Talks 2012 Given By Bernhard Thalheim

Selected and Compiled Version of All 2012 Talks
Conferences, Tutorials, Miscellaneous

2012

Bernhard Thalheim
Dr. rer.nat.habil.
Prof. @ Christian Albrechts University at Kiel, Germany
Department of Computer Science
Information Systems Engineering Group

(*) Kolmogorov Professor h.c. @ Lomonossov University Moscov, Russia
Overview

(1) Analysis-Driven Data Collection, Integration and Preparation for Visualisation, EJC 2012, Prague, 6.6.2012

(2) Co-Design of Structure, Functionality, Distribution, and Interaction, Eötvös Loránd University of Sciences, Kolloquium, Budapest, 4.7.2012

(3) Database and Information Systems Theory, Eötvös Loránd University of Sciences, Kolloquium, Budapest, 5.7.2012


(6) Pattern-Based Construction of Large Information Systems Schemata, Kolloquium, Technical University Tampere at Pori 8.2.2012


(8) Query and Answer Forms for Sophisticated Database Interfaces, NLDB 2012, Groningen, 5.6.2012


(10) The Extended Entity-Relationship Modelling Language HERM, Eötvös Loránd University of Sciences, Kolloquium, Budapest, 3.7.2012
Not In This Collection

(13) Information Modelling and Data Mining, EJC 2012, Prague, 7.6.2012
(17) Dynamic & Generic Workflows, INDYCO workshop Kickoff meeting, Linz, 11.10.2012
(21) Science and Art of Modelling, DFG, Bonn, 5.11.2012
(22) Grounding and Basis of Models - Beyond T. Kuhn (in Russian), Kolmogorow-Lecture, Lomonossow-University, Moscow, Russia, 25.12.2012
(23) Culture, Science and Art of Modelling, Kolmogorow-Lecture, Lomonossow-University, Moscow, Russia, 27.12.2012
are given in the 2012 Almanach of the Technical Faculty of Christian-Albrechts University at Kiel.

or provided on demand.
The English and German talks which are not in this collection will be also send on demand.

Note: In 2012 our ISE group has published 29 papers.

Overall # of talks given by B. Thalheim in 2011: 24.
Furthermore 2 talks have been given by M. Berg,
2 talks by F. Förster, 2 talks H.-J. Klein, and
2 talks by O. Sörensen.

This list does not include internal talks of the ISE group.
Publication List for 2012


Publication List for 2012


Analysis-Driven Data Collection, Integration and Preparation for Visualisation

Analysis Space + Data Collectors + Inspection Charts
@ European Japanese Conference, EJC 2012, Prague

6.6.2012

Yasushi Kiyoki & Bernhard Thalheim
(1) Keio University, Fujisawa, Kanagawa, Japan
(2) Kiel University, Germany
(*) Kolmogorow Professor h.c. of Lomonossow University Moscov
Plan for this Talk

Observations for some solutions
- challenges of real-life applications
- limitations of storage and computation

Exploration intelligence based on information demand
- intelligence is a ‘wide field’
- separation of concern results in specific support

Ideas and inspirations
- learning from the past
- without forgetting reasons for this better state

Informed analysis
- since we do not need anything
  “think first before doing anything”

Analysis space depending on information demand
- to be solved: how to describe, how to support
  enabler for systematic exploration

Collectors and inspectors
- as supporting facilities
  based on event-oriented computing
An Application: Traffic Analysis
An Application

State-of-art
Analysis
Inspiration
Informed analysis
Analysis space
Collectors
Inspectors
Conclusion
The Essence of the Solution

Microdata: origin-destination-(arrival)time data (GB)
Mesodata: purged and extended by potential train runs, corrections by train weight (hitting TB)
Analysis data: dynamic railway space analyser data (ranging to many TB)
Presentation data: visual inspection data
The Main Data Source

\((\text{check\_in\_})\ o(\text{origin\_station}),\)

\((\text{check\_out\_})\ d(\text{destination\_station}),\)

\((\text{check\_out\_})\ t(\text{time\_at\_destination}))\)

<table>
<thead>
<tr>
<th>Passenger</th>
<th>Date</th>
<th>Origin</th>
<th>Destination</th>
<th>Ticket</th>
<th>6-6</th>
<th>6-7</th>
<th>7-8</th>
<th>...</th>
<th>23-24</th>
<th>Sum per day</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Routing</th>
<th>Date</th>
<th>Origin</th>
<th>Destination</th>
<th>Set time</th>
<th>RouteID</th>
<th>Order Train</th>
<th>List Of Lines</th>
<th>Train Name</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>...</th>
<th>Direction</th>
<th>List Of Transfer Stations</th>
<th>Departure From Transfer Station</th>
<th>Station Get</th>
<th>To</th>
<th>Arrival At Destination Station</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weight data</th>
<th>Date</th>
<th>Station 1</th>
<th>Station 2</th>
<th>Line Name</th>
<th>Train Name</th>
<th>Estimated number of passengers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Analysis-Driven Data Collection

6.6.2012

Kiyoki/β

State-of-art Analysis

Inspiration

Informed analysis

Analysis space

Collectors

Inspectors

Conclusion

Concept

Topic
The inspection chart for outgoing traffic from Shinjuku station
Passenger-flow (arrival-stations) departed
from Shinjyuku Station (6-9am)
The inspection chart for incoming traffic to Shinjuku station (6-9am)
An Application: Problems to Tackle

Various levels of granularity at the same time

Complex filtering at real-time; high database performance

Complex windowed aggregation nested aggregates incrementally updated

Multiplexing and demultiplexing for streams of micro data

Data mining within the stream sophisticated data mining techniques

Extraordinary sizes of the meso and analysis databases

Uncomplete micro data input by data collection facilities

Meta-data injection difficult and requires high performance
Data Exploration Intelligence

Analysis-Driven Data Collection
6.6.2012 Kiyoki/β

State-of-art Analysis
Inspiration
Informed analysis
Analysis space
Collectors
Inspectors
Conclusion

Computation: Logics, aggregation

Preparation

Coordinated requests

Adequate presentation

Tasks, portfolio, demand
Profile, context

on demand, on profile, on portfolio, on context
Inspired by Analog Computers

Monitoring techniques capturing data on need generation of differentials and integrals

Analog computer MEDA42_2_k
Inspiration: Blind Search versus Informed Search

Depth-first-search

Breadth-first search

Beam search

Best-first search

Branch and bound

A* algorithm

Evolutionary algorithm

Dynamic Programming

AO* algorithm

Hill-climbing

Trees and adversarial search

Min-Max Procedure

...
Inspired by Event-Based Computing

Data from external sources
History of values and not just only the most recently reported data.

Trigger-oriented
Data from multiple streams of data
with a different scale, granularity and scope

Data is volatile, stale or intentionally omitted for computational or data collection reasons

Real-time requirements

Agile observers (collectors) collect data within their data demand

Realisation: publish-subscribe + observe + pipe and filter
Inspired by Performance Monitors

Data Collector for MS SQL Server 2008

The Kiel approach to performance analysis, monitoring based on profiles of systems, portfolio of applications calibration and inverse modelling based on data sets collected during observation periods used for performance forecasting (see EJC’2010)
Analysis Profiles, Demands and Portfolio

Profile support for the analysis

- Analysis profile
- Data profile
- Group profile

Information demand support depending on specifics

- Demand, need, resulting quality
- Focus and scope
- Context framework
- Formulation, language

Portfolio support for the analysis

- Tasks: characteristics, execution, result, control
- Data sets to be considered
- Background information and metadata
- Restrictions: subtasks, environment
Be however Careful with Aggregation of Data

Aims and assumptions

Grouping of data, records, objects based on selections

Summarisability depending on completeness and disjointness assumptions

Detection of misleading aggregations and prevention from misapplication

Invariance properties of aggregation operators

Harmonisation of aggregation and abstraction without forgetful handling

Preservation or transformation of semantics or explicit semantics transformation

Derivation of repair functions in the case of damages
Multi-Layered Architectures: Special Case of Sensor Databases

Various levels of granularity at the same time

Complex filtering at real-time

Complex windowed aggregation including nested aggregates

Multiplexing and demultiplexing for streams

Data mining within the stream e.g. frequent item sets, pattern matching, similarity search, learning

Stream combination and joins on observation and demand

Windowed queries as a second-layer generic query

Unreliable and uncontrollable input due to the input devices

Meta-data injection for later summary evaluation
The Data Exploration Cycle as Starting Point for the Data Analysis Advisor

Problem and goal formulation

Real-life system

Structure and function analysis
System definition (isolating abstraction)

Empirical data

Data space formation / formalisation

Conceptional model

Formalisation/Algorithmisation (generalising abstraction)

Formal model

Programming

Computer model

Model checking

Applying results

Results

Model verification

Model validation
A Solution with High Potential: The Event-Oriented Observer System

Analysis space model with explicit specification of data need for analysis

Analysis-driven filtering and summarisation

Data collectors based on analysis portfolio

Inspection charts
**Analysis Space**

**Analysis interest** based on goals, intentions, targets

**Analysis scenario** as a small ‘workflow’ with
candidate facts, measures, dimensions, and hierarchies;
list of possible queries

**Analysis data demand** necessary for completing analysis tasks
with the right granularity, level of detail

**Analysis escort data** from other resources for improvement

**Analysis correction and calibration data** for quality analysis

**Analysis side data** for explanation of the analysis results

**Analysis context** restricting or simplifying the analysis
Analysis Demand

Analysis situation through comparing initial situation with potential outcome

Analysis suppositions and necessary data for support or falsification

Analysis concept space supporting ‘semantification’ of the task and of results

Analysis phenomena

Analysis data models as model suite

Analysis agents for executing the analysis

Analysis termination and goal tests whether a given state or state set satisfies the goals
Analysis Portfolio

Analysis plan with steps/tasks, tools/instruments, analysis space exploration tools (e.g., informed search), adaptation/transformation, and quality control

Analysis task with subtasks (initial state, potential goal states, applicable state transformations, constraints), objectives, executables, and restrictions

Analysis hypotheses space

Analysis algorithmics applicable for analysis

Analysis data with input views $V_C$ and functions for potential inspection charts $C$ based on the meso databases (shuffling direct source, escort, side, calibration data)

Analysis potential outcome and potential failures
Data Collectors

Collector contract based on the policy, e.g., subscribe what is published

Collector definition as active media type $C = (S_C, \Sigma_C)$

$S_C = collector (Structure, CollectorQueryForm(ViewImportingExportingFeatures))$

input views $(V_{C,1}^I, q_{ExportDB_1}), \ldots, (V_{C,n}^I, q_{ExportDB_n})$

queries mapping data from $V_{C,i}^I$ to $S_C$

output views $(V_{C,1}^O, q_1), \ldots, (V_{C,m}^O, q_m)$

defined on $S_C$

functionality $F_C$

Collector activity based on activation time, redirection to other collectors after time points have been reached
Collectors in the Application

CREATE TRIGGER /* FOR INSERT EVENT INTO MICRO-DATABASE
  /* POPULATE CORRESPONDING INBOUND IN COLLECTORS
  /* POPULATE CORRESPONDING OUTBOUND IN COLLECTORS
  ON INSERT (d,a,t) INTO MICRO-DB
  INSERT INTO a-INBOUND (d,a,t)
  INSERT INTO d-OUTBOUND (d,a,t) END;

State-of-art Analysis
Inspiration Informed analysis
Analysis space
Collectors
Inspectors
Conclusion
**Inspection Space**

**Input data, output data and meta-data**

**Analysis programs for deriving data,**

**Inspection charts for presentation of (computed) data**

**Recharge functions for injection of modified data**

**Analysis quality agreement** for coordination of the data between the collector management system and the inspection management system.

**Realisation of inspection spaces through MVC**

<table>
<thead>
<tr>
<th>MVC</th>
<th>model</th>
<th>view</th>
<th>controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>inspection MS</td>
<td>dataset supporter</td>
<td>inspection chart</td>
<td>transformation engine</td>
</tr>
</tbody>
</table>
A General Model of an Inspection Chart

Inspired by the Molnar/Lenz/β data cube model.

Data cell model with related data collections similar to the data cube model (cells with attributes that represent derived values) as a visualisation model.

Cell visualisations for visual representations of complex structured data (classical object-relational, hierarchies, geometric shapes, texts, links, ...)

Cell topology for interrelationship among the cells, redundancy and shared data sets

Cell traversal functions generalising classical cube operations

Constraints and restrictions for interrelated data among cells

Conceptualiser adds themes to the visualisation pipe (from micro, meso and analytical data to visualisation data)
Summarising

- Applications might require their specific architecture that is more effective than the classical architecture

- Data analysis is seldom digging or browsing or uninformed search; it is more often hypothesis/suggestion-based. Inspection charts can thus be specified in advance.

- Event-based data collection is an alternative to data hill building. Open field surface mining (with data hills generated) versus underground mining.

- DBMS support this approach as well.

- Analysis portfolio as a systematic way of exploration.

- Collectors for eager/lazy data collection within the profile of interest.

- Inspection charts as a generalisation of spreadsheets.
Thank you! Arigatou! Danke!
Co-Design of Structure, Functionality, Distribution, and Interaction

Eötvös Loránd University of Sciences
Budapest

4.7.2012

Bernhard Thalheim
Technologie der Informationssysteme
Institut für Informatik, Christian-Albrechts-Universität zu Kiel, BRD
Kolmogorow-Professor e.h. der Lomonossow-Universität Moskau
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*
Handling Large Schemata Through Layered Architectures
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

Why Co-Design?
Layering
Distribution
Interactivity
Methodology
Media Types
Open
References

"Partial reality"

Part of reality

Things of reality

Observed property

Predicator

"Topic"

Foundation of decisions

Modeling decision

Context

Revision during the development process

acts within

under usage

Usage of theory

Revision during the development process

"Schema" as result and partial point of view of a database development process

"Partial reality"

Exactness

Confidence

Modeler

Reference model

Concept

Topic

Information

Systems

Co-Design

4.7.2012

B. Thalheim
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 Topic</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


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)

\[ \text{Distribution} := (\text{Service (Informational process, Service Manager, Competence)}, \text{ExchangeFrame (Architecture Collaboration Style, CollaborationPattern)}) \]

Services on the basis of generalized views (media types)

Interactivity as 4-tuple

\[ \text{Interaction} := (\text{StorySpace, Actors, MediaObjects, Presentation}) \]

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)
Modern Information Systems

Database systems as the kernel

dbs = dbms + { db }

- Structuring of databases (structure + static integrity)
- Functionality based on programming by generic functions

Information systems based on database systems and coping with the user perspective:

- Users stories, scenarios within the story space
- Users views on content based on media types
- Context (users, content, functionality, environment, provider, history)

Collaborating information systems coping with component systems

- Components providing services over networks
- Collaboration for task solution

Nowadays: Co-Design of Four Dimensions
Languages (syntax √, semantics ??, pragmatics ????)

- Languages for specification
  - Structures (object-relational) √
  - Functions DML-based ????
  - Distribution system-based
  - Interaction
- Languages of tools
  - 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
Conceptualisation: Oneness of Schema(ta)

Depending on the viewpoint

Leading, governing schemata in UML (13 of 142)

Class diagram with associated object, package, composite structure, deployment, and distribution diagrams

State machine diagram with associated use case, activity diagrams

Interaction diagram with associated communication, sequence, timing diagrams

or consider the \textbf{ER}-backed layered conceptual modelling

(1) (Extended) Entity-Relationship schema with H(igher-order)ER algebra, HERM logics as the governing or kernel schema

(2) Business process model, e.g., BPMN

(3) Distribution styles, profiles, pattern with communication + cooperation + coordination

(4) Storyboards and stories as the often neglected dimensions
Separation of Concern

(a) Properties essential for things of consideration in the application domain

(b) Log as a volatile but essential piece of data
   (b1) docket / superimposed schemata
   (b2) source
   (b3) time

... ...

resulting in

- binding schemata for references
- implicit exclusion via explicit exclusion
- special viewpoints (with importance in the development and deployment processes)
Different Scope - Need - Demand

- Business user, application-oriented scopes, local processes, representation in an adequate form
- Scope-oriented representation for reasoning of a singleton user (object-centred model style (XML style))
- Data representation at different abstraction and granularity levels (raw, cleansed/settled, stored, aggregated, analysis data)
- All necessary information for implementation, e.g., usage profile and portfolio, set cardinalities, performance demands with performance tolerances, denormalisation and normalisation opportunities, semantics enforcement profiles

Salami-slice-oriented conceptual models as the main style.

Delay all other information to implementation (logical/physical) levels!

Conceptual modelling is the programming of the future!!!!
Three-Model-Conceptualisation

Natural model: representing reality as it is

**m-schema** as an **object-centred representation**:

- my car ⇔ my car product ⇔ my car model ⇔
- my car brand ⇔ my car manufacturer ⇔ my product ⇔ my product registration

thing of reality with scope, role, cohesion/adhesion, context, function, ‘personality’, competing viewpoints, with ‘natural’ semantics

Universal model: ER-architecture with inner (abstract) structure and scoping profiles, explicit shuffling among schema dimensions, folders and mappers, with full semantics and enforcement styles

Implementation model: class-separation schema with **central -view tower** architecture, performance information as an **class-centred representation**, with rigid implementation semantics

Information (iso-)morphism among the three schemata

Mapping theory with the universal model as governor.

Refinement of ER schemata (see Qing/β @ ER’2011)
Conceptualisation: Solution

Assumption nowadays: Oneness of conceptual schema

instead however

**Application-oriented conceptual schema**: typically compactified;
with different viewpoints for roles of stakeholders;
zooming out, scaling, scoping

**Universal (real) conceptual schema**: represents the conceptualisation;
based on many-dimensional separation of concern

**Realisation-oriented conceptual schema**: depending on implementation policy, style and implementation platform;
also object-wise realisation
Abstraction Layer Models

Separation by Level of Detail

Application domain layer concerned with description of the application

Requirements acquisition layer concerned with prescription of system requirements

Business user layer concerned with behaviour or users, their demands to the system

Conceptual layer concerned with specifications (schemata) that describe the system

Implementation layer concerned with logical and physical (specifications and) programs

Deployment layer concerned with introduction, usage, maintenance, evolution of the system
Abstraction Layer Models

Separation by Level of Detail

Application domain layer concerned with description of the application

Requirements acquisition layer concerned with prescription of system requirements

Business user layer concerned with behaviour or users, their demands to the system

Conceptual layer concerned with specifications (schemata) that describe the system

Implementation layer concerned with logical and physical (specifications and) programs

Deployment layer concerned with introduction, usage, maintenance, evolution of the system
Classical dichotomy: Human-computer systems and information systems

**Description/prescription layer**
- Application area description
- Requirements prescriptions
- WIS description and prescription
- Information system specification
- Presentation system specification
- WIS specification
- Information system specification
- WIS and information system specification

**Conceptual layer**
- Design Refinement
- WIS description
- Information system specification
- Presentation system specification
- WIS specification
- Information system

**Implementation layer**
- Implementation Transformation
- Presentation system
- Web information system

**Why Co-Design?**
- Layering
- Distribution
- Interactivity
- Methodology
- Media Types
- Open
- References
Dichotomy of human-computer systems and the software systems

Why Co-Design?

Layering
Distribution
Interactivity
Methodology
Media Types
Open
References

Conceptual layer

Implementation layer

Description/prescription layer

Design Refinement

Application area description

Requirements prescriptions

WIS description and prescription

Presentation system specification

Information systems specification

Web information system

Content

Information

Concept

Topic
### The Software Engineering Quadruple

- **Application domain description**
- **Architecture blueprint**
- **Software specification**
- **Requirements prescription**

The ‘holy’ triade so far extended by **context**

- Application domain **description**
- Requirements **prescription**
- Software **specification**

---

**Why Co-Design?**

- **Layering**
- **Distribution**
- **Interactivity**
- Methodology
- Media Types
- Open
- References
Distributivity Specification Through DistrLang

Service $S = (\mathcal{I}, \mathcal{F}, \Sigma_S)$ specifying

- Informational processes $\mathcal{I} = (\mathcal{V}, \mathcal{M}, \Sigma_T)$ specifying
  - the content $\mathcal{V}$ based on media types ("what"),
  - the service manager $\mathcal{M}$ ("how"), and
  - the competence $\Sigma_T$ through a set of tasks ("for what")

- Service characteristics $\mathcal{F}$ specifying the organization frame ("how"),
  - the parties ("who") and the context ("whereby")

- Quality of service $\Sigma_S$ agreeing on the quality and motivation ("why")

Exchange frame specifying

- Architecture drafting the general engine ("where")
- Collaboration style drafting the flow ("when")
- Collaboration pattern describing the functionality ("how", "whereby")

as a generalization of distributed systems, communication systems,
  groupware systems, and collaboration architectures
Conceptual Modelling of Collaboration

Collaboration Triangle Relating Communication, Coordination, and Cooperation

Communication supports collaboration which requires coordination to arrange tasks for cooperation. Collaboration creates opportunities for communication which generates commitments that are managed by coordination.

