such as the ladder- or grid techniques. In these cases, a hierarchical structure of the application domain is formed by asking questions designed to move up, down, and across the hierarchy.

### 4.3 Semi-Formal Representation of Life Cases

Although it is sufficient for life cases to be stated in natural language, we may use a semi-formal representation instead using the following template:

- **Life case name**
- **Outcome description**
- **List of user tasks**
- **List of problems**
- **General characterisation**
- **List of objectives**

| Life case flow | (general graphical description) |
| Milestones | (graph of milestones) |
| Content consumed | (consumed content items) |
| Content produced | (produced content items) |
| Themes | (class of intents) |
| Figures | (actors list) |
| Expected profile | (general profile description) |
| Collaboration | (general collaboration description) |
| Context | (general context description) |
| Time | (temporality limitations) |
| Place | (assignment of places) |
| Legacy | (names of documents) |
| WIS | (general WIS context) |
| Representation | (general behavior) |
| Approaches | (general description of approaches) |

This template captures the following components of life cases:

**Characterisation**: Life cases are characterised on the basis of strategic issues, the problem statement, background and objectives for the life case, the methodology that is used for solving the life case, and by describing the basic modules that are used for solving the life case. The characterisation is harmonized with intentions, tasks and goals.

**Life case flow**: The life case flow combines the observations we made, the processes involved, and the data which are consumed or produced. The life case flow is mapped to a scenario.

**Figures**: We develop profiles, especially those of individuals, as part of the WIS utilization portfolio, and interpersonal coherence specifications.

**Context**: Time and location is explicitly described for life cases. The applicability of life cases is restricted by regulations, laws, and orders. These restrictions are seldom given in an explicit form. In addition, the context of life cases is given by the provider, the intended audience, the utilization history, and the availability of data due to the technical environment. We also use this information for the context specification.

**Requirements**: Life cases are restricted by habits, general approaches, good practices, and boundary conditions for their application. They presuppose experiences and skills of the users involved.

Note that life case we intend to support by a WIS can be completely different in real life. Sometimes we need a complete reorganisation of the business activities. In this case we should not map the real life case to a suite of associated scenarios, but rather envision...
a better organisation of the tasks and goals and then map these to a new envisioned hypothetical life case.

Let us now outline briefly how life cases can be used in a pragmatic way to specify a story space. More precisely, we just want to derive a prototypical scenario. Other scenarios may result from other life cases, and the scenarios collected this way can be integrated and then be subject to further refinement. Let us concentrate on the life case relocation again.

Example 4.4 In the case of the relocation of a person the steps identified in Example 4.1 give immediately rise to scenes in a scenario, i.e. in addition to an entry scene, say start, we obtain scenes for change of address data, change of data for associated persons, registration data, change of specific data, change of data for social aid. For a start we only have a single actor citizen.

In principle, the visit of any of these scenes is optional, which gives rise to a classification of actors into citizens with children, citizens with pets, foreign residents, etc. The life case of a citizen with children gives rise to a scenario for the change of data for associated persons, while the life case of a citizen with pets gives rise to a scenario for the change of specific data, etc.

5 Conclusion

In this paper we introduced the concept of life cases as a contribution to supporting pragmatics of storyboarding, which closes a gap in our development methodology. The general idea is to start from real life observations and to characterize them in a semi-formal way. This gives rise to prototypical scenarios, tasks, roles and user profiles by abstraction. Several of such prototypes can then be integrated and refined to obtain the desired storyboard.

The concept of life cases is new and original. It has already successfully been applied in our web information system projects, e.g. for information services (e.g., www.cottbus.de), for entertainment systems (e.g., DaMIL), and for community services (e.g., SeSABS). Use cases in UML (Conallen, 2003) and business use cases (Robertson and Robertson, 2006) share some similar intentions, but are far too simplistic to capture the same information as the novel life cases. However, life cases still contribute only a part of story board pragmatics. They have to be complemented by user models, which should give rise to a deeper understanding of actors and contexts. Both topics will be addressed next and published in due time.

REFERENCES


Facets of Media Types

Klaus-Dieter Schewe¹, Bernhard Thalheim²

¹ Massey University, Information Science Research Centre
Private Bag 11 222, Palmerston North, New Zealand

² Christian Albrechts University Kiel, Department of Computer Science
Olshausenstr. 40, D-24098 Kiel, Germany

kdschewe@acm.org thalheim@is.informatik.uni-kiel.de

Abstract. The concept of media type is central to the codesign approach to Web Information Systems (WISs). By means of views on some database schema it permits a separation of global content from local content that is to support particular scenes of a WIS. By means of various extensions such as cohesion, hierarchies, style options and associated operations it enables interface abstraction and adaptivity to users, channels and end-devices. It can further be used for modelling collaboration, session support and contextual information, and by means of an associated dynamic logic provides the basis for formal reasoning about a WIS design. In this paper we give a brief survey of the potential of media types as a very powerful abstraction mechanism for WISs.

1 Introduction

A Web Information System (WIS) is a data-intensive system that is accessible via the world-wide web (WWW). As such the design of such systems has attracted a lot of attention by researchers in the area of conceptual modelling, and various modelling languages and design frameworks have been developed so far [2–4, 6, 7, 15, 16], all having their individual merits but none giving a complete answer, how all the challenges in WIS modelling (see [13]) are addressed.

In this paper we give a brief survey of the potential of the concept of media type, the central notion in the codesign approach to WISs [15]. The naming of the concept relates to the common understanding of “media” as means for mass communication suggesting that the WWW has become such a means. “Types” then refer to the classification and formal abstraction from content and functionality of such communication means.

As such a media type is first an interface abstraction related to and extending the notion of dialogue type introduced in [14]. The rough idea is that the global data content of a WIS is captured by means of an instance of a conceptual database schema, while content needed at the interface is modelled by means of views. As emphasized in [13] this includes the modelling of the navigation structure, which in various other approaches is treated as a separate add-on feature. Furthermore, same as with dialogue types operations expressing the
functionality of the WIS, are coupled with the views. In this way, media types capture form-based interfaces [5, 9] in an abstract way.

In doing so, media types provide a perfect mechanism to refine the scenes and actions of the storyboard that result from analysing and modelling the usage of the WIS [15], thus linking the conceptual model to the identified systems requirements. By means of associated style options [11] they also provide the mechanism for system implementation including a structural part (XML clusters), a behavioural part (scripts) and a layout part associated with page grids.

However, media types are more than just interface abstractions. By means of hierarchical versions they subsume OLAP-like features allowing users to switch between representations at various levels of granularity. By means of cohesion they allow an automatic split of information adhering to the restrictions imposed by channels, end-devices and user preferences. In this sense media types enable derivable adaptivity. Furthermore, as outlined in [12] consistency and personalisation with respect to content and functionality can be achieved by logical reasoning about media types.

Most WISs are centred around the actions of individual users, whereas the need may arise to support collaboration. This can also be captured by media types using associated exchange frames [1] that support cooperation, communication and coordination within user groups. This is related to the fact that media objects, i.e. instances of media types, can be used to capture information about sessions, which are linked to complex scenes in the storyboard.

Finally, media types permit dealing with the often mentioned phenomenon of “loss in cyber space”, which actually is a loss of navigation context. Adapting an idea from Contextual Information Systems [18] we obtain query macros that allow a user to traverse back a path in the storyboard and to explore alternative access paths, respectively [8].

2 Interface Abstraction

In the following let $S$ denote some conceptual database schema. The introduction of media types in [15] was based on the higher-order Entity-Relationship model (HERM) [17], but we already emphasised that the choice of data model is of minor importance. A model (such as HERM) with richer structuring mechanisms will simplify the definition of views, while a poorer model – even if it has the same expressive power – will require more sophistication in the queries that define views.

Furthermore we assume a type system comprising base types and various type constructors, e.g.

$$t = b \mid \emptyset \mid (a_1 : t_1, \ldots, a_n : t_n) \mid \{t\} \mid [t] \mid \langle t \rangle \mid (a_1 : t_1) \oplus \cdots \oplus (a_n : t_n)$$

would define a type system with not further specified base types $b$, a trivial type $\emptyset$, and constructors for records, finite sets, lists and multisets, and disjoint unions. Usually, we assume that among the base types there is a type $URI$ representing abstract surrogates for URIs.
The semantics of such types is primarily defined by their domains, i.e. with each type \( t \) we associate a set of values \( \text{dom}(t) \). For a base type the domain is assumed to be countably infinite. For the trivial type the domain is a singleton set \( \{1\} \). For constructed types we use the usual recursive definitions. Furthermore, we associate basic (fixed) operations with the type system, e.g. projection on record types, union for sets, etc., but we dispense with listing them here.

The type system forms the basis for the definition of content type expressions. We simply extend the type system permitting \( r : M \) to appear in lieu of a base type, with \( r \) and \( M \) being names for a reference and a media type, respectively, both taken from some unspecified countable pool of names. Replacing \( r : M \) by URI results in a proper type, called content representation type.

The core of a media type \( M \) is defined by an interaction type, which comprises a content type expression \( c(M) \) (with corresponding content representation type \( t_M \), a query \( q_M \) defined on \( S \) with output type \( (\text{id}: \text{URL}, \text{value}: t_M) \), and a finite set \( Op_M \) of operations – more formal details can be found in [15]. An interaction schema \( I \) is a finite set of interaction types that is closed under references, i.e. whenever \( r : M \) appears in a content type expression \( c(M') \) with \( M' \in I \), then \( M \in I \) must also hold.

The semantics of interaction types and schemata is defined by means of the defining queries. Take an instance \( db \) of the database schema \( S \) and execute the queries \( q_M \) for all \( M \in I \), which will result in sets \( db(M) \) with values that are actually pairs \( (u, v) \) with a URL \( u \) and a value \( v \) of type \( t_M \). We request that whenever a URL \( u' \) appear in such a value corresponding to the reference \( r : M' \) in the content type expression \( c(M') \), then \( db(M') \) must contain a value \( (u', v') \). Furthermore, URLs must be globally unique. In this way we obtain an abstraction from the structure at the interface. The value \( (u, v) \in db(M) \) represents the content \( v \) at the abstract URL \( u \) including references to other units of content, i.e. including the navigation structure. In its simplest form this is abstraction from web pages, but as we shall see later it can be more than that. As emphasised in [15] and discussed with respect to different query languages, the abstract definition of interaction types requires the use of query languages that are able to create abstract identifiers, i.e. the URLs, and in many cases will need a fixed-point construction.

Operations associated with an interaction type permit updating the underlying database and to open and close interaction objects. For this, we require a signature describing input- and output-parameters as well as a selection type, and an operation body that can be defined by means of usual programming language constructs. The selection type specifies, which part of the content value must be selected as a precondition for the operation being executed. In [9] we illustrated that this form of interaction types, which is more or less the same as the dialogue types in [14] captures form processing in a natural way and on a reasonably high level of abstraction, but it is not limited to this.

For this, consider storyboarding, the precursor of conceptual modelling in the codesign approach to WISs. A storyboard consists of three parts: (1) the story space describing scenes, the transition between scenes, actions associated with
these transitions, and the plot, i.e. the detailed action scheme, (2) the actors describing roles, right and obligations associated with roles, user profiles, and preferences associated with these profiles, and (3) the tasks describing meaningful chunks of WIS usage, the actors involved, their goals, the scenes associated with tasks, and the actions required for task execution. Scenes are hierarchically organised in the sense that each non-elementary scene gives rise to a scenario, which is composed in the same way as the whole story space, in particular giving rise to sub-scenes.

The basic interface abstraction mechanism described above associates a media type with an elementary scene leading to page abstraction as outlined above. However, there is no formal reason not to associate media types also with non-elementary scenes, thus leading to interface abstraction on a higher-level. We will discuss the benefits of this high-level interface abstraction in the following sections.

Before doing this, let us first take a look at some other facet of media types. Media types are extended interaction types, and one of the extensions refers to layout and playout style options. For the layout the primary question is how the abstract content of a media object, i.e. a value of type $t_M$, is to be represented. Exploiting that types give rise to a lattice, this first leads to a decomposition by means of a Sperner set of types $t_{M1},\ldots,t_{Mk}$ with $\bigcup_{i=1}^{k} t_{Mi} = t_M$. Using canonical projections from $\text{dom}(t_M)$ to each $\text{dom}(t_{Mi})$ we obtain values $v_1,\ldots,v_k$ that jointly represent the content of some WIS unit. For each of the types $t_{Mi}$ a style option specifies how records, sets, lists and multisets are to be represented using maybe tables, itemized lists, forms or other layout elements, which colours are to be used, which font families and sizes are to be used, which adornments such as background images should be added, etc. In the same way the playout of operations is handled, i.e. how are operation names to appear, how is input from the user collected using forms or dialogue boxes, how is correctness of data entry ensured, etc.

With respect to the actual system implementation, the content representation type of a media type gives rise to a cluster of XML templates, while the defining query has to be mapped to a query resulting in a cluster of XML documents. The style options then give rise to translators from XML to HTML, while the operations give rise to scripts implemented in the preferred scripting language. In this way, most of the actual implementation of the WIS results from page generation with the actual content defined by the current database instance.

Furthermore, media types permit the easy use of the container metaphor. By using this metaphor the content representation type $t_M$ is only considered as the type describing the content at the server-side, while the demand at the client side may be expressed by a different type $t$. In this case, the meet $t_M \sqcap t$ defines the part of the content that is to be shipped in a “container” to this client. Any missing part has to be obtained from other media types. Style options on the client side then define how the content of various containers is to be assembled.
3 Adaptivity

Two other extensions of interaction types (turning them into media types) address the granularity of the represented data, and the adaptivity to users, channels and end-devices. The former one adopts ideas from on-line analytical processing (OLAP), while the latter one addresses splitting and condensation of information. Both extensions exploit the type lattice.

In order to address granularity we take the lattice defined by the supertypes $t' \leq t_M$ holds. In particular, $t_M$ is the smallest type in that lattice. Now take a subset $\mathcal{H}(M)$ of types in the lattice that defines a tree with root $t_M$ – this will be called a set of hierarchical versions. In earlier versions of our work we actually requested a chain instead of a tree. Obviously, be means of canonical projections from $\text{dom}(t_M)$ to $\text{dom}(t)$ for each $t \in \mathcal{H}(M)$, we obtain for each value $v$ of type $t_M$ and each $t \in \mathcal{H}(M)$ a version $v_t$ of type $t$. Furthermore, due to the fact that we use a tree, we also get canonical up and down navigation operations between these versions. As $t_M$ itself represents very condensed information, the hierarchical versions $v_t$ represent lighter versions. Which version is the preferred one to start with depends on the preferences of the user. We should remark that there is no need to define specific style options for the hierarchical versions, as these carry over by means of the meet operation in the type lattice. In the same way, other OLAP-like operations, e.g. slicing and dicing are already defined by the content type expression, and there is no need to specify them.

With respect to adaptivity, we start from restrictions on the amount of data that is to be presented to a user. A value $v$ of type $t_M$ corresponds to the full amount of data represented by the media type $M$, but this may not be needed or requested by a user, or considered to be not suitable due to network or end-device restrictions, e.g. restricted band-width, limited presentation capacity in videotext or simply user preference. The extension to interaction types dealing with these restrictions is cohesion, which permits a controlled splitting of a value $v$ of type $t_M$ into a sequence of values $v_1, \ldots, v_\ell$ of types $t_1, \ldots, t_\ell$ such that

$$\bigcup_{i=1}^{\ell} t_i = t_M, v_i$$

is the canonical projection of $v$, i.e. the decomposition is lossless, each $v_i$ contains a reference to $v_{i+1}$, and each $v_i$ is a higher importance than $v_{i+1}$. As discussed in [15] this can be achieved in basically two different ways, either by means of a cohesion pre-order, which extends the order defined by the type lattice to a total pre-order, or by means of proximity values for each pair of types in a specified maximal antichain in the type lattice.

The main difference between the two approaches to cohesion is that proximity values use an a priori decomposition of $t_M$, i.e. an anti-chain of types $t_1, \ldots, t_k$ such that $\bigcup_{i=1}^{k} t_i = t_M$. Then the types in the cohesion sequence are either these a priori given types or result from applying the join operator to some of them. The guideline for defining the types in the cohesion anti-chain is that the joint type must be maximal within the limitations defined by the end-device, channel
or user preferences, and the sum of proximity values must be maximal. In this it is guaranteed that those \( t_i \) with the highest cohesion will stay together.

The alternative of using a cohesion pre-order does not assume an a priori decomposition of \( t_M \). Then the guideline for selecting \( t_1 \) and thus define the value \( v_1 \) is to be maximal within the limitations defined by the end-device, channel or user preference and also maximal with respect to the cohesion pre-order. The process is then repeated recursively with a maximal type \( t' \) complementing \( t_1 \), i.e. \( t' \sqcup t_1 = t_M \).

Formal details of hierarchies and cohesion were presented in [15]. It is not too difficult to see that cohesion, hierarchies and style options are orthogonal extensions that can be mutually combined, i.e. for each hierarchical version the cohesion decomposition can be applied, and for each value in a cohesion sequence (for whatever hierarchical version) the style options can be applied.

4 Session and Context Support

As already sketched above media types enable interface abstraction non only on the level of elementary scenes, but can also be used for non-elementary scenes in the storyboard. In doing so media types can be used to support the concept of a session. Informally, a session comprises all scenes visited by a user from entering the systems until leaving it. Exploiting the hierarchies of scenes, a session can be represented itself by a non-elementary scene. That is, entering a session refers to the creation of the corresponding media object, which is linked to the elementary media objects used for defining subscenes. Technically, we model this by links from the media types associated with the elementary scenes to the overarching session scene. Such links can be modelled as being hidden, i.e. they actually do not appear as part of the navigation structure. The session object may represent data that is carried around through the session. A typical example is the shopping cart in e-commerce systems. It is deleted by means of garbage collection, i.e. when no more media object exists that links to it.

Another use of media types for complex scenes is the support of navigation context, for which the basic idea from contextual information bases has been adopted [18]. According to this a context is a set of object identifiers each having several names, and each of these names may be coupled with a reference to another context. There may be names for objects that are not referencing to other contexts. More formally, a context \( C \) is a finite set of triples \((o, n, r)\), where \( o \) is an object identifier, i.e. a value of some base type \( ID \), \( n \) is a name, i.e. a value of type \( STRING \), and \( r \) is either a reference \( \rightarrow C' \) to a context \( C' \) or \( nil \), the latter one indicating that there is no such reference. We write \( C = \{ n_1 : o_1 \rightarrow C_1, \ldots, n_f : o_f \rightarrow C_f \} \). If there is no reference for the \( i \)'th name, i.e. we have \((o_i, n_i, nil)\) we simply omit \( \rightarrow C_i \) and write \( n_i : o_i \).

The idea of working with contextual information bases is that a user queries them and thus retrieves objects. In order to describe these objects in more details s/he accesses the context(s) of the object, which will lead to other objects by
following the references. In addition, a particular information encoded by the name is associated with each of these references.

Bringing together media types and contextual information bases, the obvious questions are: What are the object identifiers that are required in contextual information bases, if we are given media types? What are the references that are required in contextual information bases? Is it sufficient to have names for describing objects in a context or should these be replaced by something else?

The natural answer to the first question for generalising the notion of object identifiers in contexts is to choose the unique URLs of the media objects. Supporting navigation context requires access to path information, so we may want to reference back to the various media objects that we have encountered so far. These media objects are placed in several contexts, one of which is the right one corresponding to our path. However, we may also have different references, which lead to different contexts. So, the contexts we asked for in the second question are just the contexts for the media objects, which themselves are represented by media objects for paths, i.e. again non-elementary scenes.

As to the third question, we definitely want to have more information than just a name. Fortunately, the concept of media type is already based on the assumption of an underlying type system. Thus, we simply have to replace the names by values of any type permitted by the type system. Having defined such extended contexts, we can exploit query macros to traverse back a path in the story board and to explore alternative access paths, respectively.

However, one important aspect of media types is the use of classification abstraction. Conceptually, we do not define a set of media objects, but we generate them via queries defined on some underlying database schema. Therefore, we also need a conceptual abstraction for contexts. In order to obtain this conceptual abstraction, we assume another base type $Context$, the values of which are context names. Instead of this, we could take the type $URL$, but in order to avoid confusion we use a new type.

Then a context consists of a name $C$, i.e. a value of type content, a type $t_C$ and a defining query $q_C$, which must be defined on the media schema, i.e. the set of media types, such that

$$\{(id : URL, value : t_C, reference : Context), q_C)\}$$

defines a view. Thus, executing the query $q_C$ will result in a set of triples $(u, v, r)$, where $u$ is the URL of a media type, $v$ is a value of type $t_C$, and $r$ is the name of a context. If this context is undefined, this is interpreted as no reference for this object in this context.

5 Collaboration

In [1] we started an investigation of collaboration frameworks for distributed WISs addressing the problem that we may have tasks that are to be solved by collaboration of several users, e.g. using group work in e-learning systems.
Collaboration is understood as combining cooperation, communication and coordination. The ground idea for capturing collaboration in WISs therefore is to define another extension of media types by means of exchange frames, which combine an exchange architecture addressing communication and cooperation, and a coordination specification, which consists of supporting programs, cooperation style coordination facilities, roles of collaborators, their responsibilities and rights, and the protocols they rely on.

Exploiting again the idea of media types supporting complex scenes we can abstract from the actor representing an individual user to actors representing a group of users. The exchange architecture for cooperation and communication within a group can be modelled by means of a workspace, which again is nothing but a media object associated with the group action, and consequently media object used by individuals in the group contain hidden links to the group object.

6 Reasoning about Media Types

Following our previous work in [12] we can formalise the operations associated on media types using the following language of abstract programs:

- 1 and 0 are abstract programs meaning skip and fail, respectively.
- An assignment $x := \text{exp}$ with a variable $x$ and an expression of the same type as $x$ is an abstract program. The possible expressions are defined by the type system. In addition, we permit expressions $\{P\}$ with a logic program $P$, assuming that $P$ contains a variable $\text{ans}$. The expression $\{P\}$ is interpreted as the result of the logic program bound to $\text{ans}$.
- If $p$, $p_1$ and $p_2$ are abstract programs, the same holds for the iteration $p^*$, the choice $p_1 + p_2$ and the sequence $p_1 \cdot p_2 = p_1p_2$.
- If $p$ is an abstract program and $\varphi$ is a condition, then the guarded program $\varphi p$ and the postguarded program $p\varphi$ are also abstract programs.
- If $x$ is a variable and $p$ is an abstract program, then the selection $@x \cdot p$ is also an abstract program.

On these grounds we introduce a higher-order dynamic logic, where the order comes from the intrinsic use of the set constructor and the logic programs in queries. In fact, instead of using logic programs with a semantics defined by fixed-points, we could use directly higher-order logic enriched with a fixed-point operator. As a consequence, we may consider a logic program $P$ as a representative of a higher-order logical formula, say $\varphi_P$. If $\{P\}$ is used as the right-hand side of an assignment, then it will correspond to a term $\text{Ians.}\varphi_P$ denoting the unique $\text{ans}$ satisfying formula $\varphi_P$. That is, all conditions turn out to be formulæ of a logic $\mathcal{L}$, which happens to be a higher-order logic with a fixed-point operator. From the point of view of expressiveness the fixed-point operator is already subsumed by the order, but for convenience we do not emphasise this aspect here. Furthermore, by adding terms of the form $\text{I}_x.\varphi$ with a formula $\varphi$ and a variable $x$ all assignments in operations are just “normal” assignments, where the left-hand side is a variable and the right-hand side is a term of $\mathcal{L}$.
We now extend $\mathcal{L}$ to a dynamic logic by adding formulae of the form $[p]\varphi$ with an abstract program $p$ and a formula $\varphi$ of $\mathcal{L}$. Informally, $[p]\varphi$ means that after the successful execution of $p$ the formula $\varphi$ necessarily holds. In addition, we use the shortcut $\langle p \rangle \varphi \equiv \neg [p] \neg \varphi$, so $\langle p \rangle \varphi$ means that after the successful execution of $p$ it is possible that the formula $\varphi$ holds. Using our recursive definition of abstract programs the following rules apply to $[p] \psi$ for a complex abstract program $p$:

\[
\begin{align*}
[1] \psi & \equiv \psi \\
[0] \psi & \equiv 0 \\
[x := t] \psi & \equiv \psi\{x/t\} \text{ (substitute all free occurrences of } x \text{ in } \psi \text{ by } t) \\
[p_1 p_2] \psi & \equiv [p_1][p_2] \psi \\
[p_1 + p_2] \psi & \equiv [p_1] \psi \land [p_2] \psi \\
[p^*] \psi & \equiv \text{ the weakest solution } \varphi \iff \psi \land [p] \varphi \\
[\varphi p] \psi & \equiv \varphi \rightarrow [p] \psi \\
[p \varphi] \psi & \equiv [p](\varphi \rightarrow \psi) \\
[\forall x \cdot p] \psi & \equiv \forall x. [p] \psi
\end{align*}
\]

As outlined in [12] we can formulate proof obligations for the operations that result from the specification of the story space, in particular for consistency with respect to static and dynamic integrity constraints on the underlying database schema. Furthermore, we can extend WIS personalisation in the light of dynamic logic.

7 Conclusion

In this survey we outlined in compact form the potential of the concept of media type emphasizing its use for interface abstraction supporting adaptivity to users, channels and devices, collaboration among user groups, session and context support, and serving as the conceptual bridge between identified requirements, the usage model of the WIS, the layout and playout, and the implementation on the basis of XML clusters and scripts. The versatility of media types are a major reason for the success of the codesign approach to WIS design as demonstrated in various large-scale applications. Furthermore, it is worth exploring the concept of media types in more depth, in particular with respect to the logical inferences as started in [12] on the basis of dynamic logic, the connection to deontic constraints on the storyboard and their refinement to the level of media types [13], and their embedding in layout and playout by means of screenography [10].

References


Towards Semantic Wikis: Modelling Intensions, Topics, and Origin in Content Management Systems

Gunar Fiedler, Bernhard Thalheim
Department of Computer Science, Christian-Albrechts University at Kiel
Olshausenstr. 40, 24098 Kiel, Germany
{fiedler, thalheim}@is.informatik.uni-kiel.de

Abstract. Content management is the process of handling information within an organization or community. Therefore, content management systems have to provide generic functionality for generation, extraction, storage, and exchange of digital assets. Because of the heterogeneity and complexity of content, a sufficient semantical and user-oriented annotation of content is crucial. Although semantical annotation by metadata and ontologies together with reasoning support has been extensively studied for a long time, commercially available content management systems provide only basic support for semantic modelling. Conceptual aspects of content users and support of user specific intensions are neglected. In this paper we will analyze the mismatch between the requirements of content management and semantical description and propose a data model for content which treats semantic information not only as describing metadata but incorporates the data itself, the intension behind the data, the usage of data and the origin of data on the same level.

1 Introduction

Content Management

Content in its actual definition is any kind of information that is shared within a community or organization. In difference to data in classical database systems content usually refers to aggregated macro data which is complex structured. Structuring of content can be distinguished:

- The structure of the aggregated micro data is preserved but micro data was combined to build larger chunks of information. Examples are scientific data sets such as time series of certain measurements. There is a common (or even individual) structuring and meaning for each sampling vector but the compound of all sampling vectors adds additional semantics.
- The structure of content is only partially known. A typical example is the content of Web pages: structuring is known up to a certain level of detail which may also be varying within one instance.
- Content may be subsymbolic, such as pictures, videos, music or other multimedia content.
Aggregation of content usually takes place by combining reusable fragments provided by different sources in different formats such as texts, pictures, video streams or structured data from databases. Content is subject to a content life cycle which implies a persistent change process to the content available in a content management system (CMS).

Currently, many systems claim to be content management systems. A recent overview of the German market (www.contentmanager.de, viewed June 12th, 2007) reveals hundreds of products related to tasks of content management. Most products are related to Web content management. These products organize content for Web pages with a strong orientation on editorial components such as texts and pictures.

The more generic ones agree in a major paradigm: the separation of data management and presentation management. Data management reflects the process of supporting content creation, content structuring, content versioning, and content distribution while presentation management grabs the data for delivering it to the user in various ways. Only content which is generated following this separation can be easily shared, distributed, and reused.

Following new trends and developments in Web technologies, e.g., in the context of Web 2.0 or the Semantic Web the automated processing of content becomes more and more important. Because content represents valuable assets it may be reused in different contexts (content syndication) or has to remain accessible for a long time.

The semistructured or even unstructured nature of content requires annotations to enable search facilities for content. Expressing semantics in a machine interpretable way has been under investigation since the early days of artificial intelligence, see e.g., [12] for a survey of knowledge representation techniques such as logical theories, rule-based systems, frames or semantic nets. Today systems handle semantical descriptions as metadata describing certain content instances. There are different ways for associating data and metadata:

- A conceptual, logical, or physical schema is defined and instances are created according to this schema. This is the usual way for classical databases. The modelling language strongly restricts the capabilities of this description facility. Common languages such as Entity-Relationship Modelling or UML focus on structural properties with support of selected integrity constraints.

- Defining a schema is not applicable (or only in a restricted way) to semistructured or unstructured content. For that reason content instances are annotated. An annotation is a triple \((S, P, O)\) where \(S\) denotes the subject to be annotated, \(P\) a predicate denoting the role or purpose of this annotation, and \(O\) the object (or resource) which is associated with \(S\). The vocabulary for annotations is organized in ontologies and thesauri. A typical language for expressing annotations in the context of the Semantic Web is the Resource Description Framework (RDF, [20]) while the Web Ontology Language OWL ([19]) may be used to express semantic relationships between the concepts and resources used for annotation. There exist myriads of ontologies and parameter definitions for different application domains such as the Dublin Core parameters [3]) for editorial content.

Content Management and Semantic Annotations

Semantic annotation in current content management systems is usually restricted to preselected ontologies and parameter sets. Rich conceptual data models are only available in more sophisticated systems. Because most generic CMS are focused on Web content management
semantic annotation is usually restricted to editorial parameters. Specialized content management systems which are adapted to certain application domains incorporate preselected and tailored ontologies. Especially for XML-based content there exist several annotation platforms which incorporate semantical annotation either manually or semi-automatically; see [11] for a survey on available platforms.

Automated processing of semantical metadata is usually restricted to search facilities, e.g., searching for the author of an article. Because ontologies are preselected for most systems a full-featured reasoning support is usually not available. Especially for OWL ontologies there are reasoning tools based on description logics such as Racer ([7]) or FaCT which enable T-box (but also A-box) reasoning about semantic relationships between annotation concepts.

Applying generic semantical annotation and classical reasoning facilities to content management suffers from several drawbacks:

- Content as aggregated macro data is only partially analysable. The purpose of metadata is the description of properties which cannot be concluded from the data itself. The very simple annotation frame of \((S, P, O)\) triples does not allow one to express complex properties. For that reason this information has to be kept in the underlying ontology by defining appropriate concepts. The support of user-specific concepts increases the size of the ontology significantly and makes reasoning support even harder. Ad hoc definitions of user-specific concepts is not supported in this annotation model.

- Annotation with respect to arbitrary ontologies implies general purpose reasoning support by the system. Reasoning for even simple languages suffers from its high computational complexity (e.g., NEXPTIME for the restricted OWL-DL dialect, [9].) Dealing with high worst-case complexities implies a small size of input data but this is a contradiction to expressible ontologies and the definition of content as complex structured macro data. Especially the size of content instances is a crucial factor because A-box reasoning is a critical point for automated content processing ([8].)

But there are advantages, too:

- Usually, it is possible to distinguish between different points of view on content instances. Not every property is important while looking from every point of view. The macro data may encapsulate and hide properties from its aggregated micro data. Reasoning about the properties of the compound can be separated from the properties of the elements as well as the properties of interconnections between content instances.

- Typical application scenarios determine important properties and suggest evaluation strategies. So ontologies may be decomposed to enable a contextualized reasoning, e.g., on the basis of Local Model Semantics ([6]). Local reasoning may rely on a language that is just as expressive as needed in this context. Contexts relying on less expressive languages may support automated reasoning while contexts relying on more expressive languages may be used for manually interpreted information. Soundness and completeness of the reasoning process are not of primary interest as long as the reasoning result is acceptable in the application domain.

- The separation between annotations relying on common knowledge, user-specific annotations and (especially) usage-specific annotations reduces the size of incorporated ontologies significantly.

- If semantic annotations themselves are given a more sophisticated internal structure reasoning can be adapted to the requirements of the application domain.
The major disadvantage of current semantic description in content management is the treatment of knowledge over content instances as metadata on a secondary level in a strongly restricted language. In the following sections we will introduce a data model for content which handles the semantic part on the same level as the content itself and gives additional structure to the semantic description. We will start with the definition of content chunks as semantically enriched content instances in Section 2. In Section 4 we will introduce the notion of a schema for content chunks to incorporate typical functionality of content management systems such as content generation, content delivery, or content exchange.

As an example for a content management system we will take a look at a (simplified) Web shop application which sells products to customers via a website. The usual functionality should be supported: customers are able to search and browse for products, manage their profiles, shopping carts, and wish lists and order products.

2 Content Chunks

If we consider the HERM ([14]) schema fragment in Figure 1 we see a modelled list of products.

This modelling reveals certain structural properties such as attributes of the entity types and the connection between the list and the products. But the purpose of this model is missing. What kind of list was modelled: a shopping cart, a wish list, a list of stock items? Why was it modelled? What does the modeler expect? All this information is missing in the modelled fragment but is crucial if content instances of this schema are processed: if the list represents a shopping cart, pricing information may be collected. If it represents a wish list, there may be the need for additional functionality for discovering related products. It is obvious that expressing all this information by \((S, P, O)\) annotations will increase greatly complexity of each content instance and prevents a sophisticated semantic handling.

Modelling the semantics behind the data needs as much attention as modelling the structural properties of content. For that reason we propose a content data model which integrates structure, intension, usage, and origin on the same level. We start with the definition of content instances in this model.

**Definition 1.** A content chunk is a tuple \(\mathcal{C} = (D, I, T, O)\) where \(D\) is the structural representation of a content instance, \(I\) an intensional description, \(T\) a topic description expressing the usage of content, and \(O\) the specification of the context where the content instance is used. The state of a content system is a finite set of content chunks.

Figure 2 visualizes the four dimensions of content chunks. Each content chunk expresses the association identifying ‘what (structure) is used by whom (origin, context) in which way (topic) under which assumptions and thoughts (intension)’. In the following paragraphs we will refine these notions.
The Structure Dimension

The structural part of a content chunk reflects the classical notion of a content instance. Depending on the nature of the content data may be represented using an instance of a database schema formulated in ERM or UML, a semistructured resource such as a XML document, or a subsymbolic resource such as a picture.

Definition 2. Let $\mathcal{L}$ be a set of (supported) modelling languages, $S_L$ the set of all schemata expressible with a certain modelling language $L \in \mathcal{L}$ and $\Sigma_S$ the set of all possible states of a certain schema $S$. The structural component $D$ of a content chunk $\mathcal{C}$ is a triple $(L, S, I)$ denoting a modelling language $L \in \mathcal{L}$, a schema $S \in S_L$, and an instance $I \in \Sigma_S$.

In our example, $(\text{'HERM'}, s, i)$ is the structural part of a content chunk if $s$ denotes the schema in Figure 1 and $i$ an instance which associates e.g., the entity type $\text{List}$ with the entity set $\{\{\text{No} : 1\}\}$, the entity type $\text{Product}$ with the set $\{\{\text{No} : 134, \text{Descr} : \text{Book}, \text{Price} : 16.99\}, \{\text{No} : 521, \text{Descr} : \text{CD}, \text{Price} : 9.95\}\}$, and the relationship type $\text{consistsOf}$ with the relationship set $\{\{\text{List.No} : 1, \text{Product.No} : 134, pos : 1\}, \{\text{List.No} : 1, \text{Product.No} : 521, pos : 2\}\}$.

The structure dimension of content chunks is based on the theory of media types [1]. Media types [2] combine views and their functions into one type. Media types may be linked to other media types. For instance, we may distinguish input data for the workflow, retrieval data for the workflow, output data of the workflow, display data suites for each stage of the workflows, and escorting data supporting the understanding of each stage of the workflow.

