Conceptual Modeling of Internet Sites

Part 3.1: Codesign of structuring, functionality, and interactivity

Conceptual Modeling of Internet Sites
Wintersemester 2014, Gastprofessor@FH Hagenberg, Österreich

15.10.2014

Prof. Dr. Bernhard Thalheim
Information Systems Engineering Group
Computer Science Institute
Kiel University, Germany
Survey

(1) Websites as further development of other presentation media

(2) Theory of media objects

- Codesign of structuring, functionality, and interactivity
- Modelling content
- Modelling of abstraction
- Modelling of website functionality
- Adaption of a website to portfolio, profile, and context
- Refinement

(3) Systematic story boarding
(4) Production of websites
(5) Fine tuning of websites
Overview

- Co-Design - why?
- Structuring (the classical and the non-classical case)
- Functionality [behavior] (the hidden programmer’s cave)
- Advanced views and media types (the long waited link)
- Interactivity (playout of scenarios, actors and interfacing)
- Making co-design working (handling complexity well-educated)
- References, conferences, open problems

Maximal exploitation of database theory and technology for intelligent information systems design support
Modeling is out

SAP Chief Manager 1999: **Modeling is out**!

but: $> 20,000 + > 42,000 + > 87,000 + > 3,500,000$

but **relational schema redundancy** in the SAP data schema: $> 4.5$

but large **runtime and performance problems**

but huge **integration problems**

but hyper-huge **development problems**: instead of integration development once more

but until 1999: **no documentation** on R/3

but: heavy maintenance, installation and extension problems

hyper-redundancy in SAP R/3, e.g., more than 75 address relations

simple update ("change the zip for one street") may take 2-3 days or weeks

SAP database system is initialized within a two weeks time frame and not less
Stone-age Computer Engineering

handicraft programming
except computer game industry
everybody programs from scratch

why pupils are programming websites?
Solution A: Script languages
Modeling of small programs
Meta-modeling

No Engineering Yet

No Engineering Science Yet
Why Do We Need Co-Design

Support of work is the killer requirement
Ease of work is the bargain requirement
Efficiency of work trigger company’s choices
High utility is the shopping argument
Data are not the kernel
Functions, procedures, triggers should support work
Users need to interact in changing exchanges
Users are different (configuration, exception, adaptation support)
Context of work is changing (high flexibility, adaptation to environment)
Large variety of very different users due to computers omnipresence
The Knowledge Gap on Database Design Decisions

- "Partial reality"
- Part of reality
- Things of reality
- Observed property
- "Topic"
- Predicator
- Foundation of decisions
- Modeling decision
- Revision during the development process
- "Schema" as result and partial point of view of a database development process
- "Context"
- acts within
- under usage
- Modeler
- Reference model
- Exactness
- Confidence

Overview
Modelling?
Structuring
Functionality
Interactivity
Methodology

Information

Concept

Topic
Languages (syntax √, semantics ??, pragmatics ????)

- Languages for specification
  - Structures
  - Functions
  - Distribution
  - Interaction
  (object-relational) √
  DML-based ????
  system-based
  interface, XML ???

- Languages of tools
  may be also UML

- Language and wording conventions of developers and teams

- Educational gap

- Binarization, weak types, pointers and other non-sense

Payoff: Babylonian Language Utilization

- Mismatches: structure / function, structure / static constraints, static constraints / maintenance, dynamic constraints in implementations

- Interfaces are going to be developed later

- Distribution in the Las Vegas approach
Application Systems are Task-Oriented

- task: specification of goal-oriented actions;
  subtasks: workflow with activities, restricted by conditions, data and context (organization, policy, environment, channel, ...)

- participatory design (task analysis, user-oriented)
  mapped to essential design (data and function analysis)
  three spaces: task world, system space, interaction space

- website as services (scenario of activities which can be called by the user with supporting protocols and delegation facilities)
  request, indication, response, confirmation
  (a)synchronous, layering, resource-sharing, error-robustness, communication support, (temporary) workspace

- generic user model and profile
Requirements to Co-Design

- Oneness
- Full fledged theory underneath
- Consistency, interpretability
- Scoping to various aspects
- Reasoning facilities
- Translatability
- Extensibility, scalability, integrability, performance
- Team work support, tools
- Methodology
### State Of The Art So Far

<table>
<thead>
<tr>
<th>Concept</th>
<th>Used in practice</th>
<th>Theoretical background</th>
<th>Earliest layer of specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structures</td>
<td>well done</td>
<td>well developed</td>
<td>strategic</td>
</tr>
<tr>
<td>Static semantics</td>
<td>partially used</td>
<td>well developed</td>
<td>conceptual</td>
</tr>
<tr>
<td>Processes</td>
<td>somehow done</td>
<td>parts and pieces</td>
<td>requirements</td>
</tr>
<tr>
<td>Dynamic semantics</td>
<td>some parts</td>
<td>parts and glimpses</td>
<td>implementation</td>
</tr>
<tr>
<td>Services</td>
<td>implementations</td>
<td>ad-hoc</td>
<td>implementation</td>
</tr>
<tr>
<td>Exchange frames</td>
<td>intentionally done</td>
<td>nothing</td>
<td>implementation</td>
</tr>
<tr>
<td>Interfaces</td>
<td>intuitive</td>
<td>nothing</td>
<td>implementation</td>
</tr>
<tr>
<td>Stories</td>
<td>intuitive</td>
<td>nothing</td>
<td>implementation</td>
</tr>
</tbody>
</table>