Communication act view based on sending and receiving
Concurrent view based on commonly used data, functions, and tools
Cooperation context view combines the context of cooperation, i.e. portfolio to be fulfilled, the cooperation story and the resources that are used.
Explicit Collaboration

Communication via exchange of messages and information as only one of the perspectives demands cooperation and generates commitments that are managed by coordination choice of media, transmission modes, meta-information, conversation structure and paths, restriction policy

Coordination via management of individuals, their activities and resources as the dominating perspective generates communication and arranges tasks for cooperation pre-/post-articulation of tasks; management of tasks, objects, and time; loosely ... tightly integrated activities, enabled, forced, blocked

Cooperation as production taking place on a shared space as the workflow or life case perspective creates opportunities for coordination and demands communication storyboard-based interaction, mapped to (generic, structured) workflows production, manipulation, organization of contributions through media objects

Awareness is fostered by each of the aspects and mediates each of the aspects
Support for Collaboration

media type: views + functionality + adaptation + presentation

Active media types: in parallel
open(content); inform(proprietor, usage)

Self-protecting media types: content based on queries and supported view
protocol: contact(proprietor, possessor, usage);
obtain(proprietor, token); provide(media type, token)

Communication protocols based on service (distributed ADT), signals, shared
variables, sender/reponder ASM (signature (e.g., signal), phases (via small
ASM)), and timer
represented through SDL or message sequence chart or other protocols

Security techniques against passive/active sniffing, trust exploitation, viruses,
downloadables, OS holes, hacking

Control techniques for focusing, access, user authorization, (password)
protection, biometrics, content/concept/topic security, firewalls, hiding/anonymizing/translating, identity management

Privacy enhancing techniques based on virtual private networks, key encryption,
secure transaction, and corporate policy on security, privacy and control, cookie cooker
# Services for Distribution:

**Informational Processes**

<table>
<thead>
<tr>
<th>Media types</th>
<th>Services manager</th>
<th>Competence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw media type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extensions (unit/order)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Co-/Ad-hesion</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hierarchy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Playout</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kind</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Co-ordination</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Co-operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course insertion view</th>
<th>Chair priority order</th>
<th>Course-centered</th>
<th>Term order</th>
<th>Mac-XSL5</th>
<th>Daily refresh</th>
<th>Deliver only</th>
<th>None</th>
<th>None</th>
<th>Input main course data</th>
<th>Select or insert</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course negotiation view</td>
<td>Assisstant order</td>
<td>Course-centered</td>
<td>Workload order</td>
<td>Mozi-XSL5</td>
<td>Weekly refresh</td>
<td>Exchange</td>
<td>View others</td>
<td>Other assistants</td>
<td>Assignment</td>
<td>Accept or reject</td>
</tr>
</tbody>
</table>

| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

---

**Why Co-Design?**

**Layering**

**Distribution**

**Interactivity**

**Methodology**

**Media Types**

**Open**

**References**
## Services for Distribution:

### Services Characteristics

<table>
<thead>
<tr>
<th>Charateristics</th>
<th>Party</th>
<th>Organization</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Responsible person</strong></td>
<td>Inform, provide</td>
<td>Insert new, select</td>
<td>[request, deadline]</td>
</tr>
<tr>
<td><strong>Assistant at chair, assigned (course)</strong></td>
<td>Negotiate</td>
<td>Own plan</td>
<td>[request, meeting]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Time slot</th>
<th>Hierarchy</th>
<th>Synchronization</th>
<th>Coordination</th>
<th>Media types</th>
<th>Environment</th>
<th>Range of variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party</td>
<td>[request, deadline]</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>Last 3 terms lectures</td>
<td>Linux, PC, browser</td>
<td>Full</td>
</tr>
<tr>
<td>Organization</td>
<td>[request, meeting]</td>
<td>none</td>
<td>colleagues</td>
<td>None</td>
<td>Chair schedule</td>
<td>Linux</td>
<td>History, profile, adaptation</td>
</tr>
</tbody>
</table>

---

**Why Co-Design?**

**Layering**

**Distribution**

**Interactivity**

**Methodology**

**Media Types**

**Open**

**References**

---

**Content**

**Information**

**Concept**

**Topic**
Services for Distribution:

**Quality requirements**

- **General requirements** (enforced by the entire system)
  - ubiquity, e.g., 7-24-60-60 frame
  - security, including all aspects
- **User requirements** (for interactivity and distribution support)
  - interpretability (meaningful content, adaptive, context-sensitive)
  - consistency (of the story, of the content, of the dialogues)
  - view consistency (within a media object suite)
- **System requirements** (applied to implementations)
  - scalability (handling large system configurations as well as small)
  - durability (of data)
  - robustness (against any kind of (user/system) errors)
  - performance (at least for usability)
Exchange Frames for Distribution:

Architecture, e.g. Database Farms

as generalization of federated systems and multi-database systems
Exchange Frames for Distribution:

Another Architecture Supporting Distribution

Exchange support system

- Communication (asynchronous | synchronous)
  - Sender Receiver
  - Stub
  - Function manager

Cooperation
- Work process manager
- Organizations manager
- Working space

Coordination
- Task manager
- Coordinator
- Version manager

Media object suite management system
- Object suites
- Association manager
- Consistency manager

Exchange = Cooperation + Coordination + Communication

Collaboration system supporting groups, workspace, media object suites
Exchange Frames for Distribution:

**Collaboration Style**

- supporting programs (session, user, payment management)
- data access pattern (release, sharing, remote)
- model of collaboration (P2P, ...)
- collaboration interplay (partner, mapping, rules)

**Collaboration Pattern**

- access / configuration
- event processing
- synchronization
- concurrent/parallel execution
Typical Collaboration Pattern

Proxy collaboration: partial system copies (remote proxy, protection proxy, cache proxy, synchronization proxy, etc.)

Broker collaboration: coordination of communication either directly, through message passing, based on trading paradigms, by adapter-broker systems, or callback-broker systems

Master/slave collaboration: tight replication in various application scenarios (fault tolerance, parallel execution, precision improvement; as processes, threads; with(out) coordination)

Client/dispatcher collaboration: for name spaces and mappings

Publisher/subscriber collaboration (observer-dependents paradigm) active/passive subscribers or passive with their subscription profile

Model/view/controller collaboration: e.g. three-layer architecture of database systems
Portfolio and Task Specification

Party portfolio: ⟨party portfolio name⟩
Task: ⟨general description⟩
Characterization: ⟨general description⟩
  Initial state: ⟨characterization of initial states⟩
  Target state: ⟨characterization of target states⟩
Profile: ⟨profile presupposed for solution⟩
Instruments: ⟨list of instruments for solution⟩
Collaboration: ⟨collaboration style/pattern⟩
Auxiliary: ⟨list of auxiliary conditions⟩
Execution: ⟨list of activities, control, data⟩
Result: ⟨final state, target conditions⟩
Party involvement: ⟨general description⟩
Role: ⟨description of role⟩
Part: ⟨behavioral categories/stereotypes⟩
Collaboration: ⟨general description⟩
Communication: ⟨protocols, services and exchange⟩
Coordination: ⟨contracts and enforcement⟩
Cooperation: ⟨flow of work⟩
Restrictions: ⟨general description⟩
Party restrictions: ⟨general description⟩
Environment: ⟨general description⟩
# Coordination Contracts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract</td>
<td>⟨name⟩</td>
</tr>
<tr>
<td>Based on</td>
<td>general conditions</td>
</tr>
<tr>
<td>Parties</td>
<td>⟨general description⟩</td>
</tr>
<tr>
<td>Proprietor</td>
<td>⟨...⟩</td>
</tr>
<tr>
<td>Possessor</td>
<td>⟨...⟩</td>
</tr>
<tr>
<td>Trustee</td>
<td>⟨...⟩</td>
</tr>
<tr>
<td>Arbiter</td>
<td>⟨...⟩</td>
</tr>
<tr>
<td>Subject matter</td>
<td>⟨Media object suite⟩</td>
</tr>
<tr>
<td>Exchange</td>
<td>⟨binding obligations, permissions⟩</td>
</tr>
<tr>
<td>Computation</td>
<td>⟨obligations, permissions⟩</td>
</tr>
<tr>
<td>Distribution</td>
<td>⟨obligations, permissions⟩</td>
</tr>
<tr>
<td>Monitoring</td>
<td>⟨managers: recognizer, ⟩</td>
</tr>
<tr>
<td></td>
<td>⟨states, timer, constraint scanner⟩</td>
</tr>
<tr>
<td>Notification</td>
<td>⟨obligations, permissions⟩</td>
</tr>
<tr>
<td>Correlation</td>
<td>⟨protocols, obligations, permissions⟩</td>
</tr>
<tr>
<td>Considerations</td>
<td>⟨legal conditions⟩</td>
</tr>
<tr>
<td>Enforcement</td>
<td>⟨actions, termination⟩</td>
</tr>
</tbody>
</table>
Coordination Specification

Coordination profile: ⟨name⟩

Based on: general conditions

Formation: ⟨general description⟩

Contract: ...

Lifespan: ...

Contract variant: ...

Parties: ⟨names⟩

Organization: ⟨names, general description⟩

Infrastructure: ⟨name, general description⟩
Party Specification

Party: ⟨names⟩
  Characteristics: ...
  Profile: ...
  Roles: ...
  Rights: ...
  Relations: ...

Part: ⟨general description⟩
Organization: ⟨general description⟩
Infrastructure: ⟨general description⟩

Party profile: ⟨party profile name⟩
Information demand: ⟨general description⟩
Utilization pattern: ⟨general description⟩
Specific utilization: ⟨general description⟩
Party context: ⟨general description⟩
Party Group Specification

organizations, e.g., groups:

Organization: ⟨name⟩
  Synchronization: ...
  Stories: ...
  Hierarchy: ...
  Time slot: ...
  Task distribution: ...
  Coordination: name

Infrastructures: ⟨names⟩
Services

Service: ⟨name⟩

Based on: general conditions

Media types: ⟨general description⟩

Raw media type: ...

Extensions: ...

Unit: ...

Order: ...

Co-/Adhesion: ...

Hierarchy: ...

Playout: ...

Services manager: ⟨general description⟩

Kind: ...

Communication: ...

Coordination: ...

Cooperation: ...

Competence: ⟨general description⟩

Task: ...

QoS: ...
Context of a Service

Context: ⟨name⟩

Media types: ...

Environment: ...

Range of variation: ...

Context in general: Enrichment by super-typing

Pragmatical context (situational, physical environment, social, policy, time)

Website context (provider, supporter, SW/HW stakeholder)

Explicit context (Story space)

Syntactic
  verbal context

Extra-syntactic
  auxiliary correlates

Media type suite, meta-information

Actors, profile, payment, ...

Potential environment, information system, scenes, tasks, roles

Intention, theme, occasion, mission, purpose

Current scenario, history, current environment, user, goals, particular, culture
Communication/Coordination Infrastructure

- Agenda
- Scheduled_in
- Item
- Contribution
- Session manager
- Work/meeting session
- User
- Channel status
- Channel buffer
- Channel
- Event handler
- Event handler kind
- Message
- Log File
- Process
- User interface

Why Co-Design?
Layers of a Typical Collaboration System

Cooperation Layer
- Cooperation space/workspace: workspace control, awareness, notifications, security over user functions
- Communication space: (a)synchronous, multicast/broadcast, protocols, standard

Coordination Layer
- Coordination space: operation management, session management, shared resources management, users management

Communication Layer
- Media object unit manager
- Coordination and contracting system
- Communication support system

generalising Ochoa, Guerrero, Fuller, Herrera: Infrastructure of Groupware Systems
# Exchange Frames for Distribution

**Workspace and Group Collaboration Using the Trader Approach**

<table>
<thead>
<tr>
<th>Trader exchange</th>
<th>Communication</th>
<th>Cooperation</th>
<th>Coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architecture</strong></td>
<td>Asynchronous client</td>
<td>Cooperation manager, Media suite management system</td>
<td>Coordination manager</td>
</tr>
<tr>
<td><strong>Formation</strong></td>
<td>Protocol: trader ▸ P2P</td>
<td>Contract, TA</td>
<td>FIFO schedule</td>
</tr>
<tr>
<td>Model</td>
<td>Login</td>
<td>Hierarchical</td>
<td>Sharing, lifespan</td>
</tr>
<tr>
<td>Release</td>
<td>Stub, error track</td>
<td>Public, personal space</td>
<td>Replica of suites</td>
</tr>
<tr>
<td>Data exchange</td>
<td>Event, open space</td>
<td>Reactor, proactor</td>
<td>Trade, book, pay</td>
</tr>
<tr>
<td>Interplay</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Processing</strong></td>
<td>Message passing</td>
<td>Trader</td>
<td>FIFO</td>
</tr>
<tr>
<td>Model</td>
<td>IP5</td>
<td>Workspace, user billing managers</td>
<td>Session manager</td>
</tr>
<tr>
<td>Programs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data exchange</td>
<td>Push through, completion token</td>
<td>Scoped 2-phase locking</td>
<td>Half-sync among, full-sync with DBS</td>
</tr>
<tr>
<td>Interplay</td>
<td>IP5 sequencing</td>
<td>Involve on demand</td>
<td>2-phase commit</td>
</tr>
</tbody>
</table>
Experiences with Web Information Systems

- Many WISs are still developed in an ad-hoc way, not using appropriate methodology:
  - page not found or does not contain what was expected, page overloaded
  - link points to general entry point ignoring context
  - search produces heaps of pages, in one of which the desired information may be found
  - repeated information, repeated requests for data entry
  - etc.
Need for WIS Development Methods

- In general, the need for methodological development arises, if systems tend to become large, if content changes frequently, if complex tasks are to be supported, if multiple users use the system.

- WIS often serve purposes such as booking, selling, edutainment, etc., i.e. users are customers: giving up and turning away is highly undesirable.

- The web is an attractive (and fast) business channel for communication, advertisement, commerce: systems have to serve the business goals.
Existing WIS Development Methods

- Most WIS development methods (OOHDM, ARANEUS, HERA, WISM, WebML, UWE, Co-Design, etc.) concentrate on the data-intensive aspect: views on some underlying database schema

- All methods support the generation of pages and layout

- Some (Co-Design) emphasize also the modelling of users/actors, tasks and action schemes/plots

- Some (WISM, Co-Design) emphasize personalisation

- Strategic modelling is rarely addressed (except Co-Design)
Differences from IS: Task-Centredness

- 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
Differences from IS: Usage-Centredness

- Identify essential user and business purposes
- Understand targeted users by roles in relation to site
- Understand tasks in terms of user intentions and needs
- Prioritize user roles and user tasks in terms of expected frequency and business importance
- Engineer site design to fit user and business priorities

Thus: Questions to answer

- What types of people will be visiting the site?
- What’s the relationship between the users and the site?
- What are the users trying to do?
- What do the users need from the site in order to accomplish their goal?
- How do we organize the site content to help them succeed?
Key Questions

- **Who** will be using the system?
- **When** will the system be used?
- **Where** is the information system used?
- **What** is represented in the system?
- **How** will the system be used?
- **Why** is the system used?
- **What** is the policy, intention, goal, and aim of the **provider**?

Additional dimension in the Zachman model are:

**Competency:**

**Time:** (schedule; delay)

**Environment:** (context; technical and organizational)

**Quality:** (in which quality; with which guarantees)

**Runtime characteristics:** (adaptation; exceptions; delays)

**Collaboration:** (with whom)
Storyboarding an Old Issue?

E.g., great script and movie: *Tootsie, Witness, Back to the Future*
story + characters + topic + pictures + dialogues

- well structured (set-up, development, midpoint scene, climax, resolution)
  job-less, changing sex, job success, friendship, love, overwhelming problems,
  happy end
  + brief and focused, constraint enforcement through dimensions

- top-down (expose, outline of scenes, treatment, plot, production)

- using all elements (catalysts, action point, attraction, barrier, complication,
  contrast, momentum, motives, pay-off, reversal, sequence, A-story, B-story,
  subjective point-of-view, subplot, twist)

<table>
<thead>
<tr>
<th>motivation</th>
<th>action</th>
<th>goal</th>
<th>conflict</th>
</tr>
</thead>
</table>

A well known story:

The wild swans: \(ib^1a^1c^1A^1B^4C \leftarrow \{Sch^1H^1Z^1||sch^7H^7Z^9\}W^4L^1\downarrow\)
\(V^1[Sch^1H^1Z^9 \equiv R^4] \times 3\)
Separation of Concern in the Web Utilisation Space by the Specification Hexagon

User and intention

Context
- Technics
- organisation
- WIS context

Content
- Data
- objects
- knowledge

Presentation

Functionality

Storyboard
- Stories
- tasks

Web IS collaboration
- group content
- collective identity

Goal, application area
- profile,
- information demand

Interfaces
- depending on the environment

Navigation
- search
- work

Why Co-Design?
Layering Distribution
Interactivity
Methodology
Media Types
Open
References
**From Web 1.0 to Web 3.0**

**Web 1.0:** author driven, publish/provide/story/support or advertise/wait/attract/react/retain for users: inform/subscribe/obtain/answer/come back.

**Web 3.0:** asset driven, content-asset centered, provides additionally linguistic semantics, technology combines: artificial intelligence, automated reasoning, cognitive architecture, composite applications, distributed computing, knowledge representation, ontology (computer science), recombinant text, scalable vector graphics, semantic Web, semantic Wiki, software agents.
Web x.0 and the Co-Design Hexagonal Dimensions

Web x.0: Towards sophisticated web engineering

Context
- Technics
- organisation

Content
- Data
- objects
- knowledge

User and intention
- Goal, application area
- profile,
- information demand

Storyboard
- Stories
- tasks

Functionality
- Navigation
- search
- work

Presentation
- Interfaces
- depending on the environment

Web 2.0
- collaboration
- group content
- collective identity

Why Co-Design?
Layering
Distribution
Interactivity
Methodology
Media Types
Open
References

Web 2.0 allows context injection and is user-centered and story-centered
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

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

\[
\begin{align*}
\text{if precond}_{ru_i} \text{ and event then actions and CommitState}_{ru_i} &= \text{toCommit} \\
\text{if CommitState}_{ru_i} &= \text{toCommit and postcond}_{ru_i} \text{ then Commit}_{ru_i} \\
\text{if CommitState}_{ru_i} &= \text{toCommit and } \neg \text{postcond}_{ru_i} \text{ then Undo}_{ru_i}
\end{align*}
\]

Interaction space - space of all possible interaction stories

System space - glass box of system (all runs integrated)
**Interaction Modeling**

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

- \( V \) - scenes, \( E \subseteq V \times V \) - transitions
- media object assignment \( \lambda : V \to \{\text{MediaObj}\} \)
  - representation through media objects
  - with permitted rights, permitted roles, obligations of actors
- MediaObject (Container, ManipulatRequests, SuppliedFunctions)
- activity assignment \( \kappa : E \to \{\text{Activity}\} \)
  - 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** $v \in \text{Alg(DialogueStep)}$

- basic control: sequence $;$, parallel split $\mid$, exclusive choice $\mid$, synchronization $\mid^{\text{sync}}$, simple merge $+$
- advanced branching and synchronization control: multiple choice, multiple merge, discriminator, n-out-of-m join, synchronizing join
- structural control: arbitrary cycles $\ast$, 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: on event if $\alpha$ doScene $v$ 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>name</td>
</tr>
<tr>
<td>variants</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>application</th>
</tr>
</thead>
<tbody>
<tr>
<td>applicability</td>
</tr>
<tr>
<td>usability profile</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>structuring: structure, static constraints</td>
</tr>
<tr>
<td>interactivity: story space, actors, media objects, representation</td>
</tr>
<tr>
<td>context: tasks, intention, history, environment, particular</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>implementation</th>
</tr>
</thead>
<tbody>
<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

sub-unit

supplied process

enabled actor

transition according to dialogue scene expression

dialogue step

involved actors

story scene sequence

media object

representation style

context, task

involved actors

story scene sequence

media object

representation style

context, task
### dialogue step

<table>
<thead>
<tr>
<th>header</th>
<th>container</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>content</td>
</tr>
<tr>
<td>title</td>
<td>text</td>
</tr>
<tr>
<td></td>
<td>multimedia object</td>
</tr>
<tr>
<td></td>
<td>graphics</td>
</tr>
<tr>
<td></td>
<td>picture</td>
</tr>
<tr>
<td></td>
<td>video</td>
</tr>
<tr>
<td></td>
<td>audio</td>
</tr>
<tr>
<td></td>
<td>functionality</td>
</tr>
<tr>
<td></td>
<td>operations</td>
</tr>
<tr>
<td></td>
<td>algorithmic object</td>
</tr>
<tr>
<td></td>
<td>tayloring style</td>
</tr>
<tr>
<td></td>
<td>ordering by hierarchies</td>
</tr>
<tr>
<td></td>
<td>adhesion</td>
</tr>
<tr>
<td>adaptation</td>
<td>interaction style</td>
</tr>
<tr>
<td></td>
<td>control style</td>
</tr>
<tr>
<td></td>
<td>actors applicable</td>
</tr>
</tbody>
</table>
Storyboards: Example Wikis

forming a Wiki team with three roles (evaluator, member, supporter)
modelling the Wiki collaboration story
modelling the intended result
Wiki Specification Based on Content Chunks