Media objects may be structured, semi-structured, or unstructured by the media types. They are data that are generated from underlying databases, ordered, hierarchically representable, tailorable to various needs and enhanced by functionality for its usage. Since users have very different needs in data depending on their work history, their portfolio, their profile and their environment media types are packed into containers. Containers provide the full functionality necessary for the application and use a special delivery and extraction facility. The media type suite is managed by a system consisting of three components:

Media object extraction system: Media objects are extracted and purged from database, information or knowledge base systems and summarized and compiled into media objects. Media objects have a structuring and a functionality which allows to use these in a variety of ways depending on the current task.

Media object storage and retrieval system: Media objects can be generated on the fly whenever we need the content or can be stored in the storage and retrieval subsystem. Since their generation is usually complex and a variety of versions must be kept, we store these media objects in the subsystem.
**Media object delivery system:** Media objects are used in a large variety of tasks, by a large variety of users in various social and organizational contexts and further in various environments. We use a media object delivery system for delivering data to the user in form the user has requested. **Containers** contain and manage the set of media object that are delivered to one user. The user receives the user-adapted container and may use this container as the desktop database.

This understanding closely follows the data warehouse paradigm. It is also based on the classical model-view-control paradigm. We generalize this paradigm to media objects, which may be viewed in a large variety of ways and which can be generated and controlled by generators.

**The Topic Dimension**

The topic part of a content chunk is the conceptual counterpart to the presentation facilities of content management systems. Available systems offer template mechanisms (e.g., based on XSLT or scripting languages such as PHP or JSP) which transform a content instance to a physical representation ready for delivery through an output channel, e.g., HTML Web pages, e-mails, or PDF documents. Instead of coding presentation on the level of rendering templates a more abstract approach should be used. Topic maps ([10, 17]) provide the general data structure for a user-dependent view on content on the conceptual level. Expressing content via topic maps fulfills the following tasks during content delivery:

- The data structure is transformed to local vocabulary, e.g., according to a corporate identity or internationalization. In our example attribute names may be changed to language dependent labels. The prices of our products may be converted to local currencies or may be recalculated according to different tax regulations.

- The content is embedded into the usage context. The onion approach for a stepwise generation of delivered content ([15, ?]) defines different kinds of embeddings depending on the profile of a user (characterization of the properties of the user such as language, skill level, or preferences) and portfolio (tasks which have to be, should be, and can be fulfilled by the user, see [5].) This information is obtained from the specifications of workflows and storyboards for interaction and added to the topic map as supplementary content. There are different kinds of supplementary content:
  - **static content**, e.g., the logo of the company or statically linked elements such as advertisement banners,
  - **decorative content** which is dependent on the current usage context but has no meaning to the application such as contextual help or integrated services such as a contextual weather forecast,
  - **additionally delivered content** such as information about manufactures or links to related products in our Web shop example, and
  - **navigational events** such as navigational links allow the user to interact with the system.

- Multiple topic maps may be merged for multi-modal applications.
Supplementary content is incorporated in the topic map by parameterized queries on the set of content chunks and user specifications which characterize the occurrences of topics defined in the topic map. These queries are evaluated during content delivery and produce the topic map which can finally be rendered.

The topic part of a content chunk in our example may be the topic map depicted in Figure 3. This topic map reflects our product list in the context which supplies additional information on these products. This topic map can be transformed to a physical representation (e.g., a HTML page) using the usual techniques mentioned above.

![Figure 3: Topic Map for the ‘Product List’ Content Chunk](image)

Topic parts of a content chunk thus serve for several purposes: to represent ideas or understandings; to represent a complex structure of content chunks and their related chunks; to communicate complexes of content chunks; to aid understanding by explicitly integrating new and old content chunks; and to assess understanding or diagnose misunderstanding.

The Origin Dimension

To express content syndication information about the origin of content has to be stored. The provenance of data was already studied on the instance level ([2, 22, 1]) especially for scientific data sets. We can adapt these results for our purposes. We choose a finite set $\mathcal{C}$ from a universe $\mathcal{U}$ of contexts. Each context in $\mathcal{C}$ represents a point of view on the application area under consideration. These points of view may be different points of view of the same user or may belong to different users. Because all these contexts are views on the same universe of discourse they are related: data, intensions, and topics may be exchanged between contexts. Actions in one context may affect other contexts.

**Definition 3.** Let $\mathcal{U}$ be a universe of contexts and let $\mathcal{C} \subset \mathcal{U}$ be a finite set of contexts. Further, let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be a set of content chunks. The origin part of a content chunk $\mathcal{C}$ is a tuple $(c, A)$ with a context $c \in \mathcal{C}$ where the content chunk $\mathcal{C}$ resides and a set $A$ of content chunks which are considered to be the ancestors of this chunk. The graph implied by this ancestor relationship between content chunks has to be acyclic.
Connections between content chunks enable the exchange and transformation of data, intensions, and topics between different contexts. In our example we may define a content chunk representing our product list together with a topic map for rendering a shopping cart. By adapting the topic map as well as the intension we may construct a content chunk which renders an order confirmation.

The origin dimension also reflects the purpose of the content chunk. Content chunks have a function such as to give an instruction, to control behaviour, to transmit ideas in a memorable form, to enhance social cohesion, or to give users a better understanding. For example, a content chunk that reflect a piece of knowledge may start with a mystery that leads to a conflict situation, may continue with an explanation of the solution or of the discovery as the turning point and may conclude with a resolution of the conflict situation. Context injection must thus be an integral element for content chunk processing.

Our origin model for content chunks extends the usage thesis of [?] that mainly reflect the communication act between a sender and receiver with their intentions, backgrounds, cultures and relationships. Usage context should also consider excluded receivers, value of content chunks to receivers, groups or societies. Content chunks are thus generically [?] enhanced by context, refined by intended specifics and instantiated by their specific usage.

The Intension Dimension

The intention dimension is based on concepts. They are the building blocks in human thinking and reasoning, and as such highly flexible. They can be general or specific, concrete or abstract, natural or technological, artistic or scientific. They can apply to things that are real or imaginary. They provide a help for distinguishing between things we observe in the world, or ideas such as truth and falsity, appearance and reality and continuity and discontinuity. Abstract concepts are useful for characterisation of observations, thoughts and expressions. Typical abstract concepts are truth and falsity, sameness and difference, wholes and parts, subjectivity and objectivity, appearance and reality, continuity and discontinuity, sense and reference, meaningful and meaningless and problem and solution. They govern different kinds of human thinking at a fundamental level.

Concepts are vital to the efficient functioning of semantic Wikis. They are organised bundles of stored knowledge which represent an articulation of events, entities, situations, and so on experience. Concepts are necessary for an understanding, for the organisation, for sharing Wikis and for communication. We may assume a simple association between the components of Wikis and concept. The associations may form a complex multi-dimensional network. They may be of specific types such as kind-of, is-part-of, is-used-for and of variable strength. Associations typically correspond to concepts of a more schematic kind than the concepts which they serve to connect.

The classical approach to concepts is based on description of necessary and sufficient criteria for content-concept association. We notice however that most concepts characterising content chunks cannot be captured by means of a set of necessary and sufficient features. Many natural concepts are fuzzy and contextually flexible. Therefore we need to extend the approaches typically assumed for formal semantics to natural semantics. Additionally, the association of content to concepts must not be strict. Some content may be a better example to a concept than other content.

The prototype approach for concept-content association is also limited. Ratings or se-
lections of prototypes are strongly context dependent, e.g., culture dependent and actor dependent. Prototypes are given with certain preference, frequency, sample extraction, learning background, level of verifiability, and under time pressure. The degree of association may vary over time, may be dependent on the concrete usage, and bound by the representation language chosen. Prototype content may also be more or less specific or general for concepts.

Concepts are typically expressed through propositions. The meaning has typically two parts: an element of assertion and something that is asserted. What is asserted is called proposition. The simplest type of proposition consists of an argument and a predicate. Semantical units or propositions are interrelated by entailment. Entailment is different from material implication and relates propositions by forward propagation of truth and backward propagation of falsity. Propositions can be contraries, contradictories, or independent. They may belong to a category or genre of expression, are given in a certain style or manner, are often based on stereotypical norms of expression, depend on ideas and values that are employed to justify, support or guide the expression, reflect aspects of culture or social order, are shaped according to the community that uses them, and are configured by theories or paradigms.

We also may distinguish between the existential approach to meaning based on a correlation of expressions in a language with aspects in the world. The intentional approach associates some kind of representation with concepts as the main constituents of the sense and depends on the cultural context. Whenever content is difficult to interpret then we need to consider concepts, deep structures, unconscious foundations, hidden symbols, annotations or underlying pattern supporting it. If content seems to transparent then we do not need to look for these things. It is often surprising how much background information is necessary for understanding content even such content that appear on the surface to be wholly transparent. There are various connotations and denotations that content may have. We have to consider the arrangements and laws for constructing content phenomena (langue) as well as the various instances that are constructed by constructors and laws (parole). Content can be coded in various ways, e.g. based on different representation such as text or multimedia elements. Content can be differently categorized and organised. We may use conventions that draw on common forms of knowledge. Furthermore, we need to devise different ways for understanding and for association of concepts to content.

The intension of a content chunk expresses the purpose of the content as well as meanings and thoughts about the content chunk. Thoughts about some object may be expressed using a general description frame. A sophisticated and generic frame was given by Zachman in the context of specifications in software engineering ([23, 13]): each thought is expressed by formulating the facets who, what, when, where, how, and why. Each facet is specified by a concept. A concept itself is a small logical theory. We base our notion of concepts on intensional logics, especially on a restricted version of Transparent Intensional Logic (TIL) introduced by Tichý ([4]). TIL introduces the notions of intensional functions which map modalities (time, place, object identity, possibility, etc.) to values and intensional constructions building the intension of more complex expressions out of its components.

In our example we may introduce the concepts of customers, products, shopping carts, wish lists, product orders, etc. The concept shopping cart implies an intension of what a shopping cart is: it is a list of products selected from the offers in our Web shop. These products may be purchased in the future.

TIL analyses the intension of a concept down to the objectual base (calculating valuations of the intension behind the sentence ‘Products are associated with a description and a price’ considers all possible valuations of product, description, price, associated with and even and


in ‘one shot’). This is not the natural way of thinking. We modify the TIL approach in the following way:

- We introduce different types of individuals in the objectual base. TIL defines a single class \( \iota \) of individuals. Introducing multiple (disjunct) classes \( \iota_1, \ldots, \iota_n \) together with operations and predicates (such as ordering relations) corresponds to the definition of data types for attributes in classical database modelling. As defined in TIL there is at least one class \( o \) of truth values \( (true, false) \) with the usual operations and predicates. The intension behind these operations and predicates is no longer transparent in terms of TIL.

- We support different types of modalities. TIL is specialized on modalities object identity and time and defines each intensional function on these modalities. Because there are other modalities (especially the possibility of a fact) and some intensions may be expressed in a more compact way if e.g., the time modality is omitted we will define intensional functions over arbitrary modalities from a given universe \( \Omega \) of modalities.

- The objectual base consists of all first order types defined in TIL:
  - \( \iota, o, \) and \( \omega \) are first order types,
  - each partial function \( \alpha_1 \times \alpha_k \rightarrow \beta \) with first order types \( \alpha_i \) and \( \beta \) is a first order type,
  - nothing else is a first order type.

**Definition 4.** An intensional function is a function \( f : \omega_1 \times \omega_k \rightarrow \alpha \) mapping possible worlds \( (w_1, \ldots, w_k) \) to instances of a first order type \( \alpha \). An intensional function is called non-trivial if there are two possible worlds \( (w_1, \ldots, w_k), (v_1, \ldots, v_k) \) with \( f(w_1, \ldots, w_k) \neq f(v_1, \ldots, v_k) \).

All first order types which are no intensional functions are called extensions.

Intensional functions can be used to express the usual type constructors: classes can be represented by their characteristic function, attributes by functions mapping to individuals, associations between objects by functions mapping to object identities.

In contrast to TIL we consider different kinds of intensional functions. The first kind is defined in a non-transparent way. Typical examples are extensional objects such as operations and predicates on the objectual base. Other non-transparent functions may be obtained from external data sources. For example, the concept of a customer may be represented as a characteristic function over modalities \( \omega \) (object identity) and \( \tau \) (time): \( isCustomer : \omega \times \tau \rightarrow o \). The valuation of this function may be determined by coupling the function with the customer database of our Web shop: \( isCustomer(w, t) = true \) if and only if the object with identifier \( w \) is registered as a customer in our customer database at time \( t \). Non-transparent intensional functions may be evaluated but do not reveal the internal structure of the valuation or their relationship to other intensional functions.

The second kind of intensional function is built in a transparent way: an intensional construction is used to relate valuations of the function with valuations of other first order types. Tichý introduced four types of constructions: variables of type \( \alpha \), trivialization (using objects in constructions), composition (application of values to a function) and closure (creation of functions.)

**Definition 5.** We consider a single context \( c \in C \). We organize intensional functions on strata in the following way:
• Operations and predicates on the objectual base (such as boolean connectives) as well as all non-transparent intensional functions and all intensional functions imported from contexts other than \( c \) are considered to be functions on stratum 0.

• Let \( k \) be an intensional construction with free variables \( x_i \) and a total mapping \( p : \mathcal{X} \rightarrow \mathcal{F} \) from variables \( \mathcal{X} = \{x_1, \ldots, x_n\} \) to intensional functions \( \mathcal{F} = \{f_1, \ldots f_m\} \) where the stratum of \( f_j \) is at most \( s - 1 \). The intensional function constructed by \( k \) is considered to be a function on stratum \( s \).

The layering of intensional functions implies the independence of intensional functions on lower strata from intensional functions on higher strata and especially from their usage in constructions. This enables the determination of valuations of intensional functions on higher strata by first fixing the valuations of intensional functions on lower strata. This restricts expressiveness with respect to TIL. The strict monoton layering may be relaxed to constructions out of functions from the same stratum. Functions can be lifted to higher strata by using identity constructions, so we will allow the direct assignment of functions to strata higher than given by the definition.

Intensional constructions represent the terminological knowledge in traditional ontologies. Constructors such as ‘is-a’, ‘part-of’, or ‘union-of’ represent a fixed, preselected, and not configurable set of intensional constructions.

Building intensions by intensional constructions does not associate valuations of this intensional function with concrete objects. Beside intensional (T-box) reasoning based on constructions, properties of valuations of intensional functions have to be revealed.

**Definition 6.** Let \( c \) be a context, \( \mathcal{F} = \{f_1, \ldots, f_n\} \) a set of intensional functions, \( L \) a logical language and \( T \) a theory with sentences from \( L \) formulating the knowledge about the valuations of \( \mathcal{F} \) with respect to the layering of intensional functions in \( c \). The tuple \( \mathfrak{B} = (\mathcal{F}, T) \) is called a concept in context \( c \).

In our Web shop example we might consider intensional functions \( \text{isCustomer} : \omega \times \tau \rightarrow o \), defined in a non-transparent way as mentioned above. Assuming that a customer will remain to be a customer for all the time we can express this in our small theory about customers:

\[
\text{isCustomer}(w, t) \implies (\forall t' > t)(\text{isCustomer}(w, t'))
\]

In another example shopping carts (\( \text{isShoppingCart} : \omega \times \tau \rightarrow o \)) might become an order list (\( \text{isOrderList} : \omega \times \tau \times \eta \rightarrow o \) for possibilities \( \eta \)):

\[
\text{isShoppingCart}(w, t) \implies (\exists n' \in \eta, t' \in \tau)(\text{isOrderList}(w, t', n'))
\]

With the definition of concepts we can finally construct content intensions:

**Definition 7.** Let \( \mathcal{S} \) be a set of facets (e.g., according to the Zachman framework). A content intension is a set of functions \( i : \mathcal{S} \rightarrow \mathfrak{B} \) mapping facets from \( \mathcal{S} \) to concepts from \( \mathfrak{B} = \{\mathfrak{B}_1, \ldots, \mathfrak{B}_n\} \) in the current context.
3 Query Facilities for Content Chunks

The definition of content chunks as combinations of data, intension, topic, and origin enables several kinds of query facilities in content management systems. In the rest of the paper we use $D$ for the set of all structure definitions in the state of the CMS, $I$ for the set of all defined intensions, $T$ the set of all defined topic maps, and $C$ for the set of all contexts. Typical examples for query functions are:

- Structural queries remain unchanged. Depending on the modelling language(s) the usual query facilities are present, e.g., return all products from a certain product list.
- The function $\text{explain} : D \rightarrow 2^I$ returns all intensions associated with a certain data instance.
- $\text{sample} : I \rightarrow 2^D$ returns all data instances associated with a certain intension.
- $\text{express} : D \times I \times C \rightarrow 2^T$ returns all topic maps associated with the given data object under the given intension in the given context.
- $\text{understand} : T \times C \rightarrow 2^{I \times D}$ returns data instances together with an intension for the given topic map and context.
- $\text{find} : C \rightarrow 2^C$ returns the contexts which incorporated content from this context.
- T-box reasoning in a generalized fashion is available by evaluating the intensional constructions. There is additional reasoning support, as depicted in Figure 4.

Properties of a concept relevant within a context are expressed in small local theories. We do not assume that this theory is a complete description of the concept but reveals relevant aspects. Concepts may be imported by other contexts while possibly different properties may become important. This is expressed by associating a different theory to the corresponding intensional function. For example, in a certain context it may be important to have a conception about the time when a person became a customer.
additional intensional function $customerRegistrationDate: \omega \rightarrow \tau$ may be introduced on a stratum lower than $isCustomer$ while the local theory of the concept $customer$ is enhanced by the constraint

$$(\forall w \in \omega)(\forall t < customerRegistrationDate(w))(isCustomer(w, t) = false)$$

Evaluation of properties follows this construction strategy:

- First, the theory locally defined within the current context is used to prove the desired property.
- If the local proof was not successful, the intensional construction is investigated and reasoning is delegated to original contexts where the concept was imported from.

It is obvious that reasoning in this fashion does not ensure decidability but enables the delivery of precalculated relevant aspects which may not be accessible by pure intensional reasoning.

4 Content Schemata

In Section 2 we defined the building blocks of content as arbitrary tuples $(D, I, T, O)$. Considering typical application scenarios of content management systems arbitrary associations can be restricted to support additional content management functionality:

- There are relationships between intensions and structural properties. Reasoning about intensional properties is reflected by certain values of the associated data instances. For example, reasoning about prices should be reflected by appropriate attributes in the structural definition. Non-transparently defined intensional functions must be directly computed from data.
- Information expressed in a topic map should be related to the underlying data and vice versa.
- Information can only be expressed or understood if there is an appropriate intension. On the other side, every intension should be expressible.
- Content which is imported from different contexts may not be completely revised but transformed.
- Not every intensional construction should be allowed. To restrict complexity a configurable set of construction templates has to be defined which incorporates the conceptual theories from the sources to build theories in the target context.

Restrictions may be expressed by constraint relations between the four dimensions of content chunks. To support content management functionality a mapping approach is better. There are three general tasks which have to be fulfilled during content management: content is created, selected content is delivered to the user, and content is exchanged between different contexts. Content creation in terms of our data model is the mapping of a topic map within a context to combinations of an intension and a data instance. Content delivery is the mapping between a data instance and an intension within a context to a topic map. Content translation maps content chunks from one context to another.
**Definition 8.** A content schema is a tuple \((generate, deliver, exchange)\) with a function \(generate : T \times C \to 2^{D \times I}\), a function \(deliver : I \times D \times C \to T\), and a function \(exchange : D \times T \times C \times C \to D \times I \times T \times C\).

These functions are defined for each context separately. First, a set of base intensions is defined. These base intensions rely on concepts (such as customer or shopping cart) which may be defined transparently or non-transparently. These base intensions are associated with a data schema \((L, S)\) (where \(L\) is a modelling language and \(S\) is a schema expressed in this language), a topic map template incorporating the data by associated data queries and a data template defining the data instance by queries on the topic map.

**Definition 9.** Let \(\{k_1, ..., k_n\}\) be a set of intensional constructions. An intensional construction template is a tuple \((\{k_1, ..., k_n\}, p, m)\) with intensional constructions \(k_i\) for each facet in the intension specification frame, a parameter assignment specification \(p : X \to B\) mapping variables from \(B = \{x_1, ..., x_k\}\) to concepts from \(B = \{B_1, ..., B_l\}\) restricting valuations for variable substitutions in \(\{k_i\}\) to the given concepts, and a merging function \(m\) which creates

- the logical theory \(T\) out of the theories associated to \(X\),
- the data schema out of the data schemata of \(X\),
- the topic map template out of the topic map templates of \(X\), and
- the data template out of the data templates of \(X\).

The definition of the data schema, the topic map template, and the data template implies the content generation and content delivery functions. The creation of the logical theory out of other concepts is given by a compatibility relation between models of these theories as defined by the Local Model Semantics framework ([6]).

5 Semantic Wikis: Enabling Collaborative Content Annotation and Foundation

Communities form an interacting group of various actors in a common location, common intension, and common time. They are based on shared experience, interest, or conviction, and voluntary interaction among members with the intension of members welfare and collective welfare. They can have more or less structure and more or less committed members.

A wiki is a collaborative web site set up to allow user editing and adding of content by any user who has access to it. Wikis have changed access and habits of internet users. The right and wrong usage of wikipedia is already widely studied in literature, e.g. the journal *First Monday* provides a detailed study of Web 2.0 in issue 13, March 2008 and of wiki misuse in more than a three-score papers in various issues. The main functionality provided for wikis is

- management of content chunks with their structure, intention, origin and topic,
- annotation management for users depending on their role, rights and obligations,
- explanation, exploration and knowledge elicitation and gathering support for readers of content chunks,
- presentation of information in a variety of ways and for a variety of environments,
- user management depending on the roles, functions and rights a user may have on the content,
• security and safety management for integrity of content and information, and

• history and evolution management that allows to show the development of the wiki and to restore previous versions.

We concentrate the remaining part of the paper to the first three main functionalities of wiki systems. These three functionalities are backed by our approach to semantic wikis. Presentation may also include generic adaptation to the user environment and features for marking content [?]. Wiki systems should integrate features that have been developed for customer management. Wikis are generally designed with a functionality that makes it easy to correct mistakes. Since this functionality is a target for attacks on content and on the system, wiki systems are extended by security and quality management features. Thus, they provide a means to verify the validity of recent additions, changes, corrections, replacements etc. to the content. History and development information can be maintained through docket [?] and the diff feature that highlights changes between two revisions. Wiki systems are special web information systems. They support information seeking life cases [?, ?], are based on storyboards for creation and consumption of information [?] and require a sophisticated user profile and portfolio model [?].

Wiki systems share and encourage several features with generalized content management systems, which are used by enterprises and communities-of-practice. They are maintained, developed and enriched by communities of leisure, interest or practice. Community members are allowed to instantly change the content (usually Web pages.) There are some minimal requirements to the content chunk for wikis. The name or annotation of a content chunk is typically embedded in the hyperlink and interlinked with other chunks. Content chunks can be partially created or edited at anytime by anyone (with certain limitations for protected articles). They are editable through the web browser. Their evolution history is accessible through a history/versioning vie, which also supports version differencing (“diff”), retrieving prior versions and summary of most recent additions/modifications. Additionally, easy revert of changes is possible. We can extend this conception of Wikis and look forward on how functionality of Wikis may evolve by incorporating topically annotated and intensionally founded content.

Semantic wikis¹ enhance content that is displayed in the web with fully considered and perfected concepts or verified knowledge and with user annotation or topics. It thus formalises the notion of wikis enhanced by ontologies [?], clarifies the knowledge basis and provides a basis for using data from the web in a form that corresponds to the user demands, their information portfolio and their personal profile.

Using Communities for Content Generation

Content creation and content annotation are resource-intensive processes. Introducing a user- and usage-centric approach to content handling as presented in this paper, these processes can be distributed through a social network, adapting the notions of the Web 2.0 initiative. One of the most severe requirement to wiki evolution is trustworthiness of the wiki. Everybody

¹Our concept of semantic wikis should not be mistaken as a concept of Web 3.0. Web 3.0 or semantic web aims in annotation of content on the basis of an ontology which is commonly accepted by a community. Proposals such as the Semantic MediaWiki add a fixed annotation to concepts similar to tagging in websites such as delicio.us.
who uses wiki systems such as wikipedia observes that the competence level, the education profile, the work portfolio, the biases, the intensions (e.g., trolling) and the psychographical level of wiki creators has a huge impact on the quality of a wiki. Wikis that can be created by almost everybody are typically of lower quality than those that can only be created by experts in the area. As an example we accessed the entry ‘Dresden’ in wipipedia.org. In the average, each second sentence was not completely correct.

Wiki systems are considered to be special content management systems which allow the user to instantly change the content (usually Web pages.) We can extend this notion to semantically enriched content:

• Content may be loaded into the system. This reflects the usual process of editing pages in a Wiki. The load process results in stored data instances in the CMS which can be extracted via search templates or associated with metadata in the usual sense (e.g., editorial parameters).

• Data instances may be associated with intensional descriptions such as copyrights and access rights.

• The user may annotate the content after searching for it, e.g., making recommendations on products in our Web shop example. A recommendation can be expressed by an additional intension on the content chunk expressing that the current user interprets the data as a product recommendation. The local theory expresses the fact, that this user has bought these products or might buy these products in the future.

• Another user may explore the notion of a ‘recommendation’ from the context of the first user if he sees the same data instance and looks for associated intensions. Afterwards, this user may use this concept to annotate other data instances.

• Users may refine the local theory of a concept to incorporate knowledge which was hidden so far.

• Users may associate new topic maps to content to create different (e.g., localized) versions.

Modelling Wiki Functionality Based on Content Chunks

Beside supporting content generation and annotation by social networking, semantically and user-specifically enriched content chunks are the base for modelling collaboration within a network of users. Collaboration is seen ([?], 21) as a process of interactive knowledge exchange by several people working together towards a common goal. Collaboration can be characterized ([15]) by three facets: communication, cooperation, and coordination. The communication facet defines the exchange protocols of content between users. The cooperation facet relies on the workflow of the collaboration by specifying who (actor) has to deliver which results to whom. The coordination facet defines the task management and synchronization between the collaborating partners to fulfill the cooperation goals.

Collaboration in social networks is usually defined in an implizit and decentralized way, so classical workflow management systems with fixed process definitions cannot be applied. The content data model defined in this paper can be used to annotate content with the specified collaboration frame to express
- the history of content and content changes,
- the purposes of content changes and content usage,
- future tasks on content chunks and therefore
- access rights on content chunks.

In the context of collaboration the specification of users becomes important. Conceptually, users are handled by a set of concepts \( \mathcal{A} \) called actors (such as administrator, moderator, registered user, inexperienced user, guest user, etc.) Actors define roles of users and therefore imply a grouping on the set of users. According to our definition of concepts each actor is associated with a small logical theory expressing the properties which are common to all users in the user group of the actor.

Communication between users takes place by topics. Topic map fragments have to be defined in the content schema to express tasks in the collaboration frame characterized above. Figure 5 shows an example for a topic map concerning a product within our Web shop. A typical task which can be done by users is to write a comment. The topic map for expressing the product incorporates only comments fulfilling the intension of a \textit{proofread comment}. To write a comment the topic map is merged with a topic map requesting comments from users. Occurrences of these topics are linked with dialog specifications that offer access to the CMS services to fulfill the desired task.

![Figure 5: Topic Map with Task Specification](image)

These topic map fragments which express task specifications are associated with an intension according to our collaboration frame (who wrote a comment on which product.) in the content schema. Coordination is done by expressing obligations (e.g., adoptions of [18]) on the content chunk in the local theory of the intension, e.g., a moderator has to proofread the comment after the comment was written and before the comment is published. For that reason there is a topic map defining the proofreading task for moderators which can be merged with the topic map associated with the intension of commented products. This merging process creates the intension of a \textit{proofread comment}, characterized by the fact in the local theory that at a time point \( t \) the proofreading task took place:

\[
isProofReadComment(w) := (\exists w', t < now)(\text{moderator}(w, t) \land \text{proofread}(w, w'))
\]
From Wikis to Semantic Wikis: Extended Functionality

The new query functionality on content with topic annotations and intensional foundations enables additional high-level functionality for content management systems in social environments to support more sophisticated content management services:

- Sophisticated and automatically generated graphical user interfaces such as WYSIWYG editors rely on a user-centric topic-based content annotation to provide information in the right fashion at the right time.
- Community services such as contributions to communities, joining communities, meetings, publications, or community organization as well as their integration can be intensionally modelled.
- Online collaboration support active discussion and interaction among participants as well as content exchange.
- Content changes within a collaborating environment may be conflicting. Expressing the purposes of changes may help to solve these conflicts.
- Task annotations support modelling of interaction scenarios and coordination facilities such as schedules to enable project management functions.
- Secretarial functions such as filtering or ordering can be intensionally expressed and enforced by appropriate topic map definitions.
- Blackboard facilities support tracing of tasks, members, schedules, and documents. Report functions may be incorporated for non-members of the community.
- Ad hoc (and implicit) communities are supported. Members can conduct their own communities, interests, tasks, and portfolios by defining private workspaces.
- Asynchronous as well as synchronous collaboration is supported depending on the handling of annotations and intensions.

6 Conclusions

In this paper we are introducing a data model for content management systems which handles content as associations between the data itself, the intension behind the data, the usage of data and the origin of data. Content annotation is treated according to its purpose: terminology which is common sense in a community is shared in ontologies. Concepts which are only relevant to certain users or in certain situations are defined locally and removed from the global ontologies to make reasoning about global terminology easier. Local concepts may be exchanged and adapted between different usage contexts. For that reason concepts are seen not only as notions from an ontology but as small logical theories. Additionally, intensional annotation is separated from usage annotation to allow different expressions of the same data under the same intension.

Because of the reduced size of annotation ontologies, the local definition of concepts and the suggested evaluation strategies according to the origin definitions of the content, the separation of concerns within the data model allows a better automated reasoning support than simple \((S, P, O)\) annotation frameworks although decidability as well as soundness and completeness of the reasoning process cannot be guaranteed. The user-centric approach together with the facility of explicitly incorporating and exchanging hidden knowledge into
local theories behind a concept ensure the usability within known application scenarios when automated reasoning fails. Adding local and user-specific semantics to content chunks is a prerequisite for distributing content over social networks and therefore extends current Web 2.0 technologies in a natural way. While today Wikis support open communities mainly interested in free-form changes of content, Semantic Wikis may also support transaction oriented communities with the need of at least partially controlled collaboration.

References


A Conceptual View of Web-Based E-Learning Systems

Klaus-Dieter Schewe¹, Bernhard Thalheim²,
Aleksander Binemann-Zdanowicz², Roland Kaschek¹,
Thomas Kuss³, Bernd Tschiedel⁴

¹Massey University, Department of Information Systems &
Information Science Research Centre
Private Bag 11 222, Palmerston North, New Zealand
[k.d.schewe|r.h.kaschek]@massey.ac.nz
²Christian Albrechts University Kiel
Department of Computer Science and Applied Mathematics
Olshausenstr. 40, D-24098 Kiel, Germany
[thalheim|binemann]@is.informatik.uni-kiel.de
³Freiburg University, Center for Cognitive Science, Germany
thomas@cognition.iig.uni-freiburg.de
⁴Brandenburgian Technical University at Cottbus
Department of Computer Science,
Universitätsplatz 1, 03044 Cottbus, Germany
tschied@informatik.tu-cottbus.de

Abstract. Starting from a general framework for web-based e-learning systems
that is based on an abstraction layer model, this paper presents a conceptual mod-
elling approach, which captures the modelling of learners, the modelling of courses,
the personalisation of courses, and the management of data in e-learning systems.
Courses are modelled by outline graphs, which are further refined by some form
of process algebra. The linguistic analysis of word fields referring to an application
domain helps to set up these course outlines. Learners are modelled by classifying
value combinations for their characteristic properties. Each learner type gives rise
to intentions as well as rights and obligations in using a learning system. Intentions
can be formalised as postconditions, while rights and obligations lead to deontic
constraints. The intentions can be used for the personalisation of the learning sys-
tem to a learner type. Finally, the management of data in an e-learning system is
approached on two different levels dealing with the content of individual learning
units and the integrated content of the whole system, respectively. This leads to
supporting databases and views defined on them.

Keywords: e-learning systems, course modelling, learner modelling, personalisa-
tion, content management

Computing Reviews subject classification: [K.3.1] Distance Learning [H.5.3]
Web-based Interaction [H.3.5] Web-based Services

1. Introduction

More and more education institutions aim at providing e-learning sys-
tems, and more and more people take advantage of these offers as a

chance to advance their knowledge. It is envisioned that there is a huge market potential for learning technologies. Therefore, there is an increasing demand for adequate e-learning systems of highest standards, and a requirement for powerful techniques supporting agile and consistent learning. Expectations, chances and quality requirements are high. In particular, it still has to be proven that e-learning systems can provide the same (or nearly the same) teaching quality as contact classes with human teachers and adequate resources. In order to meet these high expectations professional design and development support for e-learning systems will be a crucial success factor.

The challenge in e-learning addresses at least two parts: the dissemination of information and the concept of teaching. In this article we concentrate on the first aspect, which reduces the problem to

- deciding, which information is to be presented in which form;
- enabling learners to chart individual routes through this information in accordance to their preferred learning style;
- adapting the information to different learner types.

We acknowledge that the concept of teaching is wider. For instance, didactics is more than outlining the content of a course. There is a considerable discussion in education regarding this wider perspective. However, learning technology has not yet reached a stage, where all the desiderata of e-learning systems can be supported. Our work emphasises the conceptual design of feasible systems and thus will demonstrate what can already be done with available technology, which is already more than is available in many systems on offer.

Our research interest concerns mainly those e-learning systems that are offered via the world-wide web. These e-learning systems can be considered as specific web-based information systems with a focus on the provision of knowledge to learners. In analogy to the B2C and B2B patterns that are well known in electronic commerce, we may use a pattern T2S for e-learning systems, where T represents a teacher as the service provider and S represents a student as the service consumer. More precisely, following the classification pattern in (Thalheim and Düsterhöft, 2000; Thalheim and Düsterhöft, 2001), we have a pattern $T^kS^l$ with $k$ standing for knowledge as the service that is provided, and $l$ standing for learning as the major activity supported by the system.

In this article we start from a general framework for web-based e-learning systems that is based on an abstraction layer model (Kaschek et al., 2003; Kaschek et al., 2004b). We present a conceptual modelling approach, which captures the modelling of courses, the modelling of
learners, the personalisation of courses, and the management of data in e-learning systems. This exceeds the capability of approaches in (Atzeni et al., 1998; Ceri et al., 2003; Conallen, 2003), which are based on a hypertext-extension of traditional development methods for data-intensive systems, and approaches such as (Van Duyne et al., 2002), which concentrate only on the presentation aspect.

Many learning systems that are currently offered are based on curriculum sequencing, where the learner has to follow a well-defined sequence of learning steps. To some extent this follows the principles of didactic preparation as described in detail in (Kerres, 2001). This is not always adequate, as learning should be better based on active request, i.e. e-learning systems should support self-organised learning on demand. Systems should restrain the learners as little as possible, and offer instead as much support as possible to support their individual learning styles. Consequently, it is important to anticipate the behaviour of learners and to design systems according to their needs. This includes outlining courses in such a way that the sequence of learning units, and the style of presentation is personalised to the type of learner. More abstractly speaking, for each learner type we have to anticipate how they will navigate through the system. Each possible sequence of learning units followed by a learner corresponds to a particular course outline, so the most challenging problem is to determine these sequences and to describe them in an abstract and integrated way. We will show that this can be modelled by some form of process algebra.

This approach is similar to storyboarding in web information systems (Binemann-Zdanowicz et al., 2004; Kaschek et al., 2004b; Schewe and Thalheim, 2001), which has been applied to e-commerce (Binemann-Zdanowicz et al., 2003a; Schewe et al., 2002; Thalheim et al., 2003), e-learning (Binemann-Zdanowicz et al., 2003b; Jantke et al., 2003; Rostanin et al., 2002), and information services (Feyer et al., 2000). The SiteLang process algebra has been introduced in (Thalheim and Düsterhöft, 2001). A short description of outline graphs also appeared in (Binemann-Zdanowicz et al., 2004).