**Late Specification, Inflexibility, and Unmaintainability**

extension, change management and integration become a nightmare
Languages: Extended ER + DistrLang + SiteLang

Structuring on the basis of an extended ER model that is based on hierarchical predicate logic

Functionality on the basis of HERM/LC with HERM-algebra, HERM/QBE, query forms and transactions with some kinds of dynamic integrity constraints, behavior GCS integrity enforcement instead of rule triggering pitfalls

Interactivity in integrated form based on SiteLang description of dialogue scenes, stories, story space (actors, scenario, dialogue steps)

Distribution through service specification and exchange frames

Translation and transformation methods for compilation of design into other models (logical, physical) or XML see B. Thalheim, Entity-Relationship Modeling. Springer, Berlin, 2000

Development and engineering methods for pragmatism (see my homepage)
Constructs of the Co-Design Languages

Structuring as pair

Structuring := (Structure, Static Constraints)
Structure as (marked) expression on constructors

Functionality as pair (Operations, Dynamic constraints)

Functionality := ((StateChange ∪ Retrieval)Operations , DynamicConstraints)

Operations on the basis of the HERM algebra (for modification and retrieval)
providing a language for generalized views (media types)

...
Constructs of the Co-Design Languages

Distributivity as pair (Services, Exchange Frames)

Distribution := (Service (Informational process, Service Manager, Competence),
ExchangeFrame (Architecture Collaboration Style, CollaborationPattern))

Services on the basis of generalized views (media types)

Interactivity as 4-tuple

Interaction := (StorySpace, Actors, MediaObjects, Presentation)

StorySpace as graph of scenes and activities
Actors are abstractions of user groups
MediaObjects are used by actors and are based on generalized views (media types)
Extended Entity-Relationship Models

**Syntax** of the model

Structuring as **consistent** extension of the classical entity-relationship model

Behavior specification on the basis of generic operations forming the algebra and on the basis of ASM semantics

Interacting of users with the information engine on the basis of their specific scope on the database

Environment: technical context, organizational context, distribution

Views, containers for delivery of information to the user and for accepting information from the user

**Semantics** via hierarchical predicate logics can be extended by pragmatics (variety of semantics)

**Pragmatism** based on the codesign methodology
Structuring in HERM

Structuring - extension of the ER model
- strict set semantics (no pointer semantics)
  1. Complex attributes, entity, relationship, cluster types, types of higher order,
  hierarchical schemata for structuring
  2. Static integrity constraints

Basic data types - parameterized types of the DBMS
  integer := (IntegerSet, {0, s, +, -, *, ÷, }, { =, ≤ })

Attribute type induced on basic data type system
  Name : (FirstNames<(FirstName,use)>, FamName,
  [NameOfBirth,] Title:{AcadTitle} ⊓ FamTitle)
  Contact : (Phone({AtWork}, private), email, ...)
  DateOfBirth :: date
  AcadTitel :: titleType

Entity type - product of attribute types with at least one direct identification
  Person : (Name, Address, Contact, DateOfBirth,
  PersNo: StudNo ⊓ EmplNo, ...)

Structuring in HERM

Relationship type via hierarchical construction

Unary type: Role, specialisation

InGroup : (Person, Group,
            Time(From [,To]), Position)

DirectPrerequisite : (hasPrerequisite: Course, isPrerequisite : Course)

Professor : (Person, Specialization)

Cluster type - disjoint (labelled) union

especially for generalization

JuristPerson : Person \cup\ Company \cup\ Association

Group : Senat \cup\ WorkingGroup \cup\ Association

Constructors \times, \cup, \emptyset construction complete

usually: \times, \cup, < . >, \{ . \}, \mathcal{P}

with underlying type system

and generic operations

plus construction of operations through structural recursion
A Sample Schema

Course \rightarrow Semester \rightarrow Professor \rightarrow Person

Program \rightarrow \{\text{proposed Course}\} \rightarrow \text{Kind}

\text{Course held} \leftrightarrow \text{TimeFrame}

\text{Proposal} \rightarrow \text{Time(Proposal, SideCondition)}

\text{Teacher} \rightarrow \text{Inserted} \rightarrow \text{Responsible4Course}

\text{Request}
Constraints

Static integrity constraints are formulas of the hierarchical predicate logic (with abbreviations)
in “normal definition frame” (may be deontic or strikt)

Keys for identification of objects (esp. entity types)

\[ \text{Key(Person)} = \{ \{ \text{PersNo} \}, \{ \text{Name}, \text{DateOfBirht} \} \} \]

Relationship types with attributes

\[ \text{Key(InGroup)} = \{ \{ \text{Person}, \text{Group}, \text{Time} \} \} \]
\[ \text{Key'(InGroup)} = \{ \{ \text{Person}, \text{Group}, \text{Time}, \text{Position} \} \} \]

\[ \text{Key(Lecture)} = \{ \{ \text{Course}, \text{Semester} \}, \{ \text{Semester}, \text{Room}, \text{Time} \}, \{ \text{Semester}, \text{Teacher}, \text{Time} \} \} \]

Keys defined by the component construction

\[ \text{Name(FirstNames<(FistName,use)>, FamName)} \]

at least one key is ‘inherited’ from the component
Constraints

Functional dependencies for functional associations among groups of attributes

\[ \text{plannedCourse} : \{\text{Sem, Time, Room}\} \rightarrow \{\text{Program}, \text{Prof, Course}\} \]

\[ \text{plannedCourse} : \{\text{Prof, Sem, Time}\} \rightarrow \{\text{Course, Room}\} \]

\[ \text{proposedCourse} : \{\text{Semester, Course}\} \rightarrow \{\text{Prof}\} \]

and other relational constraints

e.g. Domain constraints

\[ \text{Semester.Description} \in \{\text{WS, SS}\} \times \{x|x+1|x \in 1980..99, 2000..03\} \]