Wiki $\mathcal{W} = (\mathcal{I}, \mathcal{S}, \Sigma_\mathcal{W})$ specifying

Content chunk processes $\mathcal{I} = (\mathcal{V}, \mathcal{M}, \Sigma_T)$ specifying
the content chunks $\mathcal{V}$ based on media types ("what"),
the wiki manager $\mathcal{M}$ ("how"), and
the competence $\Sigma_T$ through a set of tasks ("for what")

Wiki storyboard $\mathcal{S}$ specifying the organization frame ("how"),
the parties ("who") and the context ("whereby")

Quality of wiki $\Sigma_\mathcal{W}$ agreeing on the quality and motivation ("why")

Exchange frame specifying

Architecture drafting the general engine ("where")
Wiki collaboration style drafting the flow ("when")
Wiki collaboration pattern describing the functionality ("how", "whereby")

as a generalization of distributed systems, communication systems,
  groupware systems, and collaboration architectures
Support for Wikis

content chunks based on media types: views + functionality + adaptation + presentation

Active media types: in parallel
open(content); inform(proprietor, usage)

Self-protecting media types: content based on queries and supported view
protocol: contact(proprietor, possessor, usage);
obtain(proprietor, token); provide(media type, token)

Communication protocols based on service (distributed ADT), signals, shared
variables, sender/reponder ASM (signature (e.g., signal), phases (via small
ASM)), and timer
represented through SDL or message sequence chart or other protocols

Security techniques against passive/active sniffing, trust exploitation, viruses,
downloadables, OS holes, hacking

Control techniques for focusing, access, user authorization, (password)
protection, biometrics, content/concept/topic security, firewalls, hid-
ing/anonymizing/translating, identity management

Privacy enhancing techniques based on virtual private networks, key encryp-
tion, secure transaction, and corporate policy on security, privacy and control,
cookie cooker
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
- Confirmation
- Submission chair data
- Assignment of courses to members
- Negotiation of assignments by members
- Formulation of side conditions

Why Co-Design?
Layering
Distribution
Interactivity
Methodology
Media Types
Open
References
Dialogue Steps for Event etc. Search

event search scene

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

Why Co-Design?
Layering
Distribution
Interactivity
Methodology
Media Types
Open
References
Storyboard with $[1,1] - [n \gg 1,m]$ Workflow

Supervised Solution of Exercises

**Supervisor**
- close group
- develop new assignments
- define new assignments
- check answers
- collect answers

**Exercise Team Member**
- new assignments
- answering sheet
- revised assignment
- discussion & evaluation
- quit group
- next assignment

**SO-CRATE**
- new pending tricks
- hints & tricks
- outdated hints & tricks

Parallel actions of users: dotted separation

Workspace for communication between parties

History for reentering the scene

Rules are based on scenario (and didactic, pedagogical and psychological profile)

Persistent media object suite recording long-running sessions

Variant: KoPra with (problem solver, corrector, advisor)-scenarios
Sequenced Scenes

Sequenced Competitive Solution Scene

- Selection of problem type
- Analysis of learners selection
- Selection of appropriate algorithm
- Selection of appropriate attributes
- Selection of target attribute
- Code upload & installation
- Finding a solution for missing data
- Computation of associations through mining
- Submission of solution for competition
- Evaluation of learners competition results

Competitive Solution Scene

- Task delivery step
- Collection of learners data
- Preparation of learners data
- Formulation of learners hypotheses
- Computation of associations through mining
- Submission of competitive solution
- Evaluation of submitted solution
- Inspection of sample solution & comparison
- Reminder of learning element on hypotheses
- Repetition of solution for competition
Story Space

Compiling all stories within a story space

Why Co-Design?
Layering
Distribution
Interactivity
Methodology
Media Types
Open
References
Underpinning Stories with BPMN Workflows

Why Co-Design?
Layering
Distribution
Interactivity
Methodology
Media Types
Open
References

Content
Information
Concept
Topic
Paper Submission and Reviewing System: Dialogue Scene

\[ v_1 = \sigma_{PaperID=pID}(PaperBody) \]
\[ \alpha = \text{on LinkPaperSubmit if deadline = ok} \land \text{Login = ok} \]
\[ \quad \text{accept on submitConfirmed} \land \lnot \text{collectError} \]
\[ \beta = \text{on submitPaper if paperBody = ok} \text{ accept on extension = ok} \]
\[ \gamma = \text{accept on } \lnot \text{transferError} \]
\[ \delta = \text{on submitPaper if 1 accept on confirmed} \]
\[ A = \text{actor[pID,pwrd]} \]
Modelling Life Cases by Stories

Digicult: Representing successful behavioural pattern

Mapping behaviour of users with full option space
Intelligent representation of information and knowledge spaces
Adaptation to the user, current situation, context, ...
Logistics for content
Formal Specification of Scenarios

Scenario: \langle name of the scenario \rangle

Scenes: \langle list of scene names \rangle

Start Scene: \langle scene name \rangle

Final Scenes: \langle list of scene names \rangle

Actions: \langle list of action names \rangle

Transitions: \langle list of transitions \rangle

Process Expression: \langle SiteLang process \rangle
Formal Specification of Scenes

Scene: ⟨scene name⟩
Actions: ⟨list of action names⟩
Roles: ⟨list of role specifications⟩
User Types: ⟨list of user type names⟩
Defining Scenario: ⟨scenario name⟩
Acceptance Condition: ⟨condition⟩
Formal Specification of Actions

Action: \( \langle \text{action name} \rangle \)

Precondition: \( \langle \text{condition} \rangle \)

Postcondition: \( \langle \text{condition} \rangle \)

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

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

Enabling Event: \( +\langle \text{list of actions} \rangle \)

\( -\langle \text{list of actions} \rangle \)

Triggering Event: \( +\langle \text{list of actions} \rangle \)

\( -\langle \text{list of actions} \rangle \)

Manipulation Requests: \( \langle \text{list of manipulation requests} \rangle \)
Formal Specification of Tasks

Task: \langle \text{name of the task} \rangle
Goal: \langle \text{condition} \rangle
Subtasks: \langle \text{list of subtask names} \rangle
Problem: \langle \text{problem statement} \rangle
Actions: \langle \text{list of action names} \rangle
Precondition: \langle \text{condition} \rangle
Event: \langle \text{action} \rangle \text{ by \langle \text{role name} \rangle}
Scenario: \langle \text{scenario name} \rangle
Relationship: \langle \text{description} \rangle
Description: \langle \text{textual description} \rangle
Participants: \langle \text{list of users} \rangle
Required Content: \langle \text{list of content items} \rangle
Produced Content: \langle \text{list of content items} \rangle
Result: \langle \text{textual description} \rangle
Starting situation: \langle \text{list of situation parameters} \rangle
Closing condition: \langle \text{acceptance constraints} \rangle
Execution context: \langle \text{textual characterization} \rangle
Enables: \langle \text{list of task names} \rangle
Specialises: \langle \text{list of task names} \rangle
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 multipoint 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

Application domain layer

Scoping

Requirements acquisition layer

Variating

Business user layer

Designing

Conceptual layer

Implementing

Implementation layer

Structuring specification

Distribution specification

Dialogue specification

Functionality specification
Making Co-Design Working: General Description

Application domain or strategic 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")
nowadays: Product feature catalog
Making Co-Design Working

Requirements Specification

**Requirements acquisition** layer:

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

1. Separation into system components
2. Sketching the story space
3. Sketching the view suite
4. 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))

1. Transformation of conceptual schemata into logical schemata, programs, and interfaces
2. Development of logical services and exchange frames
3. Developing solutions for performance improvement, tuning
4. Transformation of logical schemata into physical schemata
5. Checking durability, robustness, scalability, and extensibility

Program library
The Onion Approach

presentation engine

container engine

media object engine

view handler

DBS

DBMS

virtual V 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

XML scene onion

container onion

media object onion

XML suite

Concept

Information

Topic

Content

References

Methodology

Media Types

Open

Interactivity

Distribution

Layering

Why Co-Design?
(WIS-E 1): Application Domain Description and Requirements Statement

## Process Purpose: Goals and Subject

<table>
<thead>
<tr>
<th>Goals and subject</th>
<th>Application domain description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Agreement for development</td>
</tr>
<tr>
<td></td>
<td>Project scope: Milestones, financial issues</td>
</tr>
<tr>
<td></td>
<td>Clarification of development goals (intentions, rationale)</td>
</tr>
<tr>
<td></td>
<td>Sketch of requirements</td>
</tr>
</tbody>
</table>

## Process Outcomes: Work Products as Process Results

<table>
<thead>
<tr>
<th>Developed documents</th>
<th>Stakeholder contract: goal information, concept sketch, product functionality, story space, views on product data, view collaboration sketch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Official and contracting section</td>
<td>Comparison with products of competitors</td>
</tr>
<tr>
<td></td>
<td>Evaluation of development costs</td>
</tr>
<tr>
<td>Internal section</td>
<td>Planning of goals, development strategy and plan, quality management</td>
</tr>
<tr>
<td>Developed documents</td>
<td>Development documents on product components and quality requirements with base practices, generic practices and capabilities, estimation of efforts</td>
</tr>
</tbody>
</table>
### (WIS-E 1): Application Domain Description and Requirements Statement

**Process Outcomes: Work Products as Process Results**

<table>
<thead>
<tr>
<th>Developed documents</th>
<th>Application domain description section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information analysis</td>
<td>missions and goals of the WIS, brand of the WIS</td>
</tr>
<tr>
<td></td>
<td>general characterization of tasks and users</td>
</tr>
<tr>
<td></td>
<td>general characterization of content and functions</td>
</tr>
<tr>
<td></td>
<td>description of WIS context</td>
</tr>
<tr>
<td>Intensions of the web information system,</td>
<td>catalog of requirements</td>
</tr>
<tr>
<td></td>
<td>scenarios, scenes, actions, context and life cases</td>
</tr>
<tr>
<td></td>
<td>user profiles and user portfolio</td>
</tr>
<tr>
<td></td>
<td>actor profiles and actor portfolio, personas</td>
</tr>
<tr>
<td>Business rule model and storyboards</td>
<td>scenarios, scenes, and actions</td>
</tr>
<tr>
<td>WIS utilization portfolio</td>
<td>life cases and context</td>
</tr>
<tr>
<td>Metaphor description</td>
<td>scenarios, activities</td>
</tr>
<tr>
<td></td>
<td>supporting content and functions</td>
</tr>
<tr>
<td></td>
<td>non-functional requirements, context space</td>
</tr>
<tr>
<td></td>
<td>base metaphors, overlay metaphors</td>
</tr>
<tr>
<td></td>
<td>metaphor integration and deployment</td>
</tr>
</tbody>
</table>
(WIS-E 1): Application Domain Description and Requirements Statement

Base Activities and Steps

1. Analyze strategic information
   - Specify mission and brand
   - Characterize in general tasks and users
   - Characterize in general content and functions
   - Describe WIS context

2. Derive intensions of the WIS,
   - Obtain general requirements
   - Extract life cases
   - Describe scenarios, scenes, actions, and context
   - Describe user profiles and user portfolio
   - Derive actor profiles and actor portfolio, personas

3. Extract business rule model and storyboards
   - Develop scenarios, scenes, and actions
   - Specify life cases and context
   - Eliciting metaphors

4. Revise business rules of the application
   - possibly with reorganization models

5. Compare new business rules with known visions

6. Compare with products of competitors
**WIS-E 1): Application Domain Description and Requirements Statement**

Base Activities and Steps

<table>
<thead>
<tr>
<th>Development of application domain description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7. Derive WIS utilization</td>
</tr>
<tr>
<td>Describe scenarios to be supported</td>
</tr>
<tr>
<td>Describe activities based on word fields</td>
</tr>
<tr>
<td>Describe supporting content</td>
</tr>
<tr>
<td>Describe supporting functions</td>
</tr>
<tr>
<td>Describe non-functional requirements</td>
</tr>
<tr>
<td>Describe the context space</td>
</tr>
<tr>
<td>8. Develop metaphor</td>
</tr>
<tr>
<td>Describe base and overlay metaphors</td>
</tr>
<tr>
<td>Find metaphor integration</td>
</tr>
<tr>
<td>Develop templates for deployment</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Precondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contracted collaboration of all partners</td>
</tr>
<tr>
<td>Real necessity</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Postcondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descri. of application domain accepted and consistent</td>
</tr>
<tr>
<td>New business rule model accepted</td>
</tr>
</tbody>
</table>
Abstraction Layers and SPICE Model Categories in WIS Co-Design

Process

identifies modifications

is investigated by

identifies capabilities and risks

Process Assessment

Process Improvement

motivates

enables

enables

Capability Determination

Why Co-Design?
Layering
Distribution
Interactivity
Methodology
Media Types
Open
References
SPICEing Co-Design: Towards Optimizing Processes

Why Co-Design?
Layering
Distribution
Interactivity
Methodology
Media Types
Open
References

Distribution

- space restrictions due to technical limits
- modeling alternatives
- models fulfilling given requirements
- local structural model space

Interactivity

- model space supported by the modeler’s skills
- best practice
- space of integrated models

Model Space Restrictions during Co-Design
Evaluated Development by Rapid Prototyping and Executable Specifications

**executable specifications based on Model Driven Software Development**

automatic transformation of specifications on higher layers of abstraction to executable programs without manual transformation steps
Media Types, Media Object Suite

- Raw media types = $(cont(M), sup(M), view(M), op(M))$
  - content type $cont(M)$, set of supertypes $sup(M)$,
  - $view(M) = Q(S_{inp}, S_{outp})$ HERM view
  - generic functions $op(M)$ for changing the database

- Attached operations: (signature, selection type, body)
  - selection type - supertype of $cont(M)$
    - 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 = $(cont(C), layout(C), kind(C))$
  for shipping and representation
Layout and Playout: The General Picture

- Users
- Profiling
- Wrap
- Cut/glue
- Raw media objects
- Extract
- Main data
- Escort data
- Users
- Profiling
- Wrap
- Cut/glue
- Raw media objects
- Extract
- Main data
- Escort data

Information Systems Co-Design 4.7.2012 B. Thalheim

Why
Co-Design?
Layering
Distribution
Interactivity
Methodology
Media Types
Open
References
Media Objects

- personalized media objects
- filtration, summarization, scaling
- database schema
- static, dynamic
- information containers
- enabled manipulation requests
- supplied processes
- processes

Why Co-Design?
Layering
Distribution
Interactivity
Methodology
Media Types
Open
References
Example: Elli’s Juice trader

- use an e-commerce application as a running example for this part

- **Content:**
  - Elli’s Juice trader is going to deliver grapes, champagners, nectars, juices, blended drinks and concentrates (esp. Cocktail, Limonade, Sweet and Vitaminized juice) via the internet
  - **catalogue / specials:** there shall be a catalogue containing the offered products as well as occasional special offers
  - **news:** additional information about grapes, fruits, grape yards (also for Cocktail, Limonade, Dried juice and Concentrate) shall be provided
  - **self-info / news:** there shall be information about the shop itself, historical and actual information about certain grapes, grape producers, etc.
  - **news:** there shall be special information about best sellers, new products, etc.
  - **blackboard:** comments from (please) clients shall be made available on the web
Example: Elli’s Juice trader

- **Functionality:**
  - **cart:** the website shall allow to walk through the shop, collecting and dropping bottles to be bought
  - **search:** internal search will be supported as well as a link to a standard search engine, if a specific product cannot be found
  - **special:** there will be a permanent offer to become a member of Elli’s Club with special conditions and offers
  - **order form:** it shall be possible to order grape or Cocktail, etc. in advance, if it is not available at the moment
  - **comments:** comments about products and service can be delivered at any time

- **Usage:**
  - **roles:** there will be clients (normal and club members), product and marketing agents and just the vendor keeping track on quantities on stock
  - **profiles:** clients may be just curious, bargain-oriented or connoisseurs
Example: Elli’s Juice trader

Entry Page

- Order Form
- Cashier
- Catalogue
- Authentication
- Floor plan
- Search
- Specials
- Comments
- Account
- Cart
- News
- Self-Info
Content Modelling

- first introduce extended views and raw media types
- for this we assume an underlying database schema $S$, i.e., a collection of database 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 $r_i$,
  - a set $\text{attr}(E) = \{a_1, \ldots, a_m\}$ of attributes, each associated with a data type $\text{dom}(a_i)$ as its domain,
  - and a key $\text{id}(E) \subseteq \text{comp}(E) \cup \text{attr}(E)$
- the database types $E_i \in S$ are on levels lower than $k$ with at least one database type of level exactly $k - 1$. 
Data Types

- assume an underlying type system (data types) defined as, e.g.,
  \[ t = b \mid (a_1 : t_1, \ldots, a_n : t_n) \mid \{ t \} \mid [ t ] \]

- \( b \) represents an arbitrary collection of base types:
  - \( \text{CARD} \) for non-negative integers, \( \text{INT} \) for integers, \( \text{FLOAT} \) for floating-point numbers
  - \( \text{CHAR} \) for characters, \( \text{STRING} \) for character strings
  - \( \text{MONEY} \) for decimals, \( \text{DATE} \) 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.

- \( (\cdots), \{\cdot\} \) and \([\cdot]\) are constructors for records, sets and lists

- the semantics is defined by assigning sets of values to each type
Example

- grape
- mixed_drink
- blended_drink
- nectar
- producer
- Cocktailery
- grape yard
- region
- publication
- country
- news
- package
- offer
- bottle
- order
- delivery
- client

Why Co-Design?
Layering
Distribution
Interactivity
Methodology
Media Types
Open
References

Information
Systems
Co-Design
4.7.2012
B. Thalheim
Extended Views

- a view $V$ on a database schema $S$ consists of a view schema $S_V$ and a defining query $q_V$, which transforms databases over $S$ into databases over $S_V$

- extend the query language in a way that allows to create URLs and links (create facility):

  - `create_urls` transforms a set $\{v_1, \ldots, v_m\}$ of values into a set $\{(u_1, v_1), \ldots, (u_m, v_m)\}$ of pairs with new created URLs $u_i$ of type `URL`

  - `create_urls` also transforms a list $[v_1, \ldots, v_m]$ of values into a list $[(u_1, v_1), \ldots, (u_m, v_m)]$ of pairs with new created URLs $u_i$ of type `URL`

  - `create_url` transforms a value $v$ of any type into a pair $(u, v)$ with a new URL $u$
Raw Media Types

- extension by escort information: supertyping mechanism.

- a \textit{raw media type} has a name $M$ and consists of
  - a \textbf{content data type} $\text{cont}(M)$ (the place of a base type may be occupied by a pair $\ell : M'$),
  - a finite set of \textbf{supertypes} $\text{sup}(M)$ of raw media type names $M_i$,
  - a \textbf{defining query} $q_M$ with create-facility such that $(\{t_M\}, q_M)$ defines a view
  - a set of \textbf{operations} $\text{op}(M)$ for accessing and changing the underlying database

- finite closed sets $C$ of raw media types define \textit{content schemata}

- defining a raw media type may be regarded as a three-phase procedure using \textit{filtration} and \textit{summarization} and \textit{operation attachment}
Example

Filtration (simple view)

New Zealand red grapes from 1995 on plus grape yards, regions plus orders plus offers, packages

Information System Co-Design
4.7.2012
B. Thalheim

Why Co-Design?
Layering Distribution Interactivity Methodology Media Types Open References

Content Information Concept Topic
Example (cont.)

Summarization
(computable view)

≥ 30 \% morello
average price per liter and year consumed liters

colour = ‘red’
year ≥ 1995
{..., (morello, ≥ 30),...}
{(price, year)}
{(consumption, year)}
Functionality Modelling: Attached Operations

- on operation on a raw media type $M$ consists of
  - an operation signature: name, input-parameters, output-parameters
  - a selection type which is a supertype of $cont(M)$
  - and a body which is defined via operations accessing the underlying database
- several standard operations are of particular interest:
  - ...

Functionality Modelling: Attached Operations

- ... 