We also address the problem how to discover the best outline graphs. We observe that an atomic learning unit is always centered around a single learner activity by the learner, and this activity can be described by a verb. Therefore, the idea is to analyse verbs. Fortunately, the number of verbs in a language such as English is relatively small, so it is not an intractable approach. In particular, we suggest to analyse the word fields of verbs, which combine aspects of morphology, i.e. the forms of the written or spoken word, phonology, i.e. the sound of the spoken word, syntax, i.e. the construction of sentences using the word
forms, semantics, i.e. the meaning(s) of the word, and pragmatics, i.e. the usage of the word in written or spoken language. This gives rise to determining the data needed to perform the activity, the guidance how to perform the activity, and the explanation of the effect of the activity.

The use of linguistic analysis for the design of web information systems has already been proposed in (Ravenscroft and Matheson, 2001; Thalheim and Dürsterhöft, 2004) based on computational linguistics (Hausser, 2001). In addition, the use of metaphors has been proposed in general in (Thalheim and Dürsterhöft, 2000) and investigated in (Schewe et al., 2002) for applications in e-banking, and in (Rostanin et al., 2002) for e-learning.

Furthermore, the quality of e-learning systems crucially depends on the designers’ understanding of the learners and their needs. Therefore, it is necessary to first obtain an idea of the expected learners. This may lead to certain learner profiles. Such profiles may be determined by the different goals of the learners, their different intentions, their different behaviour, their information needs, their levels of required support, etc. However, we go further and discuss dimensions that are closer to the particular interest of modelling learners instead of users of arbitrary web information systems or banking customers. However, similar to the existing work found in the literature we will obtain a learner space. We then discuss how points in this space can be combined into learner types.

Our approach adopts the whole-person approach from (Martinez, 2001) and the work on user profiling by Kaschek et al. (Kaschek et al., 2004b) as far as we base it on learner dimensions. Also (Hübscher, 2001; Merrill, 1983; ONTO-LOGGING Consortium, 2002) discusses learner modelling. A short description of learner modelling appeared in (Kaschek et al., 2004a).

Furthermore, each learner type will give rise to intentions as well as rights and obligations in using a learning system. Intentions can be formalised as postconditions, while rights and obligations lead to deontic constraints. The intentions can be used for the personalisation of the learning system to a learner type. We will discuss personalisation based on subgraphs, but there are other techniques such as providing different granularities through splitting and merging units. Reduction to subgraphs can be formally supported by Kleene algebras with tests (Kozen, 1997). Splitting of units based on cohesion pre-orders was investigated in (Feyer et al., 2000) and briefly described in (Binemann-Zdanowicz et al., 2004).

Finally, we address the problems associated with the system’s content, and partly with its functionality. We assume that learner types
have been determined, and a course outline consisting of learning units and navigation links between them has been set up. While these already determine the functionality in a rough way, our interest is looking at the system from a more systems-oriented perspective and asking how we can support the learning units. As e-learning systems are data-intensive systems, we adopt the approach by Feyer et al. in (Feyer et al., 2000), which approaches the data management problem in web information systems in an integrated way. According to this we obtain two levels of data management: a global level that is captured by a global database schema, and a local level that is captured by views, which represent the data content of the learning units. For both levels conceptual data models such as the Higher-Order Entity-Relationship model (Thalheim, 2000) are useful.

Among the more specific literature on e-learning, especially web-based e-learning systems very little attention is paid to the aspect of content modeling. Bergstedt et al. in (Bergstedt et al., 2003) study similarities between content management systems (CMSs) and e-learning systems and conclude that using CMSs in e-learning would improve the quality of e-learning systems. The work in (Qu and Nejdl, 2003; Sessink et al., 2003) investigates the metadata standard SCORM. Sessink et al. in (Sessink et al., 2003) identify the need for supporting databases that are not in the SCORM 1.3 standard. Qu and Nejdl in (Qu and Nejdl, 2003) use XML to encompass incompatibilities between RDF and SCORM metadata. Mohan and Brooks in (Mohan and Brooks, 2003) address the problem of discovering learning objects from the semantic web and argue that this will improve learning. However, this depends on the semantic understanding of the web, which despite the merits of the research efforts in this area is still beyond reality. A short description of our approach to data modelling in e-learning systems appeared in (Rostanin et al., 2004).

We present our general framework in Section 2. Then we address the modelling of courses via outline graphs in Section 3, followed by a discussion of linguistic analysis in Section 4 as a means for setting up adequate course outlines. Section 5 is then devoted to the classification of learners following the whole-person approach from (Martinez, 2001). In addition, we discuss intentions, rights and obligations that are associated with learner types. These link the learner types to the course outlines. Furthermore, intentions of learners can be used to personalise learning system to the needs of the learners. This will be described in Section 6. Finally, in Section 7 we describe how the data in the various learning units can be modelled by using a two-level approach. We conclude with a brief summary in Section 8.
2. General Design Considerations

E-Learning Systems are large web-based information systems, and as such they share a lot of similarities with web information systems in general. Consequently, systems can be largely designed along the same principles that apply to traditional information systems. However, a particular focus has to be put onto the fact that the system will be web-based, and that its major purpose is to support learning.

This leads to important questions such as “Who are the learners?”, “Which learner intentions and behaviour shall be supported?”, “Which technical devices will be used by the learners?”, etc. We will now look at these questions into more detail focussing on five different aspects: purpose, usage, content, functionality and presentation.

2.1. Aspects of E-Learning Systems

The purpose aspect is a very general one centered around a mission statement for the system. The primary question is: what is the purpose of the system? In e-learning systems the major purpose may be to provide learning material to students including useful hints and links to supplementary literature. A second related question is this: Are there several minor purposes as well? In commercial e-learning systems a minor purpose may be to attract students to book further courses, in which case the system would be a mixture of an e-learning and an e-commerce system. A third question associated with purpose concerns their time-scale. Some of the intentions may be long-term others short-term.

Once some clarity with respect to the purpose of the e-learning system has been obtained, the question arises by whom and how the system will be used. As a web-based system it is usually open, so it is important to anticipate the behaviour of the learners. Therefore, it is necessary to first obtain an idea of the expected learners. This may lead to certain learner profiles. Such profiles may be determined by the different intentions of the learners, their different behaviour, their information needs, their levels of required support, etc.

So, the activity of learner profiling will lead to a list of profiles of expected learners who are to be supported by the system. This influences the content of the system’s pages, their logical organisation, the enabled navigation links between these pages, and maybe even their presentation. More abstractly speaking, for each learner profile we have to anticipate how they will navigate through the system. Each possible sequence of pages followed by a learner corresponds to a particular
course outline, so the most challenging problem is to determine these sequences and to describe them in an abstract and integrated way.

The usage of an e-learning system depends on whether the control of the learning process is left to the learner or the system. In both cases, however, it is assumed that the learners are willing to learn and match the required prerequisites. Learners normally enter the system more than once continuing a specific learning programme. This requires some authentication mechanism, especially if the learning progress is controlled by the system. Quality criteria are set by the teaching quality, and in the end, by the increase of knowledge on the side of the learners.

The content aspect is central to the development of the system, as it concerns the question: “Which information should be provided?”, which is coupled with the problem of designing an adequate database. However, the organisation of data that is presented to the learner via the web-interface is significantly different from the organisation of data in a database. So, organising the data content of the system means to investigate the decomposition, structuring and classification of data in such a way that the course outline(s) can be adequately supported.

Thus, modelling the content of a system has to be addressed on at least two levels: a logical level leading to databases, and a conceptual level leading to the content of pages. Both levels have to be linked together. Furthermore, in both cases abstraction mechanisms should be used. While such abstraction mechanisms are established in the area of databases, they are still a matter of research for web-based systems and in particular for e-learning systems.

Modelling content must take into account that information must be presented in different ways to different learners. This depends on the learner profile, the communication channel, and the available devices. Modelling content has to provide mechanisms to tailor the content automatically according to these parameters.

The functionality aspect is coupled with the question, whether the e-learning system should be passive or active. A passive system would only allow a learner to navigate through the pages without any activity. In these cases the major problem associated with functionality is to set up an adequate navigation structure.

In an active system, however, information would also be required from the learner. From a conceptual point of view, the main purpose of functionality modelling is to identify functions that are available to support the activities of the learners, which were identified in the course outline. Such functions can be system-specific functions in order to process learner input or general support functions.
The functionality of e-learning systems mainly supports the navigation through the learning material. In contrast to other types of web-based systems, this navigation is a long-term process with usually many interruptions. More sophisticated systems would provide system-driven repetition and feedback. Also, personal information needs can be supported by providing an interface to e-mail.

Finally, the presentation dimension concerns the realisation by web pages. This depends on the support of technical end-devices such as computer screens, television, cell phones, etc. and set layout preferences. This is partly done in accordance to results from learning psychology trying to direct the attention of the learners to the most relevant parts first. This will give them the impression of a well-organised system.

2.2. ABSTRACTION LAYERS IN E-LEARNING SYSTEMS

In this section we present a framework for the design of e-learning systems. The framework is based on an Abstraction Layer Model, which is illustrated by Figure 1. From top to bottom we identify a definition layer, a didactic layer, a tutorial layer, a presentation layer, and an implementation layer.

The general ideas of this model are as follows. We identify several layers of abstraction:

- The top layer is called the definition layer. It is used to describe the system in a general way: What is the purpose? What are the course goals? Who are the expected learners?
The next lower layer is called the didactic layer, which is used to concretise the ideas gathered on the definition layer. This means to get a clearer picture of the different learners and their profiles. The major part of this layer, however, deals with the outline of the course. That is to identify course units and paths through these units.

The central layer is the tutorial layer. Whilst the didactic layer did not pay much attention to technical issues, they now come into play. The various units appearing in the course outline have to be analysed and integrated, so that each unit can be supported by a combination of some data content with some functionality.

The next lower layer is the presentation layer which is devoted to the problem of associating presentation options. Finally, the lowest layer is the implementation layer, which addresses all aspects of the physical implementation.

On each layer except the definition layer we identify two dimensions for the description of the system: focus and modus. The focus dimension distinguishes between local and global components; the modus dimension distinguishes between static and dynamic components. Global and static components will be addressed by a repository specification, which covers all aspects of central data storage and retrieval. Global and dynamic components will be addressed by a functionality specification, which covers all operation on the stored data.

Local and static components will addressed by a content specification, which covers all aspects of the content of course units. Finally, local and dynamic components will be addressed by a unit specification, which covers the learner activities associated with the learning units.

Each layer is associated with layer specific modelling tasks. The transition from the definition to the didactic layer is associated with developing the course outline(s) and the identification of learner profiles. The transition from the didactic layer to the tutorial layer is associated with conceptual modelling, which addresses database modelling, operations modelling, view modelling, and unit content modelling. This adds more details to the design, as we look at single course units, whereas the composition of the course as such has been dealt with on the didactic layer.

The transition to the presentation layer is associated with the definition of presentation styles. Finally, the transition to the implementation layer is associated with all implementation tasks.
3. Course Modelling

Let us now look at ways to describe the navigation of learners through an e-learning system, for which we may exploit finite, directed graphs for this purpose.

Thus, an outline graph is a finite, directed graph $G = (V, E)$, i.e. $V$ and $E$ are finite sets with $E \subseteq V \times V$. The vertices, i.e. the elements of $V$, are called learning units, and the edges, i.e. the elements of $E$ are links between these units.

Take for example a course dealing with e-learning systems. Then we might have learning units such as Introduction, Learner Profiling, Course Outlining, Data Management, Adaptivity, Style Definition, and Implementation. Figure 3 gives a rough picture of the corresponding course outline with links naturally represented by edges.

3.1. Outline Graphs

With each learning unit $u \in V$ we associate a view $C^u$ describing the data content of this learning unit. In addition, we may also associate learner types $LT^u_1, \ldots, LT^u_n$ with the learning unit. These indicate that the learning unit is suitable for learners of this type only. We will describe learner types in Section 5.

Each link $\ell \in E$ from $u_1$ to $u_2$ corresponds to a possible transition from the source learning unit $u_1$ to the target learning unit $u_2$. Such a transition should be triggered by an action initiated by the learner. This action can simply be a navigation, but in general we may think of more complicated actions. Therefore, each link is associated with the name of...
an action that can be issued in that learning unit. In addition, it is also associated with a data type expressing the information communicated from learning unit $u_1$ to learning unit $u_2$.

Actions on a learning unit may depend on the successful completion of the learning unit by the learner. According to our view this is part of the action specification, and should be left for further refinement of the outline.

From a more conceptual point of view we model learner interaction with an e-learning system according to two primitives: transition between learning units and using the functionality offered at a given learning unit. This allows for a two-step modelling procedure to be applied. First at a coarse-grained level learning units and navigation links are modelled. Then in a refinement step the actual activities of the learners are added. This procedure allows for a good separation of concern with respect to personalisation and localisation.

At first the usage of the system by a particular learner type is modelled. This already reduces the complexity of the problem. Then the learner behaviour is dealt with locally, i.e. with respect to a given learning unit, which further reduces complexity. The price for this advantage is the need to integrate the various outlines and the data used within them.

Instead of emphasising the transitions between learning units and the triggering learner activities as sketched in Figure 3, we may want to emphasise the data communication between learning units. As we assume that for any two learning units $u_i, u_j \in V$ there is at most one link from $u_i$ to $u_j$, the outline graph can be represented by its adjacency matrix $A = \{a_{i,j}\}_{1 \leq i, j \leq n}$ with

$$a_{i,j} = \begin{cases} 1 & \text{if there is a link from } u_i \text{ to } u_j \\ 0 & \text{else} \end{cases}$$

Using this representation has the advantage that we get rid of crossing labelled edges in diagrams. However, we use it in a modified form.

First we use a table $T$ with $n$ columns and rows to represent the matrix. We further fill the table’s cells $T(i, i)$ for all $i = 1, \ldots, n$ with the name of learning unit $u_i$ and fill the table cell $T(i, j)$ with the labels attached to the edge from $u_i$ to $u_j$.

The learner types associated with the learning unit $u_i$ will be added at the right side of the table in the form of an additional column. This extended adjacency matrix will be called the communication matrix of the course outline.

Figure 3.1 contains a sketch of the communication matrix for the course outline from Figure 3. We abbreviated the names of the learning
units in the obvious way. As we did not yet indicate, which learner activities trigger the navigation links, we only put a \( \bullet \) into the corresponding cell of the matrix. Similarly, as we left the learner types open so far, we could only add \( LT_x \) into the corresponding cell of the matrix.

### 3.2. Algebraic Modelling of Course Outlines

Let us take now a closer look at the process algebra SiteLang from (Thalheim and Düsterhöft, 2001). So, let \( S = \{s_1, \ldots, s_n\} \) be a set of learning units, and let \( A = \{\alpha_1, \ldots, \alpha_k\} \) be a set of (atomic) actions. Furthermore, assume a mapping \( \sigma : A \rightarrow S \), i.e. with each action \( \alpha \in A \) we associate a learning unit \( \sigma(\alpha) \).

This can be used to define inductively the set of \textit{learning processes} \( P = P(A, S) \) determined by \( A \) and \( S \). Furthermore, we can extend \( \sigma \) to a partial mapping \( P \rightarrow S \) as follows:

- Each action \( \alpha \in A \) is also a learning process, i.e. \( \alpha \in P \), and the associated learning unit \( \sigma(\alpha) \) is already given.

- 0 and 1 are learning process, for which \( \sigma \) is undefined. 1 is a content-less learning process, while 0 stands for a failed learning process.

- If \( p_1 \) and \( p_2 \) are learning processes, then also the \textit{sequence} \( p_1 \cdot p_2 \) is a process. Furthermore, if \( \sigma(p_1) = \sigma(p_2) = s \) or one of the \( p_i \) is 1, then \( \sigma(p_1 \cdot p_2) \) is also defined and equals \( s \), otherwise it is undefined.
− If \( p_1 \) and \( p_2 \) are learning processes, then also the choice \( p_1 + p_2 \) is a learning process. Furthermore, if \( \sigma(p_1) = \sigma(p_2) = s \) or one of the \( p_i \) is 1, then \( \sigma(p_1 + p_2) \) is also defined and equals \( s \), otherwise it is undefined.

− If \( p \) is a learning process, then also the iteration \( p^* \) is a learning process with \( \sigma(p^*) = \sigma(p) \), if \( \sigma(p) \) is defined.

− If \( p \) is a learning process and \( \varphi \) is a boolean condition, then the guarded learning process \( \varphi \cdot p \) and the post-guarded learning process \( p \cdot \varphi \) are learning processes with \( \sigma(\varphi \cdot p) = \sigma(p \cdot \varphi) = \sigma(p) \), if \( \sigma(p) \) is defined.

We also use conjunction \( \varphi \cdot \psi \), disjunction \( \varphi + \psi \) and negation \( \bar{\varphi} \) for boolean conditions. Furthermore, let 1 represent ‘true’ and 0 ‘false’. Then the set \( P \) of learning processes carries the structure of a Kleene algebra with tests (Kozen, 1997), i.e. the following rules hold:

− \( + \) and \( \cdot \) are associative, i.e. for all \( p, q, r \in P \) we must have \( p + (q + r) = (p + q) + r \) and \( p(qr) = (pq)r \);

− \( + \) is commutative and idempotent with 0 as neutral element, i.e. for all \( p, q \in P \) we must have \( p + q = q + p \), \( p + p = p \) and \( p + 0 = p \);

− \( 1 \) is a neutral element for \( \cdot \), i.e. for all \( p \in P \) we must have \( 1p = p \);

− for all \( p \in P \) we have \( p0 = 0p = 0 \);

− \( \cdot \) is distributive over \( + \), i.e. for all \( p, q, r \in P \) we must have \( p(q + r) = pq + pr \) and \( (p + q)r = pr + qr \);

− \( p^*q \) ist the least solution \( x \) of \( q + px \leq x \) and \( qp^* \) is the least solution of \( q + xp \leq x \), using the partial order \( x \leq y \equiv x + y = y \).

We further adopt the convention to write \( pq \) for \( p \cdot q \), and to assume that \( \cdot \) binds stronger than \( + \), which allows us to dispense with some parentheses. It is also known that the double use of \( \cdot \) for sequence and conjunction and of \( + \) for choice and disjunction, respectively, does not cause any problems. Obviously, if we restrict to boolean conditions only, then the laws of Boolean algebras apply to conjunction, disjunction and negation.

Furthermore, the association of learning units with learning processes implies that we also have sorts. As processes associated with different scenes express concurrency, we claim that \( p_1p_2 = p_2p_1 \) holds for all \( p_1, p_2 \in P \) with \( \sigma(p_1) \neq \sigma(p_2) \), which leads to a many-sorted Kleene algebra with tests (Schewe and Thalheim, 2004).
**Example 1** The following expression may express a course outline with actions $\alpha_i$ and conditions $\varphi_i$.

\[
\alpha_1((\varphi_0\alpha_2 + \varphi_1\alpha_3(\alpha_5 + 1)\varphi_3 + \varphi_2\alpha_4(\alpha_6 + 1)\varphi_4)\varphi_5)(\alpha_7\varphi_6 + \alpha_{13}\varphi_7)
\]

\[
(\varphi_6\alpha_8(\alpha_8 + 1)\alpha_9\alpha_{10}\alpha_{11}\alpha_{12}\varphi_8 + \varphi_7\alpha_8\alpha_8^*\alpha_{14}\alpha_{15}\alpha_{16}\alpha_{11}\alpha_{17}(\varphi_{12}\alpha_{18}\alpha_{19})^*\varphi_{12}\alpha_{18}\varphi_9)
\]

\[
\alpha_{20}(\varphi_{10} + \varphi_{11})
\]

In addition we may have the following assignment of learning units $s_i$ to the actions

\[
\begin{align*}
\sigma(\alpha_1) &= s_1 & \sigma(\alpha_2) &= s_2 & \sigma(\alpha_3) &= s_2 & \sigma(\alpha_4) &= s_3 & \sigma(\alpha_5) &= s_2 \\
\sigma(\alpha_6) &= s_3 & \sigma(\alpha_7) &= s_4 & \sigma(\alpha_8) &= s_4 & \sigma(\alpha_9) &= s_4 & \sigma(\alpha_{10}) &= s_4 \\
\sigma(\alpha_{11}) &= s_4 & \sigma(\alpha_{12}) &= s_4 & \sigma(\alpha_{13}) &= s_5 & \sigma(\alpha_{14}) &= s_5 & \sigma(\alpha_{15}) &= s_5 \\
\sigma(\alpha_{16}) &= s_5 & \sigma(\alpha_{17}) &= s_5 & \sigma(\alpha_{18}) &= s_5 & \sigma(\alpha_{19}) &= s_5 & \sigma(\alpha_{20}) &= s_6
\end{align*}
\]

with the learning units

\[
\begin{align*}
s_1 &= \text{Introduction} & s_2 &= \text{Learner Profiling} & s_3 &= \text{Course Outlining} \\
s_4 &= \text{Data Management} & s_5 &= \text{Style Definition} & s_6 &= \text{Implementation}
\end{align*}
\]

### 4. Linguistic Analysis

Linguistic analysis summarises the activities and techniques that are applied to analyse natural language descriptions of learners’ activities. The major source of information used for developing e-learning systems is a description of curriculae, which may be delivered in the form of texts or verbally.

Linguistic analysis takes such natural language descriptions and uses them for analysing the activities of the learners: Which are these activities? Does the description indicate any sequencing or continuation? What data is needed for or used by the activities? What are the relationships between these data?

As learner activities can be described by verbs, we suggest to analyze the corresponding word fields. We will show that word fields are a valuable source for centering the outline graph around the central learner activities.
4.1. Word Fields

According to (Hausser, 2001) a word is an abstract concept, which in order to become concrete needs a word form carrying the grammatical variants. Word fields are a much more general notion than word forms. Word fields combine different aspects:

- morphology, i.e. the forms of the written or spoken word,
- phonology, i.e. the sound of the spoken word,
- syntax, i.e. the construction of sentences using the word forms,
- semantics, i.e. the meaning(s) of the word, and
- pragmatics, i.e. the usage of the word in written or spoken language.

Our restriction to written language communication makes considering phonology obsolete. Under this premise we understand a word field to consist of its intext, its context, its semantics and its pragmatics.

- The intext combines morphology and syntax. We are mainly interested in the latter one, especially for verbs. In this case we may ask for the basic syntactical form, various mandatory and optional arguments, variants of the arguments, extensions to the basic syntax, and relationships and constraints between the arguments of the word when it is used in sentences.

For instance, the verb “to teach” uses a mandatory argument “something” and an optional argument “to somebody”. The basic syntactical form “Someone teaches something to somebody” expresses an action, specifies the actor and the receiver, and an object used in the action.

- The context refers to application areas. According to the context we may identify related words, concepts related to the arguments, categories of actors and actions, and further constraints.

For instance, in a learning context the verb “to teach” may be used in a sentence such as “The teacher teaches e-commerce to third-year students”. In this case we conclude that the object of the action is subject to an assessment of whether (s)he has learnt and understood the learning object “e-commerce”. It also implies that a follow-on action, i.e. the study by the student, is expected.
The semantics refers to the meaning of a word in a particular context. For example, the already used sentence “The teacher teaches e-commerce to third-year students” includes the meaning of the teacher delivering and explaining material and the student working through it and reflecting it in a critical way.

The pragmatics refers to how the word is used in the application context.

4.2. Word Fields and Outline Graphs

Using word fields for the design of dialogues has already been suggested in (Thalheim and Düsterhöft, 2004). We suggest a compositional approach to outline graph design based on generalised phase structure grammars with applicability rules such as immediate dependence and linear precedence rules (Hausser, 2001).

We propose to exploit word fields in an analytic way starting with the syntactic analysis of sentences. For this of course we also need a sophisticated parsing theory (Hausser, 2001) to obtain the syntax trees. Within such a tree we can identify the main verb and the used arguments. The verb can be used as a descriptor for the main activity. The word field of the verb will tell us, which arguments are expected. This will give us hints for detecting the required information and the learners involved.

Finally, the various constraints indicate relationships to other activities, e.g. follow-on activities described by other sentences or relationships to objects used in these other sentences. This gives rise to determine links in the outline graph and the communicated data associated with these links. It may also indicate that the learning unit is still too broad and needs to be refined. For instance, we may exploit the specialisation of verbs or objects that are used as their arguments to the refinement of learning units.

4.3. Metaphors

According to (Thalheim and Düsterhöft, 2000) metaphors are language expressions used in an uncommon language context. Properly applied, they can simplify the task of communicating complex ideas and result in enthusiastic users.

We already pointed out that generating the best learner support depends on having a model of the learners, i.e. being able to group learners into reasonably constructed learner types. The e-learning system must be designed such that it stimulates questions by the learners.
that can be answered relatively easily. Metaphors can help in this regard. They can help learners to understand what they can, should, or should not attempt to do next, and allow the learner to master his/her own learning style.

Human communication is largely metaphorical. It is likely that the lack of metaphors in traditional human-computer interaction is responsible for some of the problems of this interaction. Improving human-computer interaction by augmented use of metaphors has the potential to reduce the number of communication barriers as well as their implications.

Recall that computer applications have at least three language levels:

- **tool language:** This is the language in which the learner interface signals the customer the semantics of its functionality.

- **discourse language:** This is the language used in the universe of discourse to identify problems, their solutions and quality criteria of all of them.

- **metaphor language:** This language helps the learners understand the state of affairs in the universe of discourse and what interface functions can be used to achieve their goals.

The generation of metaphors could be supported by logically decomposing the learning space into a small number of domains that appear homogeneous with respect to the offered functionality. This permits relating the domains to learner types and expected learner activities. The generation of metaphors appears then as being connected to finding characterising names for the relationships between learner types, expected learner activities and domain.

### 5. Learner Modelling

In this section we focus on the learner model following an inspiration by the whole-person-approach from (Martinez, 2001). In this approach a comprehensive set of human factors for learner modelling is considered. This leads to conceptual modelling of learners by learner profiles, i.e. value combinations for a list of learner characteristics. A complete list of such characteristics will of course depend on the learning domain. Therefore, we concentrate on selected reasonable characteristics.
5.1. The Learner Space

We follow (Martinez, 2001) in considering cognitive, conative, and affective personality aspects of individuals as key regarding the outcomes of their learning processes. These psychological aspects impact the learning outcome. Increasing the learning performance of most learners, if based on adaptation of e-learning systems to learners, thus appears to require learner characteristics being identified that relate to the mentioned personality aspects. We shall distinguish between characteristics of the learning style, the learner as a person, the learning task at hand, and the preferred examination style. The following lists are assumed to be non-exhaustive.

For the learning style we have the following general characteristics:

**Guidance:** The degree to which the learner prefers being guided or not in the learning process.

**Visual Modality:** The degree to which the learner prefers a presentation in visual manner. We follow (Rostanin et al., 2002) in using the characteristic modality.

**Auditory Modality:** The degree to which the learner prefers a presentation in auditory manner.

**Textual Code:** The degree to which learners with visual learning style prefer having access to text.

**Illustrational Code:** The degree to which learners with visual learning style prefer having access to images.

**Example:** The degree to which the learner prefers learning deductively or inductively.

Characteristics that refer to the learners are:

**Persistency:** The degree to which the learner in general can or cannot mentally cope with new material.

**Retentivity:** The degree to which the learner in general can or cannot memorize learned material.

**Computer literacy:** The degree to which the learner has or has not acquired skills in using modern computing infrastructure for a task at hand.

**Curiosity:** The degree to which the learner in general is or is not fond of learning new material.
Learning task related characteristics are the following:

**Prerequisites:** The degree to which the learner has or has not learnt the required subjects.

**Performance:** The degree to which the learner has or has not performed well regarding background subjects, required subjects or subjects in general.

**Motivation:** The degree to which the learner is or is not interested in efficiently mastering the learning task at hand.

**Confidence:** The degree to which the learner believes to be capable or incapable of successfully mastering the learning task.

Characteristics that apply to the examination style are:

**Kind:** The degree to which the learner either prefers his knowledge or his skills or a combination of both being checked.

**Control:** The degree to which the learner prefers his increment in knowledge or skills being checked.

**Evaluation strategy:** The degree to which the learner prefers his solution to an examination problem being evaluated right after its submission or whether he / she prefers all problem solutions being evaluated at the same time.

**Feedback:** The degree to which the learner prefers receiving full explanations of the assessments of his / her problem solutions or whether he / she prefers being notified of the correct results only.

We believe that the chosen grouping of characteristics is obvious and that the characteristics we have defined are obvious and plausible candidates for characteristics in real learning tasks. We further believe that at least extreme scores in scales for the characteristics result in manifest options in designing and presenting learning units. We thus do not discuss these issues here in more detail. Clearly, persistency and retentivity are cognitive; curiosity and motivation are conative; and guidance, evaluation strategy and feedback are affective personality aspects. Thus indeed our characteristics for the key personality aspects gives at least a hint on how to break them down and make use of these for e-learning systems.
5.2. Formalising the Learner Space

In order to describe learner profiles we identified general and subject-specific characteristics of learners. So formally, we obtain a non-empty set $C$ of learner characteristics. In addition, for each of these learner characteristics we obtain a linearly ordered set of values, which from now on will be called the scale of the characteristic. Formally, for each characteristic $c \in C$ we have a scale $S(c)$ and a linear order $\leq$ defined on it.

For example, if the characteristics are abstraction, perception and memory, we may use a numerical scale, say the integer interval $[0, \ldots, 5]$, for each of them.

- A low value for abstraction indicates a learner who needs more concrete examples in order to be able to understand the learning material, whereas a higher level indicates a better ability of abstract thinking.

- A low value for perception indicates a need for visualisation, while a higher value indicates that the learner can cope with textual and maybe even formal writing.

- A low value for memory indicates that the learner often needs to review larger parts of the learning material, whereas a higher value indicates that this is not needed or that the learner only needs a brief reminder in a very condensed form.

The learner space $LS$ is the cartesian product of the scales, i.e.

$$LS = \prod_{c \in C} S(c).$$

Thus, each element of the learner space is a tuple, and each component of this tuple indicates the value of a certain learner characteristic. That is, the learner space captures our knowledge about the different combinations of learner characteristics.

Figure 5.2 illustrates a learner space with two characteristics, i.e. $C = \{c_1, c_2\}$, and the scales $S(c_1) = \{v_1, v_2, v_3, v_4\}$ with $v_1 < v_2 < v_3 < v_4$, and $S(c_2) = \{v'_1, v'_2, v'_3\}$ with $v'_1 < v'_2 < v'_3$. The learner space $LS = S(c_1) \times S(c_2)$ contains 12 tuples.

5.3. Learner Types

Thus, the learner space is an adequate way to represent the different learners. However, we have also seen that there are not so few characteristics of learner types, and for each of them we obtain quite a
few different values to be taken into account. This bears the risk of a
combinatorial explosion of the learner space.

In order to personalise a system to learners, we would have to parameterise each learning unit and each navigation link by the elements of the learner space or provide functions that map units and links between them to a personalised course outline. This is only feasible, if either the learner space is small or the personalisation will be the same for many elements of the learner space.

For instance, if we have 10 different characteristics and each comes with only two different values in their scale, then we would already have $2^{10} = 1024$ elements in the learner space. This is far too much.

Therefore, we have to bundle points in the learner space, i.e. instead of attempting to personalise the system to each combination of characteristic values, we would personalise it only to learner types. In general, a learner type should correspond to several points in the learner space. We now discuss two different ways to combine elements of the learner space to learner types.

A convex learner type $LT$ is a cuboid in $LS$. In other words, if $C = \{c_1, \ldots, c_n\}$, then take intervals $[v_i, v'_i] \subseteq S(c_i)$ for all scales $i = 1, \ldots, n$ and define

$$LT = \prod_{i=1}^{n} [v_i, v'_i] \subseteq LS.$$

Look again at Figure 5.2. Here we defined six convex learner types:

- $LT_1 = \{v_1\} \times \{v'_1, v'_2\}$
- $LT_2 = \{v_2\} \times \{v'_2\}$
- $LT_3 = \{v_2, v_3\} \times \{v'_1\}$
- $LT_4 = \{v_1, v_2\} \times \{v'_3\}$
- $LT_5 = \{v_4\} \times \{v'_1\}$
- $LT_6 = \{v_3, v_4\} \times \{v'_2, v'_3\}$

In other words, for learners with value $v_1$ for characteristic $c_1$ we do not make a distinction between the values $v'_1$ and $v'_2$ anymore, as this will not have an effect on the system design.
Alternatively, we may apply aggregation functions to the learner space. In this case we assume that the scales are sets of integers, e.g. intervals $S(c_i) = [m_i, M_i]$. An aggregate function on the learner space is an integer valued function $f : LS \rightarrow \mathbb{N}$.

An aggregate learner type is a subset of $LS$ of the form

$$LT = \{ \ell \in LS \mid m \leq f(\ell) \leq M \}$$

for some integer interval $[m, M]$.

For instance, in Figure 5.3 we have taken the integer scales $S(c_1) = \{0, 2, 3, 5\}$ and $S(c_2) = \{0, 1, 4\}$. Then the aggregation function is a simple addition, i.e. $f(v, w) = v + w$. We then define four aggregate learner types:

$$LT_1 = \{ (v, w) \in LS \mid 0 \leq f(v, w) \leq 1 \}$$
$$LT_2 = \{ (v, w) \in LS \mid 2 \leq f(v, w) \leq 3 \}$$
$$LT_3 = \{ (v, w) \in LS \mid 4 \leq f(v, w) \leq 6 \}$$
$$LT_4 = \{ (v, w) \in LS \mid 7 \leq f(v, w) \leq 9 \}$$

These learner types are illustrated as well in Figure 5.3.

5.4. Intentions, Rights and Obligations

The presence of different learner types indicates different usage of the system. An obligation of a learner type specifies what a learner of that type has to do. A right specifies what a learner of that type is permitted to do. Both obligations and rights together lead to complex deontic integrity constraints. We use the following logical language $L$ for this purpose:

- All propositional atoms are also atoms of $L$. 

Figure 5. Aggregate Learner Types
− If \( \alpha \) is an action on learning unit \( s \) and \( r \) is a learner type associated with \( s \), then \( O \ do(r, \alpha) \) is an atom of \( L \).

− If \( \alpha \) is an action on learning unit \( s \) and \( r \) is a learner type associated with \( s \), then \( P \ do(r, \alpha) \) is an atom of \( L \).

− If \( \alpha \) is an action on learning unit \( s \) and \( r \) is a learner type associated with \( s \), then \( F \ do(r, \alpha) \) is an atom of \( L \).

− For \( \varphi, \psi \in L \) we also have \( \neg \varphi, \varphi \land \psi, \varphi \lor \psi, \varphi \Rightarrow \psi \) and \( \varphi \Leftrightarrow \psi \) are also formulae in \( L \).

The interpretation is standard. In particular, \( O \ do(r, \alpha) \) means that a learner of type \( r \) is obliged to perform action \( \alpha \), \( P \ do(r, \alpha) \) means that a learner of type \( r \) is permitted to perform action \( \alpha \), and \( F \ do(r, \alpha) \) means that a learner of type \( r \) is not allowed to perform action \( \alpha \).

**Example 2** If action \( \alpha_8 \) represents an indispensible part of the course, e.g. a compulsory assignment, each learner will have to take it, once one of the actions \( \alpha_7 \) or \( \alpha_{13} \) have been executed (condition \( \varphi_6 \) or \( \varphi_7 \)). Furthermore, assume that in case of \( \varphi_7 \) the learner is obliged to execute actions \( \alpha_{16} \) and \( \alpha_{17} \), i.e. we obtain the deontic constraints

\[
\varphi_6 \lor \varphi_7 \Rightarrow O \ do(\text{learner}, \alpha_8)
\]

and

\[
\varphi_7 \Rightarrow O \ do(\text{learner}, \alpha_{16}) \land O \ do(\text{learner}, \alpha_{17})
\]

The intention of a learner can be expressed by goals, which can be modelled by postconditions to the learning space, which itself is described by an expression in the algebra discussed in Section 3.

6. Course Personalisation

Let us now discuss some techniques for the personalisation of outline graphs: reduction to a subgraph, and splitting of learning units.

6.1. Reduction to a Subgraph

According to the definition of outline graphs each learning unit is associated with a set of learner types. As a consequence, given a certain learner type only those learning units that are associated with it, are accessible for a learner of this type. This leads to the definition of the subgraph spanned by a learner type.
Formally, let $LT$ be some learner type, and let $G = (V,E)$ be an outline graph. Then the set of learning units that are accessible to learners of type $LT$ is

$$V(LT) = \{ u \in V \mid u \text{ is associated with } LT \}.$$ 

The outline graph $G(LT) = (V(LT), E(LT))$ with

$$E(LT) = E \cap (V(LT) \times V(LT))$$

defines the subgraph spanned by $LT$, i.e. we only consider those links, for which the starting and the target learning unit are in the set of learning units that are accessible to learners of type $LT$.