Cardinality constraints are restrictions of combinatorics with (min,max)-notation

\[ \text{card(DirectPrerequisite, hasPrerequisite)} = (0,2) \]

\[ \text{card(DirektVoraussetz, isPrequisite)} = (3,4) \]

\[ \text{card(plannedCourse, Course Sem Prof)} = (0,3) \]

\[ \text{card(proposedCourse, Sem Prof)} = (3,5) \] or \[ (0.5) (!!!) \]
Classes and Instances

Classes are sets of (markable) objects of corresponding type

Entity class is the basic class of objects

\[ \beta: \langle (Karl, add), (Bernhard, call) \rangle, \text{Thalheim, \{Prof., Dr.rer.nat.habil, Dipl.-Math.\}}, \text{BTU Cottbus}, \]

\[ \langle \{ +49 355 692700, +49 355 692397 \}, +49 355 824054 \rangle, \]

thalheim@informatik.tu-cottbus.de, 10.3.52, 637861)

(DBIV, Database theory, CS Programm, mandatory, 2+0+0, certificate)

Relationship classes for association of objects of component
classes expanded by properties (through attributes)

\[ \text{Prof}\beta: ( 637861, \text{Database and information systems} ) \]

\[ \text{Senator3}\beta: ( 637861, \text{Senat, (1995,1998), Dekan}) \]

\[ \text{Senator5}\beta: ( 637861, \text{Senat, (2000), Chair}) \]

\[ \text{PrerequDBIVMain}: (DBIV, DBI) \]

\[ \text{CourseDBIVSS02}: (DBIV, SS2002, 637861, SR1, Mo. first pair) \]

\[ \ldots \]

\[ \ldots \]
Classes and Instances

Classes are sets of (markable) objects of corresponding type ....

Cluster classes for disjoint (!), i.a. virtual union of objects of the component classes

{ Senator2\pi: (889721, Senat, (1998,2000), Chair),
Senator5\beta: (637861, Senat, (2000), Chair),
DBIS: (Database and information systems, 637861, IfI),
CBnet\beta: (CottbusNet e.V., 637861, Member, Services Group ) }

Classes are extended by generic operations insert, delete, update, retrieve
may have additional methods
static integrity constraints must remain valid
inherent integrity constraint: existence of components
Classes and Instances

Representation and Storage

possibly with weak value-based OID or label

- either as full objects (with all properties (XML)) or as
- class-separated object (similar to snowflakes) [object-relational representation] or
- hybrid
Functionality

Operations + dynamic integrity constraints

Operations as a generalization of relational algebra

Basic operations: union, difference, intersection, projection, selection, various types of join operations, product, (un-)nesting, power-set, aggregation

Manipulation operations (insert, delete, update) as specific basic operations

Retrieval operations as type-constructing basic operations restricted to components or sub-schemata or unrestricted

Workflows on the basis of transaction models and on control models

Dynamic integrity constraints defined in a temporal logic

Abstract state machine semantics for state changes

Transition constraints restricting state changes caused by operations

General pre- and post-conditions for state changes
### Intensional Functionality Specification

**Business Case:** *Enter information on lectures after being requested*

<table>
<thead>
<tr>
<th>Caller</th>
<th>Organizational unit</th>
<th>Help and auxiliary information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request to responsible person</td>
<td>Chair</td>
<td>Courses, programs, rooms</td>
</tr>
</tbody>
</table>

**Actions**

<table>
<thead>
<tr>
<th>Step</th>
<th>Responsible Person</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Entry</td>
<td>Responsible person of chair</td>
<td><strong>Main information entry</strong>&lt;br&gt;&lt;br&gt; (*Classification</td>
</tr>
<tr>
<td>2. Confirmation step</td>
<td>Responsible person and members of chair</td>
<td><strong>Proofreading, correction, extension for requests, main and other data</strong></td>
</tr>
<tr>
<td>3. Submission step</td>
<td>Responsible person of the chair</td>
<td><strong>Archive in chair folder ; send data to caller; publish at chairs internal page</strong></td>
</tr>
</tbody>
</table>
HERM Query Algebra

Operations are defined on the basis of structural recursion \( \text{op} = \text{src}[e, g, \sqcup] \)
with a value \( e \) of type \( t' \), a function \( g : t \rightarrow t' \) and a function \( \sqcup : t' \times t' \rightarrow t' \):

\[
\text{src}[e, g, \sqcup](\emptyset) = e \\
\text{src}[e, g, \sqcup](\{x\}) = g(x) \quad \text{for } x \text{ of type } t, \\
\text{src}[e, g, \sqcup](X \sqcup Y) = \text{src}[e, g, \sqcup](X) \sqcup \text{src}[e, g, \sqcup](Y) \quad \text{for } X, Y \text{ of type } \{t\} ,
\]
e.g., \( \text{filter}(\phi) = \text{src}[\emptyset, \text{if \_then\_else} \star (\phi \times \text{single} \times (\text{empty} \star \text{triv})), \sqcup] \),

\( \text{sum} = \text{src}[0, \text{id}, +] \)