- several standard operations are of particular interest:
  - **Generalization**: generation of aggregated data, especially in the case of insufficient space, i.e., roll-up, slicing, grouping
  - **Specialization**: used for querying the database in order to obtain more details for aggregated data, e.g., drill-down
  - **Reordering**: used for the rearrangement of data, e.g., pivoting, dimension destroying, pull, push, rotate
  - **Browsing**: useful in the case that information cannot be presented completely
  - **Linking**: useful whenever the user is required to imagine the context or link structure
  - **Surveying**: used for the graphical visualization of the content
  - **Searching**: attached to enable the computation of ad-hoc aggregates
  - **Join**: used for the construction of more complex raw media objects
Content Tailoring

- four extensions turning a raw media type into a media type

- **unit extension**: introduce measure units

- **order-extension**:
  - use ordering-operators $ord \leq$ on lists and sets
  - replace set types in $cont(M)$ by list type
  - **Example**: order the previous raw media type by average price followed by percentage of Cabernet followed by year

- **cohesion / adhesion**:
  - allow a controlled form of information loss
  - determine which data has to be kept together (preferably), if technical constraints or user preferences require to split the raw media type

- **hierarchical versions**:
  - flatten or deflatten along dimensions (analogous to OLAP)
  - allow condensed or detailed information presentation
Cohesion / Adhesion

- define a partial order $\leq$ on content data types, which extends sub-typing:
  - for any expression $exp$ we have $exp \leq OK$
  - for link expressions we have $(\ell : M') \leq (\ell : M'')$ iff $M''$ is a direct or indirect (via transitive closure) supertype of $M'$
  - for record expressions we have $(a_1 : exp_1, \ldots, a_m : exp_m) \leq (a_{\sigma(1)} : exp'_{\sigma(1)}, \ldots, a_{\sigma(n)} : exp'_{\sigma(n)})$ with injective $\sigma : \{1, \ldots, n\} \to \{1, \ldots, m\}$ and $exp_{\sigma(i)} \leq exp'_{\sigma(i)}$
  - for list and set structures we have $\{exp\} \leq \{exp'\}$ (or $[exp] \leq [exp']$, respectively) iff $exp \leq exp'$ holds
  - let $sup(cont(M))$ be the set of all content expressions $exp$ with $cont(M) \leq exp$
Adhesion Preorder and Proximity Values

- Preordering:
  - an adhesion preorder on \( M \) is a preorder \( \preceq_M \) on \( \text{sup}(\text{cont}(M)) \) extending the order \( \leq \)
  - smaller types with respect to \( \leq \) represent a preferred data collection (for the case of unavoidable split)
  - Example: keep name, year and grapes for a grape more ‘important’ than grape yard or region or additional information

- Proximity:
  - choose an antichain \( \exp_1, \ldots, \exp_n \) with respect to \( \preceq \)
  - this represents a possible split of information
  - 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 higher the proximity value, the more do we wish to keep the components together
  - Example: a proximity value of 0.2 for (consumption) to all other expressions in an antichain indicates that we are like to separate this part
Hierarchical Versions (1/2)

- consider dimension hierarchies as in OLAP
- the hierarchy is already implicitly defined by the component or link structures
- flattening of dimensions results in information growth, its converse in information loss
- **flattening**:  
  - if \( E' \) is a component of \( E \) corresponding to the role \( r \), then we may replace \( E \) by \( \text{flat}_r(E) \) defined as follows:
    
    \[
    \begin{align*}
    \text{comp}(\text{flat}_r(E)) &= \text{comp}(E) - \{r : E'\} \cup \text{comp}(E') \\
    \text{attr}(\text{flat}_r(E)) &= \text{attr}(E) \cup \text{attr}(E') \quad \text{and} \\
    \text{id}(\text{flat}_r(E)) &= \begin{cases} 
    \text{id}(E) - \{r : E'\} \cup \text{id}(E') & \text{for } (r : E') \in \text{id}(E) \\
    \text{id}(E) & \text{else}
    \end{cases}
    \end{align*}
    \]

  - flatten occurrences of links \( \ell : M' \) in content data types \( \text{cont}(M') \) by simply substituting \( \text{cont}(M') \) for \( \ell : M' \)

  - the resulting raw media type will be denoted as \( \text{flat}_\ell(M) \)

- **Example**: a hierarchy in our example database is grape yard/cocktailery — region — country
Hierarchical Versions (2/2)

- **deflattening:**
  
  - any subset $P \subseteq \text{comp}(E) \cup \text{attr}(E)$ allows to replace $E$ by $\text{raise}_P(E)$ and a new database type $E_{\text{new}}$ defined as follows:

    \[
    \begin{align*}
    \text{comp}(\text{raise}_P(E)) &= \text{comp}(E) - P \cup \{r_{\text{new}} : E_{\text{new}}\}, \\
    \text{attr}(\text{raise}_P(E)) &= \text{attr}(E) - P \quad \text{and} \\
    \text{id}(\text{raise}_P(E)) &= \begin{cases} 
    \text{id}(E) - P \cup \{r_{\text{new}} : E_{\text{new}}\} & \text{for } P \cap \text{id}(E) \neq \emptyset \\
    \text{id}(E) & \text{else}
    \end{cases}.
    \end{align*}
    \]

    \[
    \begin{align*}
    \text{comp}(E_{\text{new}}) &= P \cap \text{comp}(E), \\
    \text{attr}(E_{\text{new}}) &= P \cap \text{attr}(E) \quad \text{and} \\
    \text{id}(E_{\text{new}}) &= \begin{cases} 
    \text{id}(E) & \text{for } \text{id}(E) \subseteq P \\
    P & \text{else}
    \end{cases}.
    \end{align*}
    \]

- define $\text{raise}_{\text{exp}}(M)$ for a content expression occurring within $\text{cont}(M)$ by introducing a new link expression replacing $\text{exp}$

- let $\bar{H}(M)$ be the set of all raw media types arising from $M$ by applying a sequence of flat- and raise-operations to raw media types or underlying database types

- a set of hierarchical versions of $M$ is a finite subset $H(M)$ of $\bar{H}(M)$ with $M \in H(M)$
Media Types

• a media type is a unit-extended, order-extended raw media type $M$ together with an adhesion order $\preceq_M$ and a set of hierarchical versions $H(M)$

• extensions to raw media types require changes to the attached operations:
  • unit-extension: whenever values are computed, the translation of measure units has to be taken into account
  • order-extension: replace computations on sets by computations on lists
  • cohesion / adhesion: split operations as well
  • hierarchies: extend or reduce operations as well
  • in addition decide for each operation on its importance: ‘nice to have’ or ‘indispensable’
Usage Modelling

- distinguish between *user roles* and *user types*:
  - a role corresponds to specific tasks to be done on the site: normal user / client, administrator, etc.
  - roles are associated with access rights
  - a user type corresponds to different behaviour

- modeling user types:
  - finite set $\Delta$ of *user dimensions*, e.g.,
    \[
    \Delta = \{\text{goal-orientation, presentation_preferences}\}
    \]
  - each dimension $\delta \in \Delta$ has a *scale* $sc(\delta)$ (totally ordered set), e.g.,
    \[
    \begin{align*}
    sc(\text{goal-orientation}) &= \{\text{surfer, navigator, searcher}\} \\
    \text{surfer} &\leq \text{navigator} \leq \text{searcher}
    \end{align*}
    \]
  - the *set of user profiles* over $\Delta = \{\delta_1, \ldots, \delta_n\}$ is
    \[
    gr(\Delta) = sc(\delta_1) \times \cdots \times sc(\delta_n).
    \]
  - a *user type* over $\Delta$ is a convex region $U \subseteq gr(\Delta)$
Example

- roles in the bottle shop example:
  - occasional client: access to normal offers / catalogue
  - client who is a member of the club: access to special offers
  - product officer: access to add / delete offered products
  - marketing officer: access to add / delete offers and specials
  - vendor: access to update quantities on stock

- user types (as above) apply to the first two roles
Relationship to Scenarios

- consider stories, i.e., sequences of user actions
- abstract description by certain directed graphs: scenarios
- a scenario is a finite, directed graph $\mathcal{G} = (V, E)$, where the nodes are called scenes
  - associate with each scene $sc$ a view $V_{sc}$ (information consumption) and a user type $U_{sc}$
  - associate with each edge from $sc_1$ to $sc_2$ a data type (information communicated)
- scenes will be supported by media objects, but media objects are not coupled to just one scene
Figures

- use a restricted form of media types to support session objects

- formally, a *session object* $S$ consists of a data type $T(S)$, a role $r(S)$, a user type $U(S)$ and operations $op(S)$

- session objects occur as escort information on all visited pages

- these are created on entering the site and kept persistent during the whole site visit

- session objects are used to support metaphors / figures:
  - wish list, trolley, cart, cashier, order form, etc. in e-commerce
  - time table, note book, etc. for community sites
  - guide, floor plan, help desk, note book, news feeds, etc. for entertainment / edutainment

- **Example**: in the bottle shop example the cart is a session object
Presentation: The Container Metaphor

- underlying picture: selected information will be wrapped, loaded into a (generic) container, shipped to the user and then unloaded
- a container has a name $C'$ and consists of
  - a content data type $cont(C')$
  - and a presentation layout $lay(C')$
- influences on container design:
  - usage of media objects in the scenarios
  - user types
  - technical constraint: size and presentation facilities of supported presentation devices, e.g., sizes tiny, small, normal, large
Presentation: The Container Metaphor

- container parameters:
  - **capacity** of a container means restrictions to size of representable data types
  - **loadability** of a media object \((u, (M_1 : v_1, \ldots, M_k : v_k))\) into a container \(C\) holds, if the content data type \(\text{cont}(M_i)\) is a supertype of \(\text{cont}(C)\) for at least one \(i\)
  - for low capacity exploit prefetching, caching in accordance with the split due to adhesion
  - **unloadability** means the representability of the container content on a given device
  - **wrapping** of the media object \((u, (M_1 : v_1, \ldots, M_k : v_k))\) into the container \(C\) means the identification of the suitable \(M_i\) in order to load a maximum amount of information
  - **loading** of the container consists in the generation of the page replacing the parameters by the loaded information
  - we may use *style rules* for the container layout
Publications on Co-Design


- Lewerenz, J., Schewe, K.-D., Thalheim, B.: Modelling data warehouses and OLAP applications by means of dialogue objects. Proc. ER’1999, LNCS 1728, 354-368. OLAP in a consistent, powerful and simple way


Publications on Web IS Engineering

Publications on Science and Art of Conceptual Modelling


Publications on Model Suites, Evolution, Migration


Publications on Pattern Development


Publications on Component Development


Publications on Content Management


Publications on Privacy


Database and Information Systems Theory

Eötvös Loránd University of Sciences
Budapest

5.7.2012

Bernhard Thalheim
Technologie der Informationssysteme
Institut für Informatik, Christian-Albrechts-Universität zu Kiel, BRD
Kolmogorow-Professor e.h. der Lomonossow-Universität Moskau
Database and Information Systems Theory

Why
Constraints
Visuality
BPMN
Model suite
Triggering
Identity
Concepts
Open
Publications
Observations

- Right logic in the best-fitted structure, in the best complexity and micro-structure, fitted to its purpose and to the user
  - Counterexamples
    - Tuple calculus in relational database theory with the nonsense of safe formula
    - Null value logics without consideration of nil semantics and pragmatics
- State of the art:
  - Relational databases: poor structuring and superficial, incomprehensible classification of constraints
    - with many non-axiomatisation or non-decidability results
  - ER schemata: implicit structuring semantics
- Constraints with insufficient expressive power despite the high variety, e.g., cardinality constraints cannot express finiteness of cycles
Distinguishability and Identifiability

- Objects with their way of identification.
  Value-representability is a must.
- Objects with their properties (intext).
- Objects with their context
  e.g. relationships to other objects, concerns, viewpoints
- General management of worlds, e.g., relationships with properties also many-dimensional, multi-layered
- Detecting cycles is a nightmare. *Cyclic structuring* leads to non-first-order logics. Structures with abstract linking are potentially cyclic.
- The OID has already been implemented by relational databases: tuple identifier. The OID is too expressive.

Object-oriented approaches have improved and damaged culture at the same time.

Next after OO: components!!!!
**Pearls of Structuring**

*Lessons Learned for the Good*

Hierarchical structuring of types leads to a generalized first-order predicate logics.

**Optionality is difficult** and is handled by

- **weak semantics** for object sets $\mathcal{O}$ with $\text{compon}^{\text{opt}}(\mathcal{O}) \neq \emptyset$

  not true: $\mathcal{O} \models \{\alpha_1, \ldots, \alpha_m\}$ iff $\mathcal{O} \models \{\alpha_i\}$

- **strong semantics** for object sets $\mathcal{O}$ with $\text{compon}^{\text{opt}}(\mathcal{O}) \neq \emptyset$

  in general: $\mathcal{O} \not\models \alpha \rightarrow \alpha$

  $\mathcal{O} \models \alpha \rightarrow \beta$ and $\mathcal{O} \models \beta \rightarrow \gamma$ do not imply $\mathcal{O} \models \alpha \rightarrow \gamma$.

**Axiomatizability:** Implication operator $\Phi_{\models}$ reflexive, monotone, closed and compact $\iff$ deductive system $\Gamma$ $\Phi_{\Gamma} \equiv \Phi_{\models}$

$\Phi_{\models}$ inference, deduction properties and generalization invariant:

$\Phi_{\Gamma}^*(\emptyset) = \Phi_{\models}^*(\emptyset)$
Constraints

- Functional, key, inclusion and exclusion dependencies are constraints that are natural in the relational model
  - Key-based inclusion dependencies are handicapped
- Multi-valued dependencies are far better expressed in the ER model
- Cardinality constraints are overloaded
  - better treat maximal and minimal cardinality in separate systems
- Integrity enforcement is treated separately for types
- Join dependencies are representational constraints
### Static Integrity Constraints

<table>
<thead>
<tr>
<th>Layer</th>
<th>Partial identification</th>
<th>relative independence</th>
<th>existence constraint</th>
<th>redundancy constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual layer</td>
<td>functional, equality-generating</td>
<td><strong>multivalued</strong>, hierarchical, join, exclusion, tuple-generating constraint, horizontal decomposition</td>
<td>null-value-free, union constraint, numerical, cardinality</td>
<td>inclusion, exclusion constraint</td>
</tr>
<tr>
<td>Implementation layer</td>
<td>key, uniqueness, trigger, check</td>
<td>decomposition, stored procedures, trigger</td>
<td>no null, stored procedures, trigger</td>
<td>referential integrity, surrogate, capsules</td>
</tr>
<tr>
<td>User layer</td>
<td>identification</td>
<td><strong>structure</strong> !</td>
<td>no null</td>
<td>elementary facts</td>
</tr>
</tbody>
</table>

95 classes in “Dependencies in Relational Databases”!!
Optimisation of Behaviour Through Normalisation of Database Structuring

Normalisation as a vehicle for performance improvement

(A) *Redundancy* problematic whenever additional actions are required;

(B) *Blocking of management* due to the information capacity of the schema (e.g., insertion anomaly);

(C) *Information loss after database modification* whenever data are eagerly deleted (e.g., deletion anomaly);

(D) *Evolution sensitivity* and instability of the database due to changes;

(E) *Different abstractions* at the same time (e.g., views, derived attributes, logs as part of structures);

(F) *Performance problems* caused by integrity constraint maintenance (e.g., update anomalies);

*Denormalisation*, index management, inflexibility against evolutionary changes;

(G) *Expensive maintenance, operating and modification* due to consistency maintenance.
The Constraints Handling Framework

(1) Specification level: description of the constraints property, validation, policies for evaluation, specific policies, transformations of constraint properties to others;

(2) Control or technical level: application of the constraint constraint property portfolio, techniques and methods (e.g., cascade, restrict, no action, defaultify nullify);

(3) Application or technology level: management of constraint handling

(4) Establishment or organisational level: based on a methodology and constraint maintenance system.

Level five: facilities for handling satisfaction of constraints and for predicting changes of satisfaction

Level six optimises the constraint management;

Level seven uses experiences for innovation and adaptation.
Constraints: Forgotten Properties

- Functional dependencies: $\forall x_1, x_2 (\psi(x_1, x_2) \rightarrow \phi(x_1, x_2))$, expressible through equality sets, invariance properties for subsets, sensitive against supersets, pair algebra, two-tuple-implication

- Hierarchical dependencies: reasoning through the tableau calculus, not invariant for most operations, semantical gap between FD’s and MVD’s, variety of definitions

- Implicit model assumptions: component inclusion condition, identifiability whenever set semantics, cycles and finiteness, superficial structural representation

- Normalization as structural instead of behavioral optimization

- Schema transformation and equivalence
Constraints: Supplanted Properties

- Finiteness properties: finite implication, finite calculi, finite representation of potentially infinite sets, finite computation (‘safety’)
- Parallel execution: transaction based concurrent semantics, causal semantics of processes, predicative semantics for engines
- Unsafety of data: null values (overloaded, logics, computation, identification), fuzzy data with error model, aggregated macro-data versus micro-data
- Completeness of specification: infeasible for humans, normal forms based on completeness, order dependence of (normalization) algorithms, time-dependence of constraints validity, real-life constraints
- Weak constraints: deontic maintenance, temporal maintenance, default values
Normal Forms

Maintenance through keys, domain and referential constraints

Forbidden substructures used for the definition of normal forms, e.g.,

3NF: $Z \rightarrow \{A\} \in \Sigma^+, A \notin Z$, A non-key attribute $Z \rightarrow U \notin \Sigma^+$

Boyce-Codd-Normalform: $Z \rightarrow \{A\} \in \Sigma^+, A \notin Z$, $Z \rightarrow U_i \notin \Sigma^+$

4NF: $Z \rightarrow \rightarrow X \in \Sigma^+, X \not\subseteq Z$, $Z \rightarrow U \notin \Sigma^+$

5NF: $(Y_1, ..., Y_k) \in \Sigma^+, \exists j : Y_j \rightarrow U_i \notin \Sigma^+$

Project-Join-NF: $(Y_1, ..., Y_k) \in \Sigma^+, \{X \rightarrow U_i \in \Sigma^+\} \neq (Y_1, ..., Y_k)$

Inklusion-NF: $R_i[X] \subseteq R_j[Y] \in \Sigma^+, Y \rightarrow U_j \notin \Sigma^+$

Domain-Key NF (DKNF): $\alpha \in \Sigma^+, \text{non-trivial, } \Sigma_K \not\models \alpha$

more than 35 other useful normal forms have been introduced

Synthesis and decomposition algorithms
## Solved and Open Problems

<table>
<thead>
<tr>
<th>class</th>
<th>axiomatiz.</th>
<th>complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hilbert/Gentzen</td>
<td>worst</td>
</tr>
<tr>
<td>keys</td>
<td>√ Hilbert</td>
<td>√</td>
</tr>
<tr>
<td>keys (bounded domains)</td>
<td>√ Hilbert</td>
<td>√</td>
</tr>
<tr>
<td>keys (with nulls)</td>
<td>√ Hilbert</td>
<td>√</td>
</tr>
<tr>
<td>keys NF²</td>
<td>? √ LogChar (hierarch.)</td>
<td>????</td>
</tr>
<tr>
<td>fd's</td>
<td>√ Hilbert, √ Graph</td>
<td>√</td>
</tr>
<tr>
<td>Hungarian deps</td>
<td>√ LogChar</td>
<td>??</td>
</tr>
<tr>
<td>multivalued deps</td>
<td>√ Hilbert, √ Graph</td>
<td>????</td>
</tr>
<tr>
<td>hierarchical deps.</td>
<td>√ Hilbert, √ Graph</td>
<td>??</td>
</tr>
<tr>
<td>join deps.</td>
<td>√ Gentzen</td>
<td>??</td>
</tr>
<tr>
<td>cardinality constr.</td>
<td>√ Charact.</td>
<td>??</td>
</tr>
<tr>
<td>inclusion/exclusion</td>
<td>√ Hilbert</td>
<td>????</td>
</tr>
<tr>
<td>transition constr.</td>
<td>????</td>
<td>???</td>
</tr>
</tbody>
</table>

**Warning:** Hilbert-/Gentzen-kind calculi are only one option!
Problems: Implicit Model-Inherent Integrity Constraints

Typical implicit model-inherent integrity constraints:

*Component-construction constraints* existence, cardinality and inclusion of components;

*Identification constraints* implicitly used for the set constructor value-representability or weak-value representability;

*Acyclicity and finiteness* of structuring cardinality constraints may be based on potentially infinite cycles;

*Implicit meaning of constraints* , e.g., value construction, optional/mandatory existence, derived value/object, identification, union, combination/association;

*Superficial structuring* leads to representation of constraints through structures.
However: Enrichment

Lessons of implicit definition

Component-construction constraints: existence, cardinality, inclusion impact on translation and implication

Identification constraints: set, list constructors

Acyclicity and finiteness of structuring supports axiomatization and definition of the algebra

Superficial structuring leads to representation of constraints through structures

Implicit model-inherent constraints belong to the performance and maintenance traps.
Problems: Modelling Problems with Constraints

Representable by structuring e.g. join dependencies

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c}
  & X & \text{first} & \text{second} \\
\hline
Y & & & \\
Z & & & \\
\hline
\cdots & & & \\
\hline
\end{array}
\]

\[
X \rightarrow Y|Z
\]

\[
Y \cap Z = \emptyset, \quad X \cup Y \cup Z = R
\]

another relational representation by decomposition:

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c}
  & X & Y & \cdots & Z & \cdots \\
\hline
\cdots & \cdots & \cdots & \cdots \\
\end{array}
\]

Humans Do you reason abstractly?

Can you understand logics?

Do you understand your colleagues semantics?