Of course $G(LT)$ is an outline graph, and all its links are associated with the learner type $LT$. In fact, this subgraph models exactly the part of the outline graph that corresponds to the permitted navigation for learners of type $LT$.

We may also exploit the algebra from Section 3 to compute a subgraph by equational reasoning. The general approach is to formulate a problem by using equations or conditional equations. Furthermore, we obtain (conditional) equations, which represent application knowledge. This application knowledge arises from events, postconditions and knowledge about the use of the system for a particular purpose. We then apply all equations to solve the particular problem at hand.

The application knowledge contains at least the following equations:

1. If an action $p$ has a precondition $\varphi$, then we obtain the equation $\bar{\varphi}p = 0$.
2. If an action $p$ has a postcondition $\psi$, we obtain the equation $p = p\psi$.
3. If an action $p$ is triggered by a condition $\varphi$, we obtain the equation $\varphi = \varphi p$.
4. In addition we obtain exclusion conditions $\varphi\psi = 0$ and tautologies $\varphi + \psi = 1$.

The problem of personalisation according to the preferences of a particular learner can be formalised as follows. Assume that $p \in P$ represents the learner space. Then we may formulate the preferences of a user by a set $\Sigma$ of (conditional) equations. Let $\chi$ be the conjunction of the conditions in $\Sigma$. Then the problem is to find a minimal process $p' \in P$ such that $\chi \Rightarrow px = p'x$ holds for all $x \in P$. Preference equations can arise as follows:

1. An equation $p_1 + p_2 = p_1$ expresses an unconditional preference of action (or process) $p_1$ over $p_2$. 
2. An equation \( \varphi(p_1 + p_2) = \varphi p_1 \) expresses an conditional preference of action (or process) \( p_1 \) over \( p_2 \) in case the condition \( \varphi \) is satisfied.

3. Similarly, an equation \( p(p_1 + p_2) = pp_1 \) expresses another conditional preference of action (or process) \( p_1 \) over \( p_2 \) after the action (or process) \( p \).

4. An equation \( p_1 p_2 + p_2 p_1 = p_1 p_2 \) expresses a preference of order.

5. An equation \( p^* = pp^* \) expresses that in case of an iteration it will at least be executed once.

Example 3 Let us continue Example 1. Assume that we have to deal with a learner who knows \( \varphi_5 \). This can be expressed by the application knowledge equation \( x \varphi_5 = x \) for all \( x \in K \). Furthermore, assume three additional exclusion conditions:

\[
\varphi_5 \varphi_0 = 0 \quad \varphi_5 \varphi_1 = 0 \quad \varphi_5 \varphi_2 = 0
\]

Taking these equations to the first part of the expression in Example 1 we obtain

\[
\alpha_1((\varphi_0 \alpha_2 + \varphi_1 \alpha_3(\alpha_5 + 1)\varphi_3 + \varphi_2 \alpha_4(\alpha_6 + 1)\varphi_4)^\ast \varphi_5)x = \alpha_1((0 \alpha_2 + 0 \alpha_3(\alpha_5 + 1)\varphi_3 + 0 \alpha_4(\alpha_6 + 1)\varphi_4)^\ast \varphi_5)x = \alpha_1 \varphi_5 x = \alpha_1 x
\]

That is, the whole learning space can be simplified to

\[
\alpha_1(\alpha_7 \varphi_6 + \alpha_1 \varphi_7) + \alpha_1(\varphi_6 \alpha_8(\alpha_8 + 1)\alpha_9 \alpha_{10} \alpha_{11} \alpha_{12} \varphi_8 + \varphi_7 \alpha_8 \alpha_8 \alpha_{14} \alpha_{15} \alpha_{16} \alpha_{11} \alpha_{17}(\alpha_1 \alpha_2 \alpha_3 \alpha_4)^\ast \varphi_7 \alpha_8 \varphi_9) + \alpha_2(\varphi_{10} + \varphi_{11})
\]

Personalisation also means satifying the intention of a particular learner. Let this intentions be formalised by the postcondition \( \psi \in B \). Furthermore, assume that the learning space is represented by some process expression \( p \in P \). Then the problem is to find a minimal process \( p' \in P \) such that \( p \psi = p' \psi \). In order to find such a \( p' \) we have to use again the application knowledge.

Example 4 Let us continue Example 1 and look at a user with intention \( \varphi_{10} \). Then we express application knowledge by the equations \( \varphi_{10} \varphi_{11} = 0, \varphi_{10} \varphi_9 = 0 \) and \( \varphi_6 \varphi_7 = 0 \).
Then we can simplify $p\varphi_{10}$ with the expression $p$ from Example 1 step by step. First we get $(\varphi_{10} + \varphi_{11})\varphi_{10} = \varphi_{10}$, which can then be used for

$$
\varphi_6 a_8 (a_8 + 1) a_9 a_{10} a_{11} a_{12} \varphi_8 \\
+ \varphi_7 a_8 a_8 a_{14} a_{15} a_{16} a_{11} a_{17} (\varphi_{12} a_{18} a_{19})^* \varphi_{12} a_{18} a_{19} \varphi_{10} = \\
\varphi_6 a_8 (a_8 + 1) a_9 a_{10} a_{11} a_{12} \varphi_8 \varphi_{10} \\
+ \varphi_7 a_8 a_8 a_{14} a_{15} a_{16} a_{11} a_{17} (\varphi_{12} a_{18} a_{19})^* \varphi_{12} a_{18} a_{19} \varphi_{10} = \\
\varphi_6 a_8 (a_8 + 1) a_9 a_{10} a_{11} a_{12} \varphi_8 \varphi_{10}
$$

Then finally we get

$$
(a_7 \varphi_6 + a_{13} \varphi_7) \varphi_6 a_8 (a_8 + 1) a_9 a_{10} a_{11} a_{12} \varphi_8 \varphi_{10} = \\
a_7 \varphi_6 \varphi_6 a_8 (a_8 + 1) a_9 a_{10} a_{11} a_{12} \varphi_8 \varphi_{10} \\
+ a_{13} \varphi_7 \varphi_6 a_8 (a_8 + 1) a_9 a_{10} a_{11} a_{12} \varphi_8 \varphi_{10} = \\
a_7 \varphi_6 a_8 (a_8 + 1) a_9 a_{10} a_{11} a_{12} \varphi_8 \varphi_{10}
$$

This means that the story space can be simplified to

$$
a_1 ((\varphi_0 a_2 + \varphi_1 a_3 (a_5 + 1) a_3 + \varphi_2 a_4 (a_6 + 1) a_4)^* \varphi_5) \\
a_7 \varphi_6 a_8 (a_8 + 1) a_9 a_{10} a_{11} a_{12} \varphi_8 a_{20} \varphi_{10}
$$

### 6.2. Splitting of Learning Units

The reduction to a subgraph obviously leads to an outline graph that is completely associated with the given learner type. However, it does not change any of the data content of the involved learning units. This problem is addressed by the splitting of learning units.

Roughly speaking, if we are given a learning unit in an outline graph that is associated with a given learner type, we may replace this unit by a whole outline graph such that all learning units in this new outline graph are associated with the given learner type. Furthermore, the combined data content of the units in this replacement outline graph should be equivalent to the data content of the original learning unit.

Formally, let $LT$ be some learner type, and let $G = (V, E)$ be an outline graph. Let $u \in V$ be a learning unit associated with $LT$. Now take another outline graph $G_u = (V_u, E_u)$ with the following properties:

- $V \cap V_u = \emptyset$, i.e. the two outline graphs have no common learning units;
- all learning units $v \in V_u$ are associated with the learner type $LT$;
The join of the data contents $C_v$ for all $v \in V_u$ is equivalent to the data content $C_u$;

- there is a distinguished starting learning unit $v_0 \in V_u$ such that all learning units $v \in V_u$ can be reached from $v_0$;

- there is a distinguished final learning unit $v_f \in V_u$, which can be reached from all learning units $v \in V_u$.

The first three properties express our intention to replace $u$ by the graph $G_u$ without losing data content. The last two properties are of technical nature to ensure that this replacement is formally correct.

So, we obtain a new outline graph $\bar{G} = (\bar{V}, \bar{E})$ with $\bar{V} = (V - \{u\}) \cup V_u$, i.e. we replace the learning unit $u$ by all learning units in $V_u$. Furthermore, a link $(u', u) \in E$ must be replaced by a link $(u', v_0) \in \bar{E}$ preserving all data types and actions associated with it. Finally, a link $(u, u') \in E$ has to be replaced by a link $(v_f, u') \in \bar{E}$ preserving all data types and actions associated with it.

However, if we want to preserve the learning unit $u$ for other learner types, we would first have to double it in $G$ ensuring that one copy is associated with $LT$, and the other copy with all other learner types.

The splitting of a learning unit requires a more detailed specification of the content of learning units. We will therefore postpone a more technical discussion of splitting to end of the Section 7.

7. Data Management

The content aspect of e-learning systems concerns the question: Which information should be provided? This is tightly coupled with the problem of designing an adequate database. However, the organisation of data that is presented to the user via the pages in an e-learning system differs significantly from the organisation of data in the database. We conclude that modelling the content has to be addressed on at least two levels: a logical level leading to databases, and a conceptual level leading to the content of pages. Both levels have to be linked together.

As pages correspond to learning units, an abstract description of such a learning unit would be a pair $(u, v)$, where $u$ is a URL and $v$ is a complex value based on the use of some type system. We decide to call such a pair a learning object. The term ‘object’ is used, because pairs constructed out of an abstract identifier and a complex value are called ‘objects’ in the context of object oriented databases.

Using classification abstraction as usual in data modelling, we obtain content types. In general, every data type can become a content type of
a learning unit. Thus, if $U$ denotes a learning unit, the learning objects of type $U$ are pairs $(u, v)$ consisting of a value $u$ of type URL and a value $v$ of the content type of $U$.

A little subtlety comes in here, which makes the definition still a bit more complicated. When a value of type URL appears inside the content value $v$, this may be a URL somewhere outside the web information system that we want to develop. However, in the case where the URL is an internal one, it will be the URL of another learning object, say of type $U'$.

Therefore, we extend content types in such a way that instead of the type URL we may also use links $\ell : U'$ with a unique link name $\ell$ and a name of a learning unit $U'$.

In order to obtain a learning object we would have to replace the links $\ell : U'$ by the data type URL first. However, the link $\ell : U'$ would force us to use only values of type URL that are URLs of the learning unit $U'$.

Learning objects support an individual learner and only provide a section of the data of the system. The question arises how the data can be described globally. In fact, we would need to combine all the content types and set up a database schema. Designing such a database schema underlies completely different quality criteria. For instance, for databases we would like to avoid redundancies as much as possible. We would also have to pay much attention to providing fast and concurrent access to the data.

Therefore, we use a separate level defined by database types. Thus, we obtain a description of the static components on two levels: the global or database level, and the local or information unit level. On both levels we employ the same type system.

Database types are more or less defined in the same way as the content types.

7.1. CONTENT DATABASES

In order to define underlying databases we could refer to any data model. Due to its convenience we prefer to adopt a variant of the Entity-Relationship model (Thalheim, 2000). According to this model we have database types on various levels $k \geq 0$. Usually, the types of level 0 are called entity types, whereas types on higher levels are called relationship types.

A database type of level $k$ has a name $E$ and consists of

- a set $\text{comp}(E) = \{r_1 : E_1, \ldots, r_n : E_n\}$ of components with pairwise different role names $r_i$ and names $E_i$ of database types,
− a set \( \text{attr}(E) = \{a_1, \ldots, a_m\} \) of attributes, each associated with a data type \( \text{type}(a_i) \), and

− and a key \( \text{id}(E) \subseteq \text{comp}(E) \cup \text{attr}(E) \), such that the database types \( E_i \in \mathcal{S} \) are all on levels lower than \( k \) with at least one database type of level exactly \( k-1 \). In the following we often write \( E = (\{r_1 : E_1, \ldots, r_n : E_n\}, \{a_1, \ldots, a_m\}, \text{id}(E)) \) to denote a database type. The first component in this triple is the component set \( \text{comp}(E) \). The second component is the attribute set \( \text{attr}(E) \). The third component is the key \( \text{id}(E) \).

We use this notation to associate two data types with each database type \( E \). These data types are called the \textit{associated data type} of \( E \) and the \textit{associated key type} of \( E \). These types are defined as follows:

− The \textit{associated data type} of \( E \), denoted as \( \text{type}(E) \), is
  \[
  (r_1 : \text{key-type}(E_1), \ldots, r_n : \text{key-type}(E_n),
  a_1 : \text{type}(a_1), \ldots, a_m : \text{type}(a_m))
  \]

− The \textit{associated key type} of \( E \), denoted as \( \text{key-type}(E) \), is defined analogously with the difference that only those \( r_i \) and \( a_j \) are considered that occur in \( \text{id}(E) \).

In particular, if we have a database type \( E \) of level 0, then \( \text{comp}(E) \) is the empty set. This implies that that there are no fields labelled by \( r_i \).

Let us now conclude the presentation of the global database by defining database schemata. A database schema is simply a collection of database types. Of course, if \( E_i \) is a component of a database type \( E \), and \( E \) is defined in the schema, then \( E_i \) must also be defined in the schema.

Formally, we can define a database schema as follows:

A \textit{database schema} \( \mathcal{S} \) is a set of database types satisfying the following condition: If \( E \in \mathcal{S} \) is a database type, then for all components \( r_i : E_i \in \text{comp}(E) \) we must also have \( E_i \in \mathcal{S} \).

We define the semantics of database schemata by the collection of possible \textit{databases} prescribed by the database schema. Thus, let \( \mathcal{S} \) be a database schema. For each database type \( E \in \mathcal{S} \) we have defined an associated data type \( \text{type}(E) \) and an associated key type \( \text{key-type}(E) \). As these two are indeed data types, they define fixed sets of values, which we call the set of \textit{objects of type} \( E \) and the set of \textit{keys of type} \( E \), respectively:

\[
\text{Obj}(E) = \text{dom}(\text{type}(E)) \quad \text{and} \quad \text{Key}(E) = \text{dom}(\text{key-type}(E))
\]
As the key id\((E)\) in a database type \(E\) is a subset of \(\text{comp}(E) \cup \text{attr}(E)\), each object of type \(E\) can be projected to a key of type \(E\). Let \(O_{\text{key-type}(E)}\) denote the projection of the object \(O \in \text{Obj}(E)\) to the value of its key.

We use the sets of objects and keys for the database types \(E \in \mathcal{S}\) to define a database over \(\mathcal{S}\) as follows:

A database \(db\) over \(\mathcal{S}\) assigns to each database type \(E \in \mathcal{S}\) a finite set \(db(E) \subseteq \text{Obj}(E)\) of objects of type \(E\) such that the following conditions are satisfied:

1. Key values are unique, i.e. there cannot be two different \(O_1, O_2 \in \text{db}(E)\) with \(O_1[\text{key-type}(E)] \neq O_2[\text{key-type}(E)]\).
2. Component values exist in the database, i.e. for each \(O \in \text{db}(E)\) and each \(r : E' \in \text{comp}(E)\) there must exist some \(O' \in \text{db}(E')\) such that \(r : O'[\text{key-type}(E')]\) is part of \(O\).

### 7.2. Learning Units

The core of a learning unit is defined by a view. A view \(V\) on a database schema \(\mathcal{S}\) consists of a view schema \(\mathcal{S}_V\) and a defining query \(q_V\), which transforms databases over \(\mathcal{S}\) into databases over \(\mathcal{S}_V\).

The underlying datamodel itself is not relevant. The defining query may be expressed in any suitable query language, e.g. query algebra, logic or an SQL-variant, provided that the queries are able to create links.

This leads to the definition of data unit based on some type system. The type system must provide base types and type constructors, e.g. record, set and list type constructors. Arbitrary type expressions are built by nesting these constructors.

A data unit has a name \(M\) and consists of a content data type \(\text{cont}(M)\), which is an extended type expression, in which the place of a base type may be occupied by a pair \(\ell : M'\) with a label \(\ell\) and the name \(M'\) of a data unit, and a defining query \(q_M\) such that \((\{t_M\}, q_M)\) defines a view. Here \(t_M\) is the type arising from \(\text{cont}(M)\) by substitution of URL for all pairs \(\ell : M'\).

In order to model functionality operations are added to data units. An operation on a data unit \(M\) consists of an operation signature, i.e., name, input-parameters and output-parameters, a selection type which is a supertype of \(\text{cont}(M)\), and a body which is defined via operations accessing the underlying database.

In order to allow the information content to be tailored to specific learner needs and presentation restrictions, data units are extended.
to learning units. The most relevant extension is *cohesion*, which introduces a controlled form of information loss. Formally, we define a partial order $\leq$ on content data types, which extends subtyping in a straightforward way such that references and superclasses are taken into consideration.

If $\text{cont}(M)$ is the content data type of a data unit $M$, then let $\text{sup}(\text{cont}(M))$ denote the set of all larger content expressions, i.e. all expressions $\text{exp}$ with $\text{cont}(M) \leq \text{exp}$.

A total pre-order $\preceq_M$ on $\text{sup}(\text{cont}(M))$ extending the order $\leq$ on content expressions is called an *cohesion pre-order*. Clearly, $\text{cont}(M)$ is minimal with respect to $\preceq_M$.

Small elements in $\text{sup}(\text{cont}(M))$ with respect to $\preceq_M$ define information to be kept together, if possible. An alternative to cohesion preorders is to use *proximity values*, but we will not consider them here.

A *learning unit* is a data unit $M$ extended by operations and a cohesion pre-order $\preceq_M$. There are other extensions beyond cohesion, but these are not relevant for context modelling.

If the defining query is evaluated for a learning unit $M$, we obtain a complex value $v$ of type $t_M$. Together with a generated URL $u$ this defines a learning object of type $M$.

Cohesion enables a controlled form of information decomposition, which can be exploited for the splitting of learning units as indicated in the previous Section 6. If we want to decompose a learning unit or if we are forced to decompose according to user requirements or technical restrictions, then we may choose a minimal elements $t_1 \in \text{sup}(\text{cont}(M))$ with respect to $\preceq_M$ such that it satisfies the representation requirements. Note that if we only provide a preorder, not an order, then there may be more than one such $t_1$.

Taking just $t_1$ instead of $\text{cont}(M)$ means that some information is lost, but this only refers to the first data transfer. When transferring $t_1$, we must include a link to a possible successor containing detailed information. In order to determine such a successor we can continue looking at all content data types $t' \in \text{sup}(\text{cont}(M))$ with $t_1 \not\preceq_M t'$. These are just those containing the complimentary information that was lost. Again we can choose a least type $t_2$ among these $t'$ with respect to $\preceq_M$ that requires not more than the available capacity. $t_2$ would be the desired successor.

Proceeding this way the whole communication is broken down into a sequence of suitable units $t_1, t_2, \ldots, t_n$ that together contain the information provided by the learning unit. Of course, the cohesion pre-order suggests that the relevance of the information decreases, while progressing with this sequence. The learner may decide at any time
that the level of detail provided by the sequence \( t_1, \ldots, t_i \) is already sufficient for his/her needs.

8. Conclusion

In this article we presented a conceptual view of web-based e-learning systems. Starting from a general abstraction layer model we addressed the modelling of courses, the modelling of learners, the personalisation of courses, and the management of data in e-learning systems. Courses are modelled by outline graphs, which are further refined by some form of process algebra. This process algebra actually gives rise to a many-sorted Kleene algebra with tests, which can be used to formally reason about the whole learning space.

The linguistic analysis of word fields referring to an application domain helps to set up these course outlines. Learners are modelled by classifying value combinations for their characteristic properties. Each learner type gives rise to intentions as well as rights and obligations in using a learning system. Intentions can be formalised as postconditions, while rights and obligations lead to deontic constraints. The intentions can be used for the personalisation of the learning system to a learner type. We showed that we can use equational reasoning in the Kleene algebra to reduce the learning space to a subgraph that represents the learning needs of a particular learner type.

We approached the management of data in an e-learning system on two different levels dealing with the content of individual learning units and the integrated content of the whole system, respectively. This leads to supporting databases and views defined on them. Furthermore, we demonstrated that the adaptivity feature of the learning units can be used to set up a simple algorithmic approach to personalisation based on the splitting of learning units. Our ideas have been partly realised in the DaMiT system (Binemann-Zdanowicz et al., 2003a; Jantke et al., 2004; Jantke et al., 2003; Rostanin et al., 2002), a system that addresses learning in the area of data mining.

We believe that the integrated conceptual nature of our approach and its coupling with rigorous mathematical foundations will help to improve e-learning system in a way that learner-driven learning on demand will be better supported. However, we do not claim that our methodology is a panacea for all problems related to e-learning. We acknowledge that not all of the educational discussion in the field has entered our work nor the work of others, as the technological challenges arising from these discussions have not yet found adequate answers. In
other words, we will continue our research in order to further improve our approach.

References


Feyer, T., Kao, O., Schewe, K.-D., and Thalheim, B. (2000). Design of data-intensive web-based information services. In Li, Q., Ozsuyoglu,


LOGISTICS FOR LEARNING OBJECTS

Bernd Tschiedel, Aleksander Binemann-Zdanowicz, Bernhard Thalheim
BTU Cottbus, LS Datenbanken Informationssysteme
{bernd.tschiedel | binemann | thalheim}@informatik.tu-cottbus.de

Abstract
An auspicious technology in the area of e-Learning is the assignment of learning objects. Nowadays, learning object is a common vocabulary in the meaning of content that has to work up to small independent content units which are connected with some semantic signification. A postulating advantage of learning objects is to improve the information quality of service, the provision of content. The aspects of information quality are described in the three main principles of information logistics. The correct content in the right place to the right time. This we have realized in the DaMiT Project. Rationale was the utilization of SiteLang, a methodology originally developed for specifying entire websites and enhanced for designing web-based learning systems.

Keywords: SiteLang, e-Learning websites, adaptive e-Learning, information logistics

1. Introduction
In many scientific papers and working group specifications the fundamental insight was outlined, that a high granularity of learning units leads to higher user-adaptive systems. One guideline is to deal with a lot of different types of units from the same topic. Users, in our case learners, just want best-suited content delivered just in time and to the right place and device. They should not be confronted with wasteful content. Therefore, some services are needed to fulfill the learner’s needs. In the information logistics one finds acceptable strategies which satisfy the requirements of a delivery of the right unit of information. The goals of information logistics are well-adoptable for the e-Learning challenge.

1.1 Types of Learning
A new hope is born, since e-Learning profits from the new technologies related to the internet and permanently increasing bandwidth. A hope with high demands on human and technical resources, which are involved in making the dream true to let everybody learn at any time, everywhere and whatever. But if we simply say "learn", we face a lot of problems and solutions coming from classroom learning, which are not easy adoptable for a computer-based learning system. Let us look at only one methodical aspect of learning and
teaching, the social aspect. In [Ott01] one can find organizational types which determine the interaction and communication possibilities.

*Frontal teaching:* a teacher presents a learning unit and works on it together with the pupils, students. *Single working place:* every pupil, student is self-responsible for planning, solving, checking problem solutions. *Classmate working:* exactly two pupils, students are responsible for their product of actions. *Cooperative, collaborative working:* a group (3-6, autonomous) works on the instructions and creates an own product of actions. It is to divide into common group tasks and separated group tasks.

Not only the obvious difficulty to model these social environments but also the classical problems like adaptivity, multi-usage of content, the necessary underlayer of developing social aspects are still in the scope of research. These natural characteristics of interactive working are controlled by a teacher, who evaluates and analyzes the needs and learning history of the pupils and basing on this knowledge he generates a learning scenario, which can be adapted at learn-time according to pupils’ progress and evolving needs.

### 1.2 Information logistics

Information logistics serves for providing optimized information to users [Lie01]. Optimized means that textually correct and needed information will be served at the right time and place. The information should always be transformed depending of user preferences and communication facilities. Derived from this definition [Lie01] explains some basic principles in information logistics which also concern e-Learning.

#### Several information sources:
The use of several content sources, in dependence of the actual user needs. The user looks for additional or extended information in distributed, correlated content bases. At the moment we restrict this to one database of one domain.

#### Information on the tick:
Information at the right time depends on the value of the information and the user context. The value depends on the deliverable content in the knowledge bases. High value demands a detailed description of the content. Users of an e-Learning system may interact with the system in different roles with special characteristics. This is the primary context for the user. Thus, on these conditions the system has to choose on demand or on click what the preferred content for the user is at this moment.

#### Consideration of user preferences:
Applications of information logistics must be able to satisfy individual needs of the users. User needs must be specified for optimal operation. This must be implemented in an explicit and an implicit way. Explicit data, like preferred presentation style or difficulty, must be treated as granularly as possible. Implicit data are the user’s history, current and recorded interaction and utilization behavior.

#### Flexibility of presentation:
Users have (depending on their working place and working context) different communication devices and channels. The system automatically recognizes the user’s system, his network connection and application. In any case, the presentation style must be adapted with respect to the individual working context.

The *working place and context* are features of information logistics. They do not belong to a web-based e-Learning environment, as they depend on the internet access. The described particularities are related to the challenges our department faced during the development of the SiteLang approach.
1.3 Challenges in e-Learning

While creating a computer-based learning application you face certain challenges which have already been investigated and used in some prototypes. We also have to restrict the mentioned aspects of information logistics from above in the following aspects.

Full flexibility: E-learning services require full flexibility of learning scenarios. The set of scenarios necessary to be supported looks graphically similar to a complete graph. It is possible in such cases to use any menu point and to jump to any other dialogue step [Cau00]. The usage of a proven scenario theory for the application development is a basic assumption to realize full flexibility.

Multiple usage of content: Authors of content must be able to index, research, reselect, recombine and update existing content. There is the need of a flexible data structure in order to comply with these demands. Thus, in the area of e-learning we use the approach of learning objects, based on the Reusable Learning Object Strategy [Cis01].

Adaptivity: Learners want to get content depending on their specific information requirements and demand. There are some general user profiles in the area of content format like text or formula-oriented learners. These can be used as pre-made scenarios with already associated learning objects. To realize user adaptation fully, learning objects should be enhanced at run-time with specific user information from the current user profile.

1.4 Overview of the Paper

This paper gives a short overview of the SiteLang methodology which is being used to develop parts of the DaMiT e-Learning system. Principles of information logistics are part of this design concept. The language has an operational semantics based on entity-relationship structuring and Abstract State Machines [Gur95]. It allows to specify entire websites by means of their structuring, behavior, information support, interaction and story space. The theoretical background of this work generalizes the approaches [FOS00] and [Tha00].

2. The SiteLang Methodology

SiteLang has been developed to enable developing interactive web services in a systematic way, allowing to verify the behavior of an information service before its actual implementation. SiteLang supports the stepwise development of a website according to the CoDesign abstraction layer model [FOS00]. It allows to specify database behavior and user interaction of the system in parallel.

The major goals achieved with SiteLang are the refineability of obtained SiteLang specifications and the possibility to execute and validate an abstract specification on any abstraction level. This has been achieved thanks to the operational semantics of Abstract State Machines [Gur95] SiteLang is based on. The development of analysis methods in the SiteLang methodology is also supported by the developments in the ASM community.

The SiteLang language comprises constructs for specifying database functionality (database schema, database operations, transaction management, integrity constraints, database content), as well as user interaction (event-driven interaction model, multiple users and devices, scenes, dialogue steps with trans-
action semantics, dialogues, media objects) [DuT01] in parallel. It allows to specify distributed system architectures. The user interaction model of SiteLang is event-driven; the event frame differs from the classical ECA model and comprises well-founded transaction semantics.

SiteLang has been successfully used for specifying interactive services of a set-top-box-based TV platform which has been developed by an industrial consortium in cooperation with our database group. The positive experience of SiteLang has proven its suitability for specifying data-intensive applications. It is also suitable for specifying and validating systems in which the interaction flow is reciprocally influenced by semantic content structures.

3. The Data Mining Tutor Project

The focus of the DaMiT project is the development of a computer-based tutor system to support learning and teaching in the area of Knowledge Discovery and Data Mining. Another important focus is put on the creation of content.

The main goal of DaMiT is to realize user adaptivity. Derived sub-goals are: content generation, creating coherent and consistent content, generating a semantic network using the domain Data Mining, integrating applications for on-line data mining, keeping an architecture open for extensibility and updating, as well as obtaining a knowledge basket adoptable to other projects.

4. SiteLang Usage in the Project

A SiteLang specification of DaMiT is obtained stepwise, starting with the Motivation Layer and the Requirements Acquisition Layer. The next step is the Business User Layer. It is the basis of further modeling and is then refined onto Conceptual Layer and the Implementation Layer. As an example we briefly present the user model and then focus on content modeling.

4.1 Modeling the User

For user adaptivity it is essential to analyze the most typical characteristics, operations and expectations of users who are to work and interact with the planned web service. Some of the characteristics are attributes and operations which form the adaptive interface of the user. This concerns the area of rights, educational classification and utilization behavior. So we defined at the BUL roles (Content Provider, Teacher, Administrator, Tutor, Learner with the sub-roles Anonymous, Pseudonymous, Standard, Manager), metadata and added
functionality, e.g. Payment, Rights Management (of courses, with respect to learner groups), as well as user-role-dependent interface and system functions. The transformation process towards the Conceptual Layer led us, through the (partially depicted) ER schema, to an appropriate relational database schema.

4.2 Modeling Content

Content and content-related user interaction is modeled in SiteLang on various abstraction levels. Content can be seen as a semantic structure with associated metadata. This semantic structure is modeled in SiteLang on the Business User Layer (BUL). In the stepwise refinement process of SiteLang, the BUL is mapped onto the Conceptual Layer, the semantic structure is mapped to an appropriate relational schema. On this layer, a more thorough validation of the system behavior is possible, as we deal with well-defined database structures and a ripe story specification. The Conceptual Layer itself can be then refined to the Implementation Layer, on which implementation-related details are added, e.g. architecture, information containers exchange etc.

4.2.1 Modeling on Business User Layer. The semantic structure of content is modeled on BUL by means of a content graph \( G = (V, E, t_v, v_v, v_e, p, \Sigma) \), with \( E \subseteq V \times V, v_v : V \rightarrow \Gamma, v_e : E \rightarrow 2^F, p : V \rightarrow 2^{\text{Properties}} \).

The sets \( V \) and \( E \) represent vertices and edges. The functions \( v_v, v_e \) determine version numbers of a node or a set of version numbers to an edge, respectively. \( \Gamma \) is just a set of consecutive version numbers. The function \( p \) assigns a set of properties to a node. The type functions \( t_v, t_e \), the constraint set \( \Sigma \) and the set \( \text{Properties} \) are application-dependent. A content graph is valid if all constraints from \( \Sigma \) are fulfilled. For modeling e-learning content we define:

\[
\begin{align*}
t_v : V &\rightarrow \{\text{content, unit, topic}\} \\
t_e : E \times \Gamma &\rightarrow \{\text{deleted, requires, is_required_by, see_also, has_part, is_part_of}, \ldots\}
\end{align*}
\]

\( \text{Properties} = \{\text{lecture, course, unit, group, multimedia, obj, text obj, applet obj}\} \).

The set \( \Sigma \) contains restrictions on the content graph:

\[
\begin{align*}
\alpha_0 &:= \forall (e, \gamma) \in \text{dom}(t_v) : \gamma \in v_v(e) \\
\alpha_1 &:= \forall v_1, v_2 \in V : (v_1, v_2) \in E \Rightarrow (t_v(v_1) = \text{topic} \land t_v(v_2) = \text{topic}) \\
\vee (t_v(v_1) = \text{topic} \land t_v(v_2) = \text{unit}) \lor (t_v(v_1) = \text{unit} \land t_v(v_2) = \text{content}) \\
\alpha_2 &:= \forall e \in E : [\exists \gamma \in \Gamma : t_e(e, \gamma) = \text{deleted} \Rightarrow \forall \gamma' \in v_e(e) : \gamma' > \gamma] \\
\alpha_3 &:= \forall v \in V : (\text{course} \in p(v) \Rightarrow (\forall v' \in V : E^+(v, v') \Rightarrow \text{course} \not\in p(v'))) \\
\end{align*}
\]

The content graph is similar to the concept of semantic networks. It is, however, particularly suitable for modeling content on BUL.

4.2.2 Modeling on Conceptual and Implementation Layers. The BUL is essential for modeling semantic structures of e-Learning content. It allows to validate system behavior before the actual database schema has been modeled. In order to simulate both database behavior and user interaction, BUL needs to be refined to Conceptual Layer.

For this purpose, the content graph is mapped to a database schema. Classes of objects of the same type in the content graph are mapped to relationship types,
Figure 2. Simplified content graph

links between objects are mapped to relationship types of a higher order. The resulting schema depends on the types of links between nodes on the BUL and of their semantics with respect to learning scenarios, as well as on e-learning content type and paramount manipulation operations performed on the content. As a result we get a complete relational database schema with the corresponding metadata and the user information.

User interaction is specified explicitly on Conceptual Layer. As learning scenarios are generated along links between content objects, topic and unit nodes from the BUL are transformed to scenes and dialogue steps. The navigation structure (dialogue step specification) results from the link structure of the respective node and of its parent nodes (topics). Navigation through content is realized as an execution of a series of scenes; navigation steps through a single unit by means of dialogue steps of a single scene. Content graph modifications at run-time result in scene changes on Conceptual Layer.

Having obtained the Conceptual Layer specification, we can refine it onto Implementation Layer, e.g. by including further implementation details and by adding constructs describing system distribution.

4.2.3 Modeling Metadata. Metadata in DaMiT is modeled according to commonly known concepts like in [IMS01]. It is modeled in SiteLang on the BUL directly by means of the HERM model [DuT01]. On Conceptual Layer, the metadata schema is then transformed to a corresponding relational schema.

4.3 Content Generation

E-Learning content in DaMiT is generated according to the user’s needs and is closely related to content adaptivity. The generation is done in twofold ways. Content-to-Profile Matching: the existing content structure is matched with the user’s preferences and history. The generated content is assembled from the matching topics and units, chosen from the content graph and is presented to the user as a lesson. It is being realized in our prototype by means of parameterized views, mappings of profile-dependent generation rules.

Semantic Content Evolving: the content structure is enriched by new subgraphs built on the basis of common learning objects. The new lessons can be more valuable to the learner, as they have new semantic relationships previously not present in the content graph; they are also reusable for later usage.
Representing content generation rules on BUL is subject to further research, so that SiteLang can support specifying the rules for both generation methods mentioned above.

4.4 Content Versioning with SiteLang

In the learning process, it is essential to provide every user with the a content version corresponding to his needs. The following needs to be considered:

Content Stability: After the user has begun a course, any changes on the course content must not be released to the user as long as he has not completed it. This intuitive condition cannot be realized in a trivial manner, i.e. just by freezing the course version, as it would cause an overhead when dealing with massive amounts of users.

History Continuance: It may be necessary to keep older content versions for later usage. Therefore, it must be possible to recall any older content version at any time.

Similarity Versioning: Depending on the user’s knowledge and usage history, it may be necessary to use parallelly different versions of the same content, which are assigned to the same unit. For instance, there may exist different difficulty versions of the same content, e.g. "basic", "intermediate", "advanced". Such parallel versions may also be subject to "classical" versioning in terms of updating content, e.g. for improvement purposes.

4.4.1 Content Modification Versioning. Course modifications, i.e. content modifications, are realized on the Business User Layer by integrating a new subgraph into the existing content graph, by deleting a subgraph or by updating a node. When a course modification is performed, it must not affect yet not completed interactions related to the respective course, in order to preserve Content Stability. Also, new interactions related to that course may be started after the modification: the new version should be applied to them. The following rules define how e-Learning content versioning is realized with SiteLang by means of BUL.

Course Versioning: Each course is versioned separately from the other ones.

Extending courses: Adding a subgraph into a course (into the content graph) increases the course version number. All vertices and edges of the subgraph also get the increased version number. The new edge leading from the existing graph into the added subgraph also receives the new version number.

Removing course parts: Removing a subgraph from a course increases the course version number and adds another edge of type deleted from the subgraph’s father to the subgraph.