Operations for tuple types are defined by structural recursion on basic operations
projection \( \pi_i : t_1 \times \ldots \times t_n \rightarrow t_i \), (Cartesian) product \( \times : t \rightarrow t_1 \times \ldots \times t_n \),
reordering \( \rho \), renaming \( \kappa \)

Operations for set types are defined by structural recursion on basic operations union
\( \cup \), difference \( \setminus \), constant, singleton element,
e.g., join operation

Operations on function types are defined by structural recursion on basic operations
composition \( \star : (t_2 \rightarrow t_3) \times (t_1 \rightarrow t_2) \rightarrow (t_1 \rightarrow t_3) \), evaluation \( \text{ev} : (t_1 \rightarrow t_2) \times t_1 \rightarrow t_2 \) and abstraction \( \text{abstr} : (t_1 \times t_2 \rightarrow t_3) \rightarrow (t_1 \rightarrow (t_2 \rightarrow t_3)) \)

Conversion operations from tuple to set types, from function types to collections, etc.

Operations for URL extension using labels \( l \) extending the type system to

\[
t = b \mid l \mid t_1 \times \ldots \times t_n \mid \{t\} \mid [t] \mid l : t
\]
that is restricted to rational trees (number of different subtrees is finite)
Provide data on students who have successfully completed those and only those courses which have successfully been given or which are currently given by the student's supervisor?
Advanced Views and Media Types

**Specification Frame**

generate *MAPPING*: $\text{VARS} \rightarrow \text{OUTPUT STRUCTURE}$

from $\text{DATABASE TYPES}$

where $\text{SELECTION CONDITION}$

represent using $\text{GENERAL PRESENTATION STYLE}$

& $\text{ABSTRACTION (Granularity, measure, precision)}$

& $\text{ORDERS WITHIN THE PRESENTATION}$

& $\text{HIERARCHICAL REPRESENTATIONS}$

& $\text{POINTS OF VIEW}$

& $\text{SEPARATION}$

browsing definition $\text{CONDITION}$

& $\text{NAVIGATION}$

functions $\text{SEARCH FUNCTIONS}$

& $\text{EXPORT FUNCTIONS}$

& $\text{INPUT FUNCTIONS}$

& $\text{SESSION FUNCTIONS}$

& $\text{MARKING FUNCTIONS}$
XML as the display medium

XML specifications can be automatically generated out of this view
Co-Design
15.10.2014
B. Thalheim

Overview
Modelling?
Structuring
Functionality
Interactivity
Methodology
Content
Information
Concept
Topic

Algebraic expressions for views

Views for internet presentation as Read-Only-View

Archiv view as materialized Read-Only-View

\[\text{Slice SS00/01 with Archiv.Semester := } e(\text{Semester})\]
\[\text{for } e = \sigma_{\text{Bezeichn}=\text{SS00/01}}\]

\[\text{Archiv.Course := } e(\text{CourseHeld}[\text{Course}])\]

\[\text{Archiv.Person := } e(\text{CourseHeld}[\text{plannedCourse}[
\quad \text{proposedCourse}[\text{Responsible4Course : Person}]]])\]

\[\text{Program, Kind, Professor analogous}\]

\[\text{Archiv.CourseHeld := } e(\text{CourseHeld}[\text{plannedCourse}[
\quad \text{proposedCourse [ Course, Program, Teacher:Professor,
\quad \text{Responsible4Course : Person}, Kind]]})\]

Be careful: changing set of integrity constraints!

Insert view for new course proposals: als Read-Write-View with identifiable sub-views and side conditions with optional and mandatory components

Media object schema as containers based on views with container functionality and attached functions representation bound by order, adhesion, cohesion of types taylored by user profile, user environment, history, channel capacity are associated with dialogue scenes
View: Insertion view for new course proposals

- **Course**
  - retrieve, select, input
  - **Chair**
    - retrieve
  - **Person**
    - retrieve
  - **Program**
    - retrieve, select
  - **Room**
    - retrieve, select
  - **Professor**
    - retrieve, select
  - **Semester**
    - retrieve
  - **Teacher**
    - retrieve, select
  - **Wish**
    - insertedBy = "SecrKK"
  - **Proposal**
    - Time(Proposal, SideConditions)

- **Course**
  - (1,1)

- **Description** = "SS01/02"
- **Default** = Prof β
- **ShortDescr** = "DBIS"
- **InsertedBy** = "SecrKK"
- **Responsible4Course** = "β"

- **Kind**
  - retrieve, select
  - **Proposed Course**
    - input

- **Concept**
  - **Topic**
  - **Information**
  - **Content**
  - **Overview**
  - **Modelling**
  - **Structuring**
  - **Functionality**
  - **Interactivity**
  - **Methodology**
View: Insertion view for new course proposals

Course
- retrieve, select, input
- required

Semester
- retrieve description
- (1,1)
- Description = “SS07/08”

Professor
- retrieve, select
- retrieved
- default = Prof\(\beta\)

Person
- retrieve if change

Room
- retrieve, select

Proposal
- Time(Proposal, SideConditions)

Kind
- retrieve, select

Course
- retrieve

Chair
- retrieve if change
- ShortDescr = “ISE”

Co-Design
15.10.2014
B. Thalheim

Overview
Modelling
Structuring
Functionality
Interactivity
Methodology
Outlook: Media Types, Media Object Suites