Systems Are there systems which use integrity constraints beyond keys, referential integrity and domain constraints?
Problems: Computational Problems with Constraints

*Combinatorial complexity* is exponential necessary complete set of constraints for normalization, optimization
either we need to know all valid constraints (CWA) or to know constraints which are valid and invalid

*Too simple properties* simple axiomatisation, ...
“too simple” is misunderstood and taken for granted, also intentionally extended to all other cases

*Maintenance of consistency* full of mis-conceptions
triggers are only and only applicable to strict hierarchical constraint sets but active database people believe in triggering

solution: greatest consistent specialization approach to refinement of operations

*Object-identification* through identifiers has to be supported by value identification thus XML will never have a computational query algebra
Combinatorial Problems with Constraints 1

Maximal number of minimal keys (sharp)

\[
\binom{n}{\lfloor \frac{n}{2} \rfloor}
\]

(J. Demetrovics (1979))

Restricted domains

\[| \text{dom}(B_i) | \leq k \ (1 \leq i \leq n)\]

\[
k^4 < 2n + 1
\]

\[
\binom{n}{\lfloor \frac{n}{2} \rfloor} - \lfloor \frac{n}{2} \rfloor
\]

(\(\beta (1984)\))

same behavior on non-uniform domains

similar behavior in presence of null values

(Selesnev, \(\beta (1985)\))
Combinatorial Problems with Constraints 2

Functional dependencies
maximum number \( N(n) \) of basic functional functional dependencies for schemata on \( n \) attributes

\[
2^n \left( 1 - \frac{4 \log_2 \log_2 n}{\log_2 e \log_2 n} \right) (1+o(1)) \leq N(n) \leq 2^n \left( 1 - \frac{\log_2^{3/2} n}{150 \sqrt{n}} \right)
\]

minimal generating sets of fd's (\( n \) odd; \( n \) even (replace \( \frac{1}{n^2} \) by \( \frac{1}{n^{2/3} n^{8/3}} \))

\[
\left( \binom{n}{\left\lfloor \frac{n}{2} \right\rfloor} \right) + \frac{1}{n^2} \left( \binom{n}{(n+3)/2} \right) \leq M(n) \leq 2^n \left( 1 - \frac{\log_2^{3/2} n}{150 \sqrt{n}} \right)
\]

closed families of fd's

\[
2^{\left\lfloor \frac{n}{2} \right\rfloor} \leq Cl(\mathcal{F}, n) \leq 2^{\left( \frac{n}{\left\lfloor \frac{n}{2} \right\rfloor} \right)(1+o(1))}
\]

(J. Demetrovics, G.O.H. Katona, D. Miklos, \( \beta \) (1982-2006))
Combinatorial Problems with Constraints 3

rather than describing the entire set of basic functional dependencies use a relation $R^C$ which allows to reason on the set of constraints

Armstrong relations - size

$$\frac{1}{n^2} \left( \left\lfloor \frac{n}{2} \right\rfloor \right) \leq L_{\text{key}}(n) \leq \left( \left\lfloor \frac{n}{2} \right\rfloor \right) + 1$$

$$c_1 n^{k-\frac{1}{2}} \leq L_{\text{key}(k)}(n) \leq c_2 n^{k-\frac{1}{2}}$$

$$\frac{1}{n^2} \left( \left\lfloor \frac{n}{2} \right\rfloor \right) \leq L_\mathcal{F}(n) \leq \left( \left\lfloor \frac{n}{2} \right\rfloor \right) \left( 1 + \frac{c_3}{\sqrt{n}} \right)$$

(J. Demetrovics, G.O.H. Katona & Hungarians, $\beta$ (1982-2005))
Good Case however: Average Complexity of Constraints 1

**conjecture:** average relation has a key system of polynomial size

average relation: almost all minimal keys are of same length

\[ 2 \log_{|D|}(R^C) \]

Poisson distribution

- deviation is constant
- if \( X \) is of length \( 2 \log_{|D|}(R^C) \) then \( X \) is minimal
- deviation decreases with increasing domain sizes
- same behaviour for the non-Bernoulli case

small fraction of relations with large key systems

similar results for functional dependencies

(J. Demetrovics, G.O.H. Katona, D. Miklos, O. Selesnev, \( \beta \), 1995-2005)
Good Case however: Average Complexity of Constraints 2

Behavior of the Key Probability (Frequency Polygon)
**Good Case however: Average Complexity of Constraints 3**

average length $av_n(l, 2)$ of minimal keys in relations with $m$ tuples

$$|\log_2 m| \leq av_n(m, 2) \leq 2|\log_2 m|.$$ 

random relation

$$P(X \rightarrow \{A_j\}, l) = \begin{cases} 
0, & \text{if } 2\log l - \sum_{A_i \in X} H_2(\kappa_i) \rightarrow +\infty, \\
 e^{-2a-1(2^{-H_2(\kappa_j)}-1)}, & \text{if } 2\log l \\
1, & \text{if } 2\log l - \sum_{A_i \in X} H_2(\kappa_i) \rightarrow a, \\
& \text{if } 2\log l - \sum_{A_i \in X} H_2(\kappa_i) \rightarrow -\infty.
\end{cases}$$

If $X$ is a set of attributes of size definitely larger than $\frac{2\log l}{H_2}$ then $X \rightarrow \{A_j\}$ holds with high probability for any $A_j$. 
Problems: Logical Problems Restricting Reasoning based on Constraints

(un-)decidable implication

\[ \begin{align*}
&\models = \models_{\text{fin}} \text{ or not} \\
&\text{Axiomatization or not} \quad \text{Hilbert type} \quad \text{Gentzen type}
\end{align*} \]

Properties of constraint sets

- decidable for tuple-generating and equality-generating dependencies
- undecidable for embedded JD’s
- undecidable for embedded MVD’s
- undecidable for FD’s and ID’s

Axiomatizability of constraint classes

- FD’s and ID’s are not axiomatizable
  
  \emph{but weak FD’s and ID’s axiomatizable}

- JD’s not axiomatizable
  
  \begin{itemize}
    \item but tuple-generating and equality-generating dependencies are axiomatizable
  \end{itemize}
Missing Research on Constraints

“Tuple-generating” constraints are in reality existence constraints
value must (not) exist; also in domain, referenced type (inclusion/exclusion/component),
potentially also in same class (redundancy constraint)
multivalued and hierarchical dependencies
hidden properties, e.g., subset property

“Equality-generating” constraints are in reality binding constraints
value must/might (not) be equal, is potentially derivable,
generalised to \((p,q)\) constraints, used for cardinality constraints

Cycle constraints are not yet developed

cycle must/might (not) exist, cycles in values (‘marriage’, ‘friend’), cycle in types, cycles in value sets

Logical gap between existence and binding constraints

Derived constraints for derived structures

Kinds of constraints, e.g., explicit declaration, coupling, semantic association,
semantic unit, structural representability
# The Power of Functional Dependencies

![Diagram](image.png)

<table>
<thead>
<tr>
<th>Functional dependencies</th>
<th>Schema after decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A} → {B}</td>
<td>Schema 1</td>
</tr>
<tr>
<td>{A} → {B} → {C}</td>
<td>Schema 1'</td>
</tr>
<tr>
<td>{A} → {B, C}</td>
<td>Schema 1</td>
</tr>
<tr>
<td>{A} ↔ {B} → {C}</td>
<td>Schema 1, Schema 1'</td>
</tr>
<tr>
<td>{A} ↔ {B, C}</td>
<td>Schema 2</td>
</tr>
<tr>
<td>{A} ↔ {B} → {C}</td>
<td>Schema 3</td>
</tr>
<tr>
<td>{A} → {B} ← {C}</td>
<td>Schema 3</td>
</tr>
<tr>
<td>{A, B} → {C}</td>
<td>no new schema</td>
</tr>
<tr>
<td>{A} ↔ {B}</td>
<td>Schema 1, Schema 1'</td>
</tr>
</tbody>
</table>
**FD-Axiomatisation for Tuple and Set Constructors**

Given $\mathcal{X}, \mathcal{Y}, \mathcal{Z} \subseteq \text{Sub}(T)$

**reflexivity axiom:** \( \mathcal{X} \rightarrow \mathcal{Y} \Rightarrow \mathcal{Y} \subseteq \mathcal{X} \)

**sub-attribute axiom:** \( \{ \mathcal{X} \} \rightarrow \{ \mathcal{Y} \} \Rightarrow \mathcal{Y} \leq \mathcal{X} \)

**extension rule:** \( \mathcal{X} \rightarrow \mathcal{Y} \Rightarrow \mathcal{X} \rightarrow \mathcal{X} \cup \mathcal{Y} \)

**restricted meet rule:** \( \mathcal{X} \rightarrow \mathcal{Y} \Rightarrow \mathcal{X} \rightarrow \mathcal{X} \cap \mathcal{T} \mathcal{Y} \)

\[ \exists \mathcal{X}' \geq \mathcal{X} \exists \mathcal{Y}' \geq \mathcal{Y} : \mathcal{X}' \cdot \mathcal{Y}' = \lambda \land \mathcal{X} \sqcap \mathcal{Y} = \mathcal{X}' \sqcap \mathcal{Y}' \]

**transitivity rule:** \( \mathcal{X} \rightarrow \mathcal{Y}, \mathcal{Y} \rightarrow \mathcal{Z} \Rightarrow \mathcal{X} \rightarrow \mathcal{Z} \)

**Join (generalised $\cap$):**

\[ Z \text{ sub-attribute of } \mathcal{Y} \text{ for } \mathcal{Y}, \mathcal{Z} \in \text{sub}(\mathcal{X}): \mathcal{Y} \sqcap_X \mathcal{Z} = \mathcal{Z} \]

\[ \mathcal{X} = A\{B\}, \mathcal{Y} = A\{C\}, \mathcal{Z} = A\{D\}, \mathcal{C}, \mathcal{D} \leq \mathcal{B} : \mathcal{Y} \sqcap_X \mathcal{Z} = A\{C \sqcap_B D\} \]

\[ \mathcal{X} = A(A_1, ..., A_n), \mathcal{Y} = A(B_1, ..., B_n), \mathcal{Z} = A(C_1, ..., C_n), B_i, C_i \leq A_i (\forall i) : \mathcal{Y} \sqcap_X \mathcal{Z} = A(B_1 \sqcap_{A_1} C_1, ..., B_n \sqcap_{A_n} C_n) \]

**Meet (generalised $\cup$):**

\[ Z \text{ sub-attribute of } \mathcal{Y} \text{ for } \mathcal{Y}, \mathcal{Z} \in \text{sub}(\mathcal{X}): \mathcal{Y} \sqcup_X \mathcal{Z} = \mathcal{Y} \]

\[ \mathcal{X} = A\{B\}, \mathcal{Y} = A\{C\}, \mathcal{Z} = A\{D\}, \mathcal{C}, \mathcal{D} \leq \mathcal{B} : \mathcal{Y} \sqcup_X \mathcal{Z} = A\{C \sqcup_B D\} \]

\[ \mathcal{X} = A(A_1, ..., A_n), \mathcal{Y} = A(B_1, ..., B_n), \mathcal{Z} = A(C_1, ..., C_n), B_i, C_i \leq A_i (\forall i) : \mathcal{Y} \sqcup_X \mathcal{Z} = A(B_1 \sqcup_{A_1} C_1, ..., B_n \sqcup_{A_n} C_n) \]

Example: 

\( T^C = \{t_1, t_2\} \), \( t_1 = \{ (\text{Alf}, \text{Math}), (\text{Betty}, \text{DB}) \} \), \( t_2 = \{ (\text{Betty}, \text{Math}), (\text{Alf}, \text{DB}) \} \)

\( \models_{T^C} \{ \text{Library}\{\text{Lending(Student)}\} \rightarrow \{ \text{Library}\{\text{Lending(Book)}\} \} \}

\( \not\models_{T^C} \{ \text{Library}\{\text{Lending(Student)}\} \rightarrow \{ \text{Library}\{\text{Lending(Student, Book)}\} \} \)
FD-Axiomatisation in Extended HERM

Classical: tuple and list constructors

\[ \frac{X \rightarrow Y}{Y \subseteq X}, \frac{X \rightarrow Y}{X \rightarrow X \sqcup_T Y}, \frac{X \rightarrow Y, Y \rightarrow Z}{X \rightarrow Z} \]

Extended HERM: tuple, list, set, and multiset constructors

\[ \mathcal{X}, \mathcal{Y}, \mathcal{Z} \subseteq \text{Sub}(T) \]

Axioms

\[ \frac{X \rightarrow Y}{Y \subseteq \mathcal{X}}, \frac{\{X\} \rightarrow \{Y\}}{Y \leq X}, \frac{\{X, Y\} \rightarrow \{X \sqcup_T Y\}}{X, Y \text{ reconcilable}} \]

Rules

Extension \[ \frac{X \rightarrow Y}{X \rightarrow X \sqcup Y} \]

Transitivity \[ \frac{X \rightarrow Y, Y \rightarrow Z}{X \rightarrow Z} \]

reconcilable: have inductively the same structuring for tuple and multiset constructors or \(Y \leq X\)
or \(X \leq Y\).
\(Y \leq X\): there exist a projection from \(X\) to \(Y\).
\(X \sqcup_T Y\): lattice operation on \(T\)-subtypes.
Functional Dependencies

(Not as well understood as it seems!)

Explicit declaration of partial identification: explicitly declaring a functional association $X \xrightarrow{\text{Ident}} Y$.

Tight functional coupling: numerical constraints i.e., $X \xrightarrow{\text{Num}} Y$

Another denotation is based on cardinality constraints.

Semantic constraint specific for the given application: application has a limited scope and allows us to strengthen the constraint $X \xrightarrow{\text{Sem}} Y$.

Semantical unit with functional coupling: Semantical units are those reducts of a type that are essential in the given application $X \xrightarrow{\text{Unit}} Y$.

Structural association among units: Semantical units may allow a separation of concerns for certain elements $X \xrightarrow{\text{Struct}} Y$.
Functional Dependencies

(Not as well understood as it seems!)

\{A\} \rightarrow \{C\}: explicit and direct dependency.

\{A\} \rightarrow \{B\} is an inter-type constraint and leaves the scope of type \(T_A\).

<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Instantiation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit declaration</td>
<td>(T_B = \text{StudyProgram}, B = \text{ProgramCode}, D = \text{ProgramName})</td>
</tr>
<tr>
<td>Tight coupling</td>
<td>(T_A = \text{Student}, T_A.to.T_B = \text{MajorProgram}, T_B = \text{StudyProgram})</td>
</tr>
<tr>
<td>Semantic constraint</td>
<td>(T_B = \text{StudyProgram}, B = \text{ProgramCode}, D = \text{ResponsibleProfessor})</td>
</tr>
<tr>
<td>Semantical unit</td>
<td>(T_B = \text{StudyProgram}, B = \text{ProgramCode}, D = \text{ProgramDegree})</td>
</tr>
<tr>
<td>Structural association</td>
<td>(T_A = \text{Student}, T_B = \text{RegisteredStudent}, A = \text{PersonID}, B = \text{StudentNr})</td>
</tr>
</tbody>
</table>
Axiomatisation follows the Classical Approach

**Augmentation** e.g.,

<table>
<thead>
<tr>
<th>Trivialisation of identification</th>
<th>Augmentation of semantics scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R : X \xrightarrow{\text{Ident}} Y \sqcup_R Y' )</td>
<td>( R : X \xrightarrow{\text{Sem}} Y \sqcup_R Y' )</td>
</tr>
<tr>
<td>( R : X \sqcup_R X' \xrightarrow{\text{Ident}} Y )</td>
<td>( R : X \sqcup_R X' \xrightarrow{\text{Sem}} Y )</td>
</tr>
</tbody>
</table>

**Implication** e.g.:

\[
\begin{align*}
R : X \xrightarrow{\text{Ident}} Y, & \quad R : Y \xrightarrow{\text{Sem}} Z \\
R : X & \xrightarrow{\text{Ident}} Z
\end{align*}
\]

\[
\begin{align*}
R : X \xrightarrow{\text{Sem}} Y, & \quad R : Y \xrightarrow{\text{Ident}} Z \\
R : X & \xrightarrow{\text{Sem}} Z
\end{align*}
\]

Axiomatisation as pair algebra and by lifting among types.

Basis: subset, substructuring, implication properties of FD’s.
Key Dependencies

Descriptive meanings of the key concept depending on the levels

- **Language definition level**: uniqueness
- **External level**: uniqueness inherited
  - expressed via identification
  - existence property: secondary role
- **Conceptual level**: identification and integrity
  - surrogate concept, key dependency
- **Internal level**: uniqueness covered by identification
  - additionally of interest: restrictions like invariant values in keys, accessibility

```
Language definition level

external level

conceptual level

internal level

uniqueness

identification

integrity constraint (key dependency)

invariance

identification (surrogate)

existence

open

publications

information systems

theory

5.7.2012

b. thalheim
```
Explicit and Excluded Functional Dependencies

$X 
otightarrow Y$: the functional dependency $X ightarrow Y$ is not valid.

**Axioms**

\[ X \cup Y \rightarrow Y \]

**Rules**

\[ (1) \frac{X \rightarrow Y}{X \cup V \cup W \rightarrow Y \cup V} \]
\[ (2) \frac{X \rightarrow Y, Y \rightarrow Z}{X \rightarrow Z} \]
\[ (3) \frac{X \rightarrow Y, X \notightarrow Z}{Y \notightarrow Z} \]
\[ (4) \frac{X \notightarrow Y}{X \notightarrow Y \cup Z} \]
\[ (5) \frac{X \cup Z \notightarrow Y \cup Z}{X \cup Z \notightarrow Y} \]
\[ (6) \frac{X \rightarrow Z, X \notightarrow Y \cup Z}{X \notightarrow Y} \]
\[ (7) \frac{Y \rightarrow Z, X \notightarrow Z}{X \notightarrow Y} \]
**Multivalued Dependency**  \( X \rightarrow\!
abla\rightarrow Y|Z \)

Given a relational schema \( R = (U_R, \Sigma_R) \), some relation \( R^C \) on \( R \) subsets \( X, Y \subseteq U_R \) and \( Z = U_R \setminus (Y \cup X) \)

1. **Classical definition:**  \( R^C \models X \rightarrow\!
abla\rightarrow Y|Z \)
   
   if for all \( t, t' \in R^C \) with \( t =_X t' \) an object \( t'' \in R^C \)
   
   exists with \( t'' =_{X \cup Y} t \) and \( t'' =_{X \cup Y} t' \)

2. **Decomposition definition:**  \( R^C \models X \rightarrow\!
abla\rightarrow Y|Z \)
   
   if \( R^C = R^C[X \cup Y] \Join R^C[X \cup Z] \)

   We may further conclude: If \( R^C \models X \rightarrow Y \) then \( R^C \models X \rightarrow\!
abla\rightarrow Y|Z \)

3. **Independence definition:**  \( R^C \models X \rightarrow\!
abla\rightarrow Y|Z \)
   
   if \( (\sigma_{X=x}(R^C))[Y] = (\sigma_{(X=x)\land(Z=z)}(R^C))[Y] \)

   for all \( X \)-values \( x \in R^C[X] \) and all \( Z \)-values \( z \in R^C[Z] \)

**Proof of equivalence:**  \((1) \Rightarrow (2) \Rightarrow (3) \Rightarrow (1)\)

E.g., for \((1) \Rightarrow (2)\) we need to prove: \( R^C \supseteq R^C[X \cup Y] \Join R^C[X \cup Z] \)

Given \( t_1 \in R^C[X \cup Y] \) and \( t_2 \in R^C[X \cup Z] \) mit \( t_1 =_X t_2 \)

\( \Rightarrow \{t_1\} \Join \{t_2\} \subseteq R^C \) because \( R^C \models X \rightarrow\!
abla\rightarrow Y|Z \)
**Multivalued Dependency** $X \rightarrow\rightarrow Y|Z$

Given a relational schema $R = (U_R, \Sigma_R)$, some relation $R^C$ on $R$ subsets $X, Y \subseteq U_R$ and $Z = U_R \setminus (Y \cup X)$

(4) **Constructor** definition: $R^C \models X \rightarrow\rightarrow Y|Z$
   
   if for all $x \in R^C[X]$ with $(\sigma_{X=x}(R^C))[Y \cup Z] = (\sigma_{X=x}(R^C))[Y] \times (\sigma_{X=x}(R^C))[Z]$
   
   i.e. $\nu_Z(\nu_{Y\setminus X}(\nu_X(R^C))) = \nu_{Y\setminus X}(\nu_{Z}(\nu_X(R^C)))$

(5) **Structuring** definition: $R^C \models X \rightarrow\rightarrow Y|Z$

<table>
<thead>
<tr>
<th>$R^C$</th>
<th>${X}$</th>
<th>${Y}$</th>
<th>${Z}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>$\ldots$</td>
<td>$A_{k+1}$</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
<td>$\ldots$</td>
</tr>
</tbody>
</table>

$X = \{A_1, \ldots, A_k\}$, $Y = \{A_{k+1}, \ldots, A_m\}$, $Z = \{A_{m+1}, \ldots, A_n\}$

by a nested relation
Points of View through Separation

Example: Employee, Dependent, Project, Supplier, Product

\{ Employee \} \rightarrow \{ Department, Dependent \} | | \{ Project, Product, Supplier \}
\{ Employee \} \rightarrow \{ Dependent \} | | \{ Department, Project, Product, Supplier \}
\{ Project \} \rightarrow \{ Employee, Department, Dependent \} | | \{ Product, Supplier \}
\{ Product \} \rightarrow \{ Department, Employee, Dependent, Project \} | | \{ Supplier \}

Definition (6): MVD defined by two (Y,Z) relationship types can be generalised to hierarchical dependencies and n-ary relationship types

\begin{align*}
\text{DepBasis}(\text{Employee}, \Sigma) \setminus \{ \text{Employee} \} & = \\
& \{ \{ \text{Department} \}, \{ \text{Dependent} \}, \{ \text{Project}, \text{Product}, \text{Supplier} \} \}
\end{align*}

\begin{align*}
\text{DepBasis}(\text{Project}, \Sigma) \setminus \{ \text{Project} \} & = \\
& \{ \{ \text{Product} \}, \{ \text{Supplier} \}, \{ \text{Employee}, \text{Department}, \text{Dependent} \} \}
\end{align*}
Derivation Rules for MVD and FD