Updating courses: Updating a node in a course (a topic, a unit or a content node) increases the version number of the course and creates a new node of the same type. This new node is connected with the same nodes and by new edges of the same type as the original node; the version number of the new edges is the increased version number.

4.4.2 Maintaining Parallel Unit Versions. Similarity Versioning is an important aspect of user adaptivity on e-Learning content, as it is necessary to provide users with content of appropriate difficulty level. Therefore, multiple units with similar content of varying difficulty are grouped in a topic which is specially marked with the property unit_group. In this way, a group of units on the BUL can be identified as "similar units". The transformation into the Conceptual Layer is analogous to the one of the entire content graph.
5. Summary

In our work we have presented a methodology for specifying information logistics for e-learning applications. It is based on the operational semantics of Abstract State Machines and has proven suitable for specifying content adaptivity in the DaMiT project.

Our further research will be focused on deeper aspects of adaptive content generation. Applying and enhancing the SiteLang methodology for an enhanced modeling of content structures and navigation rules - the new challenges that arose during the development - will be subject to further development as well.

Actually it is playtime with the DaMiT prototype. Users can login as learners, switching their educational preferences for adaptive content navigation to different levels of difficulty or presentation. The system is accessible to the public under http://neumann.dfki.uni-sb.de/damit.

Acknowledgments

The authors wish to thank the active participants of the DaMiT-Project Oleg Rostanin, Martin Memmel and Klaus P. Jantke for their encouragement and support, as well as other colleagues not mentioned here by name.

This research was partially supported by the DFG, Berlin-Brandenburg Graduate School in Distributed Information Systems (DFG grant no. GRK 316).

References


Institute of Computer Science, Cottbus, 2000


[Cis01] Cisco Systems Inc.: Reusable Learning Object Strategy, Nov. 2001, Ver. 4.0


[ScT00] K.-D. Schewe, B. Thalheim: Conceptual development of internet sites, Full-day tutorial, Conf. ER’2000, Salt Lake City, 2000

Towards Generative Engineering of Content-Intensive Applications
-Submission to PRISE 2004-

Aleksander Binemann-Zdanowicz
Kiel University
Computer Science and Applied Mathematics Institute
Olshausenstr. 40, 24098 Kiel, Germany
Email: binemann@is.informatik.uni-kiel.de

Keywords  behaviour anticipation, query generation, information retrieval, metadata, model management, software specifications

1 Introduction

Aspect-oriented software development has gained much importance in the recent years [22]. The term aspect means an abstraction localizing a cross-cutting concern of a software system. The idea of aspect-oriented programming is thus to develop the class structure of an object-oriented software system in parallel with the aspects containing the cross-cutting code [26]. In parallel, the idea of generative programming has been successfully pursued as a measure for specifying software features [15]. The comprised field of application domain analysis allows to specify so called feature models reflecting common characteristics of classes of the developed software as well as the dependencies between software features.

Along with the development of web-based applications the significance of persistence of information has been paid increasingly much attention. Software engineering concepts for supporting the interaction between software behaviour viewed from the user perspective and persistent data are sought [16,3]. Content-intensive systems require a sound specification of both classes and methods of the software itself (in object-oriented software development) as well as of the underlying database model. Moreover, the complexity of interaction between the user and the content-intensive system requires an anticipation of the system behaviour on both application and database sides [3,13,5,8,11].

Unfortunately, a generic architecture for content-intensive software systems is missing. Thus, the development of a web-based information system requires the development of classes and methods as well as of the underlying database model virtually from scratch. In order to tackle this problem, an approach to specifying the user-related software behaviour is needed. On the other hand the approach has to allow to obtain the specification of the underlying database structure upon the needs of the user.

Storyboarding techniques have proven a convenient measure for specifying software behaviour. Storyboarding is a notion developed originally in linguistics...
and in movie business [16,11]. The story of interaction between the user and the website is the 'intrigue' or 'plot' of the narrative work or an account of events. Stories can be played in different scenarios. A story space is a set of scenarios. A story space [16,13,5] can be modeled by means of a many-dimensional (multi-layered) graph:

\[
\text{Story-space}(\text{Site}) = (\{\text{scene}\}, E, \lambda, \kappa)
\]

transitions \( E \subseteq \{\text{scenes}\} \times \{\text{scenes}\} \)

\( \lambda : \{\text{scene}\} \rightarrow \text{sceneDescription} \)

\( \kappa : E \rightarrow \text{events} \times \text{ifCond} \times \text{postCond} \)

A scene is an abstraction that distinguishes a certain general activity. A scene is associated with a media object, with a set of involved actors, context of the scene, applicable representation styles and a dialogue step expression for the specification of the scene. A media object is (roughly speaking) a possibly complex object generated from the database. It corresponds to an extended view on the database [28]. Typical media objects are images, sounds, videos, texts, simple web pages and so on. The context of a scene is expressed by means of meta information regarding the media objects of the scene and its usage. A scenario is a run through a system by a set of actors [13,5]. A scenario can be sequential or cyclic.

A graphical representation of an example login scene developed in the scope of the Data Mining Tutor project [13,5] is given in Figure 1.

![Diagram](image)

**Figure 1.** Dialogue Scene for Login Into the DaMiT System

During the software engineering process, database views (i.e. media objects) required by the dialogues comprised by the storyboard specification have to be defined. In order to specify these views, a metadata description of the underlying
database content needs to be provided. In many cases existing database content has to be reused for the generation of such views [6,8]. Moreover, the generated views have to be specified differently for varying user characteristics [4,13,5,8,9].

Thus, queries constituting the media objects of scenes in a software behaviour specification have to be obtained dynamically upon the metadata description of the content of the underlying database. This way of view query specification has been known e.g. in the data warehousing area [19].

The natural way of query specification is to traverse the metadata structure [23]. During the traversal of metadata the query is specified with respect to its relevant dimensions. For instance, typical dimensions known from the data warehousing area are the spatial, factual and temporal dimensions [19]. Thus, one of the challenges to be addressed in the engineering of content-intensive software systems is the specification of content metadata in such a way that this traversal can be performed generically. Despite the variety of existing standards for metadata, an appropriate approach to metadata modeling as part of the software engineering process has been missing so far.

In this work we propose a formalized solution for obtaining metadata specifications. The respective aspects related to the utilization of content in the application, e.g. by means of the specific content-related functionality offered to the user, can be easily supported.

2 Example Application

The motivation example for our approach comes from the e-learning project DaMiT (Data Mining Tutor) [8,4]. It has been realized by 10 German universities as part of the German Research Society support program 'New Media in Education'. The aim of the project was to build an interactive Web-based e-learning platform for learning Data Mining technology. The system was to implement context-aware content generation and delivery, under consideration of the skills and abilities of the learner and of his usage history. Payment functionality for billing only the personalized and needed content was to be implemented as well. The developed solution should be generic and applicable to other e-learning content. An example session of the DaMiT is demonstrated in Figure 2.

The main identified issue was to provide appropriate content for different learner types, in order to achieve the goal of learning a certain topic. The 'appropriateness' of content was initially measured by means of presentation and degree of difficulty for the learner [4].

Beyond the initial project requirements it was then figured out that customized content should comprise not only appropriate presentation and difficulty matching learner's abilities, but should also be optimally structured with respect to the efficiency of the learning process for the particular learner. Thus, for achieving the same learning goal, different learning styles should be supported, e.g. learning by examples or learning by formal proofs. Moreover, for the sake of cost efficiency it should be possible to generate different scenarios off the same content objects. The following three example learner types were identified:
**Alf Smart** prefers formal thinking and systematic working. Finds examples not crucial, requires precise explanations instead.

**Tina Creativa** thinks more in pictures than in formulae. Prefers a joyful style of learning, with many possibilities to divert and come back to main topic later on. Example-oriented and not focused on details.

**Joe Hacker** is characterized by a high degree of pattern recognition ability. Prefers examples, finds them helpful to recognize entire structures. Does not require full proofs.

Automated content generation for such different learner types (off the same content objects) could not be realized using existing approaches [14,16]. Due to the different, strongly diverting attitudes to learning and to learning material, a unified good-for-all learning scenario cannot be created. Although the learning goal is the same, different paths leading to it through the content must be generated and followed. For instance, for Joe Hacker a learning scenario of type \( \text{SimpleExample} \rightarrow \text{ComplexExample} \rightarrow \text{ComplexExample2} \rightarrow \text{Theorem} \) is optimal, whilst the same scenario is inappropriate for Alf Naseweis, who rather prefers the scenario \( \text{Theorem} \rightarrow \text{Proof} \rightarrow \text{Example} \rightarrow \text{Corollary} \).

In order to support the different utilization scenarios for varying user types, a metadata specification is needed that allows to specify the content of the particular domain of interest, i.e. in this case the e-learning material on data mining.

Having specified a metadata schema, we realize the queries constituting the media objects of our storyboard (i.e. database views) still need to be generated by hand. In order to automatize this, a query generation mechanism must be introduced. As already mentioned, we obtain our queries by traversing the metadata structure in order to settle the parameters on relevant query dimensions [19]. However, the respective procedure is specific to a particular model of content metadata, in this case the metadata of the DaMiT system.

---

*Figure 2. Screenshot of the DaMiT System*
3 Towards Generative Content Specifications

In our approach we propose a well-founded specification method for content metadata models on the basis of category theory [2]. Classes of objects [17] are thus defined by means of categories. Functions applicable to objects of a metadata model are modeled as functors. A brief excerpt of an abstract metadata model looks in our approach as follows.¹

The sorts of objects we consider here are database types, attributes, and queries. Database types may be given or derived upon existing ones [29,28]. Attributes represent 'units of information' that may be of interest for the user. Attributes can be flat or nested. They correspond to labeled subtrees of database type definitions. Groups of attributes may be distinguished in order to stress their semantic relationships. Attributes and attribute sets together with respective integrity constraints can be seen as analogous to those of the relational model. Queries are logic programs that return value sets of corresponding types.

Let the categories $\mathcal{C}_{\text{TYPES}}$, $\mathcal{C}_{\text{ATTRS}}$, and $\mathcal{C}_{\text{QUERIES}}$ be introduced. In the following we will refer to the set of objects of a category $\mathcal{C}$ by $\text{Obj}(\mathcal{C})$ and to its set of morphisms by $\text{Mor}(\mathcal{C})$.

Objects of $\mathcal{C}_{\text{TYPES}}$ are database type definition terms [28]. Morphisms $f : B \leftarrow A$ are embeddings of type $A$ in $B$. The composition $g \circ f : C \leftarrow A$ of two morphisms $f : B \leftarrow A$, $g : C \leftarrow B$ is defined as an embedding of the embedded type $B$ in $C$. An identity morphism $id_A : A \leftarrow A$ is additionally defined for each object as an intrinsic embedding of a type in itself.

Objects of $\mathcal{C}_{\text{ATTRS}}$ are attributes, uniquely identified by name. Morphisms $f : B \leftarrow A$ are inclusion relations $A \subseteq B$ [21]. The composition of morphisms is straightforward. An identity morphism $id_A : A \leftarrow A$ is given by the reflexivity of $\subseteq$.

Objects of $\mathcal{C}_{\text{QUERIES}}$ are simple queries [29]. Morphisms $f : B \leftarrow A$ determine the construction of new queries by applying query algebra operations as in [29]. The composition $g \circ f : C \leftarrow A$ of two morphisms $f : B \leftarrow A$, $g : C \leftarrow B$ is defined as a superposition of the query construction. An identity morphism $id_A : A \leftarrow A$ is defined as $(A, A, \epsilon)$. The respective associativity and identity properties of $\circ$ are obvious.

We have already identified the categories of objects necessary to express content metadata information. The relationships holding between these categories are represented as functors [2].² Thus, we have defined a category that represents the relevant content metadata information. It is then mapped to a concrete conceptual model, e.g. Entity-Relationship. This is performed in a straightforward manner by implementing categories as star schemata [31] and by implementing functors as (updatable) views on these. This way we can benefit from having specified the abstract essence of metadata on a higher level.

¹ Due to the size limitations of this abstract we focus on content metadata in the strict sense here. This work is an excerpt of a larger framework embedding the approach discussed here, though.

² For the sake of brevity we omit the functor definition here.
The above approach allows to generate queries specifying the media objects of a storyboard by means of operations known from the category theory. Thus, traversing metadata is generically realizable by means of pushouts/pullbacks, and limits/colimits [2].

4 Summary

In this work a novel approach to representing content metadata for generative engineering of content-intensive applications has been proposed. It allows to capture differing versions of application-specific metadata specifications in a unified manner.

Our approach allows to specify content queries for storyboarding content-intensive applications. A content query is obtained by means of traversing the abstract metadata representation. The respective operations are realized by means of operations developed in the category theory.

This work is embedded in the current development of a general framework for specifying content-intensive applications. A larger proof-of-concept application of the approach is planned in the future.

References

17. Hesse, W., Mayr, H. C., Highlights of the SAMMOA Framework for Object Oriented Application Modelling, Proc. of 9 Int. Conf. on Database & Expert Systems Applications, Aug. 98, LNCS 1460, pp. 353-373
32. M. Yassen, B. Thalheim. Practical Database Design Methodologies. Kuwait University, Faculty of Science, 1989.
Adapting to Changing Times and Needs
UNESCO - SEAMEO Conference, Bangkok, May 27-29, 2004

CULTURALLY ADAPTIVE LEARNING OBJECTS: CHALLENGES FOR EDUCATORS AND DEVELOPERS

Authors:

Dr. Paul Nicholson *(contact author)*
Faculty of Education, Deakin University
221 Burwood Highway, Burwood Australia 3125
phone: +613 9244 6922
fax: +613 9244 6834
pauln@deakin.edu.au

Prof. Dr. Bernhard Thalheim
Institute of Computer Science and Applied Mathematics
University of Kiel, Olshausenstr. 40.
D-24098 Kiel, Germany
thalm@is.informatik.uni-kiel.de

Abstract
As globalisation and internationalisation increasingly impact on Education, there is an emerging awareness that there are significant economic benefits in adopting pre-designed (mainly western) ‘learning objects’ as the basis of curriculum design and implementation over developing locally created curricula and content. The potential dangers of such an approach include loss of cultural heritage and diversity, and consequent social dislocation, loss of intellectual capital, and forced alignment with externally imposed curricula. In this paper we argue that the major problem facing local courseware developers is that of representing cultural artefacts in their courseware. We discuss an emerging learning-object model that can be adopted to address this problem by providing a systematic way to approach implementing and representing cultural artefacts in courseware. This also provides a basis for some aspects of computer-based cultural preservation initiatives. We argue that while this approach may have higher short term development costs, the potential long term benefits to the society far outweigh short term considerations.
Culturally Adaptive Learning Objects: Challenges for Educators and Developers

Paul Nicholson
Deakin University, Australia
pauln@deakin.edu.au

Bernhard Thalheim
University of Kiel, Germany
thalheim@is.informatik.uni-kiel.de

Abstract
As globalisation and internationalisation increasingly impact on Education, there is an emerging awareness that there are significant economic benefits in adopting pre-designed (mainly western) ‘learning objects’ as the basis of curriculum design and implementation over developing locally created curricula and content. The potential dangers of such an approach include loss of cultural heritage and diversity, and consequent social dislocation, loss of intellectual capital, and forced alignment with externally imposed curricula. In this paper we argue that the major problem facing local courseware developers is that of representing cultural artefacts in their courseware. We discuss an emerging learning-object model that can be adopted to address this problem by providing a systematic way to approach implementing and representing cultural artefacts in courseware. This also provides a basis for some aspects of computer-based cultural preservation initiatives. We argue that while this approach may have higher short term development costs, the potential long term benefits to the society far outweigh short term considerations.

Background: prior concerns with ICT-based education
It has been conservatively estimated that there will be between 30 and 80 million online students in the world by 2025. Globally, but particularly in the West, universities, schools, and school systems are investing huge resources in the development of ubiquitous online education programs in order to either compete in the global education marketplace, or to achieve as yet unrealised productivity improvements and cost savings. The investment in classroom-based Information and Communications Technology (ICT) is presumably larger but has occurred over the past 40 years and has now assumed a mainstream role in many countries. The issues we discuss in this paper (other than specific aspects of online communications) apply equally to both classroom-based and online learning.

The wide-scale investment in ICT-based online courses (in particular) has been accompanied by questions about their nature, efficacy and quality. However, while investment in ICT-based education has increased, much of this has focused on infrastructure and little attention has been paid to developing appropriate pedagogical practices. The result has been the implementation of simple instructional design models and platforms such as Intelligent Tutoring Systems that often produce ‘…visually appealing and interactive screens behind which there is a shallow representation of content and pedagogy.’ (Lesh, Byrne, & White, 2003; Murray, 1999). This has been to the detriment of quality teaching, and has led to increasing concerns about the quality of online courses. (e.g., Gartner Inc, 2001; Oliver, 2001; Symonds, 2001). Concerns about the influence of quality assurance driven models on the structure and quality of these programs, and their ability to deliver meaningful pedagogically structured learning experiences, have also been raised (e.g., McGorry, 2003; Stone Wiske, Sick, & Wirsig, 2001; Suthers, Hundhausen, & Girardeau, 2003). Many online learners feel isolated from tutors and peers, suffer from a lack of tutor support, convenient and effective interaction, and a lack of strategies for actively involving them in their online learning. Many of these concerns also apply to face-to-face ICT-based classrooms: romantic notions of good teaching and charismatic pedagogy are not supported by research into the use of ICT in education. These concerns highlight the need to pay more attention to implementing effective teaching strategies and to meeting students’ learning needs.
The concerns about quality and pedagogical issues discussed above are largely embedded in a default western educational paradigm that assumes common educational and cultural norms (e.g., Grey, 2001) in which issues of culture are largely minimized (cf. Braak, 2000) or ignored other than in some attempts to ‘localize’ content so as to make it more relevant or interesting to specific situations (e.g., Bennett, 2000). This situation has arisen because of the greater and more extensive use of ICT in education in western countries over the past 40 years. Now that the West is, and always will be, the minority player on the Internet, there are increasing pressures to examine the underpinning cultural assumptions about ICT use in Education. These inherent limitations are increasingly being recognised in a number of exploratory studies that attempt to identify what might kinds of entities might need to be adapted for the successful adoption of ICT-based learning in a range of cultural settings (e.g., Brown & Hedberg, 2001; Leonard, 2000).

The problems associated with accommodating cultural considerations in face-to-face western classrooms (n.b., an idealised notion) are both complex and problematic, even at the level of basic classroom management where issues such as cultural diversity and culturally responsive pedagogies have to be addressed (Weinstein, Tomlinson-Clarke, & Curran, 2004). In many cases individual teachers are able to adapt, mediate and facilitate students’ learning in different cultural contexts, but the level of cultural adaptation within a particular classroom depends largely on the teacher’s multicultural competence, experience and insight into potential issues, problems and effective pedagogical approaches (cf. Buzzelli & Johnston, 2002). However, when wider socio-cultural issues such as class, colour, ethnicity, gender, language, race and religion are considered, the level of complexity in such classrooms increases enormously (Kurilof & Reichert, 2003). While many teachers find such classrooms difficult to manage, others thrive in the rich multicultural milieu. In general, it is fair to say that the response from course developers and content providers has been to largely ignore cultural issues in ICT-based learning contexts and to offer products, such as courses or ‘learning objects’ (computer-based curriculum materials packaged according to particular standard) that are based on the idealised western classroom.

As globalisation and internationalisation increasingly impact on Education, there is an emerging awareness that there are significant economic benefits in adopting pre-designed and tested learning objects as the basis of curriculum design and implementation over local content development. This is becoming increasingly obvious in the educational marketplace where we see the rapid proliferation of westernised learning objects that offer the promise of reusable, low cost, and tested materials (e.g., Nevile & Smith, 2003). The potential dangers of such an approach include loss of cultural heritage and diversity, and consequent social dislocation, loss of intellectual capital, and forced alignment with externally imposed curricula.

The contemporary educational fascination with learning objects is based in economic theory, not in Education. Learning objects do have the potential to deliver significant cost savings if their cost of development can be amortised over many clients. For example, imagine if every Chinese teacher used one learning object per week – that would be an enormous market with a high rate of return on the investment needed to create the object. There would also be a related cost reduction for those educational authorities who no longer needed to create and distribute the educational resource that the learning object has replaced. Ordonez argues that ‘Governments must swim against the current in the free market competition, avoid this ‘race to the bottom’ in providing incentives and cutting on government revenues, and ensure minimum government support for social safety nets, and for basic education to start with. (Ordonez, 2001, p.96) This cannot be overemphasised – the potential for western learning objects to dominate and subvert individual cultures is enormous. As a loose analogy, Ordonez reminds us of how the ubiquitous uptake of television has impacted on peoples’ expectations of media and ‘…lowered our tolerance for anything not spellbinding’. The contemporary
manifestations of this are evident in almost every western classroom, as well as in those of many other cultures. The relevance of his point for this paper is that many, perhaps most, locally produced educational content struggles to be as ‘glossy’ and vibrant as commercially produced materials – while it may, in fact, be far superior culturally and educationally. At what cost do we discard local educational and cultural knowledge and authenticity for globalised media products?

The challenge of culture

Culture is a complex multi-faceted construct with dimensions including class, colour, customs, ethnicity, gender, language, moral codes, power-distance, religion and social conventions (and many more). How these and other elements impact on educational practice – curricula, the nature and role of the teacher and the learner etc. — is ill defined and problematic as there is no one model of ‘culture’ that can be used to compare and relate such cultural elements to educational practice. Obviously there are major influences, such as religion, an historical connection to land, and politics, that can have significant recognisable and definable effects on education, but these are the exception rather than the norm; cultural pervasion of learning can be both overt and subtle. For designers of ICT-based learning materials, identifying and accommodating cultural issues for a specific cultural group seems to currently involve addressing political and religious sensitivities, using appropriate language, and targeting preferred learning styles. For example, whereas western students are assumed to cope with constructivist learning materials, Chinese students are presumed to prefer instructivist materials and are often assumed to adapt easily to traditional western courses.

Such responses do not address the real complexity of incorporating cultural considerations into learning environments in a holistic way, but rather serve as add-ons that are generally derived from western psychological models of learning styles and education. This should not be taken to mean that they are therefore not useful starting places — indeed they are! The problem with them is simply that they are not an integral part of a more inclusive and robust model of ‘culture’ that can be mapped onto entities such as learning objects – which is what is needed to systematically identify and address cultural considerations in ICT-based environments. Unfortunately there is no such single holistic model available for such use. It’s also idealistic to expect such a comprehensive model to exist at all!

Our central argument here is that there is a need for significant development to occur in this area if there is really going to be an attempt to seriously and systematically accommodate cultural elements in software, let alone redress the issues described previously. The following example attempts to suggest some reasons why this is so. China has recently made e-learning (both online and face-to-face) a national priority area in an attempt to address concerns about future access and equity issues. Hidden in this is the issue of how to meet the needs of China’s 56 ethnic groups – each with a distinct culture, and those of the majority Han people. A ‘simple’ first-order approach might be to develop a common content repository and to then provide 56 different language versions online. However, unless the intention is to deliver a common national curriculum, including values, attitudes, culture etc., the very notion of such a repository is problematic because of the diversity of culturally-situated knowledge of each ethnic group. This is a challenge that is far beyond educators, course developers or technical experts. Its eventual solution will require partnerships between centres of cultural expertise, education, politics, computer scientists and various technical folk.

The need for agile models

The social sciences inherently accommodate complexity and uncertainty in ways that are quite different from the physical sciences. This is evident in the kinds of models that each
produces, and how they get produced, with the social sciences embracing a more inclusive range of research paradigms and methodologies. In the context of this paper, Agile Models are particularly attractive for attempts to model cultural artefacts. These are models that ‘..fulfil their purpose, are understandable, and are sufficiently accurate’ and which are ‘...an abstraction that sufficiently describes one or more aspects of a problem, or a potential solution to a problem.’ (Ambler, 2001a, 2001b) As such, they allow for iterative development and improvement over time as researchers and developers increase their understandings of the artefacts they are modelling.

The really significant aspect of building culturally-adaptive ICT-based learning materials around agile models is exactly this issue — they embrace both a realistic perspective on the complexity of the task and allow for its ongoing improvement by all kinds of cultural and technical experts in common sense ways. This is arguably in stark contrast to many ICT developments where precise definitions and boundaries are locked in at the start of the development process which then simply implements those agreements. For cultural and educational academics and other experts, agile models facilitate both an entrée and a developmental pathway for them to work in partnership with technical personnel. Many of the issues discussed at the beginning of this paper have their origins in the failure of such a dialogue to occur. This apparently simple issue may well be one of great importance if wide-scale cultural adaptation is to occur in a meaningful, partnership-based model.

Tentative steps in designing for culture

In the following we give an example of how combining research from Computer Science and Education can provide more holistic understandings of the kinds of issues that need to be identified and modelled in developing culturally-adaptive ICT-based learning environments. This is not yet an example of an agile cultural model, but it will hopefully provide the underpinning technical framework around which such a model might be developed.

1. Educational perspectives:

When educators create classroom-based learning environments, they are generally focused on content, pedagogy and assessment. Somehow these seem to be diminished in ICT-based learning environments where the nature and use of the technology itself often dominates design processes and pedagogical aspects are minimised (cf. Cornu, 2001; Nicholson & White, 2001; Watson, 2001). This is often due to the large technical overheads involved in developing courses in commercial learning management systems (e.g., WebCT) ‘swamping’ technically inexperienced educators; it’s often seen as relief just to get basic course materials into the system. There is also the common problem that most of these commercial systems don’t have pedagogically-oriented design and development processes for use by educators in developing their online courses. In response to this and related issues, the Layered-design Model (Nicholson & White, 2001) was developed to help educators to focus on important learner-oriented aspects of course development by drawing their attention to the layers above the basic technical design. Figure 1 shows the basic components (layers) of this model.

Figure 1: Basic elements of the layered-design model. (Nicholson & White, 2001)

| Culturally-sensitive assessment layer |
| Learner Expertise Layer               |
| Interaction Layer                     |
| Basic Design Layer                    |
Each ‘layer’ represents a different set of design decisions and foci (see Table 1). By functionally separating design decisions into the categories of basic design, interaction, learner expertise and culturally sensitive assessment, we join technical considerations with their operationalisation in the classroom, decouple the ways in which learners interact with content, teachers and each other, and how they develop to their maximum potential in this environment. In this layers it is necessary to consider issues such as pedagogy, mentoring, scaffolding and coaching, all of which help to ensure a learner-centred, active learning environment is created on top of the basic technology. In previous works, we have articulated this new approach to designing learning environments that is particularly attractive for online learning (Nicholson, 2004; Nicholson & White, 2002).

Table 1: The layered-design model for E-Learning (Nicholson & White, 2001)

<table>
<thead>
<tr>
<th>Layer</th>
<th>What it focuses on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic design</td>
<td>Selection and implementation of technology; choosing the instructional design architecture; content selection or development; curriculum design.</td>
</tr>
<tr>
<td>Interaction</td>
<td>Identifying the models that will guide the ways in which users interact with students, staff &amp; content; process facilitation design—mentoring models, use of discussions etc.</td>
</tr>
<tr>
<td>Learner expertise</td>
<td>Elements that contribute to the learners’ optimal cognitive development.</td>
</tr>
<tr>
<td>Culturally sensitive assessment</td>
<td>Developing an assessment protocol that allows for a diverse range of cultural and educational responses.</td>
</tr>
</tbody>
</table>

The power of this model is not in its ability to prescribe a particular instructional design, but rather, to keep the instructional design process focusing on designing for learners. The focus on culturally sensitive assessment is currently a key element of our strategy to ensure that the whole design process does not revert to a traditional instructional design exercise, and that at a very basic level, we begin to address cultural issues. To add more extensive cultural elements (as discussed above) to this model requires that the items in each of the layers be interpreted in culturally appropriate ways. Of course this requires extensive cultural insights into the targeted culture, and even with such insight, this is not an easy task to do. This is a key area where educators and cultural experts need to have extensive ongoing partnerships. This challenge is even more daunting when it becomes obvious that this process also needs to be systematized and implemented in ways that can be meaningfully implemented and operationalised in software environments; scholarly, hand-crafted learning environments are too slow and expensive to create, and may not be readily scaled up in learning management systems. A fundamental problem is that educators often do not have the requisite technical knowledge to bridge the gap between their educational understandings, desires and intents for learning environments, and those of the technically savvy courseware developers. There is a urgent need to find ways to bridge this knowledge and epistemological divide.

In many ways, the layered-design model is a step in this direction as it can not only guide educators in their course development, but also forms a conceptual framework for a generalised learning object (at a course level) that contains higher-order entities such as scaffolding and coaching. This begins to have use for both educators and developers as it provides a reference framework that can be understood by both. However, there is still the issue of actually implementing the aspects of the model in software. A possible strategy to address this issue is to here is a real need for partnerships between educators and computer scientists to develop ways of modelling and implementing such ‘soft’ educational constructs as
coaching, mentoring and scaffolding in ways that current development systems, and educators, can make use of.

2. Technical perspectives
From a computer science perspective, creating a learning object that incorporates cultural elements is essentially the same as creating one without cultural elements – identifying, structuring and linking together the various entities in a structured database. However, the difference between them is that nature and complexity of cultural entities and their relationships with one another are vastly more complex than the typical instructional entities we find in contemporary learning objects. Technically, the challenge revolves largely around being able to model the higher-order entity-relationships that characterise many educational entities.

Developments in this direction have arisen from concerns about knowledge, information and learning environment models, some of which are similar to those discussed above, from within the computer science field (e.g., Binemann-Zdanowicz, Thalheim, & Tschiedel, 2004). Serendipitously, some of these developments can be mapped onto the layered-design model, creating the possibility of creating entity-relationship models of some educational constructs such as coaching, mentoring and scaffolding that might form the basis of further exploration of their potential to be used in creating culturally-adaptive versions. However, the complexity of this challenge should be underestimated, as the following example shows. Teachers often provide cognitive scaffolding (Vygotsky, 1962) to help students to overcome difficulties with specific content and ideas. This takes the form of short-term support strategies that will help the student to better understand the ideas in question. With the aid of this temporary support (hence the term ‘scaffold’), the student’s learning can progress. In addition to the teacher’s initial diagnosis of a problem, scaffolding consists of providing the learner with increased support in the form of tools (such as concept maps, computers, books), peer support, adult guidance, or particular information management and synthesis strategies. After this support has been given, and the student has begun to learn the material, the level of support is reduced (‘faded’) so that the learner again operates independently with this newly acquired knowledge. To do this well is difficult in a face-to-face classroom; to do this in an online environment, for example, is highly complex. It is even more complex to attempt to automate or systematize this in an LMS or learning object because so much of this simple concept revolves around the teacher’s ability to diagnose problems and to select and implement appropriate scaffolding mechanisms. The implications of trying to develop culturally-sensitive scaffolding models — assuming that this is a valid thing to do — require that we also address the complex cultural issues discussed above.

Conclusions
There is great potential for major advances in the development of culturally-adaptive learning environments and associated learning objects. Such advances will depend on targeted and well-funded partnerships between educators, course developers and computer scientists. While the expected development costs are high (but with low usage costs), their economic cost must be measured against the societal and socio-cultural benefits of adopting a culturally adaptive model.

Globally, minority groups are seeking recognition of their diversity at the same time as globalisation is threatening to homogenise large parts of the world’s populations. The issues raised in this paper provide a starting point for realising these and other kinds of cultural aspirations by providing a starting point for implementing a culturally sensitive development framework for E-Learning in accordance with UNESCO’s vision for the transformative use of technology in education.
References


Abstract. Edutainment web information systems must be adaptable to the user, to the content currently available, to the technical environment of the user, and to the skills, abilities and needs of the learner. This paper provides conceptions that support this kind of adaptivity and sophisticated support.

1 Introduction

1.1 Edutainment Web Information Systems

E-learning is now nearly as popular as e-commerce and e-business. The providers range from universities over non-profit organisations to professional training institutions. Furthermore, sites of museums, exhibitions, etc. can be counted as learning web information systems. In general, every data-intensive information system that is realised in a way that users can access it via web browsers will be called a web information system (WIS).

The major intention of a learning WIS should be to support learning. In the case of so-called edutainment systems the provided knowledge is usually easy to grasp. The message is that learning can be fun. The usage of a learning system depends on whether the control of the learning process is left to the user or the system. In both cases, however, it is assumed that the users are willing to learn and match the required prerequisites.

The content of a learning site depends on the area that is to be taught. Learning sessions are used as structural means, and navigation through these sessions may be organised in linear form or as a directed acyclic graph [ST05].

Systems may be completely passive allowing only material to be read or downloaded. Other systems may involve upload mechanisms for assessment of the learning progress. According to the progress made, a system may even provide feedback to its user. We concentrate our work on active learning.

The functionality of learning sites mainly supports the navigation through the site, i.e., the navigation through the learning material. In contrast to other systems, this navigation is a long-term progress with usually many interruptions. More sophisticated systems provide system-driven repetition and feedback. Apparently, the functionality of such systems is still a matter of research. Also, personal information needs can be supported by providing an interface to e-mail.
1.2 Achievements and Challenges of Edutainment WIS

A large number of websites provide learning and edutainment services has already been developed. Examples of technology-supported learning include computer-based training systems, interactive learning environments [DN03], intelligent computer-aided instruction systems [RK04, BGW92], distance learning systems, and collaborative learning environments. Developers have learned a number of lessons. Edutainment should neither be considered to be yet another presentation from classical instructionistic learning nor a means for presentation of content that has been used for lecturing.

Edutainment should not be understood as the enrichment of learning by multimedia facilities but must concentrate on content that is presented in a pleasing form, that is easy to use and to understand, that is enriched by functionality which corresponds to the content, and that allows to control the progress in learning.

Edutainment might support everybody to participate in learning activities on any place at any time within any team. Learning cultures will however limit this globalisation claim. Learner have their profile and their portfolio.

Main kind of learning stories are complementary learning, self-organised learning and continuing education on demand. Learner must thus be supported in developing their own learning space and in understanding and developing their learning abilities.

Controlling and assessment cannot be completely automatised. Each user is different from the other. The context differs. The learning history is different. At the same time, learners often need an immediate feedback.

Edutainment can be supported by a number of devices ranging from computers and PDA’s to mobile phones. We thus distinguish between electronic learning (e-learning) systems and mobile learning (m-learning) systems. Mobile phones and mobility in general are changing people’s way of working, communication and learning. The differences between formal learning, informal learning and way of working will diminish. The real added value of m-learning is in the area of informal learning whereas e-learning may cover formal and informal learning.

Scenarios and content are multifaceted, depend on the presentation device, and must be adaptable to the learner, to the learning community. The type of learning will be different from classical ones. The style of learning changes to pro-active formal learning with self study content, virtual classroom and trainer based facilitation. Learning is social, arguing, reflecting, articulating, and debating with others.

Majority of learning in the web is informal learning. It is based on ad-hoc information sharing and on communications and collaboration. The functionality of edutainment WIS is also based on functions such as participation, contribution, and annotation.

1.3 Scope of this Paper

This paper introduces some new conceptions we have already used in our edutainment WIS projects: DaMiT (a data mining tutoring workbench), KoPra (cooperative learning of database programming), and Learning Lusitia (continuing life-long learning for engineers and alumni). We extend the classical concept of learning objects [LOM00, LTS00] to open learning objects, show how scenario development [For94] leads to sophisticated support for didactics, and provide an insight into development of appropriate functionality. Additionally we show how brands of edutainment WIS, actor specification, learning scenarios, content and functionality can be developed for sophisticated edutainment WIS.
2 The Five General Characteristics of Edutainment WIS

2.1 The Brand of Edutainment WIS

The pattern or brand of the edutainment WIS $P^{W2U^A}$ (Provider $P$, knowledge $W$, user $U$, activities $A$) generalises the classical who-to-whom pattern (e.g. B2B). It is specialized for edutainment WIS:

Provider: Providers are currently mainly educational institutions or educational communities. In future, we will observe commercialisation of education. So, main providers are going to be companies. If a provider is a singleton person then this provider plays the role of the teacher.

Product dimension: Since control and assessment of learning progress is a still unsolved issue and appropriate presentation of complex information is often not feasible, edutainment WIS concentrate on easy-to-understand information or easy-to-grasp knowledge. The associated auxiliary scenarios are based on functions such as validate, control and advice. Therefore, the brand can be characterised by $P^{K2C^{learn},\text{validate},\text{control},\text{advice}}$.