Theoretical Basis

- Raw media types = \( (\text{cont}(M), \text{sup}(M), \text{view}(M), \text{op}(M)) \)
  - content type \( \text{cont}(M) \)
  - set of supertypes \( \text{sup}(M) \)
  - \( \text{view}(M) = Q(S_{\text{inp}}, S_{\text{outp}}) \) HERM view
  - generic functions \( \text{op}(M) \) for changing the database
- Attached operations: (signature, selection type, body)
  - selection type - supertype of \( \text{cont}(M) \)
    - e.g. generalization/specialization, reordering, browsing, linking, surveying, searching, join
- Media type: raw media type + unit extension
  + order extension + cohesion/adhesion + hierarchical versions
- Usage modeling: usage dimensions, scales, user profiles, user kind
- Container = \( (\text{cont}(C), \text{layout}(C), \text{kind}(C)) \)
  for shipping and representation
Rule Triggering May Fail

Rule triggering is nice but

- termination problem
- confluence property (Church-Rosser)

and it may fail:

Consider a simplistic example: three classes $A, B, C$

integrity constraints: $A \subseteq B, A \subseteq C$ (inclusion constraints)

$B \parallel C$ (exclusion constraint)

rule triggering system:

\[
\begin{align*}
i_A(x) & \rightsquigarrow i_A(x); i_B(x), & i_A(x) & \rightsquigarrow i_A(x); i_C(x), \\
d_B(x) & \rightsquigarrow d_B(x); d_A(x), & d_C(x) & \rightsquigarrow d_C(x); d_A(x) \\
i_B(x) & \rightsquigarrow i_B(x); d_C(x), & i_C(x) & \rightsquigarrow i_C(x); d_B(x)
\end{align*}
\]

general compression rules:

\[
\begin{align*}
i_T(x); d_T(x) & \rightsquigarrow d_T(x), & d_T(x); i_T(x) & \rightsquigarrow i_T(x)
\end{align*}
\]

applying this ECA set to the operation $i_A(x)$ allows to derive either

\[
\begin{align*}
i_C(x); d_B(x); d_A(x) \text{ or } & \\
i_B(x); d_C(x); d_A(x) \text{ or } & d_B(x); d_C(x); d_A(x)
\end{align*}
\]

Therefore: be careful during design of constraints and structures!

GCS result: FAIL
Can The Designer Repair ECA Faults?

\[ R_1 = (A :: INT, C :: INT) \]
\[ R_2 = (B :: INT, D :: INT) \]

\[ \mathcal{I}_1 \equiv R_1[A] \subseteq R_2[B] \quad \text{(inclusion dependency)} \]
\[ \mathcal{I}_2 \equiv R_2 : D \rightarrow B \quad \text{(functional dependency)} \]
\[ \mathcal{I}_3 \equiv R_1[C] \parallel R_2[D] \quad \text{(exclusion dependency)} \]

\[ r_1 = \text{ON insert}_{R_1}(a, c) \]
\[ \quad \text{IF } a \notin R_2[B] \quad \text{THEN insert}_{R_2}(a, ?) \]
\[ r_2 = \text{ON delete}_{R_2}(b, d) \]
\[ \quad \text{IF } b \in R_1[A] \land b \notin R_2[B] \quad \text{THEN delete}_{R_1}(b, ?) \]
\[ r_3 = \text{ON insert}_{R_2}(b, d) \]
\[ \quad \text{IF } (b', d) \in R_2 \land b' \neq b \quad \text{THEN fail} \]
\[ r_4 = \text{ON insert}_{R_1}(a, c) \]
\[ \quad \text{IF } c \in R_2[D] \quad \text{THEN delete}_{R_2}(?, c) \]
\[ r_5 = \text{ON insert}_{R_2}(b, d) \]
\[ \quad \text{IF } d \in R_1[C] \quad \text{THEN delete}_{R_1}(?, d) \]

\[ \text{insert}_{R_1}(a, c) \quad \text{AFTER} \quad \text{INSERT} \quad a \notin R_1[A] \]
\[ \text{AFTER} \quad \text{INSERT} \quad c \notin R_2[D] \]
Loving or Hating ECA?

Size of triggers: Triggers tend to become large, incomprehensible, resist maintenance and cause the *trigger crisis*.

Rule triggering is safe iff the structuring (structure and static integrity constraints) is strictly hierarchical (stratifiable) \(\text{this property is undecidable}\).

Effect preservation of rule triggering systems is undecidable.

- Insert operation: \(\text{Ins}(db, set) \sqsupseteq db\)
- Delete operation: \(\text{Del}(db, set) \sqsubseteq db\)

Practical advice of DBMS: Whenever a problem occurs, disable rule triggering.

- Sybase: at most one trigger per event and per relation
- The sad example: LAUBAG lost their data in 1996

All execution models fail: For each execution model and a correctness condition

- a set of types and a set of constraints exists such that the ECA enforcement does not satisfy the correctness condition

Therefore: either use GCS or be careful during design of constraints and structures!
Integrity Enforcement via GCS instead of ECA

Basic idea: Enhance (specialize) [basic] operations of the DBMS in such a way that the application of the operations to a consistent database will lead to a consistent database again.

Restriction: Use the smallest enhancement.