$X, X', X'', Y, Y', Z, Z', V, W \subseteq U$

All sets within an MVD constitute a cover of $U$

Axiom:

$(1_F) \quad X \cup Y \rightarrow Y$  \quad $(1_M) \quad X \rightarrow \emptyset|Z$

Rules

$(21_F) \quad \frac{X \rightarrow Y}{X \rightarrow Y}$  \quad (FD – MVD reduction)

$(22_F) \quad \frac{X \rightarrow Y, Y \rightarrow Z}{X \cup V \cup W \rightarrow V \cup Z}$  \quad (FD transitivity)

$(21_M) \quad \frac{X \rightarrow X'|Y|X'' \cup Z}{X \cup X'|X'' \rightarrow Z|Y}$  \quad (MVD weakening)

$(23_M) \quad \frac{X \cup V \rightarrow Y|Z, X \rightarrow Y \cup Z|V}{X \rightarrow Y|Z \cup V}$  \quad (MVD root reduction)

$(3_{F,M}) \quad \frac{X \cap Y \rightarrow Y \setminus X|X \setminus Y, X \rightarrow Z}{X \cap Y \rightarrow Y \cap Z}$  \quad (FD pushback)

$(1_F), (1_M), (21_F), (22_F), (23_M), (3_{F,M})$ are complete and correct

for functional and multivalued dependencies
Derivation Rules (structurally)

Axiom

\[ X \cup Z \rightarrow X \]

Root reduction

\[ Y(X) \rightarrow X \cup V \]
\[ (X)Z \]
\[ Y \cup Z(X) \rightarrow X \]
\[ (X)V \]

Weakening

\[ X \cup Y(X) \rightarrow X \]
\[ (X)X' \cup Z \]

Tree restructuring

\[ Y(X) \rightarrow X \]
\[ (X)Z' \cup V \]

\[ Y \cup Y'(X) \rightarrow X \]
\[ (X)Z' \cup Z' \]
\[ Y \cup Z(X) \rightarrow X \]
\[ (X)Z' \cup Z' \cup Y' \]
Referential Constraints

Generalising foreign key constraints

Relational understanding $\pi_{\text{list}(A)}(\alpha) \subseteq \pi_{\text{list}(C)}(\beta)$

1. table / relation type $\alpha$ with primary key $A$
   - either artificial ID or surrogate key or value key
2. $B$ is a key in $\beta$
3. $C$ can be property based

generalise to $\text{view}_1,\text{list}(A)(\alpha) \subseteq \text{view}_2,\text{list}(C)(\beta)$

meaning:
- reference as a pointer to already existing data
- reference as a binding means
- reference with master/slave and other protocols

Differentiate introduction and reference
Cardinality Constraints are Global Constraints

Trip
\{ 1 \} \rightarrow \text{starts_in} \rightarrow \text{City} \rightarrow \text{visits}
\{ 3,4,7 \} \rightarrow \text{corrected: \{ 3 \}}
\{ 1,2,3,6 \} \rightarrow \text{corrected: \{ 6 \}}
\{ 2,3,5,6 \} \rightarrow \text{corrected: \{ 2 \}}
Open Issues

A maximally reduced list.

Coherence constraints, e.g., Zip

Define/use constraints resulting in sophisticated treatment of object identity

similar to mirror typing in hierarchical DBMS

Redundancy maintaining constraints

Explicit handling of incompleteness of constraint set in specifications

Separation of concern for constraints

Explicit specification of enforcement

SPA framework (sensing, processing, actuation)

see also the SDKB 2008 list, list in the HERM book
Questions for the Next Decade

- Complexity of key systems in nested relations
- Complexity of other constraint classes:
  - Multi-valued, hierarchical, join dependencies
  - Inclusion and exclusion constraints
  - Satisfiability of constraint sets
- **Graphical reasoning systems**
  - What is known on combinatorial properties of weak constraints?
  - Average complexity of Armstrong relations?
  - Average complexity of functional dependencies?
  - Approaches based on combinatorics or probability theory
“Remembering the Future”

- Semantics with explicit consideration of structuring and type system variants
  (e.g., multivalued dependencies representing structuring and decomposition)
- Combinatorial complexity for holistic consideration
  (i.e., exponential ('worst case'))
- Ability for expressing constraints by users
  (Too natural and not state constraints; difficult to express constraints; abstraction; semantics
  by different users; everyday life logics, different interpretations (strong, ‘normal’ (logics),
  weak))
- Computational facilities in systems
  (keys, referential integrity constraints, domain constraints, implementation (physical and/or
  logical) constraints, views, CASE tool biases)
- Too simple properties (also for axiomatisation) are misleading
  anything in the world has its price we have to pay at the minimum; complex things will remain to be
  complex
- Logical systems are in general complex
  (not decidable or axiomatisable, derivation complexity, ...)

Why
Constraints
Visuality
BPMN
Model suite
Triggering
Identity
Concepts
Open
Publications

Content

Information
Systems
Theory
5.7.2012
B. Thalheim
The Simplicity of Graphical Reasoning

Darwen FD rule

\[ X \rightarrow Y_0 Y_1, \ Y_1 Y_2 \rightarrow W \]

\[ XY_2 \rightarrow Y_0 Y_1 W \]
Graphical FD Reasoning

\[ (S) \quad \frac{Y \to B}{YC \to B} \]

\[ (Q) \quad \frac{Y \to A, Y \leftrightarrow B}{YA \leftrightarrow B} \]

\[ (R) \quad \frac{YA \to B, Y \leftrightarrow B}{Y \leftrightarrow A} \]

\[ (T) \quad \frac{Y \to A, YA \to B}{Y \to B} \]

\[ (P) \quad \frac{YC \leftrightarrow B}{Y \leftrightarrow B} \]

\[ (\square) \quad \neg(Y \to B, Y \leftrightarrow B) \]

Resulting rules for graphical reasoning
The Simplicity of Graphical Reasoning

\[ U_R = \{ A, B, D, F, G, I \} \]
\[ \sigma_R = \{ A \rightarrow IG, D \rightarrow FG, IAB \rightarrow D, IF \rightarrow AG \} \]

Classical synthesis algorithms:

\[ R_1 = (\{ A, G, I \}, \{ A \rightarrow GI \}) \]
\[ R_2 = (\{ A, F, I \}, \{ A \rightarrow I, FI \rightarrow A \}) \]
\[ R_3 = (\{ A, B, D \}, \{ AB \rightarrow D \}) \]
\[ R_4 = (\{ D, F, G \}, \{ D \rightarrow FG \}) \]

This normalisation not minimal
Instead of \( R_1 \) take \( R'_1 = (\{ A, G \}, \{ A \rightarrow G \}) \).

\( R_2 \) is not in BCNF. It cannot be split into two relation schemata.
The Simplicity of Graphical Reasoning

Easy exploration

Is D in any minimal key?
What are the minimal keys?
AB, BC, BE, CF?
Is Date correct?

Difficult but possible

What are alternative reductions?
Which dependencies are central?
Another FD Example

Why
Constraints
Visuality
BPMN
Model suite
Triggering
Identity
Concepts
Open
Publications

Information
Systems
Theory
5.7.2012
B. Thalheim
Another FD Example
Another FD Example

A directly reduced set

Another irreducible set

Derivation of \( \{A, C, D\} \rightarrow \{B\} \) as two step procedure
Graphical FD !!Set!! Reasoning

|       | |Y|=1 | |Y|=2 |
|-------|---------|---------|
| (S)   | ![Diagram](image1) | ![Diagram](image2) |
| (T)   | ![Diagram](image3) | ![Diagram](image4) |

**Why Constraints**

**Visuality**

**BPMN**

Model suite

Triggering

Identity

Concepts

Open

Publications

---

Information Systems Theory 5.7.2012 B. Thalheim
Additional cardinalities:

\[
\text{card}(\text{Flies}[\text{Gate, Time, Flight}], (\text{Gate, Time})) = [0,1] \\
\text{card}(\text{Flies}[\text{Gate, Time, Flight}], (\text{Flight, Time})) = [0,1] \\
\text{potentially card}(\text{Flies}[\text{Time, Flight}], (\text{Flight})) = [0,1]
\]

As FD’s:

\[
\{ \text{Flight Time} \rightarrow \text{Pilot}, \text{Time Pilot} \rightarrow \text{Flight}, \text{Flight} \rightarrow \text{Time}, \\
\text{Gate Time} \rightarrow \text{Flight}, \text{Flight Time} \rightarrow \text{Gate} \} \models ?
\]

Is this set complete or do we need other constraints?
The Triangular Representation

Trivial or redundant FD’s not represented, eg.:

- \( \text{Flight} \rightarrow \text{Flight}, \text{Flight} \rightarrow \text{Flight Time} \)
- \( \text{Flight} \rightarrow \text{Time Pilot} \iff \{ \text{Flight} \rightarrow \text{Time}, \text{Flight} \rightarrow \text{Pilot} \} \)

Example ternary case:

\[
\begin{array}{c}
\text{Flight} \\
\text{Pilot} \\
\text{Time}
\end{array}
\]

Initial set of FD’s: filled circles

- \( \text{Flight Time} \rightarrow \text{Pilot} \) 2D
- \( \text{Time Pilot} \rightarrow \text{Flight} \) 2D
- \( \text{Flight} \rightarrow \text{Time} \) 1D

There is a sound and complete axiomatisation!
Graphical Reasoning for Ternary Cases

Sound and complete implication system: \((A \neq B; A, B \notin Y)\)

(S) Extension: \(\frac{Y \to A}{YB \to A}\)  
\(Y \to A\) is extended by \(B\)

(T) Rotation: \(\frac{Y \to A, YA \to B}{Y \to B}\)  
\(Y \to A\) is rotated around \(Y\)  
\(\Rightarrow\) supported by \(YA \to B\)

Example ternary case:

Implied set of FD’s: empty circles

\(\vdash_{(S)} Flight\ Pilot \to Time\)

\(\vdash_{(T)} Flight \to Pilot\)
**Incorporating Negated Dependencies**

Sound and complete implication system: \((S), (T) + (P)\)

\[
\begin{align*}
\text{(P)} & \quad \frac{YB \rightarrow A}{Y \rightarrow A} \\
\text{(Q)} & \quad \frac{Y \rightarrow B, Y ightarrow A}{YA \rightarrow B} \\
\text{(R)} & \quad \frac{Y \rightarrow B, YA \rightarrow B}{Y \rightarrow A}
\end{align*}
\]

Reduction | Extension | Rotation

Example ternary case:

- **Flight** \(\rightarrow\) **Time** \(\rightarrow\) **Pilot**
- **Time Pilot** \(\rightarrow\) **Flight**
- **Flight** \(\rightarrow\) **Time**
- \(\text{×} \quad \text{Time} \leftrightarrow \text{Pilot}\)
- \(\vdash (S) \circ \text{Flight Pilot} \rightarrow \text{Time}\)
- \(\vdash (T) \circ \text{Flight} \rightarrow \text{Pilot}\)
- \(\vdash (R) \times \text{Time} \leftrightarrow \text{Flight}\)
Graphical Reasoning for Ternary Cases

Example ternary case:

The closed set:

- Flight Time \rightarrow Pilot
- Time Pilot \rightarrow Flight
- Flight \rightarrow Time
- Flight Pilot \rightarrow Time
- Flight \rightarrow Pilot

The result is one of the 14 different ternary cases (schema / relationship types).

What about raising the number of attributes?
The Tetrahedral Representation

A quaternary case has \(\binom{4}{3} = 4\) nested ternary cases forming the surface of a tetrahedron.

Edges correspond to the \(\binom{4}{2} = 6\) nested binary cases, each shared between two triangles.

The initial set:

- \(\text{Flight Time} \rightarrow \text{Pilot}\)
- \(\text{Time Pilot} \rightarrow \text{Flight}\)
- \(\text{Flight} \rightarrow \text{Time}\)
- \(\text{Gate Time} \rightarrow \text{Flight}\)
- \(\text{Flight Time} \rightarrow \text{Gate}\)
The Quadratic Representation

Ternary case - triangle 2D
Quaternary case - square 2D

Example complete case:

The initial set:

- Flight Time → Pilot
- Time Pilot → Flight
- Flight → Time
- Gate Time → Flight
- Flight Time → Gate

Key features: 'points towards', 'parallel'
Graphical Reasoning by the ST Algorithm

Example (full):

FD’s with minimal left sides:
- Flight $\rightarrow$ Time
- Flight $\rightarrow$ Pilot
- Time Gate $\rightarrow$ Pilot
- Flight $\rightarrow$ Gate
- Time Pilot $\rightarrow$ Gate
- Time Pilot $\rightarrow$ Flight
- Gate Time $\rightarrow$ Flight

The result is one of the 165 different quaternary cases (schema / relationship types).
The Pentagonal Representation

A quinary case has \( \binom{5}{4} = 4 \) nested quaternary, \( \binom{5}{3} = 10 \) nested ternary and \( \binom{5}{2} = 10 \) nested binary cases.

Derivation by extension:

- \( C \rightarrow A \)
- \( BC \rightarrow A \)
- \( CD \rightarrow A \)
- \( CE \rightarrow A \)
- \( BCD \rightarrow A \)
- \( CDE \rightarrow A \)
- \( BCE \rightarrow A \)
- \( BCDE \rightarrow A \)

Key features: 'points towards', 'parallel'
### Feasibility

<table>
<thead>
<tr>
<th>#Attrs</th>
<th>#Sets</th>
<th>#Cases</th>
<th>#Sets*</th>
<th>#Cases*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>45</td>
<td>14</td>
<td>61</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>2271</td>
<td>165</td>
<td>2480</td>
<td>184</td>
</tr>
<tr>
<td>5</td>
<td>1373701</td>
<td>14480</td>
<td>1385552</td>
<td>14664</td>
</tr>
<tr>
<td>6</td>
<td>75965474236</td>
<td>?</td>
<td>75973751474</td>
<td>?</td>
</tr>
</tbody>
</table>

- Two sets belong to the same case (relationship/schema type) if they differ only in the order of attributes.
- * means zero-dimensional FD’s are allowed (eg. $\emptyset \rightarrow \text{Pilot}$).
Foundations of BPMN Semantics

Operational semantics by reduction rules for execution of the language
opportunity: constructing an interpreter

Axiomatic semantics by correctness assertions that describe how
to draw conclusions about the input/output interface of a program
opportunity: verifying a program

Denotational semantics by a valuation function that maps a pro-
gram into a mathematical object which is considered as its meaning
opportunity: reasoning about its properties

Transformational semantics based on mappings and semantics of the target machine
requires thorough knowledge
- on mappings, their dependencies and interactions, and
- on target machine
Yet an Example of a Workflow Diagram

A BPMN diagram taken from literature with many problems
Why not Petri Nets?
Why not Petri Nets?

BPMN gateway
Why not Petri Nets?
The Pisa-Kiel BPMN-ASM Project

ASM semantics: extensible semantical framework for business process modelling notations

Guiding understanding of BPMN constructs based on ASM

Proposals for

- improvement of BPMN
- rigid semantics of all BPMN concepts
- treatment of all underspecified, overspecified, ... concepts

Systematic framework based on separate behavior from scheduling issues

Description of behavior directly in business process terms

Case study for complex business processes

  e.g., conference paper submission and reviewing system, conference management

  (muComs ∪ BYU ⊇ EasyChair ∪ ....)
Specifics of the ASM Approach

State model based on static and dynamic functions

All rules fire in parallel if they are enabled parallel and continuous execution

Conflicting updates are resolved by disabling some of rules that can be fired

Locations as an abstraction of storage

Turing/Church hypothesis with scaling abstraction
Fireable workflow transition

at node for its execution:

\[
\text{WorkflowTransitionInterpreter} = \\
\text{let } \text{node} = \text{selectNode}(\{ n \mid n \in \text{Node} \text{ and Enabled}(n) \}) \\
\text{let } \text{rule} = \text{selectWorkflowTransition}(\{ r \mid \\
\text{Fireable}(r, \text{node}) \}) \\
\text{rule}
\]

Separation of behaviour from scheduling

Execution of all fireable instance nodes

\[
\text{WorkflowTransition}(\text{node}) = \\
\text{if EventCond}(\text{node}) \text{ and CtlCond}(\text{node}) \\
\text{and DataCond}(\text{node}) \text{ and ResourceCond}(\text{node}) \text{ then} \\
\text{DataOp}(\text{node}) \\
\text{CtlOp}(\text{node}) \\
\text{EventOp}(\text{node}) \\
\text{ResourceOp}(\text{node})
\]
AND-Split Gateway (Fork)

If \((Data|Event)\text{Cond}\)

\[
\text{ANDSplitGateTransition}(node) = \text{WorkflowTransition}(node)
\]

where

\[
\text{CtlCond}(node) = \text{Enabled}(in)
\]

\[
\text{CTLOP}(node) = \\begin{align*}
\text{let } t &= \text{firingToken}(in) \\
\text{CONSUME}(t, in) \\
\text{PRODUCEALL}\{(\text{andSplitToken}(t, o), o) \mid o \in \text{outArc}(node)\}
\end{align*}
\]

\[
\text{DATAOP}(node) = \text{//performed for each selected gate}
\]

\[
\text{forall } o \in \text{outArc}(node) \quad \text{forall } i \in \text{assignments}(o) \quad \text{ASSIGN}(to_i, from_i)
\]

Separation of rule schemes from concrete rules
**AND-Join Gateway**

\[
\text{If } (\text{Data|Event})\text{Cond}
\]

\[
\begin{array}{c}
\vdots \\
\text{in}_i : t \\
\text{CtlCond}_i \\
\vdots \\
\text{out} : t \\
\text{OutCond}
\end{array}
\]

\[
\text{ANDJOINGATETRANSITION}(\text{node}) = \text{WORKFLOWSYNTONIZATION}(\text{node})
\]

where

\[
\text{CtlCond}(\text{node}) = \text{forall } \text{in} \in \text{inArc}(\text{node}) \text{ Enabled}(\text{in})
\]

\[
\text{CTLOP}(\text{node}) =
\]

\[
\text{let } [\text{in}_1, \ldots, \text{in}_n] = \text{inArc}(\text{node})
\]

\[
\text{let } [\text{t}_1, \ldots, \text{t}_n] = \text{firingToken}(\text{inArc}(\text{node}))
\]

\[
\text{CONSUMEALL}((\text{t}_j, \text{in}_j)) \mid 1 \leq j \leq n
\]

\[
\text{PRODUCE}(\text{andJoinToken} \{\text{t}_1, \ldots, \text{t}_n\}, \text{out})
\]

\[
\text{DATAOP}(\text{node}) = \text{forall } i \in \text{assignments}(\text{out}) \text{ ASSIGN}(\text{to}_i, \text{from}_i)
\]

**Be careful: BPMN assumes strict local consideration!**
The Token Problem in the BPMN Standard

For each uncontrolled Sequence Flow a “Token” will flow from the source object to the target object. To facilitate this discussion, we will employ the concept of a “Token” that will traverse the Sequence Flow and pass through the Flow Objects in the Process. The behavior of the Process can be described by tracking the path(s) of the Token through the Process. A Token will have a unique identity, called a TokenId set, that can be used to distinguish multiple Tokens that may exist because of concurrent Process instances or the dividing of the Token for parallel processing within a single Process instance. The parallel dividing of a Token creates a lower level of the TokenId set. The set of all levels of TokenId will identify a Token. A Start Event generates a Token that must eventually be consumed at an End Event (which may be implicit if not graphically displayed). The path of Tokens should be traceable through the network of Sequence Flow, Gateways, and activities within a Process.

If an upstream Inclusive OR produces two out of a possible three Tokens, then a downstream Inclusive OR will synchronize those two Tokens and not wait for another Token, even though there are three incoming Sequence Flow (see Figure 9.25).

Email from a colleague: The most interesting and, in my opinion, advanced paper on that is the paper of Frank Pullman and Matthias Weske from BPM’2005 where the authors used lambda calculi to derive workflow pattern semantics. The only problem they had then was related to OR-JOIN operator (gateway) while it was not possible to express memory of the generated tokens.

Comparing to what you propose I see that Pullman and Weske had problem with defining semantics of tokens and this was crucial for the overall workflow patterns semantics. Therefore I would expect that you in a clear and unambiguous way define semantics of tokens and that, in my opinion, would be a real step forward in defining semantics of BPMN.
The Token Problem: Acyclic Case

The BPMN Standard 1.0: All the Tokens that were generated within the Process must be consumed by an End Event before the Process has been completed.

The Process will be in a *running* state until all Tokens are consumed.