User dimension: Users of edutainment WIS are mainly private people. They may be pupils or students, people seeking for continuing education, workers in companies with specific portfolio, or just people interested in auxiliary information. Users can be also groups. The main behavior of users is characterised by the role of the learner or of student.

Activity dimension: Activities are currently centered around learning, searching for content, collecting content, and solving exercises. Activities also include to ask questions, to act in teams for problem solving, and to discuss issues associated with the learning material.

Edutainment (learning) sites $(B^{K2C^{learn}}, C^{K2C^{learn}})$ are then specialized to the brands:

- Teacher $^{content\text{_chunks}}$ 2Student
- Teacher $^{knowledge}$ 2Student
- Teacher $^{knowledge}$ 2Student Group
- Teacher $^{Wisdom}$ 2Student

2.2 Actors in Edutainment

Edutainment WIS are currently mainly or exclusively supporting the pupil or student actor. The behaviour of actors might however be more complex:

- Pupils obtain knowledge through teachers, their schedules, and their abilities. They need guidance, motivation, and control.
- Collaborating or cooperating students act in a collaboration depending on a cooperation profile and rights and roles.
- Communication partners exchange content, discuss and resolve questions, seek for hints or for better understanding.
- Supporting and motivating partners are users with control, motivation and supporting functions.
Teachers act in various roles, obligations, rights, and in a variety of involvement.

The nature of the activities that constitute teaching depends more on the age of the persons being taught than on any other one thing. Users are different in their history. We distinguish between learner that initially seek to increase their knowledge and skills and users that are seeking continuing education. The first group of learners needs some guidance. Therefore, the learning style is based on pedagogic teacher-based learning. The latter are self-organised. We call then andragogic learners. The differences between the two learning styles are given in the following table:

<table>
<thead>
<tr>
<th>andragogic self-organized learning</th>
<th>pedagogic teacher-based learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>independent learner</td>
<td>dependent learner</td>
</tr>
<tr>
<td>self-regulated learning</td>
<td>not necessarily self-regulated learning</td>
</tr>
<tr>
<td>self-motivated learning</td>
<td>need to be motivated</td>
</tr>
<tr>
<td>reflective</td>
<td>needs help for learning</td>
</tr>
<tr>
<td>arguably</td>
<td>needs help for learning</td>
</tr>
<tr>
<td>analytical</td>
<td></td>
</tr>
<tr>
<td>situated in real world context</td>
<td>structured engaged with knowledge</td>
</tr>
</tbody>
</table>

2.3 Learning Scenarios and Stories

Modelling of e-learning scenarios is more difficult due to the variety of didactic approach, to the variety of learners, to the variety of tasks and the context of the site. For this reason, modelling of dialogue scenes and dialogue steps is more generic than modelling of simple interaction steps that usually lead to deterministic runs through the story space. We, thus, distinguish a number of associations among steps and scenes depending on the content of the learning element.

Our solution to this challenge is based on generic parameters that are instantiated depending on the learner, the history, the context etc. Each learning unit is specified by a context-free expression with a set of parameters. These parameters are instantiated depending on the learner profile, the learner task portfolio, the media objects, the learner computational environment, the data to be used in exercises or algorithms, the presentation environment, and the available and accessible learning elements.

Learning scenarios (learn) are classically based on general learning styles:

**Sequenced learning** is based on a curriculum sequencing in active or passive scenarios based on classical pedagogical approaches. Typical established pedagogical approaches are conditioning, operated learning, model learning, cognitive approaches. Sequencing is extensively studied based on classical didactics. Active scenarios model behaviour of active actors. A typical metaphor to be applied is the shopping bag. Passive scenarios did not get the success that has been expected during the 80ies and 90ies. Tutoring systems are applicable for advisory systems, for help desks, or for training physical skills.

**Interactive learning** is either based on self-organized or content-sequenced or habit-regulated or publish-subscribe scenarios.

**Group learning** is based on a cooperative setting (integrated sub-tasks, black boarding) or in a collaborative setting (cooperation, discussion, development of solution by all members).
Sequenced learning is currently mainly based on didactic approaches, i.e. on instructions, classical content and classical hermeneutics. Due to the sequential way dependencies among learning units are rather simple. Learning scenes use reception techniques, explication of the learning content, context association, deriving understanding and interpretation, and finally integrating the essence of the content into the knowledge of the learner.

### 2.4 Content in Edutainment

Content used in edutainment WIS is mainly easy-to-grasp and easy-to-understand information or knowledge. Most important associated scenarios are validate, control and advice. Content is given in a large variety. We need therefore to consider the kind of content, the activities that can be associated with the content, the characterisation and annotation of content, and finally the quality characteristics of content. Edutainment content delivery, storage, retrieval, and extraction is still an open research issue.

Edutainment WIS provide content to learners depending on their learning task, their personality, their working environment, their learning history and the policy of the content provider. This challenge is more complex than the challenge to generate “content”. Content can be represented by media objects. Content for learning must have high adaptivity. We distinguish a variety of content or generally knowledge:

- knowledge and abilities for orientation, e.g., explanation, presentation, history, facts, surveys, overview;
- knowledge and abilities for application, skills, abilities rules, procedures, principles, strategy, and laws;
- knowledge and abilities for explanation, e.g., ‘why’-knowledge (proof, causal) and ‘what’-knowledge (definition, description, argument, assumption, reflection);
- knowledge and abilities on sources, e.g. archives, documents, citation, reference, and links;
- knowledge and abilities for solving problems, e.g., sample solutions, analogs, training solutions, discovery solution, and examination.

All media objects can be provided independently of the learner. We need however to consider also differences. Background knowledge leads to different speed and reception by the learner. Work abilities and habits influence current work. The learning style must be considered in many facets. The social environment is based on cultural and psychological differences. The history of the learning process should be considered if we want to avoid annoying repetitions. The learning portfolio influences occasion, intention and motivation. The learning object presentation is or is not acceptable depending on the profile of the learner. The learning environment is modelled by many technical facets. Content change management allows to provide content with or without refresh. The payment profile may result in content reduction.

Learning objects are elements of a new type of computer-based instruction influenced by the object-oriented paradigm of computer science. Learning objects are defined here as any media object which can be used, re-used or referenced during technology supported learning. Examples of learning objects include multimedia content, instructional content, learning objectives, instructional software and software tools, and persons, organizations, or events referenced during technology supported learning. The IEEEs Learning Technology Standards
Committee has developed an internationally recognized definition of “learning object”: “any entity, digital or non-digital, that can be used, re-used, or referenced during technology supported learning”. This definition is extraordinarily broad.

A large variety of learning elements is used in edutainment WIS. Course elements are lecture notes and comments. Exercise material may be given in a textual or in an interactive form. Illustration material is often based on animations, complex multimedia elements or on links. Algorithms can be provided in an executable form, as virtual machine or via an interface to the server. Input data for algorithms can be provided with the learning elements or through other elements in the network.

2.5 General Functionality Based on Word Fields

During requirements capture WIS design and development we use natural language descriptions for analysing the activities of the users: What are these activities? Does the description indicate any sequencing or continuation? What data is needed for or used by the activities? What are the relationships between these data? As user activities can be described by verbs, we suggest analysing the corresponding word fields.

A word field [Kun92, SSS90, BZKK+05] is a linguistic system in which similar words that describe the “same” basic semen and are used in the same context are combined to a common structure and data set. In contrast to common synonym dictionary, word fields define the possible and necessary actors, the actions and the context. Word fields can be used for verbs, nouns and adjectives.

Functionality of edutainment WIS is also determined by the main word fields we observe for learning. Main word fields applied in learning are:

**Learn:** Learning is a very complex activity. It includes to gain knowledge or understanding of or skill in by study, instruction, or experience. Additionally, learning is associated with memorizing, to come to be able to perform some task, and to know this ability. Learning is based on obtaining content and discover the concepts behind. It is also based on facilities for annotation, ordering, and integrating. A user obtains the role of a learner or student. Learners are usually supported by other actors who teach and instruct. Learners determine content with certainty, usually by making an inquiry or other effort. They check the content, find out whether it is useful or they need additional content.

**Know:** Learning is based on skills, abilities, and knowledge. It target on their improvement. The improvement should be measurable. The learning success is then examined. To know means to be cognizant or aware of a fact or a specific piece of information and to possess knowledge or information about. It may include also to know how to do or perform something. The learner obtains firsthand knowledge of states, situations, emotions, or sensations. The change in knowledge is acknowledged and recognized by other actors who can accept to be what is claimed. The word field know used in learning is different from one used in identity or information WIS.

**Master:** Learning is usually intending to master problems and to become completely proficient or skilled in an area. This mastership is closely related to practise and experiment with the new knowledge. The learner has a firm understanding or knowledge of and is on top of of a problem.
Skill: Skills are abilities that have been acquired by training, e.g. abilities to produce solutions in some problem. A user acting as a learner is possibly trained until he/she obtained these skills.

Study: The learner is engage in study and undertakes formal study of a subject. Studying requires an endeavor and a try. In learning WIS studying is based on read in detail especially with the intention of learning. Therefore the presentation of the material and the storyboard is an essential part. Studying does not only mean to view content but to check over, to check up, to con, to examine, to inspect, and to survey it. This activity is performed attentively and in detail. Learners need to mind, to perpend, to think (out or over), and to weigh. Therefore, users need time and workplaces. Studying can be performed by oneself without any teacher, supporter, or observer. Studying is often based on the existence of a specific workplace and of a specific workspace. Learners may thus have a study place and workplaces which are given to them, which might be rented and which can be exported to the user.

We observe that these word fields have a more complex structure. They lead to functions which require support functions. The difference to the word fields discussed for business WIS is their iterative application. This kind of behaviour has already been observed for community services. Additionally, learning can be performed within temporal communities. Other related word fields are discover, ascertain, catch on, determine, teach, educate, judge, evaluate, advise, innovate and discuss.

Learning word fields can be combined and thus form a didactic story. Learning word fields are going to be combined with reasoning word fields such as analogise, analyse, cause, classify, conjecture, counter-example, formalise, generalise, sharpen, specialise, unknown, and weaken. General activity word fields that appear in any story are folded into our learning word fields. Typical general activities are characterised by answer, comment, compose, decompose, define, draw, effect, example, extend, fact, inquire, instantiate, and reduce.

3 Novel Concepts of the Edutainment WIS DaMiT, KoPra, and Learning Lausitia

The DaMiT system [JMR+03] supports sequenced and interactive learning. Interactive learning is still an open research issue. We have developed and extensively used an implementation in the KoPra project [SBZ04]. Learners act in a collaborative setting depending on their needs and the goals of the learning program in the project Learning Lausitia.

3.1 Supporting Didactics by SiteLang

The website description language SiteLang has been introduced in [TD01]. A scenario is an application-oriented view of parts of the storyboard. The storyboard consists of a set of scenarios, one of which is the main scenario, whereas the others define scenes, a plot specified by a SiteLang process, a set of roles, a set of user types, a set of tasks each associated with a goal, and a set of constraints comprising deontic constraints for the rights and obligations of roles, preference rules for user types, and other dependencies on the plot. We may distinguish four main kinds of general scenarios for the three learning styles:

Content scenarios are centered around the content chunks or content suites to be delivered to the learner, to be received by the learner, and to be integrated into his/her knowledge.
Control scenario are based on assessment or control of the success of learning. They involve at least two different actors: learners and controllers. A third actor involved may be the advisor.

Workspace scenario are auxiliary scenario that help the learner to organise the learning material, that enhance the learner content space with memos, with excerpts, own solutions, and collaboration notes.

Collaboration scenario support the learner to learn in groups, to communicate with their partners and with other actors such as teachers and advisors, to coordinate activities and to cooperate during solution of exercises, content reception and development of solutions.

These four general scenarios can be seen as relatively independent scenarios that can be combined with each other depending in the learning style. We develop the story space by the didactic scenario quadruple in Figure 1 on the basis of relatively independent scenarios and in the second step through their combination.

Teaching: Teaching is a complex activity that goes beyond instruction, that includes a process of formal training, that is based a body of specialized knowledge, and satisfies a set of standards of performance intellectual, practical, and ethical. Teaching intends to learner to know something, to know how to accustom to some action or attitude, and to know the consequences of some action.

Educating: Education is similar to teaching but more intention based. It includes training by formal instruction and supervised practice especially in a skill, trade, or profession. It targets to persuade or condition to feel, believe, or act in a desired way. Learners are mentally, morally, or aesthetically educated especially by instruction.

Judging and evaluating: Edutainment also is often oriented to form an opinion about through some weighing of evidence and testing of premises and to decide a matter as a judge. Learner are trained in determining or fixing the value of a matter, in determining the significance, worth, or condition of usually by some kind of appraisal and study.

Advising and discussing: Advising and discussion scenarios are similar to those we have already considered for community WIS. Advising means to be able to use the background
knowledge of the learner or the knowledge given by the content chunks for generation of advices or information or for consulting. Discussion techniques are applied in edutainment whenever we need to investigate by reasoning or to argument or to discourse about in order to reach conclusions or to convince. Discussion is an exploration technique that implies a sifting of possibilities especially by presenting considerations pro and con.

**Innovating:** Innovation introduces a new idea, new content or a new activity in order to effect a change.

These different learning scenarios can be compiled or integrated within the story space. We use for integration, ordering and hierarchical presentation of scenarios the theory of KAT’s. This combination is necessary whenever we need to consider different didactic approaches:

- **Critical-constructive didactics** see learning through interaction via goals.
- **Learn-theoretical didactics** consider learning through dialogues between actors.
- **Goal-oriented learning** is based on a separation of the intention space into subspaces with sub-processes.
- **Cybernetical didactics** uses regulated processes for story development.
- **Critical-constructive didactics** are based on interactions, repetitions, and obstruction.
- **Curriculum planing** extends classical sequenced or blended learning by schedules, goals, and steps.

### 3.2 Storyboard Pattern

Scenario typically proceed in one or (usually) more scenes. Scenario are describing the ways how the work is performed and are based on didactics, goals, user purposes, and content. A storyboard contributes to achieving a purpose or goals.

Scenes are composed of other scenes. We assume that scenes can be hierarchically formed based on basic scenes. Typical basic scenes are the information seeking scene, the collaboration scene, the assessment scene, the result integration scene and the problem solution scene. Scenes can be composed with another scene based on sequential, parallel, alternative etc. composition operators.

It seems to be obvious that these scenes have a common structure and a common behaviour. We thus can extract general stories or scenarios. These general stories use general scenes or scene pattern. Patterns represent recurring solutions to software development problems within a particular context. These scene pattern are refined to the concrete scenes.

Let us consider a typical scene pattern that involves a number of learners. Problem solution scenes are one of the most complex scenes in edutainment WIS. They typically consist of an orientation or review subscene, of a problem solution scene performed in teams, and of a finalisation subscene. Problem solution scenes are composed of a number of subscenes that can be classified into:

- Review of the state-of-affairs: The state of the problem solution is reviewed, evaluated, and analysed. Obligations are derived. Open problem tasks can be closed, rephrased or prioritised.
• Study of documents and resources: Available documents and resources are checked whether they are available, adequate and relevant for the current scene, and form a basis for the successful completion of the scene.

• Discussions and elicitation with other partners: Discussions may be informal, interview-based, or systematic. The result of such discussions is measured by some quality criteria such as trust or confidence. They are spread among the partners with some intention such as asking for revision, confirmation, or extension of the discussion.

• Recording and documentation of solutions: The result of the scene is usually recorded in one solution proposal.

• Classification of solutions, requirements, results: Each result developed is briefly examined individually and in dependence of other results from which it depends and to which it has an impact.

• Review of the problem solution process: Once the result to be achieved is going to be recorded the solution is examined whether is has the necessary and sufficient quality, whether is must be revised, updated or rejected, whether there are conflicts, inconsistencies or incompleteness or whether more may be needed. If the evaluation results in requiring additional scenes or subscenes then the scene or the subscene is going to be extended by them.

This classification is based on the general problem solution framework discussed in [PP45]. Problem solution scenes are usually iterative and thus cyclic as displayed in Figure 2. The general frame to problem solution is then based on the problem solution or search problem:

Problem characterisation with abstracting from non-essential parts;

Context injection for simplification of the problem and of the solution;
Tools and instruments for solution based on constructors, associations, collections, and classification;

Specification/prescription/description as the results of the problem solution problem.

Analysis and solution formation and problem solution are two specific activities that depend on each other. Development aims in forming solutions as well as discovering inconsistencies and conflicts and resolving incompleteness. Solution formation is based on the mapping of properties, requirements or phenomenons into WIS solutions. Solutions may have alternative solutions. These alternative solutions can be used at a later scenes instead of the given one. Typical techniques for problem solution and formation are exploration and experimentation, skeptical evaluation, conjecturing and refuting. Additional techniques are investigative ones that use resources.

In the evaluation phase we show whether the problem solution result is correct, and if so what kind of tests etc. ought to be devised. We do not require completeness since this is a relative issue.

Solution validation is the process of inspection of each solution with reference to solutions it is based on. It is based on the application area description and checks whether the right solutions are developed. Validation produces a report that constitutes the correctness and that produces the extensions and updates necessary for the base solutions.

Verification analyses solutions in order to ascertain whether what has been developed satisfies a number of obliged properties. It checks whether the solutions developed so far are correct according to some correctness criteria and according to proof or verification techniques that are currently applicable. Techniques may be informal, i.e., based on verbal arguments or tests, qualitative, i.e. based on abstraction and qualitative reasoning, or formal, i.e. based on model checking and proof techniques.

Each of the subscenes can be refined depending on the application, the user, the content, collaboration and the control:

User refinement: Each user has a personal profile and a task portfolio. The website should only use those dialogue scenes the learner is assigned to.

Learning objects refinement: Learning objects are under constant change. Whether this change is shown to the user depends on the user profile. However, in general refinement to available content is generated for the user.

Learning objects refinement: Learning objects have prerequisites, support learning goals, and are associated to other learning objects. These associations must be made available by the system.

Scenario refinement: The edutainment WIS also supports an adaption of the entire story currently requested by the user depending on the user and the completion of tasks.

Usage refinement: Users are annoyed whenever they did not complete a task and they must begin from scratch after resuming. For this reason, the edutainment WIS must support also refinement to the current usage.

It is obvious that such refinement cannot be generated by random application of refinement rules. We observe however that this can be layered. The approach to layering used in the
system is displayed in Figure 3. We generate first the learning scene together with its learning objects for each of the dialogue scenes. Next we extend the set of learning objects by all associated objects. This set of learning objects cannot be delivered to the learner in any order, e.g. prerequisites must be shown first. Some of the learning objects may be shown in any order, i.e., in parallel. Next we filter this set against the storyboard and generate now learning object sequences depending on the scenes. Now we are able to take into consideration the user refinement such as the technical equipment, the channel information such as capacity, the web browser currently in use. Based on this information, we can adapt the website to the current refinement. Finally, we may now enhance the scenes to the specific demands.

3.3 Open Learning Units for Content

Classically learning objects are composed elements. Comparing the existing standards such as LOM, SCORM we extract the following basic units:

**Learning elements** are basic components providing the content for singleton learning steps. Typical learning elements are definitions, remarks, proofs, lemmata, illustrations, motivational remarks etc. Learning elements may be associated with learning intentions and require basic skills and knowledge from the learner. These requirements form the context of learning elements. Semantics context specifies the prerequisites and the pieces of knowledge that can be learned. Execution context restricts the utilisation of learning elements, e.g., the environment.

**Learning modules** are the main supporting media objects for lectures. They may be enhanced by indexing, annotation or search functionality. Actors may only call an entire module. Modules may consist of learning elements. Typical composition expressions are regular expressions. The utilisation of learning modules may be based on a number of execution styles such as blackboard execution. Learning modules are typically not materialised.

This distinction is too rough for practical usage. Learner may stop learning in any module, may resume learning at a later stage and may define their own way of visiting modules. We thus need a sophisticated, flexible and adaptable mechanism for structuring and compositions of learning modules. We distinguish between learning elements that are simply specific media objects with an additional characterisation, and learning units that can be understood
as combined learning elements, which are dependent on the learning element and thus only updateable through the learning element.

Learning elements can be understood as simple media objects such as text elements, video clips, images etc. Learning units are composed of learning elements, cf. Figure 4.

For composition of learning units we extend the specification by metadata, additional functionality, specific scenarios, specific representation styles, filters for playout, a characterisation of learning units spaces that can be associated, and context. The onion playout facility based on XSL rules [TD01] supports the generation of the right content, at the right time, in the right representation, by the right costs, and within the right learning history for each learner.

Learning elements and learning units are commonly characterised by

- a name of the unit,
- a general annotation called header content,
- metadata that provide additional information on the unit, and
- associations among the media objects.

Additionally, learning units are characterised by

- expression hierarchically combining learning elements or learning units into the given one,
- reusability conditions describing the degree of ease with which constituent media objects may be individually accessed and reused,
- common function describing the manner in which the unit is generally used,
- extra-object dependence describing whether the unit needs information (such as location on the network) about learning units other than itself,
- functions of algorithms and procedures within the media object,
• potential for inter-contextual reuse describing the number of different instructional contexts in which the learning unit may be used, that is, the unit’s potential for reuse in different content areas or domains, and

• potential for intra-contextual reuse describing the number of times the unit may be used within the same content area or domain.

The above discussed theory of learning elements, learning units and learning modules is now becoming the state-of-the-art in most edutainment WIS. We are additionally interested in support of learning scenarios which are adaptable in most of the forms discussed above. In order to cope with such requirements we introduce the concept of open learning units. These objects are learning units extended by

parameters instantiable by information used in general learning modules such as prerequisites, supporting knowledge for better understanding of the unit, and associated units,

parameters for links to other content elements, comments,

parameters for replacements strategies depending on availability of content objects,

parameters for associating units to ontology objects for flexible classification of learning objects and bottom-up collection of learning scenarios,

parameters for learner profile integration through which the open learning unit can be replaced by objects that are more appropriate for the current learner type,

parameters keeping track on the learning history and which can be used for adaptation of the unit to the learner’s history, and

parameters used to keep track on the payment profile of the learner.

Open learning units can be specialized to learning units by instantiating all parameters. In this process, user-specific learning units are generated by step-wise instantiation, extension and specialization of the given open learning unit. This procedure is based on rules which can be specified as attribute grammar rules and which can be transformed to XSL rules. These XSL rules are used to transform the given learning unit that is given as XML document to more specific XML documents. We need, however, to clarify whether an arbitrary specialization order is applicable.

Furthermore, the representation style can be added. This adaptation approach has been extensively discussed in [TD01]. Since we change step-by-step the learning unit to be transferred to the learner we call this generation approach the onion generation style.

The delivery of open learning units is based on container functions that support derived learning units by filters which support

• enrichment of each unit by other learning elements or learning units,

• contraction of units to essential material, e.g., for repetition of units already visited, and

• cut of learning units to new units according to the restrictions that are applicable.
This approach has already been used in the DaMiT system. Since DaMiT was centered around data mining material and has been mathematically oriented, a number of specific filters have been developed:

- Repetition filters contracted the units to essential definitions and to the informal statements of theorems in the unit.
- Definitions filters provided the definitions of the together with all associated definitions used in the statements and theorems of the unit.
- Examples filter compiled examples in the unit to consecutive examples with references to statements and definitions.
- Quick reminder filter are extracting those elements of the units that are marked as absolutely essential.
- Proposition-and-theorems filters summarised all theorems and statement of a unit together with sketches or ideas of their proofs.

3.4 Enhancements of Edutainment Stories

Edutainment WIS authoring is considered to be one of the most difficult tasks. This difficulty is increased by the current approach to develop content and stories from scratch. Classroom teaching does not follow this approach. It is mainly based on sequenced or curriculum-ruled learning approaches. These approaches are easier to use due to the high level of reuse, due to the high-level of similarity within the teaching scenario, and due to the homogeneity of teams after the class has been formed. The last advantage cannot be used in edutainment WIS since the auditory is very heterogeneous, is changing over time, does not follow any timed schedules, has a brought variety of preliminary knowledge and is self-organised. The first two advantages may however be incorporated into edutainment as well.

Meanwhile it is well acknowledged that general purpose, general content and general storyboard systems are not feasible. Since any area of knowledge has its specific approaches to learn that knowledge we need specific storyboards for edutainment systems. General purpose learning systems are replaced by specific purpose systems, e.g. learning application of certain knowledge. Any content needs its specific representation. Therefore, we must specialize from one hand side and must be very general for any kind of story required. We shall show that our approach supports these requirements.

Authors of content for edutainment may base their storyboard on general “pattern” of scenario that might be useful for the given topic. This general scenario can be then refined by the author. We are interested in such scenario that can be composed from given scenarios. So, the author selects first a general scenario or a storyboard. Next he/she uses pattern for refinement of their scenes. Educational material is assigned to scenes based on the storyboard. This material is modelled by media types [ST04]. Finally, collaboration, control and workspace scenario are folded into the developed scenario.

Let us demonstrate the integration based on the experience we gained in one of our edutainment WIS project. Data mining is one of the challenging topics. First, users must learn in a rather interactive form. Second, the outcome of the learning process must be evaluated and interpreted. Third, the data used for data mining are often private or secured data. Therefore,
the user should not transfer the data. Instead the user must understand how to prepare for the data mining process. Additionally the demand for data mining appears as a part of the daily business of users.

In a large project integrating research groups in ten German universities and application groups in half-dozen German software companies we developed the edutainment system DaMiT (data mining tutoring) that educates users to such extent that they can evaluate whether the results of decision support systems are generating correct results and whether results will lead to correct conclusions. The system supports decision learning

- by learning basic and advanced topics on data mining on demand,
- by applying data mining algorithms to test data for training of users, and
- by finally applying these data mining algorithms to data of the application engineer.

Surprises, data warehousing, and complex applications often require sophisticated data analysis. The most common approach to data analysis is to use data mining software or reasoning systems based on artificial intelligence. These applications allow to analyse data based on the data on hand. At the same time data are often observational or sequenced data, noisy data, null-valued data, incomplete data, of wrong granularity, of wrong precision, of inappropriate type or coding, etc. Therefore, brute-force application of analysis algorithms leads to wrong results, to losses of semantics, to misunderstandings etc.

The storyboard in Figure 2 gives only a general description of the main part of the data mining storyboard. It must be enhanced to become a complete story for learning data mining. We thus need general frameworks for data analysis beyond the framework used for data mining.

The enhancement procedure may be based on general story descriptions. In our application approaches known from mathematics for the general mathematical problem solving have been used for derivation of a data analysis framework:

**Elaboration of needs:** Modelling of the tasks and problems and their data requirements.

**Elaboration of opportunities:** Selection of possible appropriate analysis algorithms, categorisation of their outcome and pitfalls within the task and problem scope, development of an application frame for application of the chosen algorithms.

**Extraction, transformation, and loading of data:** Categorisation, extraction of macro- and meta-data, adaption of the data to the analysis needs and modelling of data semantics and pragmatics.

**Problem solution:** Extraction, transformation and loading of macro-data for the chosen analysis algorithms, including cleansing and adaption of the data.

**Refinement of problem solution:** Application, stepwise refinement and correction of the analysis algorithms.

**Interpretation of analysis:** Modelling of the obtained analysis results with their semantics and pragmatics.
The general data analysis framework has been developed for data mining tasks. It considers modelling of data analysis algorithms, description of their requirements to data, meta-data of their functionality for analysis and transformations of the data.

This general framework can be now refined in a number of ways. A typical refinement is displayed in Figure 5. Users learn the opportunities of data mining based on case studies. They explore good and bad studies, get an explanation based on the theory background, and become familiar with modelling techniques.

![Figure 5: The scenes for elaboration of opportunities for active learning in the DaMiT system](image)

The framework is classically enhanced by scenario that support training of users to data mining based on exercises, examples, case studies etc.

3.5 **Context Space Enhancements**

When determining context we already know the edutainment scenarios we would like to support, the intentions associated with the WIS, the user and learner characterisation on the basis of profiles and portfolios, and the technical environment we are going to use. These restrictions enable a more refined understanding of context within a WIS.

[MST05] characterises a WIS by six intertwined dimensions: the intentions, the usage, the content, the functionality, the context, and the presentation. We must thus relate context to the other dimensions. As presentation resides on a lower level of abstraction, it does not have an impact on context. Content and functionality will be used for context refinement. The user model, the specified edutainment scenarios, and the intention can be used for a disambiguation of the meaning and an injection of context. In doing so we distinguish the following facets of context:

**Learner context:** The WIS is used by learners for a number of tasks in a variety of involvements and well understood collaboration. These learners impose their quality requirements on the WIS usage as described by their security and privacy profiles. They need additional auxiliary data and auxiliary functions. The variability of use is restricted by the learner’s context, which covers the learner’s specific tasks and specific data and
function demand, and by chosen involvement, while the profile of learners imposes exceptions. The involvement and collaboration of learners is based on assumptions of social behaviour and restrictions due to organisational decisions. These assumptions and restrictions are components of the learner’s context.

**Storyboard context:** The meaning of content and functionality to users depends on the stories, which are based on scenarios that reflect learning scenarios and the portfolios of users or learners. According to the profile of these users a number of quality requirements such as privacy, security and availability must be satisfied. The learner’s scenario context describes what the learner needs to understand in order to efficiently and effectively solve his/her tasks in the actual portfolio. The learner’s determine the policy for following particular stories.

**System context:** The edutainment WIS is developed to support a number of intentions. The purposes and intents lead to a number of decisions on the WIS architecture, the technical environment, and the implementation. The WIS architecture has an impact on its utilisation, which often is only implicit and thus leads to not understandable systems behaviour. The technical environment restricts the user due to restrictions imposed by server, channel and client properties. Adaptation to the current environment is defined as context adaptation to the current channel, to the client infrastructure and to the server load. At the same time a number of legal decisions based on regulations, laws and business rules have been incorporated into the WIS.

**Temporal context:** The utilisation of a scene by an learner depends on his/her history of utilisation. Learners may interrupt and resume their activities at any moment of time. As they may not be interested in repeating all previous actions they have already successfully completed, the temporal context must be taken into account. Due to availability of content and functionality the current utilisation may lead to a different story within the same scenario.

This entire information forms the context space, which brings together the storyboard specification and the contextual information. Typical questions that are answered on the basis of the context space are:

- What content is required by the context space?
- What functionality is required by the context space?
- What has to be changed for the life cases, the storyboard, etc., if context is considered?

As outlined above the context space is determined by the learners, the scenarios, the WIS itself, and the time. It leads to a specialisation of the content, structuring and functionality of the scenes.

Context is associated with desirable properties of the WIS such as quality criteria and security and privacy requirements. Quality criteria such as suitability for the users or learnability provide obligations for the WIS development process. Though these criteria are rather fuzzy, they lead directly to a number of implementation obligations that must be fulfilled at later stages, i.e. within the development on the implementation layer.

For instance, learnability means comprehensibility, i.e. the WIS must be easy to use, remember, capture and forecast. This requires clarity of the visual representation, predictability,
directness and intuitiveness. These properties allow the user to concentrate on the tasks. The workflows and the discourse structure correspond to the expectations of the users and do not lead to surprising situations. They can be based on metaphors and motives taken from the application domain. In the same way other quality criteria can also be mapped to development obligations.

Other properties that may be associated with context refer to the potential utilisation for other tasks outside the scope of the storyboard. In this case we do not integrate the additional tasks into the storyboard, but instead support these tasks, if this in accordance with our intentions. For instance, we might expect further visits targeting at core concerns of the edutainment WIS.

4 Open Issues and Challenges for Edutainment WIS

Learning content must be properly handled. It turns out that this task is one of the most difficult tasks. For this reason, any edutainment WIS must be combined with an authoring WIS that supports authors during appropriate development of content.

Quality control of learning objects is necessary since low quality data harms the learning success. We may distinguish a number of reasons for low quality and development strategies for improvement of quality: Incomplete content, mixed content, wrong content, complex associations among content, and mutated content.

Functionality development for edutainment WIS also includes the development of very sophisticated supporting facilities for the actors, the content, the context, and the presentation.

Most edutainment systems are currently based on scenarios of sequenced learning. 3rd generation systems are aiming in providing best-suited content just in time to the right user, place and device with the best pricing. They challenge current technology. Research is sought on didactics, content integration and delivery, storyboarding, adaptation and context integration, and success control. Open learning objects provide a sophisticated facility for content management. The theory of open learning units should be integrated with didactics based on storyboarding, content adaptation and delivery, and content development. In future, they will be extended with context, e.g., story space, actor, user, payment, portfolio, association, history, etc. Control functionality should be provided for open learning units in the same fashion as we know it already for exercises, tests, and exams for self-control or certification.

References


Pragmatics of Storyboarding – Web Information Systems Portfolios

Anonymous

Somewhere

Abstract

A Web Information System (WIS) can be described by a storyboard, which on a high level of abstraction specifies who will be using the system, in which way and for which goals. Syntax and semantics of storyboarding have been well-elaborated. Pragmatics is the necessary complement addressing what the storyboard means for its users. The part of pragmatics concerned with usage analysis by means of life cases, user models and contexts has been dealt with before. In this paper we complement usage analysis by WIS portfolios, which comprise two parts: the information portfolio and the utilisation portfolio. The former one is concerned with information consumed and produced by the WIS users, which leads to content chunks; the latter one captures functionality requirements.

Keywords. Pragmatics, Web Information System, Information Portfolio, Utilisation Portfolio

1 Introduction

A Web Information System (WIS) is an information system that can be accessed through the world-wide-web. So far, many approaches to WIS conceptual modelling have been developed, e.g. (Ceri, Fraternali, Bongio, Brambilla, Comai & Matera 2003, De Troyer & Leune 1998, Houben, Barna, Frasincar & Vdovjak 2003, Lowe, Henderson-Sellers & Gu 2002, Rossi, Schwabe & Lyardet 1999, Anonymous 2005a), most of which are centered around content and navigation modelling, occasionally coupled with specific requirements models.

In (Anonymous 2005a) we characterised a WIS by strategic characteristics such as purpose, mission, intentions and ambience (more details in (Anonymous 2005b)), usage characteristics such as tasks, users and stories, content and functionality characteristics, context, and presentation, which leads to an abstraction layer model for WIS development. Central to this approach to WIS development is the method of storyboarding, which on a high level of abstraction specifies who will be using the system, in which way and for which goals. In a nutshell, a storyboard consists of three parts:

- a set of tasks that are associated with goals the users may have,
- a set of actors, i.e. abstractions of user groups defined by roles that determine obligations and rights, and user profiles determining preferences, and
- a story space, which itself consists of a hierarchy of scenarios describing scenes and actions, and is accompanied by a plot describing the action scheme.

Syntax and semantics of storyboarding including customisation to preferences have been well elaborated in (Anonymous 2005a, Anonymous 2007a, Anonymous 2009). However, in order to link storyboarding to the systems requirements and to provide guidelines and means to derive the complex storyboards from informal ideas about a WIS without any technical bias, this has to be complemented by pragmatics, which according to (Web 1991) is the “balance between principles and practical usage”.

In (Anonymous 2007b) we addressed the pragmatics of storyboarding focusing on usage analysis. Based on intentions we investigated life cases, user models and contexts. Life cases, capture observations of user behaviour in reality, and can be used in a pragmatic way to specify the story space. Life cases have already been envisioned in (Carroll 2004) and integrated into the entire software engineering process in (Harel & Marelly 2003). They generalise business use cases as in (Robertson & Robertson 2006). User models that are specified by user and actor profiles, and actor portfolios. They are used to get a better understanding of the tasks associated with the WIS, and the goals of users. Goals have been identified as a crucial component of requirements engineering in (Giorgini, Mylopoulos, Nicchiarelli & Sebastiani 2002). Task descriptions are also used in participatory design, e.g. in (Carroll 2004, Kensig & Blomberg 1998, O’Neill & Johnson 2004). Contexts characterise the situation, in which a user finds himself at a certain time in a particular location. For WISs we must handle different kinds of contexts and analyse the way they impact on life cases, user models and the storyboard.

In this paper we extend our work on WIS pragmatics focusing on WIS portfolios, which address the pragmatics associated with content and functionality. We distinguish between information as processed by humans and data as its carrier that is perceived or noticed, selected and organized by its receiver. Content is complex and ready-to-use data, and may be enhanced by concepts that specify the semantic meaning of content objects, and topics that specify the pragmatic understanding of users.