Greatest consistent specialization allows to derive integrity enforcing operations for the generic database functions.

reflection + predicate transformer $\Rightarrow$ greatest consistent specialization

state variable $\quad$ state space

state constraints $\quad$ extended state space

Extending GCS by weakening the order, by considering specific workflows instead of all, by distributed enforcement.
The GCS Approach

Linguistic reflection:
reflection types $S C H E M A_{rep}, C L A S S_{rep}, T Y P E_{rep}, M E T H O D_{rep},
C O M M A N D_{rep},$

$\text{insert} : S :: S C H E M A_{rep} \times C :: C L A S S_{rep} \rightarrow M E T H O D_{rep}$
$\text{delete} : S :: S C H E M A_{rep} \times C :: C L A S S_{rep} \rightarrow M E T H O D_{rep}$
$\text{update} : S :: S C H E M A_{rep} \times C :: C L A S S_{rep} \rightarrow M E T H O D_{rep}.$

one macro generic with signature $S C H E M A_{rep} \rightarrow S C H E M A_{rep}$
operation on state space $X$ -
characterized by two predicate transformers $wp(S)$ and $wlp(S)$
assign to some postcondition $\mathcal{R}$
weakest (liberal) precondition of $S$ to establish $\mathcal{R}$

$wlp(S)(\mathcal{R})$ - initial states with all terminating executions of $S$ reach final state
characterized by $\mathcal{R}$ (provided $S$ defined)

$wp(S)(\mathcal{R})$ - initial states with all executions of $S$ terminate, reach a final state
characterized by $\mathcal{R}$ (provided $S$ defined)
GCS Approach (2)

$Y$-operation $S$ on $X$

$X$-constraint $I$

Greatest Consistent Specialization (GCS) of $S$ with respect to $I$:

$S_I$ specializes $S$ ($S_I \sqsubseteq S$),

$S_I$ is consistent with respect to $I$

$S_I$ is maximal for each $X$-operation $T$ satisfying the properties: $T \sqsubseteq S_I$

specialization order $T \sqsubseteq S$

$wp(S)(true) \Rightarrow wp(T)(true)$ and

$wlp(S)(\mathcal{R}) \Rightarrow wlp(T)(\mathcal{R})$

GCSs always exist

compatible with conjunctions - universal conjunctivity

subsumption freeness

usually non-deterministic

quasi-deterministic branches of a GCS

maximal quasi-deterministic consistent specializations (MQCSs)

quasi-determinism - determinism up to the selection of some values
The Computation Of GCS

\[ R_1 = (A :: \text{INT}, C :: \text{INT}) \]
\[ R_2 = (B :: \text{INT}, D :: \text{INT}) \]

\[ I_1 \equiv R_1[A] \subseteq R_2[B] \quad \text{(inclusion dependency)} \]
\[ I_2 \equiv R_2 : D \rightarrow B \quad \text{(functional dependency)} \]
\[ I_3 \equiv R_1[C] \parallel R_2[D] \quad \text{(exclusion dependency)} \]

ON \quad \text{insert}_{R_1}(a, c)

IF \quad c \in R_2[D] \quad \text{THEN} \quad \text{Fail}
ELSE \quad \text{insert}_{R_1}(a, c) \quad \text{IF} \quad a \notin R_1[A]
\quad \text{THEN} \quad \text{insert}_{R_2}(a, d) \quad \text{WHERE} \quad d \notin R_1[C] \cup R_2[D]
Interactivity Specification Through SiteLang

Story space - space of all specified stories
Story - labeled graph of different integrated scenarios
Scenario - run through a system by a cooperating set of actors
Scene of scenario - consistent set of dialogue steps
Dialogue step - conditional actions of an enabled actor on page based on provided media objects

交互性步骤 - 条件动作的使能者在页上的依据提供的媒体对象

交互空间 - 所有可能的交互故事空间
系统空间 - 系统的玻璃盒（所有运行集成）

\[
ru_i: \quad \text{on event if precond do actions accept on postcond}
\]

\[
\text{if precond}_{ru_i} \quad \text{and event then actions and CommitState}_{ru_i} = \text{toCommit}
\]

\[
\text{if CommitState}_{ru_i} = \text{toCommit and postcond}_{ru_i} \quad \text{then Commit}_{ru_i}
\]

\[
\text{if CommitState}_{ru_i} = \text{toCommit and } \neg \text{postcond}_{ru_i} \quad \text{then Undo}_{ru_i}
\]
Interaction Modeling

**Story** as labeled di-graph $S = (V, E, \lambda, \kappa)$

$V$ - scenes, $E \subseteq V \times V$ - transitions

media object assignment $\lambda : V \rightarrow \{\text{MediaObj}\}$

- representation through media objects
- with permitted rights, permitted roles, obligations of actors

MediaObject (Container, ManipulatRequests, SuppliedFunctions)

activity assignment $\kappa : E \rightarrow \{\text{Activity}\}$

- activity = ( actor(profile) , task,
  context (equipment, channel, rights, roles, particular),
  representation (style, default, emphasis, ...))

**Scenario** - run through the system

- with cumulative history (and adaptation)
- consists of scenes
  - visited sequentially or in parallel by actors

story space - composition of sequences
Scene Specification

**Scene expression** \( v \in \text{Alg}(\text{DialogueStep}) \)

- basic control: sequence \( ; \), parallel split \( \mid \), exclusive choice \( \mid \), synchronization \( \langle \text{sync} \rangle \), simple merge \(+\)

- advanced branching and synchronization control: multiple choice, multiple merge, discriminator, n-out-of-m join, synchronizing join

- structural control: arbitrary cycles \( * \), implicit termination \( \downarrow \)

- control on multiple objects: CMO with a priori known design time knowledge, CMO with a priori known runtime knowledge, CMO with no a priori runtime knowledge; CMO requiring synchronization (synchronization edges)

- state-based control: deferred choice, interleaved parallel executing, milestone