When the Inclusive Gateway is used as a Merge, it will wait for (synchronize) all Tokens that have been produced upstream. It does not require that all incoming Sequence Flow produce a Token (as the Parallel Gateway does). It requires that all Sequence Flow that were actually produced by an upstream (by an Inclusive OR situation, for example). If an upstream Inclusive OR produces two out of a possible three Tokens, then a downstream Inclusive OR will synchronize those two Tokens and not wait for another Token, even though there are three incoming Sequence Flow (see Figure 9.25).
The Token Problem: Cyclic Case

Be careful: Underspecified, confusing and difficult

The BPMN Standard 1.0: Incoming Sequence Flow that have a source that is a downstream activity (that is, is part of a loop) will be treated differently than those that have an upstream source.
Our solution

Controller structures based on the concept of the program counter

- `new(token)` of process instance, derivable owner
- token at location of control flow, i.e. arc
- multiset of token at location
- `returnProcess(token)`
- frames of execution handling:
  - (ASM BPMN) rule
    - consumed token,
    - produced token,
    - conditions applicable,
    - methods applied,
    - locals
  - invoke or scheduler frames
- supporting infrastructure for retransmission of potential enactment
Alternative Token Model

Each token has complete knowledge on its siblings and parents.

Splits generate new children.

Joins return parent of complete join otherwise more complex model.

Special model for structures of splits and joins that do not have an equivalent Fitch structure.
Control Flow Models

- Implicit (defined by the language) or explicit (e.g. goto) control: based on structuring (and the corresponding compositional semantics), exchange interfaces (statements, messages, data, events) and subroutines
- Control of subroutines: call/return structures, recursive?, call rules, completion of subroutines, eager/lazy execution, transfer of control, uniqueness of control, current instruction and environment pointer (CIP, CEP); implicit data control by indirect transfer, binding, naming, association and reference environments, aliases, sharing conceptions; static/dynamic visibility and lifespan conceptions; block structures ... TA’s;
- Control of collaboration cases: only by orchestration
control flow on parallelism: control dependence analysis, executing multiple flows of control simultaneously, and speculative execution
Normal Forms

- Splits only at the gates
- Entry and leave for a process only at start and end events
- Activities can be separated into
  \[ \text{Activity} = \text{Task} \cup \text{SubProcess} \cup \text{IterProc} \]
  \[ \text{IterProc} = \text{Loop} \cup \text{MultiInstance} \cup \text{AdHoc} \]
- Standard looping only
- Tracking of token
  - \( \bigcirc \) catch environment for local control token
- Prenex normal form
  - also for ad-hoc processes
- Communication normal form: explicit communication only by links and messages
- Exception normal form (?)
**ASM BPMN Assumptions**

- Token are assigned to arcs in the process diagrams

\[
\text{token} : \text{Arc} \rightarrow \text{Multiset}(\text{Token})
\]

- Token are identifiable

- Execution of workflow steps is reflected at nodes

\[
\text{Enabled}(\text{in}, t) = (| \text{token}(\text{in}, t) | \geq \text{qty}(\text{in}, t))
\]

- Token enabled execution

- Support and auxiliary functions

\[
\text{CONSUME}(t, \text{in}) = \text{DELETE}(t, \text{inQty}(\text{in}), \text{token}(\text{in}))
\]
\[
\text{PRODUCE}(t, \text{out}) = \text{INSERT}(t, \text{outQty}(\text{out}), \text{token}(\text{out}))
\]
\[
\text{CONSUMEAll}(X) = \text{forall} \ x \in X \ \text{CONSUME}(x)
\]
\[
\text{PRODUCEAll}(Y) = \text{forall} \ y \in Y \ \text{PRODUCE}(y)
\]

- Context-sensitive dynamic functions
**ASM Rules**

\[
\text{PRODUCESYNC}(t, in) = \text{INSERT}(t, \text{syncToken}(in))
\]

\[
\text{CONSUMESYNC}(t, in) = \text{DELETE}(t, \text{syncToken}(in))
\]

\[
\text{PRODUCESYNCALL}(Y) = \text{forall } y \in Y \text{ PRODUCESYNC}(y)
\]

\[
\text{CONSUMESYNCALL}(X) = \text{forall } x \in X \text{ CONSUMESYNC}(x)
\]

**Split gate transition rules**

\[
\text{PRODUCESYNCALL}((t.o, i) \mid i \in \text{AllJoinArc}(o), o \in O)
\]

\[
\text{CONSUMESYNCALL}((t, i) \mid i \in \text{AllJoinArc}(o) \text{ forsome } o \in O)
\]

**Join gate transition rules**

\[
\text{PRODUCESYNCALL}((\text{joinToken}(t_1, \ldots, t_n), in) \mid in \in \text{AllJoinArc(out)})
\]

\[
\text{CONSUMESYNCALL}((t_i, in) \mid in \in \text{AllJoinArc(out)})
\]

**Synchronization counterpart**

\[
\text{CtlCondSync}(node, I) = \\
\text{forall } i \in I \text{ syncToken}(i) \neq \emptyset \text{ and } \\
\text{forall } i \in \text{inArc(node)} \setminus I \text{ syncToken}(i) = \emptyset
\]
Separation of Concern

“The Concern Space of BPMN: Control space, abstraction, dimensions, user"

“Concern space” of systems: what concerns are present, how they relate, and how they can be used for modularization.

- Encapsulation of all kinds of concerns in a software system, simultaneously.
- Overlapping and interacting concerns.
- On-demand remodularization.
- Multiple, arbitrary kinds (dimensions) of concerns.
- Separation according to these concerns simultaneously.
- Overlapping or interacting concerns.
- Connection of concerns.

http://researchweb.watson.ibm.com/hyperspace/ConcernSpaces.htm
The Concern Space of BPMN

Separation of behavior from scheduling based on explicit schedulers

Separation of orthogonal constructs beyond what BPMN envisioned

\[
\text{TaskType} = \{\text{Service, User, Receive, Send, Script, Manual, Reference, None}\}
\]

Separation of different model dimensions like control, events, data and resources

Separation of design, experimental validation and mathematical verification

see coming demo
The Concern Space of BPMN

Separation of rule schemes and concrete rules e.g. by generalising rule schemes to generic ones or patterns

\[
\text{GateTransitionPattern}(\text{node}) =
\]
\[
\text{let } I = \text{selectConsume}(\text{node})
\]
\[
\text{let } O = \text{selectProduce}(\text{node}) \text{ in }
\]
\[
\text{WorkflowTransition}(\text{node}, I, O)
\]

where

\[
\text{CtlCond}(\text{node}, I) = (I \neq \emptyset \text{ and } \forall \text{ in } \in I \text{ Enabled}(\text{in}))
\]

\[
\text{CtlOp}(\text{node}, I, O) =
\]
\[
\text{ProduceAll}(\{(\text{patternToken}(\text{firingToken}(I), o), o) | o \in O\})
\]
\[
\text{ConsumeAll}(\{(t_i, \text{in}_{i}) | 1 \leq i \leq n\}) \text{ where }
\]
\[
[t_1, \ldots, t_n] = \text{firingToken}(I)
\]
\[
[\text{in}_1, \ldots, \text{in}_n] = I
\]
\[
\text{AssignOp}(\text{node}, O) = \forall o \in O \forall a \in \text{assignments}(o) \text{ Assign}(t_o, from_a)
\]

Separation of responsibilities, rights and roles of users
Illustration of the Separation Principles: OR-Gateways

Separation of send-collect for OR-gates as either global conception or (better) local subscribing OR-joins of runtime behaviour of publishing OR-splits

Clustering of cycles as a good practice

Tokenisation of cycle counters through Tokensets as a facility to model associated token separation of pathes according to their effect

Completion of semantics for meaning of synchronisation, cycles as a graph extension with specific stratification or encapsulation structures
Results for Token Problems

![Diagram of BPMN model](image)

- **Why Constraints**
- **Visuality**
- **BPMN**
- Model suite
- Triggering
- Identity
- Concepts
- Open
- Publications
Fitch Structures for Acyclic Diagram

Separation of send-collect for OR-gates as either global conception or (better) local subscribing OR-joins of runtime behaviour of publishing OR-splits
The ‘vicious cycle’ example: explicit treatment of cycles, tokensets as an association tool, encapsulation of cycle paths, separation of send-collect synchronisation from cycle synchronisation.
The Token Problem: Cyclic Case

The BPMN Standard 1.0: Incoming Sequence Flow that have a source that is a downstream activity (that is, is part of a loop) will be treated differently than those that have an upstream source.
The BPMN Standard 1.0: Incoming Sequence Flow that have a source that is a downstream activity (that is, is part of a loop) will be treated differently than those that have an upstream source.
Lessons for Cyclic Diagrams

Level of support depends on diagram

Acyclic diagrams without OR-splits and joins: local execution with local rules

Acyclic diagrams with OR-splits and joins: local execution with send-collect information enrichment at join nodes

Cyclic diagrams with XOR-entry/leave: separation of inside cycle flow and outside cycle flow

Stratified cyclic diagrams with OR/AND-entry/leave: explicit extension of diagrams and completion of underspecified semantics

Non-stratified cyclic diagrams led to many problems, e.g., deadlocks
Orthogonal Concerns
Co-design, coexistence and co-evolution of specifications

Database support: computation in collaboration, shared resources, modification-in-place ∨ -in-private

Transactional systems: short-term, long-term, collaboration transactions

Security: integrity, availability, confidentiality also privacy

History: runtime, log, revival (recovery, restart, session),

Context: concurrency, infrastructure, architecture,

QoS, SLA: quality of use, external/internal quality

Collaboration: communication, cooperation, coordination or simply orchestration, syndication

Actors, users: shared usage beyond message exchange
Model Evolution as the Real Challenge

already evolution of one model is a challenge

models change over time

co-evolution becomes a nightmare if not planned in advance
dynamic adaptation is another challenge
Variety of UML-“languages”

Main UML Diagrams

Structure Diagram

Class Diagram  Object Diagram  Package Diagram  Deployment Diagram  Component Diagram  Composite Structure Diagram

Behavioral Diagrams

Use Case Diagram  Interaction Diagrams  Activity Diagram  State Machine Diagram

Interaction Diagrams

Sequence Diagram  Communication Diagram  Timing Diagram  Interaction Overview Diagram
Coherence of UML Diagram Clusters

Inherent integrity constraints in diagrams

\[
\text{StateChart} \left( \{ \text{IsBorrowed, IsReturned} \} \right) \subseteq \text{ClassDiagram} \left( \pi_{\text{LendingState}}(\text{Book}) \right)
\]

\[
\text{EC}_{1}^{\text{states}(\text{SC},CT)}: \text{StateChart} \left( \text{States} \right) \subseteq \text{ClassDiagram} \left( \pi_{\text{X}}(\text{RulingClass}) \right)
\]

\[
\text{State'} \not\in \text{ClassDiagram} \left( \pi_{\text{X}}(\text{RulingClass}) \right) \longrightarrow F \ \text{modify} \left( \text{StateChart} \left( \text{State, State'} \right) \right)
\]

\[
O \ \text{cascade} \left( \text{modify} \left( \text{ClassDiagram} \left( \text{RulingClass, X} \right) \right), \text{modify} \left( \text{StateChart} \left( \text{State} \right) \right) \right)
\]

\[
do \left( \text{Agent}_1, \text{modify} \left( \text{ClassDiagram} \left( \text{RulingClass, X} \right) \right) \right) \circlearrowleft \quad \text{do} \left( \text{notify} \left( \text{Agent}_2, \text{modify} \left( \text{ClassDiagram} \left( \text{RulingClass, X} \right) \right) \right) \right)
\]
Challenges in Multi-Model Environments

Explicit specification of model collaboration: interdependencies among models

Integrated development of different models: different views of the same problem or application

Co-evolution of models: exchange between models and change propagation

Combining different (e.g., graphical) representations with mathematical rigor of models

Evolution of different representations: refinements of previous models or explicit revisions of models

Management of multi-model IS development: scheduling mechanisms, rollback

Version handling for multi-model IS development: different versions

Explicit refinement and abstraction treatment: systems development abstraction layers
Model Suites

as a part of ISE@CAU model integration theory

Model structure based on model constructors starting from model kernels and model orchestration and model choreographies

- constraints and structural soundness
- constraint enforcement

Model repository for coexistence of models based on the collaboration pattern and style

- model communication generalising model protocols
- model coordination generalising model contracts
- model cooperation generalising model evolution

Model metadata as the basis for model quality management

Model evolution
Model Suite: Constituents

- set of models \( \{M_1, \ldots, M_n\} \),
- association or collaboration schema among the models,
- controllers that maintain consistency or coherence of the model suite,
- application schemata for explicit maintenance and evolution of the model suite, and
- tracers for the establishment of the coherence.

Coherence describes a fixed relationship between the models in a model suite.

Only inductive languages with compositionality principle

Concentration on discrete domains
Model Suite: Languages

Model language $\mathcal{L}$: signature $\mathcal{S}$ and a set of constructors $\mathcal{C}$

$\Sigma_{S,C}$ well-formedness conditions

Model type $\mathcal{T}_{\mathcal{S}} = (\mathcal{L}_S, \Sigma_{\mathcal{L}_S})$

language of the model and

constraints $\Sigma_{\mathcal{L}_S} \in \mathcal{L}(\Sigma_{\text{WellFormed}})$

Partial mappings $\mathbb{R}_{i,j} : \mathcal{L}_{S_i} \to \mathcal{L}_{S_j}$ among $\mathcal{L}_{S_1}, \ldots, \mathcal{L}_{S_n}$

Model $\mathcal{M}$: $\text{struct}_\mathcal{M}$ in $\mathcal{L}_S$

that obeys $\Sigma_{\mathcal{L}_S}$,

and set of constraints $\Sigma_{\mathcal{M}}$ defined in the logics of this language.
Model Suite: Model Association and Contracting

Collaboration contract among models

Collaboration

- **Communication** is used in a variety of facets as an act or instance of transmitting or a process by which information is exchanged between models through a common system.

- **Coordination** expresses the act or action of coordinating the harmonious functioning of models for effective results.

- **Cooperation** expresses the action of cooperating.

Collaboration style: supporting programs, data access pattern, style of collaboration, coordination workflows

Collaboration pattern: supporting *access and configuration, event processing, synchronization, and parallel execution*
Model Suite

Model suite type $ST = (T_{L_{S_1}}, ..., T_{L_{S_n}}, \Sigma_{L_{S_1}}, ..., L_{S_n})$

model types $T_{L_{S_1}} ... T_{L_{S_n}}$ defined on a set $\Sigma_{S_1}, ..., S_n$ of $L_{S_1}, ..., L_{S_n}$ constraints

Model suite $\mathcal{S}$
on a model suite type $ST$
models $(M_1, ..., M_n)$ of type $T_{L_{S_1}} ... T_{L_{S_n}}$
that obey $\Sigma_{L_{S_1}}, ..., L_{S_n}$

Contract on $\mathcal{C}$:

- constraints $\Sigma_{L_{S_1}} \cup ... \cup \Sigma_{L_{S_n}} \cup \Sigma_{L_{S_1}}, ..., L_{S_n}$,
- description of the enforcement mechanisms for any operation that can be used for modification of one model, and
- set of consistent evolution transformations.
Model Suite: Synchronisation and Coherence

Commuting diagrams and co-evolution

\[ M_i \xrightarrow{\text{extract } e_{i,j}} M_{i,j,\text{outp}} \xrightarrow{\text{transform } t_{i,j}} M_{j,i,\text{inp}} \xrightarrow{\text{load } l_{i,j}} M_j \]

\[ M_i \xrightarrow{\text{put}_{i,j}^*} M_j \]
\[ M'_i \xrightarrow{\text{put}_{j,i}^*} M'_j \]

\[ M_i \xrightarrow{\text{change}_i} M_i \]
\[ M_j \xrightarrow{\text{change}_j} M_j \]

\[ M'_i \xrightarrow{\text{put}_{j,i}^*} M'_j \]
\[ M'_i \xrightarrow{\text{put}_{j,i}^*} M'_j \]
Model Suite: Co-evolution

Information system model $\mathcal{M}_{IS}$

- Database structure model $\mathcal{M}_{struct}$
  - Database schema
  - Static integrity constraints

- Database function model $\mathcal{M}_{func}$
  - Database functionality model
  - Dynamic integrity constraints

- Database support models
  - Content model $\mathcal{M}_{content}$
  - Interaction model $\mathcal{M}_{interact}$
  - ... model

Changes and updates:
- $\mathcal{M}_{struct} \xrightarrow{\text{put}} \mathcal{M}_{content}$
- $\mathcal{M}_{content} \xrightarrow{\text{put}} \mathcal{M}_{struct}$
- $\mathcal{M}_{func} \xrightarrow{\text{change}} \mathcal{M}_{struct}$
- $\mathcal{M}_{struct} \xrightarrow{\text{put}} \mathcal{M}_{interact}$
- $\mathcal{M}_{content} \xrightarrow{\text{put}} \mathcal{M}_{interact}$

Open Publications

- Identity
- Concepts

Why
Constraints
Visuality
BPMN
Triggers: One of the Hereditary Diseases

Issues That Are Still Unsolved

Trigger defined through term rewriting systems

- confluence
- termination
- effect preservation

unfortunately only for strongly hierarchical systems (KDS/β).

Execution semantics for trigger set needed

Be careful: changes with DBMS

\[ C_1 \subseteq C_2, \quad C_1 \subseteq C_3, \quad C_2 \parallel C_3 \]

\[
\begin{align*}
insert_{C_1} & \rightarrow^* delete_{C_3} delete_{C_2} delete_{C_1} \\
insert_{C_1} & \rightarrow^* insert_{C_3} delete_{C_2} delete_{C_1} \\
insert_{C_1} & \rightarrow^* insert_{C_2} delete_{C_3} delete_{C_1}
\end{align*}
\]

Greatest consistent specialisation (KDS/β)
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:

- $i_A(x) \sim i_A(x) ; i_B(x)$, $i_A(x) \sim i_A(x) ; i_C(x)$,
- $d_B(x) \sim d_B(x) ; d_A(x)$, $d_C(x) \sim d_C(x) ; d_A(x)$
- $i_B(x) \sim i_B(x) ; d_C(x)$, $i_C(x) \sim i_C(x) ; d_B(x)$

general compression rules: $i_T(x); d_T(x) \sim d_T(x)$, $d_T(x); i_T(x) \sim i_T(x)$

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

- $i_C(x); d_B(x) ; d_A(x)$ or
- $i_B(x); d_C(x) ; d_A(x)$ or $d_B(x); d_C(x) ; d_A(x)$

Therefore: be careful during design of constraints and structures!

GCS result: FAIL
Can The Designer Repair ECA Faults?

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

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

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

\[ insert_{R_1}(a, c) \text{ AFTER INSERT } a \notin R_1[A] \]
\[ \text{AFTER INSERT } c \notin R_2[D] \]
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: \( \text{Ins}(db, set) \sqsubseteq db \)
- 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 (specialise) [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 specialisation allows to derive integrity enforcing operations for the generic database functions

reflection + predicate transformer ⇒ greatest consistent specialisation

state variable
state constraints

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

Linguistic reflection:
reflection types $SCHEMA_{rep}$, $CLASS_{rep}$, $TYPE_{rep}$, $METHOD_{rep}$, $COMMAND_{rep}$,

\[
\begin{align*}
\text{insert} : & \ S :: SCHEMA_{rep} \times C :: CLASS_{rep} \rightarrow METHOD_{rep} \\
\text{delete} : & \ S :: SCHEMA_{rep} \times C :: CLASS_{rep} \rightarrow METHOD_{rep} \\
\text{update} : & \ S :: SCHEMA_{rep} \times C :: CLASS_{rep} \rightarrow METHOD_{rep} .
\end{align*}
\]

one macro generic with signature $SCHEMA_{rep} \rightarrow SCHEMA_{rep}$
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 $\mathcal{R}$

\[
\begin{align*}
wp(S)(\mathcal{R}) & - \text{initial states with all terminating executions of } S \text{ reach final state} \\
\text{characterized by } \mathcal{R} \text{ (provided } S \text{ defined)}
\end{align*}
\]

\[
\begin{align*}
wlp(S)(\mathcal{R}) & - \text{initial states with all executions of } S \text{ terminate, reach a final state} \\
\text{characterized by } \mathcal{R} \text{ (provided } S \text{ defined)}
\end{align*}
\]
GCS Approach (2)

Y-operation $S$ on $X$  

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

- $S_I$ specialises $S$ ($S_I \subseteq S$),
- $S_I$ is consistent with respect to $I$,
- $S_I$ is maximal for each $X$-operation $T$ satisfying the properties: $T \subseteq S_I$

specialisation order $T \subseteq S$

$$wp(S)(true) \Rightarrow wp(T)(true) \quad \text{and}$$
$$wlp(S)(\mathcal{R}) \Rightarrow wlp(T)(\mathcal{R})$$

GCSs always exist

compatible with conjunctions - universal conjunctivity

subsumption freeness

usually non-deterministic

quasi-deterministic branches of a GCS

$maximal \ quasi$-deterministic consistent specialisations (MQCSs)

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

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

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

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

IF \( c \in R_2[D] \) THEN Fail
ELSE \( \text{insert}_{R_1}(a, c) \) IF \( a \notin R_1[A] \)
THEN \( \text{insert}_{R_2}(a, d) \) WHERE \( d \notin R_1[C] \cup R_2[D] \)
Identifiability, Identifier, Accessibility

Three kinds of identification statement

- **Identifiability** as the existence of a potential identification
- **Identification** as the declaration of a specific identification
- **Identifier** as the special implementation
- **Key access** as a special identifier

Diagram:

- **Language definition level**
  - **uniqueness**
  - **conceptual level**
    - **identification** (surrogate)
    - **integrity constraint** (key dependency)
  - **existence**
  - **external level**
    - **identification**
  - **internal level**
    - **identification**
    - **accessibility**
  - **invariance**
Equality Concepts

Functions of an object

Equality concepts

Object-equal

Shallow-equal

Referential-equal

Deep-equal

Similar

Content-equal

Functions in value-oriented ...