Thus, information is directed towards pragmatics, whereas content may be considered to highlight the syntactical dimension. If content is enhanced by concepts and topics, then users are able to capture the meaning and the utilisation of the data they receive. Analogously, functionality refers to functions offered by the system, thus highlights the dynamic aspects of the syntactic dimension, whereas utilisation is linked to the stories supporting users’ life cases, thus is directed towards pragmatics. This distinction is illustrated in Figure 1.

Accordingly, a WIS portfolio consists of two parts: the information portfolio and the utilisation portfolio. The former one is concerned with information consumed and produced by the WIS users, which leads to content chunks. We will elaborate on this in Section 2. The latter one captures functionality requirements. We will elaborate on this in Section 3 focusing on utilisation portfolios for learning and communities.
2 Information Portfolios

A WIS portfolio consists of an information portfolio and a utilisation portfolio. They are mapped to content and functionality specifications, respectively. In doing so we distinguish between content provided by the WIS and information, which is related to an actor or user.

2.1 Consumption and Production of Information

Following (Anonymous 2005a) on a high level of abstraction we may think of a WIS as a set of abstract locations, which abstract from actual pages. A user navigates between these locations, and on this navigation path, s/he executes a number of actions. We regard a location together with local actions, i.e. actions that do not change the location, as a unit called scene.

Then a WIS can be described by an edge-labelled directed multi-graph, in which the vertices represent the scenes, and the edges represent transitions between scenes. Each such transition may be labelled by an action executed by the user. If such a label is missing, the transition is due to a simple navigation link. The whole multi-graph is then called the story space.

A story is a path in the story space. It tells what a user of a particular type might do with the system. The combination of different stories to a subgraph of the story space can be used to describe a "typical" use of the WIS for a particular task. Therefore, we call such a subgraph a scenario. Usually storyboarding starts with modelling scenarios instead of stories, coupled by the integration of scenarios to the story space.

Each WIS user who enters the system with a particular goal has information needs that have to be satisfied by the system. In addition, an active WIS will also request information from its users. We use the term information consumption for the information provided by the system to its users, and information production for the information entered by a user into the system.

When a user enters the WIS, the information needs are usually not known in advance. Part of the needed information may depend on other parts, on decisions made while navigating through the WIS, and even on the information provided by the actor him/herself. That is, the information consumption and production depends on the path through the WIS, i.e. in our terminology on the story. Therefore, information consumption and production is associated with each scene of the story space. Assuming that there is a database for the data content of the WIS with database schema $S$, information consumption on a scene $s$ definitely accounts for a view $V_s$ over $S$. That is, we have another schema $S_V$ and a computable transformation from databases over $S$ to databases over $S_V$. Such a transformation is usually expressed by a query $q_V$.

- With each scene $s$ we associate a view $V_s = (S_V, q_V)$ called information consumption view. Elements of $q_V(d_B)$ for some database $d_B$ represent the information consumption of an actor.
- With each action $a$ we associate a data type $t_a$ called information production type. Values of type $t_a$ represent the information production by an actor.

Information consumption and information production of an actor for all scenes together define the information portfolio of the actor.

2.2 Information Need and Demand

We distinguish between the information need and the information demand. The former one refers to a perceived lack of something desirable or useful, while the latter one results from an act of demanding or asking.

The information need is generally related to objectives such as becoming informed. It is based on the intuitive insight that the current information and knowledge is insufficient for the task under consideration, or the necessary information cannot be easily derived from data that is currently available, or the uncertainty, indefiniteness, fuzziness, and contradictions do not permit drawing conclusions.

The information need can be considered to be subjective, but at the same time it can be the reason for a certain user visiting a website without an intention that is related to a life case. The information need is based on the conceptual incongruity in which a user’s cognitive structure is not adequate to a task, e.g. when a user recognizes that something wrong in their state of knowledge and desires to resolve the anomaly, when the current state of knowledge is less than what is needed, when internal sense runs out, or when there is insufficient knowledge to cope with gaps, uncertainties or conflicts in a knowledge area. Therefore, as the behaviour of actors is mainly related to life cases and the portfolio, we have to distinguish between information provided for support of life cases and auxiliary consumed information that is provided to visitors as a service.

The information demand is related to the portfolio under consideration and to the intents. We may distinguish between information that is necessary, desirable, or feasible. The information demand is mapped to the views defining information consumption and production for each scene of the story space as defined above. The information demand is characterised by information that is missing, unknown, necessary for task completion, and directly requested.

We can distinguish between the information demand of an actor and the information demand of a user. As actors represent groups of users, the information demand of a user contains the information demand of an actor. While
the information demand of actors is determined by the portfolio, the additional information demand of a user is determined by the user profile.

2.3 The Concept of Persona

The information demand is used to derive the information consumption of each user. This is related to the definition and meaning of information for the user based on received / requested data, which has to be organized, interpreted, understood, and integrated into his/her knowledge. In general, this would require to model the user, the specific request of the user, the ability to understand the data, and the skills, which is infeasible. However, as the information demand of actors is a subset of the one of users represented by the actor, we can use prototypes of individuals called personae to determine the information demand. In addition, we model a task-oriented life case of these individuals, and derive the information demand, data requirement, and the specific utilisation requirements.

A persona is characterized by an expressive name, profession, intents, technical equipment, behaviour, skills and profile, disabilities, and specific properties such as hobbies and habits. A persona is a typical individual created to describe the typical user based on the life cases, the context, the portfolio, and the profile. User models characterize profiles for education, work, and personality. This characterization can be extended by

- identity with name, pictures, etc.,
- personal characteristics such as age, gender, location, and socio-economic status,
- characterization of reaction to possible users error,
- specific observed behaviour including skill sets, behavioural pattern, expertise and background, and
- specific relationships, requirements, and expectations.

Example 1 Let us consider an information service of a city and focus on business people. For these we develop the following specific portfolio:

User profile: Jack-of-all-trades is a business man. He intends to visit the city through a short-term visit. He is interested in culture and history. He likes short distances and 4-star or better hotels. He is usually in a hurry. He likes good dining and talking. Additionally, he is familiar with technology.

Intention and information demand: Jack-of-all-trades visits the information site for business trip preparation. His information demand includes hotel information, spare time information for the evenings, and information on the central traffic.

Content requirements: The information demand may be mapped to general city information, information on restaurants, traffic, culture, business clubs, and good dining addresses. Therefore, the information consumption of Jack-of-all-trades must be supported by the corresponding databases. At the same time, Jack-of-all-trades requires a handy booking service.

Life case: The life case we envision includes a brief survey on the city including places to see, the selection of a convenient hotel, a survey of events of interest, the booking procedure for events, a search for good dining places, and some information what else to see and whom to talk to.

Specific utilisation: The collection of data is similar to a basket collection. Jack-of-all-trades prefers shallow navigation and fast search. He is also interested in highlights for the period he is considering.

Jack-of-all-trades can be exemplified to be a specific persona:

Personal: name: Bernhard Karłowics, age: 48, male, married, lives in northern Germany, profession: business assistant, income: around Euro50.000 per year

Robustness: errors are not taken as own errors, download time is critical

Kind of user: kind but pretentious, makes quick decisions, interested in history, culture, classical music, usually in a hurry

Specific behaviour: resumes story also after hours

Specific interactivity: works alone, with interruptions, no time for concentrated reading, final results expected

Profile: middle level manager, Masters degree in business and computer engineering, workaholic (around 60 hours per week)

Portfolio: collection of travel details with confirmation and payment, prefers hotels with four-stars or higher, checking through direct connection to booking services

Life case: preparing for travel, single use of the system, email confirmation necessary, spare time information, events of interest, dining places, business clubs

Context: business environment, typically client-server computers at workplace, occasionally mobile phone for contact while on the move

For the persona the following specification template can be used:

Persona: (persona name)
Identity: (first and last name, photo)
Demography of persona: (age, gender, location, status)
Robustness of WIS usage: (criticality of errors)
Kind of user: (general description)
Specific behaviour: (general description)
Specific interactivity: (general description)
Based On User Profile: (name)
Refined education profile: (general description)
Refined work profile: (general description)
Refined personality profile: (general description)
Based On Portfolio: (name)
Refined task: (general description)
Refined involvement: (general description)
Refined collaboration: (general description)
Refined restrictions: (general description)
Based On Life Case: (name)
Refined characterisation: (outcome description)
Refined life case flow: (general graphical description)
Refined figures: (actors list)
Refined context: (general context description)
Refined representation: (general behavior)
Based On Context: (name)
Refined persona context: (general description)
Refined storyboard context: (general description)
Refined WIS context: (general description)
Refined temporal context: (general description)
The explicit specification of personae has several benefits. They provide communication means within the development team, focus on a specific target set of actors, and help to make assumptions about the target audience. Thus, personae may augment the WIS portfolio specification, but should not be overused.

2.4 Content Chunks

In order to model the information portfolio we collect the information demand of all actors we would like to support. In addition, we can include some specific information demand of users matching with the groups of users. This information demand can be combined into a single content chunk that is demanded by all actors of similar steps in the life case. This information demand can later be modelled within a database model.

This combination turns around the viewpoint we have taken so far. We try to envision which content is necessary for which actors or users. We may start with the intention why an actor may demand a given content. The content-centered view allows us to derive a specification of steps in which a certain chunk of content is requested.

The content-centered analysis is used during brainstorming sessions in which we try to derive scenarios, intentions, and the information need of users. At the same time we can derive the service kind, utilisation, actors, presentation, content, and functions supporting this content. We can also derive directly the intersection among the content chunks, which provides a basis for the development of queries to extract the content demanded from the WIS.

Content chunks are arranged within content-extended scenarios, in which each scene is associated with a content chunk. This content chunk combines the consumption of actors, auxiliary content that is provided for the support of users, and the content that is additionally provided due to the intentions of the WIS provider and the context of the current scene. We further enhance this scenario by data that is produced by the actor or that stems from the environment.

Example 2 Let us continue Example 1 and consider the booking scene. If the actor has decided to choose a particular hotel, then we may associate with the choice action the selection of an identification for the hotel, which extends the booking scene with the content that provides information to the actor on hotels. With the choice of a hotel the actor leaves this scene and produces the selection data. In the next scene the actor can be asked about the payment.

The same scene can also be used in a different scenario, in which the actor first collects all choices in a basket and later confirms those choices, which are the most appropriate.

The information portfolio is also enhanced by information that depends on the WIS context. This can define content associated with control, which normally is internal to the WIS and not accessible by applications, content that is used to determine the transitions and to control the WIS context, e.g. for the pre- and post-scene conditions, transition conditions depending on the databases used, or assignment for collaborating actors, etc., content that might be useful as reference, e.g. meta-information on time, responsibility, and links to other WISs, or application-specific data that is not accessible by the WIS.

The content set required for each step in a life case may become too large. So we must prioritise the development of content chunks. There are two perspective for prioritising:

- In the usability perspective we evaluate the impact the content has for the the WIS portfolio. The impact is based on the occurrence the content has in steps and activities and on the number of actors requiring this content. The impact is high if the majority of users need it frequently or cannot continue their work if the content is missing.

- The economic perspective we evaluate the cost/benefit relation. Since projects have limited budget, contract commitments, resource restrictions, technological constraints, marketplace pressures, and deadlines we must limit the efforts. We may classify those content chunks which have high impact into strategic, high value, targeted and luxuries.

Example 3 The www.cottbus.de WIS provides information on hotels, booking services, etc. For the decision, which information should be provided we may use a classification along usability and economic perspective as in Figure 2. We can then assign final decisions to the evaluation such as approved (✓), under review (○), and rejected (✗).

Figure 2: The cost/benefit evaluation of the hotel data in an information service

In a similar way we may evaluate all content chunks we discovered. The evaluation is used for deciding which content is to be developed and to which extent.

Content chunks are often composed only of data, i.e. they relate to media objects on the conceptual model (Anonymous 2005a). The description of the data is based on media types, viewpoints, and adaptation facilities. Content is also based on semantics, i.e. associated concepts capturing basic pieces of knowledge or their annotations, and restricted by pre- and postconditions. Content must also be annotated using a commonly agreed vocabulary or a dictionary. This agreement is bound to actors or communities. We denote items of dictionaries by topics. The pragmatics also reflects the context of potential usage. As content is often used in a form that combines content chunks with other content chunks we also specify related content chunks. In summary, we may use the following specification template for content chunks:

Content chunk: (content chunk name)
Description: (specification)
Media types: (list of media types)
Viewpoints: (list of views over the types)
Adaptation: (description)
Scenario: (general description and knowledge)
Concepts: (list of associated concepts)
Prerequisites: (condition)
Postconditions: (condition)
Pragmatics: (general description and context)
Actor: (list of actors)
Topics: (list of associated topics)
Delivery: (general description of utilization)
Container: (names of associated containers)
Additional content: (list of content chunks)
Usage: (story space annotation)
Enabling Intention:
2.5 ContentChunks for the Entry Scene

The entry scene is associated with a specific content chunk that will give rise to the home page of the WIS. Therefore, it must be developed with greatest care. It must contain all essential information of the WIS on the setting, the environment, the characterisation, the main intentions, activities that can be played with, etc. Therefore, we must balance the list of supported tasks in such a way that they become displayable.

The entry scene concerns the initial situation, the emotional environment, and the main intentions a user may have while visiting the WIS. At the same time, the actor must understand what s/he has to expect when entering the WIS. Objectives of the entry page are to introduce the context and essential actions of the WIS, provide information on collaborating actors, show the theme of the WIS, define the kind of scenario to be played, and support an easy match with the information demand of a user or actor.

The attraction of users requires suspense in the opening, as the first impression decides whether a casual visitor continues within the WIS. Branding, navigation, content and usage must be balanced. Rules for the entry scene must be based on the characteristics of WIS, i.e. on intentions, especially objectives, aims and the target audience, on specifics of supported life cases such as need in guidance, feedback, explanation, specific content, and functionality, on complexity of content, which directly leads to the need to support surveyability, ease of use, and decomposition of content chunks, on variety of functions that must already be provided by the entry scene, on adaptation to user and actor profiles, which requires a rough separation of actors according to their education, work, personality of security profiles, on properties of portfolio such as complexity of problems, need in collaboration, and support for workspace and workplace, and on integration of context, and adaptation of content, functionality, and scenario to the context.

These rules permit the derivation of the general atmosphere and the main intention of the entry scene. Typical examples are categories such as “traditional and serious” for business and work pages, and “energetic” or “vital”, providing a flavour of progression and innovation.

**EXAMPLE 4** The entry page must accommodate the large variety of visitors and their information needs, for which we can identify a number of content items that must be provided:

- Immediately and clearly communicate the purpose to the visitor. Each visitor must be supported by descriptive wording and images that are easy and quickly to understand independent from whether the visit is the first or a repeated one. The values of the WIS must be easily detectable. Visitors’ positive impressions depend on the trustworthiness and the values of the WIS.

- Creating an identity of the WIS (branding): Users need to quickly capture whether a WIS holds a valuable promise, whether they can trust it, and what content is offered.

- Attract by content: A visitor judges a WIS within seconds of entering it. Therefore, content must be attractive, well-organised, easy to browse, and summarized.

- Personalization of content: The WIS should be tailored to the portfolio and profiles of their visitors. In this case, the system does not require users to learn and memorize its facilities.

- Provide an orientation to the visitor: Navigation must be easy to use and may be based on an exploration metaphor. Users should quickly comprehend how to get around, should not be forced to guess what can be done next.

- Balanced content and functionality: The trade-off between space used for content or functionality can be resolved by developing patterns that support fast detection of items the user is seeking for, by focusing on the tasks of the user, and by branding in a constrained form.

- Establishing a cohesive and logical layout: The most important information chunks must be immediately identifiable and located. For this the reading and recognition culture of the users must be taken into account.

The pages of a WIS, especially the home page are often designed by graphics experts who tend to use a large number of multimedia features. However, layout and play-out have to be in accordance with the rules developed above, which leads to the request to apply screenography (Anonymous 2007c, Anonymous 2008). This is, however, beyond the scope of this article.

Finally, the entry scene must be considered as one of the main advertisement instruments of the WIS. For this reason, a clear statement of the values of the WIS is the basis for deriving content that conveys these values with the users’ first impressions. Typical value properties are:

- reliability, availability, actuality, speed, responsiveness, alternatives, ease of use, managing complexity, (filtering) levels, completeness, links, flexibility, support, export, privacy, and guidance for information services, or

- user adaptation, learning styles and preferences, simplicity, support, flexibility, guidance, accessibility, consistency, motivating, clear goals, and responsiveness, or

- portability, speed, shared resources, previewing, alertness, awareness, versioning, mail management, multi-threading, report generation, feedback, mentoring, distribution, flexibility, security, and safety for community sites.

These values of the entry page are a part of the brand of the WIS and must match with the aims and objects that have already been specified for the entire WIS while capturing intentions (Anonymous 2007b). We can now combine these requirements for an abstract description of the entry scene using the following template:

**Entry scene:**
*Entry scene name*

**Entry scene value:**

**Intentions:**
*list of main supported intentions*

**Content:**
*list of main content, headlines and blurbs*

**Functions:**
*list of main functions and their representation*

**Shortcuts:**
*anchored list of shortcuts to sub-scenario*

**Links:**
*content or function anchored list of links*

**General navigation:**
*content or function anchored list of links*

**Actor:**
*list of actors, rights, roles*
3 Utilisation Portfolios

The second constituent of the WIS portfolio is the utilisation portfolio, which can be considered to be a collection of requirements for functionality and WIS utilisation in general. It describes the intentions of the users, their goals, their context, and their specific requirements, and as such is based on the life cases that were modelled before, and the profiles and the portfolio of the users and actors. Furthermore, the actor context must be taken into account. Therefore, we first discuss the utilisation portfolio and then derive its impact on the functionality required by the user.

The utilisation portfolio combines the actor or user perspective to the WIS. We already know intentions of users, their profiles, and their life cases. Users are grouped to actors for which portfolios have already been developed. Based on the portfolio and the context specification life cases were extended.

The WIS utilisation portfolio cannot be described in general for all different categories of WIS such as e-business, learning and entertainment, communities, etc., though these categories are mixed in real applications. The separation into categories cases, however, the description of the WIS utilisation portfolio. In the following we will describe the development of the WIS utilisation portfolio for learning WISs and group/community WISs.

3.1 Learning Portfolios

A large number of WISs provide learning and entertainment services. Examples of technology-supported learning include computer-based training systems, interactive learning environments, intelligent computer-aided instruction systems, distance learning systems, and collaborative learning environments. The general brand \( P^W2L^A \) of a WIS (Anonymous 2005b) has to be specialised for entertainment WISs:

Provider \( P \): For the time being providers are mainly educational institutions or educational communities. Then the provider plays the role of a teacher.

Product \( W \): Since control and assessment of learning progress is a still unresolved issue and appropriate presentation of complex information is often not feasible, entertainment WISs concentrate on easy-to-understand information or easy-to-grasp knowledge.

Users \( U \): Users of entertainment WISs are mainly students, people seeking for continuous education, workers in companies with a specific portfolio, people interested in auxiliary information, or groups of such users. The main behavior of such users is characterised by the role of the learner.

Activities \( A \): Activities are mainly centered around learning, searching for content, collecting content, and solving exercises. Activities also include to ask questions, to act in teams for problem solving, and to discuss issues associated with the learning material.

Thus, typical learning WIS brands take the form \( \text{Teacher}^W2\text{Learner}^A \), where \( W \) stands for content, chunks or knowledge, and \( A \) can be one of \{receive, respond, solve in teams, raise questions, possibly apply\}, \{learn, validate, control, advice\}, \{recognise, listen, work on it\}, \{solve exercises, ask urgent questions\}, \{discuss, get feedback, work on it\}, etc.

Entertainment WISs are currently mainly or exclusively supporting student actors. Students obtain knowledge through teachers, their schedules, and their abilities; they need guidance, motivation, and control. The behaviour of actors may, however, be more complex, in particular in case of learning groups, in which the collaboration of students depends on a cooperation profile with rights and obligations. Communication partners exchange content, discuss and resolve questions, seek for hints or for better understanding, supporting and motivating partners are users with control, motivation and supporting functions. Furthermore, teachers have various obligations and rights, and are involved in a variety of roles.

In order to derive the utilisation portfolio we analyse the word fields associated with common verbs in the learning context such as learn, know, master, study and nouns such as skill. This gives rise to stories and consequently determines the functionality requirements.

Learn: Learning is a very complex activity that includes gaining knowledge, understanding, obtaining skills, etc. by studying, instruction or experience. In addition, learning is associated with memorizing, being able to perform some task, and to know this ability. Learning is based on obtaining content, discovering the concepts behind them, annotation, ordering, and integration. Learning is associated with the role of a learner or student, who are usually supported by other actors who teach and instruct. Learners determine content with certainty, usually by making an inquiry or other effort. They check the content to find out whether it is useful or whether additional content is needed.

Know: As learning aims at improving skills, abilities, and knowledge, the improvement should be measurable and learning success examined. Knowing means to be cognizant of a fact or a specific piece of information and to possess knowledge or information about. It may also include to know how to do something. Learners obtain first-hand knowledge of states, situations, emotions, or sensations, and the change in knowledge is acknowledged and recognized by other actors.

Master: Learning usually intends to enable mastering problems and to become proficient and skilled in some area. This mastership is closely related to practising and experimenting with the new knowledge.

Study: Studying refers to the activities associated with learning, i.e. reading learning content, exercising, checking, examining, inspecting, surveying, etc. Therefore, the presentation of the learning material and the storyboard are essential. Furthermore, learners need to mind, perpend, think (out or over), and weigh, which requires time and workplaces. Studying can be performed by oneself without any teacher, supporter, or observer.

Skill: Skills are abilities that have been acquired by training, e.g. abilities to produce solutions to some problems. A learner is trained until s/he obtains these skills.

These word fields have a complex structure, which leads to functions requiring other support functions associated with related word fields such as discover, ascertain, catch on, and determine. One difference to the word fields for e-business is their iterative application. The
word fields are extended by the learning style to be supported. The three different learning styles sequenced or biomedical learning, interactive learning, and group learning result in rather different scenarios. So far, only sequenced learning approaches have received the necessary theoretical basis in pedagogical research and didactics. Interactive learning is still an open research issue.

Example 5: The KOPRA project within the German University Notebook Initiative was aiming at the support of adaptive and collaborative learning, focusing on practical training in database programming. The system is supporting communication and interaction of learners, organization and coordination of collaborating communities, free access to material depending on the progress, roles and portfolios of partners, and the stepwise development of training material. Learners act in a collaborative setting depending on their needs and the goals of the learning program.

The analysis of the word fields above highlighted the utilisation of importance of exercising of crucial part of the WIS. While often only multiple-choice exercises are supported, complex exercises have a more complex pattern. Figure 3 shows the derived scenario for exercise training for this case. Learners are allowed to choose either their own data or data provided by the system. Learners may also choose an algorithm that can be used for solving the exercise.

Using the storytelling language SiteLang (Anonymous 2005a) we can represent the entire scenario by the following expression:

\[
T; ( ( D \square ( C ; P ) ) \square ( I ; U ) ) ; \\
H ; ( R ; H ) * ; A ; ( S \boxdot skip ) ; \\
E ; J ; N ; H ; ( R ; H ) * ; A ^ { + } ; S \land
\]

with the complex actions \( T \) (task delivery), \( D \) (delivery of prepared data), \( C \) (collection of users data), \( I \) (information on applicable algorithms), \( P \) (preparation of learners data), \( U \) (code upload and installation), \( H \) (formation of learners hypotheses), \( A \) (computation of association through mining), \( S \) (submission of competitive solution), \( E \) (evaluation of submitted solution), \( J \) (inspections of sample solution and comparison with evaluations of competitors), \( N \) (preparation for next trial for solution), and \( R \) (reminder on learning element on hypothesis). Further, the symbols \( \land \) and \( \lor \) are used for denoting the entry and the termination of the whole exercise scene.

3.2 Community Portfolios

Communities and leisure groups are gathering places for registered members, who form an interacting community of various actors with common location, intentions, and time. They are based on shared experience, interests or conviction, and voluntary interaction among members towards common goals. In the case of work communities they deal with issues affecting the legal profession, and are concerned with furthering the best interests of their members. Communities often employ leadership and hosting. Membership requirements and obligations, resources and amenities, member characteristics, community policies, and activity style and pattern create a large variety of community/group WISs.

The utilisation portfolio is based on the community or group member profiles. The general brand \( P ^ { C } W _ { 2 I ^ { A } } \) must also be specialized for community and group services. The provider and the users may coincide for community WIS.

The actions depend on the kind of the community or group. The life cases of these communities and groups vary much. Nevertheless, we can identify several types of typical scenarios. Active scenarios \( \langle \text{act} \rangle \) are activities by the receiving actor(s). Information scenarios \( \langle \text{inform} \rangle \) or more specifically \( \text{ask} \) and \( \text{search} \) are mainly governed by the provider. Participation scenarios \( \langle \text{part} \rangle \) request the interaction of receiving actors. Community and group WIS are mainly following the Group2Group\( ^ { A } \) or the Group2Visitor\( ^ { inform, act } \) brand.

Discussions are another kind of action. We may distinguish chat-oriented, collaboration-based, and topic-centered discussion groups. The first group does not limit contributions. Main actions are initiate, respond, inform, summarise, and close. Collaboration-based discussions follow a collaboration pattern and preserve a collaboration style agreed in advance. The main metaphor used is the “meeting”. Main actions are open, close, propose agenda, issue discussion, contribute, and inform. Topic-centered discussion groups are based on a topic and preserve dialogue acts. Actors may be authors, responders, organisers, or supporters. Authors are typically classified and have a number of privileges, while responders respond to some data and use a response pattern and preserve a response style. Tokens are a convenient vehicle for support of topic-centered discussions often generalising communication protocols.

Example 6: The process for finding a group decision is typical example for topic-centered discussions. For instance, members of a program committee of a conference collaborate during decision making, whether a paper is to be accepted or rejected. They act either as reviewers or as ordinary members of the program committee. Reviewers of one paper may revise their review, comment on the reviews of other members, and exchange messages within the subgroup of the program committee comprising the reviewers. An ordinary member of the program committee may comment on the reviews and request a change of evaluation.

All members will make a common decision that depends on the other discussion topics, i.e. on the decisions for all papers. Opinions may be stated with certain confidence. Opinions may be distributed to all or selected members of the program committee. Authors of papers are excluded from all discussions on their papers. The actions of a member of the PC may be mandatory, optional or not permitted. So, we specify these obligations and rights using deontic constraints.

Web2.0 sites provide other examples of enhanced community or group WISs that are even more user-driven. Examples are GoogleActSense, Wikipedia, Flickr, etc. Communities may share their opinion or beliefs or partial knowledge (e.g. Wikipedia), their diary or visits (e.g. dealicio.us), their intentions of internet utilisation (e.g. 43thing.com), and more. While these communities are currently more or less leisure communities, they will evolve into large bodies requiring lead enhanced utilisation portfolios for communities.

In particular, support for collaboration and participation will be needed. This gives rise to collaboration utilisation portfolios. Collaboration means to work together towards a common goal, i.e. it combines of three aspects:

- Cooperation takes parts of actions into account that jointly define a group action.
- Coordination of actions of individual actors is necessary to achieve the desired result in a cooperative action.
- Communication between actors involved in the same group action is indispensable for coordination. It appears in a variety of facets, e.g. as an act or instance of transmitting a verbal or written message, a process by which information is exchanged between individuals through a common system of symbols, or as a technique for expressing ideas effectively (as in speech) using the technology of the transmission of information (as by print or telecommunication).
Figure 3: KoPra scenario for training with optional exercises

So, the main word fields associated for the utilisation portfolio are the following ones:

**Contribute:** A member of a community gives or supplies a document to the community and shares it with the community subject to certain obligations and responsibilities. Obligations include modification obligations such as updating and removing the document. The member of the community becomes an author and has a role as owner. The community, however, obtains or possesses the contribution. Responsibilities of the author include quality obligations for the contribution.

**Join:** Users may join a community and thus obtain a role of a member of the community. The membership requires the commitment to the coordination style and coordination pattern of the community. This commitment implies a certain behaviour and collaboration with the members of the community. For instance, members agree on regular occurrence or logical relations. The membership may be public or hidden to visitors.

**Meet:** Members of a community come together at a particular time or place, entering into conference, argument, or personal dealings with other members of the community or the general auditory. During the meeting they receive or greet in an official capacity of the community. Members may form coalitions before, during and after meetings.

**Organize:** Becoming a member of a community brings rights and obligations. The member may be permitted to develop a unit within the community and has a function during this development. In this case the member becomes a coherent functioning whole, and contributes with own profile and portfolio. Organisations often require to arrange by systematic planning and united effort.

**Publish:** The author makes his/her contributions generally known. The community decides, whether the contribution can be disseminated to the public based on the rules of the community. The contribution is then produced or released for distribution.

**Subscribe:** Communities often have their own publication lines. A member subscribes to some of them according to the rules of the community. The subscribed services are (periodically and regularly) received by the actor. The member of the community agrees to purchase and pay for securities. The subscription can also be extended to new offerings.

**Withdraw:** The members of a community should have the right to cancel the membership of the community or part of the subscriptions. In this case the user may lose the ownership of contributions s/he made. The withdrawal may require particular official procedures.

### 3.3 Portfolios for Entertainment and Gaming Systems

Web-based entertainment systems are less data-intensive than systems in the other categories of WISs. They are therefore less important for us, but we include this brief discussion of entertainment WISs for the sake of completeness.

Entertainment systems aim at evoking feelings and thoughts. They are usually combined with (personal) e-business sites. Sometimes, they are also embedded into educational sites. They are similar to real life entertainment. The general brand $\mathcal{P}^{\text{Entertainment}}$ is specialized by active scenarios ($\mathcal{A}$) with activities taken by the receiving actor(s) and by the products of entertainment. The products are usually associated with fun ($\mathcal{F}$). Therefore, the brand of entertainment (fan) sites is given by $B^2 \mathcal{V}^{\text{Ent}}$ or $V^2 \mathcal{V}^{\text{Ent}}$ with $B$ for business and $V$ for visitor.

Entertainment sites suffer from incompatibilities and unstable technology. As CD-ROM sophistication has dried out all singleton player entertainment sites, the emphasis is now on interactive or collaborating games. The aspect of collaboration is analogous to community/group WISs. The placement of functionality and content follows however the gaming approach, i.e. links to a guided tour, room or apartment metaphors for placement of items, exhibits, and setting expectations very well before breaking the rules.

Entertainment sites can be classified into a number of kinds, genres, and classes such as head and expert, develop, or fight. Kinds such as tournaments, $n$-player games, board games are specified by a number of criteria that are not subject to interpretation or understanding of the user and are not dynamic. Genres classify the experience the player gains, e.g., reasoning, reacting. Classes characterize the storyboard of the game, e.g., development, deduction, shooting, trading. Additionally, the storyboard has in entertainment a special situation: conflicts and conflict resolution.
Strategy games are the most important entertainment WIS beside action games, adventure and role games, sport games, and puzzle games. Since strategy, tactics, and role games may also be used for edutainment we discuss these in more details. They are often based on puzzle-solving scenarios where a player has certain targets such as a re-naturalization or area development target. The space of applicable steps has a high dimensionality. Applicable steps are dependent on each other and have a number of reversible, guarantee- and post-conditions and distant and side effects.

Scenarios can be based on word fields such as decide, explore and detect, and ask. They represent often complex, network and dynamic situations. The complete information on the situation cannot be retrieved by the player. Player must often follow patterns of causal and hypothetical reasoning, i.e., development of targets, collection of data and development of a model, analysis of the current situation and its opportunities, prognostic elaboration and extrapolation, planning of activities, and control of the effect. Targets and goals may have positive or negative effects. Sometimes, negative effects are preferable. Goals may be preferable or avoidable, general or specific, simple or complex, and implicit or explicit. Data have their own characterisation ranging from important in the given situation over neglectable.

This gives rise to a number of utilisation scenarios. Games are specified by

- the intention of the game by a variable and quantifiable outcome that can be valorized, e.g., the purpose for the player, the intent of the player,
- a number of transition rules with content parameters, with function parameters, and with states to be transformed,
- the player effort for the game depending on the profile and the portfolio of the player, e.g., abilities, education, tasks, collaboration, and
- the context of the game, e.g., hardware and software context, actor context, storyboard context, temporal context, organizational and social context, and provider context

The attachment of the player to the outcome may change the game behaviour. Consequences of steps the player is taking may also be under negotiation. Games are mainly linear scenarios that follow a game story. Additional information, tricks, tips, customer treatment, contracting, workspace supply, communication support, group and game rooms, help/tips, downloads, payment, conditions of usage extend the playout of the scenarios. Hobby sites use scenarios similar to information or community sites. In this case, hobby description, links, documents and other media objects are provided. Fun and humor site often do not have any scenario. They mainly consist of links and documents.

The storyboard of games and scenes can be generally specified by the scene intention in Figure 4. Players of games do not have the full control and the full knowledge of the state of the art. At the same time, they have the illusion of full control, of full vision, and of full functionality.

fascination of control, mental effects

---

The navigation is based on simple tree or graph structure whereas the structuring depends on categories/rubrics or ontology. Personal entertainment site, advice sites, and survey sites also do not have any scenario. They follow mainly approaches we discuss for corporate identity and forum. Their navigation structure is rather simple, often only a one-level structure. They follow metaphors of show-business presentation. Similarly experiment presentation sites are built. Special entertainment sites provide a group service, use a pay-per-view payment and are based on journal or presentation mechanisms. They provide a collection of media data, customer data, and contracts. Only the last case can be described through word fields such as survey, billing, condition, order, member service, and linking.

Entertainment WIS are based on a sophisticated actor specification. Actors are often highly demanding in their requirements. Their quality requirements include accessibility, art/mechanics, downloads, easy exploration, feedback, (sophisticated) functionality, guidance, immersion, integration, and (narrative) storytelling.

Actors are associated to portfolio that consists of tasks of certain degree of difficulty, with certain effects, duration for completion and benefits and penalties for their completion. Tasks are structured, ordered, and prioritized. The context is often deeply elaborated in entertainment WIS. They are providing all illusions and functions known for virtual reality.

Since virtual reality is considered to be an image of the reality, it is also characterized by history, folks, organisations, realms, and ambitions for further development.

The results of execution of a scene is related to tasks currently under consideration with their pre-determined plans and scenarios, with time management, with management of state changes, with resources, functions, and collaborating actors. The tasks are structured and ordered and follow a certain plan. For their solution a number of resources must be provided and a level of resistance must be overruled. Activities can be described by word fields for all major verbs in our language. Tasks may be open, completed, under completion, under repetition, etc. Entertainment WIS usually reward players for successful execution. Scene change follows general stories envisioned.

The involvement of actors in task completion is pre-specified on the basis of characters, i.e., the roles an actor may potentially play, the part the actor plays within the scenario, rights an actor obtains during execution, obligations an actor must obey within a given role or within a set of roles.

We expect that future edutainment WIS will inherit a number of these features and neatly integrate them into the storyboard, into the content management of the WIS, and into the support for activities of actors.

---

4 Conclusion

In this paper we extended our previous work on pragmatics of Web Information Systems. We introduced WIS portfolios, which are composed of an information portfolio and a utilisation portfolio. The information portfolio captures information consumption and production by users in the scenes of the story space. This is linked to the information needed by a user to understand the task and the information demand to perform the appropriate action. Modelling information demand of all potential users is infeasible, but the concept of persona permits to deal with prototypical users instead. The integration of information demands leads to content chunks.

Likewise, the utilisation portfolio consists of tasks users might wish to accomplish within the system, and goals they want to achieve by this. It can be considered as the necessary means for collecting functionality requirements. Different from the information portfolio the utilisation portfolio depends on the application category. Here
we focused on the categories of learning WISs and group or community WISs.

WIS portfolios complete the research on WIS pragmatics, which is an independent connector element between systems requirements and conceptual models. In this paper, due to space limitations we had to omit many details, which we plan to report elsewhere in an extended version.

References


Anonymous (2008), ‘——’. details hidden for double-blind reviewing.

Anonymous (2009), ‘——’. details hidden for double-blind reviewing.