- cancellation control: cancel action, cancel case

**FRAME**: \[ \text{on event if } \alpha \text{ doScene } v \text{ accept on } \gamma \]

representation of scenes via Montages

control and object flow specification
# Scene Specification

<table>
<thead>
<tr>
<th>scene</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
</tr>
<tr>
<td>content</td>
</tr>
<tr>
<td>variants</td>
</tr>
<tr>
<td>application</td>
</tr>
<tr>
<td>applicability</td>
</tr>
<tr>
<td>usability profile</td>
</tr>
<tr>
<td>description</td>
</tr>
<tr>
<td>structuring: structure, static constraints</td>
</tr>
<tr>
<td>implementation</td>
</tr>
<tr>
<td>implementation</td>
</tr>
<tr>
<td>associated scenes</td>
</tr>
<tr>
<td>mandatory</td>
</tr>
</tbody>
</table>
Representation of Dialogue Steps within a Dialogue Scene

dialogue scene

control(event,preCondition,acceptCondition)

next dialogue step

manipulation sub-request

enabled actor

transition according to dialogue scene expression

dialogue step

involved actors

story scene sequence

media object

representation style

context, task
<table>
<thead>
<tr>
<th>dialogue step</th>
</tr>
</thead>
<tbody>
<tr>
<td>header</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>title</td>
</tr>
<tr>
<td>container</td>
</tr>
<tr>
<td>content</td>
</tr>
<tr>
<td>text</td>
</tr>
<tr>
<td>multimedia object</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>functionality</td>
</tr>
<tr>
<td>operations</td>
</tr>
<tr>
<td>algorithmic object</td>
</tr>
<tr>
<td>tayloring style</td>
</tr>
<tr>
<td>ordering by hierarchies</td>
</tr>
<tr>
<td>adhesion</td>
</tr>
<tr>
<td>adaptation</td>
</tr>
<tr>
<td>interaction style</td>
</tr>
<tr>
<td>control style</td>
</tr>
<tr>
<td>actors applicable</td>
</tr>
</tbody>
</table>
Scene in a Storyboard

Course Data Entry Scene Extended With Internal Negotiations

Chairs Lecture Proposal Scene

- Login by chairs responsible
- Accept course demand
- Generate new course proposal
- Collect seminar proposals
- Settle data for proposal
- Settle data for seminar
- Entry of necessary data
- Auxiliary & historic data
- Formulation of side conditions
- Submission chair data
- Assignment of courses to members
- Negotiation of assignments by members
- Confirmation
- Assignment of courses to members
- Negotiation of assignments by members
- Formulation of side conditions
Dialogue Steps for Event etc. Search

event search scene

- **individual request step**
- **property-based search**
- **entry step**
- **result & clarification step**
- **map browsing step**
- **target seeking step**
- **points of interest**
- **booking step**

Dialogue Steps:

1. **individual request step**
2. **property-based search**
3. **entry step**
4. **result & clarification step**
5. **map browsing step**
6. **target seeking step**
7. **points of interest**
8. **booking step**
Formale Spezifikation von Scenarien

Scenario: ⟨name of the scenario⟩
Scenes: ⟨list of scene names⟩
Start Scene: ⟨scene name⟩
Final Scenes: ⟨list of scene names⟩
Actions: ⟨list of action names⟩
Transitions: ⟨list of transitions⟩
Process Expression: ⟨SiteLang process⟩
Beispiel aus einem Redaktionssystem
Formale Spezifikation of Scenes

Scene: \( \langle \text{scene name} \rangle \)

Actions: \( \langle \text{list of action names} \rangle \)

Roles: \( \langle \text{list of role specifications} \rangle \)

User Types: \( \langle \text{list of user type names} \rangle \)

Defining Scenario: \( \langle \text{scenario name} \rangle \)

Acceptance Condition: \( \langle \text{condition} \rangle \)
Formale Spezifikation of Actions

Action: ⟨action name⟩
Precondition: ⟨condition⟩
Postcondition: ⟨condition⟩
Roles: ⟨list of role names⟩
User Types: ⟨list of user type names⟩
Enabling Event: +⟨list of actions⟩ −⟨list of actions⟩
Triggering Event: +⟨list of actions⟩ −⟨list of actions⟩
Manipulation Requests: ⟨list of manipulation requests⟩
Formale Spezifikation von Tasks

Task: ⟨name of the task⟩
Goal: ⟨condition⟩
Subtasks: ⟨list of subtask names⟩
Problem: ⟨problem statement⟩
Actions: ⟨list of action names⟩
Precondition: ⟨condition⟩
Event: ⟨action⟩ by ⟨role name⟩
Scenario: ⟨scenario name⟩
Relationship: ⟨description⟩
Description: ⟨textual description⟩
Participants: ⟨list of users⟩
Required Content: ⟨list of content items⟩
Produced Content: ⟨list of content items⟩
Result: ⟨textual description⟩
Starting situation: ⟨list of situation parameters⟩
Closing condition: ⟨acceptance constraints⟩
Execution context: ⟨textual characterization⟩
Enables: ⟨list of task names⟩
Specialises: ⟨list of task names⟩
Application of SiteLang Specification

Deriving the Structuring Schemata

- **purpose**: structure and order information:
  - arbitrary with ambiguities, ellipses, definitions
  - heterogeneous (e.g., granularity, formats, content)
  - perspective-driven (e.g., client support)
  - intension-driven (e.g., vendor policy, marketing)

- **choice of order**:
  - order criteria: subject, tasks, usage, metaphors
  - fairly random (e.g., vendor policy) or clearly defined, easily understandable order (e.g., telephone book)
  - combine different orders

- **classification**:
  - mono- or polydimensional, mono- or polyhierarchies
Application of SiteLang Specification