Web (1991), ‘Webster’s ninth new collegiate dictionary’.
Extended Entity-Relationship Model

Bernhard Thalheim
Christian-Albrechts University Kiel, http://www.informatik.uni-kiel.de/~thalheim

SYNONYMS
EERM, HERM; higher-order entity-relationship model; hierarchical entity-relationship model

DEFINITION
The extended entity-relationship (EER) model is a language for defining the structure (and functionality) of database or information systems. Its structure is developed inductively. Basic attributes are assigned to base data types. Complex attributes can be constructed by applying constructors such as tuple, list or set constructors to attributes that have already been constructed. Entity types conceptualise structuring of things of reality through attributes. Cluster types generalise types or combine types into singleton types. Relationship types associate types that have already been constructed into an association type. The types may be restricted by integrity constraints and by specification of identification of objects defined for a type. Typical integrity constraints of the extended entity-relationship model are participation, look-across, and general cardinality constraints. Entity, cluster, and relationship classes contain a finite set of objects defined on these types. The types of an EER schema are typically depicted by an EER diagram.

HISTORICAL BACKGROUND
The entity-relationship (ER) model was introduced by P.P. Chen in 1976 [1]. The model conceptualises and graphically represents the structure of the relational model. It is currently used as the main conceptual model for database and information system development. Due to its extensive usage a large number of extensions to this model were proposed in the 80’s and 90’s. Cardinality constraints [1, 3, 4, 8] are the most important generalisation of relational database constraints [7]. These proposals have been evaluated, integrated or explicitly discarded in an intensive research discussion. The semantic foundations proposed in [2, 5, 8] and the various generalisations and extensions of the entity-relationship model have led to the introduction of the higher-order or hierarchical entity-relationship model [8] which integrates most of the extensions and also supports conceptualisation of functionality, distribution [9], and interactivity [6] for information systems. Class diagrams of the UML standard are a special variant of extended entity-relationship models.
The ER conferences (annually; since 1996: International Conference on Conceptual Modeling, http://www.conceptualmodeling.org/) are the main forum for conceptual models and modelling.

SCIENTIFIC FUNDAMENTALS
The extended entity-relationship model is mainly used as a language for conceptualisation of the structure of information systems applications. Conceptualisation of database or information systems aims to represent the logical and physical structure of an information system. It should contain all the information required by the user and required for the efficient behavior of the whole information system for all users. Conceptualisation may further target the specification of database application processes and the user interaction. Structure description are currently the main use of the extended ER model.

An example of an EER diagram.
The EER model uses a formal language for schema definition and diagrams for graphical representation of the
schema. Let us consider a small university application for management of Courses. Proposed courses are based on courses and taught by a docent or an external docent within a certain semester and for a set of programs. Proposals typically include a request for a room and for a time and a categorisation of the kind of the course. Theses proposals are the basis for course planning. Planning may change time, room and kind. Planned courses are held at the university. Rooms may be changed. The example is represented by the EER diagram in Figure 1.

![Extended Entity-Relationship Diagram for Course Management](image)

**Figure 1:** Extended Entity-Relationship Diagram for Course Management

Entity types are represented graphically by rectangles. Attribute types are associated with the corresponding entity or relationship type. Attributes primarily identifying a type are underlined. Relationship types are represented graphically by diamonds and associated by directed arcs to their components. A cluster type is represented by a diamond, is labelled by the disjoint union sign, and has directed arcs from the diamond to its component types. Alternatively, the disjoint union representation ⊕ is attached to the relationship type that uses the cluster type. In this case directed arcs associate the ⊕ sign with component types. An arc may be annotated with a label.

**The definition scheme for structures.**

The extended entity-relationship model uses a data type system for its attribute types. It allows the construction of entity types \( E \doteq (\text{attr}(E), \Sigma_E) \) where \( E \) is the entity type defined as a pair — the set \( \text{attr}(E) \) of attribute types and the set \( \Sigma_E \) of integrity constraints that apply to \( E \). The definition \( \text{def} \) of a type \( T \) is denoted by \( T \doteq \text{def} \).

The EER model lets users inductively build relationship types \( R \doteq (T_1, ..., T_n, \text{attr}(R), \Sigma_R) \) of order \( i \) \((i \geq 1)\) through a set of (labelled) types of order less than \( i \), a set of attribute types, and a set of integrity constraints that apply to \( R \). The types \( T_1, ..., T_n \) are the components of the relationship type. Entity types are of order 0. Relationship types are of order 1 if they have only entity types as component types. Relationship types are of order \( i \) if all component types are of order less than \( i \) and if one of the component types is of order \( i - 1 \). Additionally, cluster types \( C \doteq T_1 \bigcup ... \bigcup T_n \) of order \( i \) can be defined through a disjoint union \( \bigcup \) of relationship types of order less than \( i \) or of entity types.

Entity/relationship/cluster classes \( T^C \) contain a set of objects of the entity/relationship/cluster type \( T \). The EER model mainly uses set semantics, but (multi-)list or multiset semantics can also be used. Integrity constraints apply to their type and restrict the classes. Only those classes are considered for which the constraints of their types are valid. The notions of a class and of a type are distinguished. Types describe the structure and constraints. Classes contain objects.

The data type system is typically inductively constructed on a base type \( B \) by application of constructors such as the tuple or products constructor (...) or the list constructor < .. >. Types may be optional component types and are denoted by [...]. The types \( T \) can be labelled \( L \cdot T \). The label is used as an alias name for the type. Labels denote roles of the type. Labels must be used if the same type is used several times as a component type in the definition of a relationship.
or cluster type. In this case they must be unique.

An entity-relationship schema consists of a set of data, attribute, entity, relationship, and cluster types which types are inductively built on the basis of the base types.

Given a base type system $B$. The types of the ER schema are defined through the type equation:

$$ T = B \mid (l_1 : T, \ldots, l_n : T) \mid \{ T \} \mid \langle T \rangle \mid \{ T \} \mid T \cup T \mid l : T \mid N \models T $$

**Structures in detail.**

The classical four-layered approach is used for inductive specification of database structures. The first layer is the data environment, called the basic data type scheme, which is defined by the system or is the assumed set of available basic data types. The second layer is the schema of a database. The third layer is the database itself representing a state of the application’s data often called micro-data. The fourth layer consists of the macro-data that are generated from the micro-data by application of view queries to the micro-data.

**Attribute types and attribute values.**

The classical ER model uses basic (first normal form) attributes. Complex attributes are inductively constructed by application of type constructors such as the tuple constructor (..), set constructor {..}, and the list constructor < .. >. Typical base types are integers, real numbers, strings, and time. Given a set of names $N$ and a set of base types $B$, a basic attribute type $A :: B$ is given by an (attribute) name $A \in N$ and a base type $B$. The association between the attribute name and the underlying type is denoted by ::. The base type $B$ is often called the domain of A, i.e. $dom(A) = B$. Complex attributes are constructed on base attributes by application of the type constructors. The notion of a domain is extended to complex attributes, i.e. the domain of the complex attribute $A$ is given by $dom(A)$. Components of complex attributes may be optional, e.g., the Title in the attribute Name.

Typical examples of complex and basic attributes in Figure 1 are

- $Name \equiv (\text{FirstName} < \text{FirstName}>, \text{FamName}, [\text{AcadTitles}], [\text{FamilyTitle}])$, $PersNo \equiv \text{EmplNo} \cup \text{SocSecNo}$, $AcadTitles \equiv \{ \text{AcadTitle} \}$, $Contact \equiv \{ \text{Name}, \text{Login}, \text{URL}, \text{Address}, \text{Contact}, \text{DateOfBirth}, \text{PersNo} \}$, $PostalAddress \equiv \{ \text{Zip}, \text{City}, \text{Street}, \text{HouseNumber} \}$

- for $DateOfBirth :: \text{date}$, $AcadTitle :: \text{acadTitleType}$, $FamilyTitle :: \text{familyTitleAcronym}$, $Zip :: \text{string7}$, $SocSecNo :: \text{string9}$, $EmplNo :: \text{int}$, $City :: \text{varString}$, $Street :: \text{varString}$, $HouseNumber :: \text{smallInt}$

The complex attribute Name is structured into a sequence of first names, a family name, an optional complex set-valued attribute for academic titles, and an optional basic attribute for family titles. Academic titles and family titles can be distinguished from each other.

**Entity types and entity classes.**

Entity types are characterized by their attributes and their integrity constraints. Entity types have a subset $K$ of the set of attributes which serve to identify the objects of the class of the type. This concept is similar to the concept of key known for relational databases. The key is denoted by ID(K). The set of integrity constraints $\Sigma_E$ consists of the keys and other integrity constraints. Identifying attributes may be underlined instead of having explicit specification.

Formally, an entity type is given by a name $E$, a set of attributes $attr(E)$, a subset $id(E)$ of $attr(E)$, and a set $\Sigma_E$ of integrity constraints, i.e.

$$ E \models (attr(E), \Sigma_E) $$

The following types are examples of entity types in Figure 1:

- $Person \equiv \{ \text{Name, Login, URL, Address, Contact, DateOfBirth, PersNo} \}$
- $Course \equiv \{ \text{CourseID, Title, URL} \}, \{ \text{ID} (\{ \text{CourseID} \}) \}$
- $Room \equiv \{ \text{Building, Number, Capacity} \}, \{ \text{ID} (\{ \text{Building, Number} \}) \}$
- $Semester \equiv \{ \{ \text{Term, Date(Starts, Ends)} \}, \{ \text{ID} (\{ \text{Term} \}) \} \}$

An ER schema may use the same attribute name with different entity types. For instance, the attribute URL in Figure 1 is used for characterising additional information for the type $Person$ and the type $Course$. If they need to be distinguished, then complex names such as $CourseURL$ and $PersonURL$ are used.

Objects on type $E$ are tuples with the components specified by a type. For instance, the object (or entity) $(HRS3, 408A, 15)$ represents data for the $Room$ entity type in Figure 1.
An entity class $E^C$ of type $E$ consists of a finite set of objects on type $E$ for which the set $\Sigma_E$ of integrity constraints is valid.

Cluster types and cluster classes.
A disjoint union $\cup$ of types whose identification type is domain compatible is called a cluster. Types are domain compatible if they are subtypes of a common more general type. The union operation is restricted to disjoint unions since identification must be preserved. Otherwise, objects in a cluster class cannot be related to the component classes of the cluster type. Cluster types can be considered as a generalisation of their component types.

A cluster type (or “category”)

$C \doteq l_1 : R_1 \cup l_2 : R_2 \cup ... \cup l_k : R_k$

is the (labelled) disjoint union of types $R_1, ..., R_k$. Labels can be omitted if the types can be distinguished.

The following type is an example of a cluster type:

$$\text{Teacher} \doteq \text{ExternalDocent : CollaborationPartner} \cup \text{Docent : Professor}$$

The cluster class $C^C$ is the ‘disjoint’ union of the sets $R_1^C, ..., R_k^C$. It is defined if $R_1^C, ..., R_k^C$ are disjoint on their identification components. If the sets $R_1^C, ..., R_k^C$ are not disjoint then labels are used for differentiating the objects of clusters. In this case, an object uses a pair representation $(l_i, o_i)$ for objects $o_i$ from $R_k^C$.

Relationship types and relationship classes.
First order relationship types are defined as associations between entity types or clusters of entity types. Relationship types can also be defined on the basis of relationship types that are already defined. This construction must be inductive and cannot be cyclic. Therefore, an order is introduced for relationship types. Types can only be defined on the basis of types which have a lower order. For instance, the type $\text{Professor}$ in Figure 1 is of order 1. The type $\text{ProposedCourse}$ is of order 2 since all its component types are either entity types or types of order 1. A relationship type of order $i$ is defined as an association of relationship types of order less than $i$ or of entity types. It is additionally required that at least one of the component types is of order $i - 1$ if $i > 1$. Relationship types can also be characterized by attributes. Relationship types with one component type express a subtype or an Is-A relationship type. For instance, the type $\text{Professor}$ is a subtype of the type $\text{Person}$.

Component types of a relationship type may be labelled. Label names typically provide an understanding of the role of a component type in the relationship type. Labelling uses the definition scheme Label : Type. For instance, the $\text{Kind}$ entity type is labelled by $\text{Proposal}$ for the relationship type $\text{ProposedCourse}$ in Figure 1. Cluster types have the maximal order of their component types. Relationship types also may have cluster type components. The order of cluster type components of a relationship type of order $i$ must be less than $i$.

Component types that are not used for identification within the relationship type can be optional. For instance, the $\text{Room}$ component in Figure 1 is optional for the type $\text{PlannedCourse}$. If the relationship object in the $\text{PlannedCourse}$ class does not have a room then the proposal for rooms in $\text{ProposedCourse}$ is accepted. A specific extension for translation of optional components may be used. For instance, $\text{Room}$ in Figure 1 is inherited to $\text{PlannedCourse}$ from $\text{ProposedCourse}$ if the $\text{Room}$ component for a $\text{PlannedCourse}$ is missing.

Higher order types allow a convenient description of types that are based on other types. For example, consider the course planning application in Figure 1. Lectures are courses given by a professor or a collaboration partner within a semester for a number of programs. Proposed courses extend lectures by describing which room is requested and which time proposals and which restrictions are made. Planning of courses assigns a room to a course that has been proposed and assigns a time frame for scheduling. The kind of the course may be changed. Courses that are held are based on courses planned. The room may be changed for a course. The following types specify these assertions.

$\text{ProposedCourse} \doteq (\text{Teacher, Course, Proposal : Kind, Request : Room, Semester}, \text{Set2 : \{ Program \}})$,

$\text{PlannedCourse} \doteq (\text{ProposedCourse, [Reassigned : Kind] }, [\text{Reassigned : Room} ]$, 

$\text{CourseHeld} \doteq (\text{PlannedCourse, [Reassigned : Room] }, [\text{StartDate, EndDate, AssistedBy} ]$, 

$\Sigma_{\text{CourseHeld}})$.

The second and third types use optional components in case a proposal or a planning of rooms or kinds is changed. Typically, planned courses are identified by their own term-specific identification. Integrity constraints can be
omitted until they have been defined.

Formally, a relationship type is given by a name \( R \), a set \( \text{compon}(R) \) of labelled components, a set of attributes \( \text{attr}(R) \), and a set \( \Sigma_R \) of integrity constraints that includes the identification of the relationship type by a subset \( \text{id}(R) \) of \( \text{compon}(R) \cup \text{attr}(R) \), i.e.

\[
R \doteq (\text{compon}(R), \text{attr}(R), \Sigma_R) .
\]

It is often assumed that the identification of relationship types is defined exclusively through their component types. Relationship types that have only one component type are unary types. These relationship types define subtypes. If subtypes need not be explicitly represented then binary relationship types named by IsA between the subtype and the supertype are used. For instance, the type Professor in Figure 1 is a subtype of the type Person. An object (or a “relationship”) on the relationship type \( R \doteq (R_1, \ldots, R_m, \{B_1, \ldots, B_k\}) \), \( \text{id}(R) \), \( \Sigma_R \) is an element of the Cartesian product \( R_1^C \times \ldots \times R_m^C \times \text{dom}(B_1) \times \ldots \times \text{dom}(B_k) \). A relationship class \( R^C \) consists of a finite set \( R^C \subseteq R_1^C \times \ldots \times R_m^C \times \text{dom}(B_1) \times \ldots \times \text{dom}(B_k) \) of objects on \( R \) for which \( \text{id}(R) \) is a key of \( R^C \) and which obeys the constraints \( \Sigma_R \).

### Integrity constraints.

Each database model also uses a set of implicit model-inherent integrity constraints. For instance, relationship types are defined over their component types, and a (relationship) object presumes the existence of corresponding component objects. Typically only finite classes are considered. The EER schema is acyclic. Often names or labels are associated with a minimal semantics that can be derived from the meaning of the words used for names or labels. This minimal semantics allows us to derive synonym, homonym, antonym, troponym, hypernym, and holonym associations among the constructs used.

The most important class of integrity constraints of the EER model is the class of cardinality constraints. Other classes of importance for the EER model are multivalued dependencies, inclusion and exclusion constraints and existence dependencies[7]. Functional dependencies, keys and referential constraints (or key-based inclusion dependencies) can be expressed through cardinality constraints.

Three main kinds of cardinality constraints are distinguished: participation constraints, look-across constraints, and general cardinality constraints. Given a relationship type \( R \doteq (\text{compon}(R), \text{attr}(R), \Sigma_R) \), a component \( R' \) of \( R \), the remaining substructure \( R'' = R \setminus R' \) and the remaining substructure \( R''' = R'' \cap R \text{compon}(R) \) without attributes of \( R \).

The participation constraint \( \text{card}(R, R') = (m, n) \) restricts the number of occurrences of \( R' \) objects in the relationship class \( R^C \) by the lower bound \( m \) and the upper bound \( n \). It holds in a relationship class \( R^C \) if for any object \( o' \in R^C \) there are at least \( m \) and at most \( n \) objects \( o \in R^C \) with \( \pi_{R'}(o) = o' \) for the projection function \( \pi_{R'} \) that projects \( o \) to its \( R' \) components. Participation constraints relate objects of relationship classes to objects of their component classes. For instance, the constraint \( \text{card}(\text{ProposedCourse}, \text{SemesterCourse}) = (0, 3) \) restricts relationship classes for proposals for courses per semester to at least 0 and at most 3, i.e. each course is proposed at most three times in a semester. There are at most three objects \( o \) in \( \text{ProposedCourse}^C \) with the same course and semester objects. The integrity constraint \( \text{card}(\text{ProposedCourse}, \text{DocentSemester}) = (3, 7) \) requires that each docent is giving at least 3 courses and at most 7 courses. External docents may be obliged by other restrictions, e.g., \( \text{card}(\text{ProposedCourse}, \text{ExternalDocentSemester}) = (0, 1) \).

Formally, the integrity constraint \( \text{card}(R, R') = (m, n) \) is valid in \( R^C \) if \( m \leq |\{o \in R^C : \pi_{R'}(o) = o'\}| \leq n \) for any \( o' \in \pi_{R'}(R^C) \) and the projection \( \pi_{R'}(R^C) \) of \( R^C \) to \( R' \).

If \( \text{card}(R, R') = (0, 1) \) then \( R' \) forms an identification or a key of \( R \), i.e. \( \text{ID}(R') \) for \( R \). This identification can also be expressed by a functional dependency \( R : R' \rightarrow R'' \).

The lookup or look-across constraint \( \text{look}(R, R') = m.n.n \) describes how many objects \( o'' \) from \( R'''^C \) may potentially ‘see’ an object \( o' \) from \( R^C \). It holds in a relationship class \( R^C \) if for any object \( o'' \in \text{dom}(R''') \) there are at least \( m \) and at most \( n \) related objects \( o' \) with \( \pi_{R'}(o) = o' \), i.e. \( m \leq |\{o' \in \pi_{R'}(R^C) : o \in R^C \land \pi_{R'}(o) = o' \land \pi_{R''}(o) = o''\}| \leq n \) for any \( o'' \in \text{dom}(R''') \). Typically, look-across constraints are used for components consisting of one type. Look-across constraints are not defined for relationship types with one component type.

Look-across constraints are less intuitive for relationship types with more than 2 component types or with attribute types. For instance, the look-across constraint \( \text{look}(\text{ProposedCourse}, \text{DocentSemester}) = 0..7 \) specifies that for any combination of Teacher, Room, Kind, and Program objects there are between 0 and 7 Docent and Semester
The lower bound expresses that there are Teacher, Room, Kind, and Program which do not have a Docent and Semester combination.

Look-across constraints for a binary relationship type which component types form a key of the relationship type can equivalently expressed by participation constraints, i.e. \( \text{look}(R, R_1) = m_1 \ldots n_1 \) if and only if \( \text{card}(R, R_2) = (m_1, n_1) \). Similarly, \( \text{look}(R, R_2) = m_2 \ldots n_2 \) if and only if \( \text{card}(R, R_1) = (m_2, n_2) \). This equivalence is neither valid for binary relationship types which cannot by identified by their components and nor for relationship types with more than 2 components.

Participation and look-across constraints can be extended to substructures and intervals and to other types such as entity and cluster types. Given a relationship type \( R \), a substructure \( R' \) of \( R \) and \( R'' \) as above. Given furthermore an interval \( I \subseteq \mathbb{N}_0 \) of natural numbers including 0. The (general) cardinality constraint \( \text{card}(R, R') = I \) holds in a relationship class \( R^C \) if for any object \( o' \in \pi_{R^C}(R^C) \) there are \( i \in I \) objects \( o \) with \( \pi_R(o) = o' \), i.e. \( \{ o \in R^C : \pi_R(o) = o' \} \mid I \) for any \( o' \in \pi_{R^C}(R^C) \).

The following participation, look-across and general cardinality constraints are examples in Figure 1:

For any \( R' \in \{ \text{Semester, Course, Kind} \} \) \( \text{card}(\text{ProposedCourse, Semester Course Teacher}) = (0, n), \)
\( \text{card}(\text{CourseHeld, PlannedCourse}) = (1, 1), \)
\( \text{card}(\text{PlannedCourse, ProposedCourse[Semester] Room TimeFrame}) = (0, 1), \)
\( \text{card}(\text{ProposedCourse, Docent Semester}) = \{ 0, 3, 4, 5, 6, 7 \}. \)

The first constraint does not restrict the database. The second constraint expresses a key or functional dependency. The types Semester Course Teacher identify any of the other types in the type ProposedCourse, i.e. \( \text{ProposedCourse}: \{ \text{Semester, Course, Teacher} \} \rightarrow \{ \text{Request, Time, Proposal, Set2} \}. \)

The third constraint requires that any planned course must be given. The fourth constraint requires that rooms are not overbooked. The fifth constraint allows that docents may not teach in a semester, i.e. have a sabbatical. If a docent is teaching in a semester then at least 3 and at most 7 courses are given by the docent.

Look-across constraints were originally introduced by P.P. Chen [1] as cardinality constraints. UML uses look-across constraints. Participation and look-across constraints cannot be axiomatised through a Hilbert- or Gentzen-type logical calculus. If only upper bounds are of interest then an axiomatisation can be found in [3] and [4]. General cardinality constraints combine equality-generating and object-generating constraints such as keys, functional dependencies and referential integrity constraints into a singleton construct.

Logical operators can be defined for each type. A set of logical formulas using these operators can define the integrity constraints which are valid for each object of the type.

**Schemata.**

The schema is based on a set of base (data) types which are used as value types for attribute types. A set \( \{ E_1, \ldots, E_n, C_1, \ldots, C_l, R_1, \ldots, R_m \} \) of entity, cluster and (higher-order) relationship types on a data scheme DD is called schema if the relationship and cluster types use only the types from \( \{ E_1, \ldots, E_n, C_1, \ldots, C_l, R_1, \ldots, R_m \} \) as components and cluster and relationship types are properly layered.

An EER schema is defined by the pair \( \mathcal{D} = (S, \Sigma) \) where \( S \) is a schema and \( \Sigma \) is a set of constraints. A database \( \mathcal{D}^C \) on \( \mathcal{D} \) consists of classes for each type in \( \mathcal{D} \) such that the constraints \( \Sigma \) are valid.

The classes of the extended ER model have been defined through sets of objects on the types. In addition to sets, lists, multi-sets or other collections of objects may be used. In this case, the definitions used above can easily be extended [8].

A number of domain-specific extensions have been introduced to the ER model. One of the most important is the extension of the base types by spatial data types such as: point, line, oriented line, surface, complex surface, oriented surface, line bunch, and surface bunch. These types are supported by a large variety of functions such as: meets, intersects, overlaps, contains, adjacent, planar operations, and a variety of equality predicates.

The translation of the schema to (object-)relational or XML schemata can be based on a profile [8]. Profiles define which translation choice is preferred over other choices, how hierarchies are treated, which redundancy and null-value support must be provided, which kind of constraint enforcement is preferred, which naming conventions are chosen, which alternative for representation of complex attributes is preferred for which types, and whether weak types can be used. The treatment of optional components is also specified through the translation profile of the types of the schema. A profile may require the introduction of identifier types and base the identification on the identifier. Attribute types may be translated into data formats that are supported by the target system.
The EER schema can be used to define views. The generic functions insert, delete, update, projection, union, join, selection and renaming can be defined in a way similarly to the relational model. Additionally, nesting and unnesting functions are used. These functions form the algebra of functions of the schema and are the basis for defining queries. A singleton view is defined by a query that maps the EER schema to new types. Combined views also may be considered which consist of singleton views which together form another EER schema.

A view is specified over an EER schema \( D \) by a schema \( V = \{ S_1, \ldots, S_m \} \), an auxiliary schema \( A \) and a (complex) query \( q : D \times A \rightarrow V \) defined on \( D \) and \( A \). Given a database \( D^C \) and the auxiliary database \( A^C \), the view is defined by \( q(D^C \times A^C) \).

Graphical representation.
The schema in Figure 1 consists of entity, cluster and relationship types. The style of drawing diagrams is one of many variants that have been considered in the literature. The main difference of representation is the style of drawing unary types. Unary relationship types are often represented by rectangles with rounded corners or by (directed) binary IsA-relationship types which associate by arcs the supertype with the subtype. Tools often do not allow cluster types and relationship types of order higher than 1. In this case, these types can be objectified, i.e. represented by a new (abstract) entity type that is associated through binary relationship types to the components of the original type. In this case, identification of objects of the new type is either inherited from the component types or is provided through a new (surrogate) attribute. The first option results in the introduction of so-called weak types. The direct translation of these weak types to object-relational models must be combined with the introduction of rather complex constraint sets. Typically, this complexity can be avoided if the abstract entity type is mapped together with the new relationship types to a singleton object-relational type. This singleton type is also the result of a direct mapping of the original higher-order relationship type.

The diagram can be enhanced by an explicit representation of cardinality and other constraints. If participation constraints \( \text{card}(R, R') = (m, n) \) are used for component consisting of one type \( R' \) then the arc from \( R \) to \( R' \) is labelled by \( (m, n) \). If look-across constraints \( \text{look}(R, R') = m..n \) are used for binary relationship types then the arc from \( R \) to \( R' \) is labelled by \( m..n \).

**KEY APPLICATIONS**
The main application area for extended ER models is the conceptualisation of database applications. Database schemata can be translated to relational, XML or other schemata based on transformation profiles that incorporate properties of the target systems.

**FUTURE DIRECTIONS**
The ER model has had a deep impact on the development of diagramming techniques in the past and is still influencing extensions of the unified modelling language UML. UML started with binary relationship types with look-across constraints and without relationship type attributes. Class diagrams currently allow n-ary relationship types with attributes. Relationship types may be layered. Cluster types and unary relationship types allow for distinguishing generalisation from specialisation.

ER models are not supported by native database management systems and are mainly used for modelling of applications at the conceptual or requirements level. ER schemata are translated to logical models such as XML schemata or relational schemata or object-relational schemata. Some of the specifics of the target models are not well supported by ER models and must be added after translating ER schemata to target schemata, e.g., specific type semantics such as list semantics (XML) or as special ordering or aggregation treatment of online analytical processing (OLAP) applications.

The ER model has attracted a lot of research over the last 30 years. Due to novel applications and to evolution of technology old problems and novel problems are challenging the research on this model. Typical old problems that are still not solved in a satisfactory manner are: development of a science of modelling, quality of ER schemata, consistent refinement of schemata, complex constraints, normalisation of ER schemata, normalisation of schemata in the presence of incomplete constraint sets. Novel topics for ER research are for instance: evolving schema architectures, collaboration of databases based on collaboration schemata, layered information systems
and their structuring, schemata with redundant types, ER schemata for OLAP applications. Structures of database applications are often represented through ER models. Due to the complexity of applications, a large number of extensions have recently been proposed, e.g., temporal data types, spatial data types, OLAP types and stream types. Additionally, database applications must be integrated and cooperate in a consistent form. The harmonisation of extensions and the integration of schemata is therefore a never ending task for database research.

ER models are currently extended for support of (web) content management that is based on structuring of data, on aggregation of data, on extending data by concepts and on annotating data sets for simple reference and usage. These applications require novel modelling facilities and separation of syntactical, semantical and pragmatic issues. The ER model can be extended to cope with these applications. The ER model is mainly used for conceptual specification of database structuring. It can be enhanced by operations and a query algebra. Operations and the queries can also be displayed in a graphical form, e.g. on the basis of VisualSQL. Most tools supporting ER models do not currently use this option. Enhancement of ER models by functionality is necessary if the conceptualisation is used for database development. Based on functionality enhancement, view management facilities can easily be incorporated into these tools. ER models are becoming a basis for workflow systems data. The standards that have been developed for the specification of workflows have not yet been integrated into sophisticated data and application management tools.

URL TO CODE
http://www.informatik.uni-kiel.de/~thalheim/HERM.htm
http://www.is.informatik.uni-kiel.de/~thalheim/indeeerm.htm
Readings on the RADD project (Rapid Application and Database Development)

CROSS REFERENCE
I. DATABASE FUNDAMENTALS
   a. Data models (including semantic data models)
   b. Entity-Relationship (ER) model
   c. Unified modelling language (UML)

III. THEORETICAL ASPECTS
   b. Relational Theory

RECOMMENDED READING
Between 3 and 15 citations to important literature, e.g., in journals, conference proceedings, and websites.


8
Specialisation and Generalisation

Bernhard Thalheim

Christian-Albrechts University Kiel, http://www.informatik.uni-kiel.de/~thalheim/HERM.htm

SYNONYMS
refinement, abstraction, hierarchies;
clustering, grouping, inheritance

DEFINITION
Generalisation and specialisation are main principles of database modelling. Generalisation maps or groups types or classes to more abstract or combined ones. It is used to combine common features, attributes, or methods. Specialisation is based on a refinement of types or classes to more specific ones. It allows developers to avoid null values and to hide details from non-authorised users. Typically, generalisations and specialisations form a hierarchy of types and classes. The more specialised classes may inherit attributes and methods from more general ones. In database modelling and implementation clusters of types to a type that represents common properties and abstractions from a type are the main kinds of generalisations. Is-A associations that specialise a type to a more specific one and Is-A-Role-Of associations that considers a specific behaviour of objects are the main kinds of specialisations.

MAIN TEXT
Specialisation introduces a new entity type by adding specific properties belonging to that type which are different from the general properties of its more general type. Generalisation introduces the Role-Of relationship or the Is-A relationship between a subtype and its general type. Therefore, the application, implementation, and processes are different. For generalisation the general type must be the union of its subtypes. The subtypes can be virtually clustered by the general type. This tends not to be the case for specialisation. Specialisation is a refinement or restriction of a type to more special ones. Typical specialisations are Is-A and Has-Role associations. Exceptions can be modelled by specialisations.

Different kinds of specialisation may be distinguished: structural specialisation which extends the structure, semantic specialisation which strengthens type restrictions, pragmatical specialisation which allows to separate the different usage of objects in contexts, operational specialisation which introduces additional operations, and hybrid specialisations. Is-A specialisation requires structural and strong semantic specialisation. Is-A-Role-Of specialisation requires structural, pragmatical and strong semantic specialisation.

Generalisation is based either on abstraction or on grouping. The cluster construct of the extended ER model is used to represent generalisations. Generalisation tends to be an abstraction in which a more general type is defined by extracting common properties of one or more types while suppressing the differences between them. These types are subtypes of the generic type. New types are created by generalizing classes that already exist. Structural combination typically assumes the existence of a unifiable identification of all types. Semantical combination allows the disjunction of types through the linear sum of semantics. Pragmatical generalisation is based on building collections whenever applications require a consideration of commonalities.

CROSS REFERENCE
I. DATABASE FUNDAMENTALS
a. Data models (including semantic data models)

REFERENCES
Abstraction
Bernhard Thalheim
Christian-Albrechts University Kiel, http://www.informatik.uni-kiel.de/~thalheim/HERM.htm

SYNONYMS
component abstraction, localisation abstraction, implementation abstraction; association, aggregation, composition, grouping, specialisation, generalisation, classification

DEFINITION
Abstraction allows developers to concentrate on the essential, relevant or important parts of an application. It uses a mapping to a model from things in reality or from virtual things. The model has the truncation property, i.e. it lacks some of the details in the original, and a pragmatic property, i.e. the model use is only justified for particular model users, tools of investigation, and periods of time. Database engineering uses construction abstraction, context abstraction and refinement abstraction. Construction abstraction is based on the principles of hierarchical structuring, constructor composition, and generalisation. Context abstraction assumes that the surroundings of a concept are commonly assumed by a community or within a culture and focuses on the concept, turning away attention from its surroundings such as the environment and setting. Refinement abstraction uses the principle of modularisation and information hiding. Developers typically use conceptual models or languages for representing and conceptualising abstractions. The enhanced entity-relationship model schema are typically depicted by an EER diagram.

MAIN TEXT
Database engineering distinguishes three kinds of abstraction: construction abstraction, context abstraction and refinement abstraction.
Constructor composition depends on the constructors as originally introduced by J. M. Smith and D.C.W. Smith. Composition constructors must be well founded and their semantics must be derivable by inductive construction. There are three main methods for construction: development of ordered structures on the basis of hierarchies, construction by combination or association, and construction by classification into groups or collections. The set constructors \( \subset \) (subset), \( \times \) (product) and \( \mathcal{P} \) (powerset) for subset, product and nesting are complete for the construction of sets.
Subset constructors support hierarchies of object sets in which one set of objects is a subset of some other set of objects. Subset hierarchies are usually a rooted tree. Product constructors support associations between object sets. The schema is decomposed into object sets related to each other by association or relationship types. Power set constructors support a classification of object sets into clusters or groups of sets - typically according to their properties.

Context abstraction allows developers to commonly concentrate on those parts of an application that are essential for some viewpoints during development and deployment of systems. Typical kinds of context abstraction are component abstraction, separation of concern, interaction abstraction, summarisation, scoping, and focusing on typical application cases.
Component abstraction factors out repeating, shared or local patterns of components or functions from individual concepts. It allows developers to concentrate on structural or behavioral aspects of similar elements of components. Separation of concern allows developers to concentrate on those concepts that are a matter of development and to neglect all other concepts that are stable or not under consideration. Interaction abstraction allows developers to concentrate on those parts of the model that are essential for interaction with other systems or users. Summarisation maps the conceptualisations within the scope to more abstract concepts. Scoping is typically used to select those concepts that are necessary for current development and removes those concepts that do not have an impact on the necessary concepts.
Database models may cover a large variety of different application cases. Some of them reflect exceptional,
abnormal, infrequent and untypical application situations. Focusing on typical application cases explicitly
separates models for the normal or typical application case from those that are atypical. Atypical application
cases are not neglected but can be folded into the model whenever atypical situations are considered.
The context abstraction concept is the main concept behind federated databases. Context of databases can be
characterized by schemata, version, time, and security requirements. Sub-schemata, types of the schemata or views
on the schemata, are associated by explicit import/export bindings based on a name space. Parametrisation lets
developers to consider collections of objects. Objects are identifiable under certain assumptions and completely
identifiable after instantiation of all parameters.

Interaction abstraction allows developers to display the same set of objects in different forms. The view concept
supports this visibility concept. Data is abstracted and displayed in various levels of granularity. Summarisation
abstraction allows developers to abstract from details that are irrelevant at a certain step. Scope abstraction
allows developers to concentrate on a number of aspects. Names or aliases can be multiply used with varying
structure, functionality and semantics.

Refinement abstraction is mainly about implementation and modularisation. It allows developers to selectively
retain information about structures. Refinement abstraction is defined on the basis of the development cycle
(refinement of implementations). It refines, summarises and views conceptualizations, hides or encapsulates details
or manages collections of versions. Each refinement step transforms a schema to a schema of finer granularity.
Refinement abstraction may be modelled by refinement theory and infomorphisms.
Encapsulation removes internal aspects and concentrates on interface components. Blackbox or graybox
approaches hide all aspects of the objects under consideration. Partial visibility may be supported by
modularisation concepts. Hiding supports differentiation of concepts into public, private (with the possibility to
be visible to ‘friends’) and protected (with visibility to subconcepts). It is possible to define a number of visibility
conceptualizations based in inflection. Inflection is used for the injection of combinable views into the given
view, for tailoring, ordering and restructuring of views, and for enhancement of views by database functionality.
Behavioral transparency is supported by the glassbox approach. Security views are based on hiding. Versioning
allows developers to manage a number of concepts which can be considered to be versions of each other.

CROSS REFERENCE
I. DATABASE FUNDAMENTALS
   a. Entity-Relationship Model, Extended Entity-Relationship Model, Object Data Models,
      Object Role Modeling, Unified Modeling Language

REFERENCES