Deriving the Content Structure

- goal: structure the content, though often determining the navigation as well
- functionality is considered to be attached
- hierarchical structures:
  - easy to achieve, but difficult to use
  - possible use in catalogues, ontologies, etc.
  - includes directed acyclic graphs
- hypermedia structures:
  - highly flexible, but risk to loose orientation
  - the conceptual model relies on infinite (rational) structures and is difficult to understand
- underlying database structures: determinant by different design desiderata (non-redundancy, consistency, fast accessibility, etc.)
Application of SiteLang Specification

Deriving the Navigation Structure

- goal: help user to keep orientation within a site
- do not give up flexibility goal

- navigation systems:
  - hierarchical navigation via simple backtracking
  - global navigation to enable general vertical and horizontal navigation
  - local navigation extending the global one in a closed subarea
  - ad-hoc navigation via meaningful anchors and hyperlinks

- navigation aids:
  - explicit map (floor plan, structuring schema) of the site as long as this is two-dimensional
  - separation of pages into primary information and navigation information
  - use of a content description
  - use of an index / catalogue
  - markups in a navigation system: headers, meaningful anchors, meaningful icons, keywords
Application of SiteLang Specification

Extending Navigation by Exploration Techniques

- Development of the exploration techniques for complex navigation spaces
  - Fish-eye view techniques (center in scope, rest compressed)
    - 3D-fish-eye techniques
    - adorned fish-eye views
    - fish-eye views with transformation of coordinates (radial (locally or globally), orthogonal (locally or globally), 3-dimensional (implicitly or explicitly))
    - filtered fish-eye views (hierarchically or by graph structures)
  - Zoom navigation techniques
  - Non-linear visualization of navigation based on focus points or by multi-point hyperbolic representation
  - Adopting size by weight functions (measures of relevancy or importance, size of data)
- Development of visualization alternatives
Application of SiteLang Specification

Extending by Retrieval and Search Facilities

- goal: support users to find specific information:
  - ideal: system acts like a skilled, experienced librarian
  - usage factors:
    - ability to formulate search query
    - precision of expectation
- search-engine for the site?
  - only necessary, if there is a frequently changing content as in database-backed sites
  - can otherwise be replaced by sophisticated browsing / navigation
- user support:
  - explanation: connectives, string versus keyword, scope of search, etc.
  - feedback: recall, precision as in Information Retrieval
  - integration of search, browsing and navigation aids
Making Co-Design Working: Abstraction Design Layers

Co-Design
15.10.2014
B. Thalheim

Overview
Modelling?
Structuring
Functionality
Interactivity
Methodology

Content
Information
Concept
Topic
Making Co-Design Working

General Description

**Strategic** or motivation or application domain layer:

- HERM (concept map (concept)),
- HERM (functionality (feature)),
- DistrLang (contract sketch (contract, quality criteria))
- SiteLang (application story (application step))

1. Developing visions, aims and goals
2. Analysis of challenges and competitors

**Stakeholder contract** (“Lastenheft”)
Making Co-Design Working

**Requirements Specification**

**Requirements acquisition** layer:

- HERM (sketch (rough type)),
- HERM (business process (business step)),
- DistrLang (contract opportunities),
- SiteLang (story (event))

(3) Separation into system components
(4) Sketching the story space
(5) Sketching the view suite
(6) Specifying business processes

**IS development and system documentation** (“Pflichtenheft”)
Making Co-Design Working

Usage, Usability and Application Stories

**Business user** layer:

- HERM (skeleton (application type)),
- HERM (activity (action)),
- DistrLang (media types, contract),
- SiteLang (plot (theme), actors, media types)

(7) Development of scenarios of the story space

(8) Elicitation of main data types and their associations

(9) Development of kernel integrity constraints, e.g., identification constraints

(10) Specification of user actions, usability requirements, and sketching media types

(11) Elicitation of ubiquity and security requirements

Playout system specification
Making Co-Design Working

Conceptual layer:

- HERM (schema(types)),
- HERM (workflow (process)),
- DistrLang (service space, exchange frame),
- SiteLang (story space, actors, media types, presentation)

(12) Specification of the story space
(13) Development of data types, integrity constraint, their enforcement
(14) Specification of the view suite, services and exchange frames
(15) Development of workflows
(16) Control of results by sample data, sample processes, and sample scenarios
(17) Specification of the media type suite
(18) Modular refinement of types, views, operations, services, and scenes
(19) Normalization of structures
(20) Integration of components along architecture

Conceptual schemata
Making Co-Design Working

Implementation Layer

**Implementation** layer:

- o-r DDL (o-r schema (relation)),
- PL (module) language (program, trigger, sp, ...),
- Distribution specification language (distributed system (distribution, protocol, call))
- Dialog system language (presentation space (working sheet))

(21) Transformation of conceptual schemata into logical schemata, programs, and interfaces

(22) Development of logical services and exchange frames

(23) Developing solutions for performance improvement, tuning

(24) Transformation of logical schemata into physical schemata

(25) Checking durability, robustness, scalability, and extensibility

Program library
The Onion Approach

Presentation engine

Container engine

Media object engine

View handler

XML scene onion

Container onion

Media object onion

XML suite

DBS

DBMS

Virtual \lor materialized views

Update views

Survey, landmark, indexing, I/O,

Navigation, integration etc. functions

Services packages, wrapping functions,

Dialogue scene and scenario functions

Actor profiles, profile adaptation, equipment adaptation,

Channel adaptation, decomposer, style extension