... in value-based databases with keys
Object identifier

- Value-representability is a must.
- Detecting cycles is a nightmare.
- Cyclic structuring leads to non-first-order logics. Structures with abstract linking are potentially cyclic.
- The OID is too expressive.
- The OID has already been implemented by relational databases: tuple identifier.

Object-oriented approaches have improved and damaged culture.
Next after OO: components
Varieties of representations

Implementation alternatives

Object-oriented and object-relational approaches: objects are decomposed into a set of related objects

XML-based approaches: classes are hidden

Storage alternatives

Class-separated snowflake representation: an object is stored in several classes
  storage engine
  object-relational viewpoint

Full-object representation: all data associated with the object are compiled into one object
  input engine, output engine
  XML viewpoint
Identification is not for free

values - identified by themselves
objects - context-dependent identification
1-1 association of real world things and objects
real world things have an identification
most general: history
this approach is not applicable: \[ \Rightarrow \] OID
objects identifiable
existence generic operations
support for consistency

Value representability

- Object identifiers do not have a meaning to the user.
- In general, a unique identification of an object is impossible if all values and references are equal.
- It is a design decision whether each object should be identifiable.

an object in \( D \) is identifiable iff its orbit (for the automorphism group) is trivial
Value Identifiability

\( C \) a class with representation type \( T_C \)

\[ C \text{ value-identifiable : } \quad \text{Unique}(C, I_C) \]

\[ C \text{ value-representable : } \quad \text{Unique}(C, V_C) \]

all objects in each instance \( D_S \) are identifiable iff all classes in \( S \) are weak value-representable

Value Representability

\( C \) value-representable : \( \text{Unique}(C, V_C) \)

\( \exists \) proper value type \( V_C \) \( \forall D \) of \( S \) \( \exists c : T_C \rightarrow V_C \)

(1) uniqueness constraint on \( C \) defined by \( c \) holds

(2) \( \forall \) uniqueness constraint on \( C \) defined by \( c' : T_C \rightarrow V'_C \) with value type \( V'_C \) exists a function \( c'' : V_C \rightarrow V'_C \) that is unique on \( c(\text{codom}(D(C))) \) with \( c' = c'' \circ c \)
Values versus Objects

Values:
7, “Hugo”, (Seminar 9434, Keycode 1535)

Objects:
Cicero - as author, historical person
Tully - as friend

⇒

values - identified by themselves
objects - context-dependent identification

1-1 association of real world things and objects
real world things have an identification
most general: history
this approach is not applicable:
⇒ OID
Values versus Objects

⇒

(1) OID invisible for users
(2) separation value/object
(3) objects have values
(4) operations (better events) are associated with objects
(5) requirements
   (a) objects identifiable
   (b) existence generic operations
   (c) support for consistency

general object:

- identifier
- ‘set’ of values
- ‘set’ of references
- ‘set’ of methods

no definition

formal approach:

define set of objects

schema definition, instance

primary notion
Distinguishability Problem

\[ \begin{align*}
O_1 \xrightarrow{s} O_2 \\
O_3 \xrightarrow{s} O_4 \\
O_5 \xrightarrow{s} O_6 \\
O_7 \xrightarrow{s} O_8
\end{align*} \]

\[ \begin{align*}
O_2 \xrightarrow{s'} 2 \\
O_4 \xrightarrow{s'} 1
\end{align*} \]

\[ \begin{align*}
O_1 \xrightarrow{s_1} O_2 \\
O_2 \xrightarrow{s_2} O_3 \\
O_3 \xrightarrow{s_3} O_4 \\
O_4 \xrightarrow{s_1} O_1
\end{align*} \]
Distinguishability Problem

\[ O_1 \xrightarrow{s'} O_3 \quad O_2 \xrightarrow{s''} 5 \]

\[ O_4 \xrightarrow{s'} O_2 \quad O_3 \xrightarrow{s''} 5 \]

\[ 5 \xrightarrow{s'} O_3 \quad O_3 \xrightarrow{s''} 5 \]

\[ 5 \xrightarrow{s'} O_2 \quad O_1 \xrightarrow{s} O_3 \]
Reference graphs

reference graph of $C$ in $S$

1. $\exists\ v_C \in V : l(v_C) = \{t, C\}$
   - $t$ - top-level type in the structure expression $S$ of $C$

2. for each proper occurrence of a type $t \neq ID$ in $T_C$ there exists a unique vertex $v_t \in V$ with $l(v_t) = \{t\}$

3. for each reference $r_i : C_i$ in $S$ the reference graph $G_{ref}^i$ is a subgraph of $G_{ref}$

4. for each vertex $v_t$ or $v_C$ corresponding to $t(x_1, \ldots, x_n)$ in $S$ there exist unique edges $e^{(i)}_{t}$ from $v_t$ or $v_C$ respectively to $v_{t_i}$ in case $x_i$ is the type $t_i$ or to $v_{C_i}$ in case $x_i$ is the reference $r_i : C_i$
   - in the first case $l(e^{(i)}_{t}) = \{S_i\}$ for the corresponding selector name $S_i$
   - in the latter case the label is $\{S_i, r_i\}$

$S = \{C_1, \ldots, C_n\}$, $S' = \{C'_1, \ldots, C'_n\}$ another schema such that for all $i$ there exists a uniqueness constraint on $C_i$ defined by some $c_i : T_{C_i} \rightarrow T_{C'_i}$

identification graph $G_{id}$ of the class $C_i$ is obtained from the reference graph of $C'_i$ by changing each label $C'_j$ to $C_j$

Algorithm

$F(C_i) = \begin{cases} T_i & \text{for uniqueness constraint } c_i : T_{C_i} \rightarrow T_i \\ \text{undefined} & \text{else} \end{cases}$

iterate as long as possible:

1. If $F(C_j)$ is a proper value type and $ID$ occurs in some $F(C_i)$ corresponding to a reference to $C_j$ ($i \neq j$), then replace this $ID$ in $F(C_i)$ by $F(C_j)$.
2. If $ID$ occurs in some $F(C_i)$, then let $F(C_i)$ be recursively defined by $F(C_i) == S_i$, where $S_i$ is the result of replacing $ID_i$ in $F(C_i)$ by the type name $F(C_i)$. 

smallest labelled graph $G_{rep} = (V, E, l)$

$G_{id}$ of the class $C_i$ is obtained from the reference graph of $C'_i$ by changing each label $C'_j$ to $C_j$
Acyclic and cyclic schemata

- If in the reference graph $G_{ref}$ there exist uniqueness constraints for $C$ and each $C_i$ such that $C_i$ occurs as a label in $G_{ref}$, then $C$ is value-representable.
- If there exist an identification graph $G_{id}$ and uniqueness constraints for $C$ and each $C_i$ occurring as a label in $G_{id}$, then $C$ is value-identifiable.
- Value-representability is decidable for acyclic graphs.
- If there exists an acyclic identification graph the value-identifiability is decidable.
- If all explicit constraints are uniqueness constraints then value-identifiability and value-representability are decidable.
Weak value-representability

considering all references to and from a class
for identification

weak value-representable classes

\[ C - \text{weak value-representable class in } S \]
\[ \Rightarrow \exists \text{ value type } V^w_C : \text{Unique}(C, V^w_C) \]

\[ S : \text{all objects in each instance } D_S \text{ are identifiable iff all classes in } S \]
are weak value-representable
Information Demand based on User Profiles and Portfolio

Information demand support depending on specifics
- Workplace and workspace support
- Consumed and produced data
- Context demand
- Formulation, language

based on characterisation of the user, user stories

Task/work portfolio support for the user
- Tasks: characteristics, execution, result, control
- Involvement: role, part
- Collaboration: communication, coordination, cooperation
- Restrictions: subtasks, environment

Personal profile support for the user
- Work profile
- Education profile
- Group profile
Information Versus Knowledge and Data

**What do we need? INFORMATION!**

Information as processed by humans, is *data* 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.

General knowledge as sustainable, evolving, potentially durable and verifiable data that is grounded on consensus

Skills as ability to do something well

T. S. Eliot (1888-1965), The rock, 1934:

Where is the wisdom we have lost in knowledge?
Where is the knowledge we have lost in information?

β nowadays:

Where is the information we have lost in news?
Where is the information we have lost in data?
Differentiating Dimensions

The four dimensions of the knowledge space surrounded by the context dimension:

1. data dimension through content;
2. foundation dimension through concepts;
3. language dimension through topics;
4. user dimension through information;
5. context of data (content) or theories (concept) or user (information) or carrier/language (topic)

![Knowledge Space Diagram]
Knowledge Chunk

Knowledge pieces cannot be considered in an isolated form. They are composed.

Knowledge chunk $\mathcal{C}$: a suite of knowledge pieces consisting of content, concepts, topics and information.

Content chunk $\mathcal{D} = \{D_1, ..., D_n\}$
- typically given as a set of media objects

Concept chunk $\mathcal{C} = \{C_1, ..., C_k\}$
- typically given as a small ‘theory’

Topic chunk $\mathcal{T} = \{T_1, ..., T_m\}$
- typically given as a map of associated topics

Associated by generalised mappings
- interpretation: $\mathcal{D} \rightarrow \mathcal{C}$ (opposite to foundation)
- explanation: $\mathcal{T} \rightarrow \mathcal{C}$ (opposite to presentation)
- annotation: $\mathcal{D} \rightarrow \mathcal{T}$ (opposite to content delivery)

Information chunk of a user
- for a given universe of contexts $\mathcal{I}_A$ of an agent $A$
- with corresponding associations (partial)
- to content $\mathcal{D}_A$, concepts $\mathcal{C}_A$, topics $\mathcal{T}_A$ of the user
- May also be extended by the agent, ... context.
Knowledge Chunk

Knowledge pieces cannot be considered in an isolated form. They are composed.

Knowledge chunk $\mathcal{C}$: a suite of knowledge pieces consisting of content, concepts, topics and information.

Content chunk $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$

typically given as a set of media objects

Concept chunk $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$

typically given as a small ‘theory’

Topic chunk $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$

typically given as a map of associated topics

Associated by generalised mappings

interpretation: $\mathcal{D} \rightarrow \mathcal{C}$ (opposite to foundation)

explanation: $\mathcal{T} \rightarrow \mathcal{C}$ (opposite to presentation)

annotation: $\mathcal{D} \rightarrow \mathcal{T}$ (opposite to content delivery)

Information chunk of a user

for a given universe of contexts $\mathcal{I}_A$ of an agent $A$

with corresponding associations (partial)

to content $\mathcal{D}_A$, concepts $\mathcal{C}_A$, topics $\mathcal{T}_A$ of the user

May also be extended by the agent, ... context.
A sequence of real (or complex) numbers is said to converge to a real (or complex) number $c$ if for every $\epsilon > 0$ there is an integer $n_\epsilon > 0$ such that if $j > n_\epsilon$ then $|a_j - c| < \epsilon$. The number $c$ is called the limit of the sequence and we sometimes write $a_j \to c$.

If a sequence does not converge, then we say that it diverges.

In other words, a sequence can be denoted by $f(1), f(2), f(3), \ldots$. Usually, we will denote such a sequence by the symbol $(a_j)_j$, where $a_j = f(j)$.

Sequences that converge to zero are called null sequences.

A sequence which converges.

A sequence of real numbers is a function $f: \mathbb{N} \to \mathbb{R}$.

Monotone decreasing/increasing sequences.

For example, the sequence $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \ldots$ is written as $(\frac{1}{n})_n$. Keep in mind that despite the strange notation, a sequence can be thought of as an ordinary function. However, in many cases, that may not be the most expedient way to look at the situation. It is often easier to simply look at a sequence as a ‘string’ of numbers that may or may not exhibit a certain pattern.

The limit of a sequence is one of the oldest concepts in mathematical analysis. It provides a rigorous definition of the idea of a sequence converging towards a point called the limit.

A convergent sequence is bounded and the limit is unique. Cauchy sequences ....

Do not confuse this with the idea of a series defined by the result of summarising infinitely many numbers $\sum_{j=1}^{\infty} a_j$. Despite the fact that the common non-mathematical meaning of “sequence” and “series” is identical there are separate definitions of convergence for sequences and series, and separate theories for these with some important differences that you need to be aware of. An infinite series is an expression of the form $\sum_{j=1}^{n} a_j$, where $(a_n)$ is a sequence.

Limit inferior, limit superior.
(C^+) Example of a Mathematical Concept
(Annotation as Topic, Data as Content)

⟨Definition:] A sequence of real (or complex) numbers is said to converge to a real (or complex) number c if for every \( \epsilon > 0 \) there is an integer \( n_\epsilon > 0 \) such that if \( j > n_\epsilon \) then \( |a_j - c| < \epsilon \). The number c is called the limit of the sequence and we sometimes write \( a_j \rightarrow c \).

If a sequence does not converge, then we say that it diverges.

In other words, a sequence can be denoted by \( f(1), f(2), f(3), \ldots \) Usually, we will denote such a sequence by the symbol \( (a_j)_j \), where \( a_j = f(j) \).

⟨Sub-concept:] Sequences that converge to zero are called null sequences.

⟨Annotation:] A sequence which converges.

⟨Prerequisite concept:] A sequence of real numbers is a function \( f : \mathbb{N} \rightarrow \mathbb{R} \).

⟨More specific prerequisites:] monotone de/increasing sequences

For example, the sequence 1, \( \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \ldots \) is written as \( (\frac{1}{n})_n \). Keep in mind that despite the strange notation, a sequence can be thought of as an ordinary function. However, in many cases, that may not be the most expedient way to look at the situation. It is often easier to simply look at a sequence as a ‘string’ of numbers that may or may not exhibit a certain pattern.

⟨Context:] The limit of a sequence is one of the oldest concepts in mathematical analysis. It provides a rigorous definition of the idea of a sequence converging towards a point called the limit.

⟨Related concepts:] A convergent sequence is bounded and the limit is unique. Cauchy sequences ....

Do not confuse this with the idea of a series defined by the result of summarising infinitely many numbers \( \sum_{j=1}^{\infty} a_j \). Despite the fact that the common non-mathematical meaning of “sequence” and “series” is identical there are separate definitions of convergence for sequences and series, and separate theories for these with some important differences that you need to be aware of. An infinite series is an expression of the form \( \sum_{j=1}^{n} a_j \), where \( (a_n) \) is a sequence.
Transaction as concurrent operation (Elmasri/Navathe): “The execution of a program that accesses or changes the contents of the database is called a transaction. The transaction submitted by various users may execute concurrently and may access and update the same database records. If this concurrency is uncontrolled, it may lead to problems such as an inconsistent database.”

Transactions must have four properties:

- Atomicity.
- Consistency.
- Isolation.
- Durability.
Syntactic transaction and their behavior

Syntax: transaction $T$ over $(\mathcal{S}, \Sigma)$ is a finite sequence $o_1; o_2; o_3; \ldots; o_m$ of basic modification operations (insert, delete, update) and retrieval operations $\text{map}(\text{filter}(\text{join}(\ldots), \psi), \mathcal{S})$ over $(\mathcal{S}, \Sigma)$.

Semantics of application: effect of application of $T$ to $\mathcal{S}^C$ is defined as a transition constraint preserving transformation

$$T(\mathcal{S}^C) = \begin{cases} T(\mathcal{S}^C) & \text{if } T(\mathcal{S}^C) \models \Sigma \\ \mathcal{S}^C & \text{if } T(\mathcal{S}^C) \not\models \Sigma \end{cases}$$

The effect defines the granularity of the transaction. TA’s in general understood as invariant state transitions invariant according to the set of static integrity constraints.
Potential parallel execution of TA’s

$T_1, T_2$ are competing
if $\text{read}(T_1) \cap \text{write}(T_2) \neq \emptyset$ or $\text{read}(T_2) \cap \text{write}(T_1) \neq \emptyset$

or $\text{write}(T_2) \cap \text{write}(T_1) \neq \emptyset$.

$\text{read}(T_i)$ resp. $\text{write}(T_i)$: locations of objects read or written by $T_i$

Parallel execution of transactions $T_1 \parallel T_2$ is correct
if either $T_1 \parallel T_2$ are not competing
or $(T_1 \parallel T_2)(S^C) \in \equiv \{ T_1(T_2(S^C)), T_2(T_1(S^C)) \}$ for any $S^C$

Observation: If parallel execution is correct for a set of transactions
transaction execution can be scheduled entirely in parallel.

Remarks: 0. We are concerned with the conceptual definition of TA’s and not
with their implementation techniques that must satisfy a number of properties
(with a proof of validity).

1. Classically, conflict serializability expresses potential parallel execution.

2. View serializability is definable in a similar way.

3. Testing of potential parallel execution is simple (conflict graph resolution)
as long as read, write are considered and rather complex if retrieval
and modification operations (insert, delete, update) are allowed.
Proposal for an Architecture of User-Oriented Content Management Systems

Towards this century CMS

Web Playout System
- Story Space
- Stories
- Scenarios
- Actors
- Context

User Management System
- Profile manager
- Portfolio manager
- Association generator / Natural language engine

Privacy Protection System

Content Management System
- Content types
  - Structure
  - Static IC
  - (Pragmatics)
- Service
- Functionality
- Container
- Structuring
- Structure
- Dynamic IC
- (Pragmatics)

Topic Management System
- Topic manager
- Community manager
- Asset manager / Infon representer
- Concept manager
- Derivation engine
- Unit manager / Infon representer

Information Systems Theory
5.7.2012 B. Thalheim

Why
Constraints
Visuality
BPMN
Model suite
Triggering
Identity
Concepts
Open
Publications

Content base
Data base
Concept base
Topic landscape
Private database
Publications on Database Theory

Publications on Genericity

Publications on Model Suites, Evolution, Migration


Publications on Business Process Modelling & Notation


Publications on Object Identification

Publications on Enforcement and GCS

Publications on Concepts, Content, Topics

Publications on Keys

Publications on Science and Art of Conceptual Modelling

Hungarian Publications


Editing: LNCS 305, 364, 495, 2582
Domain Requirements Modelling Framework for Cross-Disciplinary Service System Development

ICCNDT’12, Bahrain

Nov 11, 2012

(Sankha Amarakoon 1), (Ajantha Dahanayake 2) & Bernhard Thalheim 3

1 Rotterdam Ophthalmologic Institute, Rotterdam, the Netherlands
2 Prince Sultan University, Dept. of CIS, Riyadh, Kingdom of Saudi Arabia
3 Christian-Albrechts-University Kiel, Germany & Lomonosov University Moscow, Russia

smamarakoon@ROI.nl adahanayake@pscw.psu.edu.sa thalheim@is.informatik.uni-kiel.de
Overview
Cross-Discipl.
Domain Req.
Service
Appl. Dom.
Concluding

Content
Information
Concept	Topic

©B. Thalheim

i.e., reaching the big triple "3"s (3, 30, 300)
(3 (habilitation students, profs, books), 30 (PhD students, editorials, projects),
300 (master/diploma students, papers, presentations))
Application Example: Ophthalmology

**Application:** develop a system that supports similarity matching between patients' eye images and images available in a repository.

**Main Issue:** representation of an ophthalmologist's way of information processing during matching images to identify the disease.

**Research issue:** integration of computational, cognitive, and sociotechnical aspects.
Application Example: Intensive Breath Care by Evita 4

IPPV (Intermittent Positive Pressure Ventilation);
SIMV (Synchronized Intermittent Mandatory Ventilation);
MMV (Mandatory Minute Volume Ventilation);
SB (Spontaneous Breathing);
CPAP (Continuous Positive Airway Pressure);
ASB (Assisted Spontaneous Breathing);
BIPAP (Biphasic Positive Airway Pressure);
BIPAPAssist (Biphasic Positive Airway Pressure Assisted);
APRV (Airway Pressure Release Ventilation);
PPS Proportional Pressure Support;
ILV (Independent Lung Ventilation);
Automatic Tube Compensation ATC;
Apnoea Ventilation;
NIV mask ventilation; ... ... ...
Overview

Visions, results, projects

(1) Necessities of real applications
   provide a model that covers all aspects

(2) Application domain and requirements elicitation
   integrate the deployment and user perspective

(3) IT systems provide services
   Mathematics, engineering, application domains

(4) Service modelling
   including application domain

(5) Mapping services to Information systems
   providing a validatable and verifiable system specification
Cross-Disciplinary Approach to Development

(1) Task- and usage-driven development of supporting means for real-life applications
   
   computer engineers neglect user requirements

(2) Systems must follow rules, culture, understanding of users, e.g., privacy, security, dependability
   
   computer engineers are not educated to take this into account

(3) Systems become far more complex, handle huge data, must be integrated, used with skills
   
   holistic understanding of applications with flexible integration

(4) Strengths, weaknesses, opportunities, and threat of systems must be understood by users before used with full benefit
   
   users must not be adapted to the systems but should continue to live

(5) Systems are going to be used by many different people with many different background
   
   cultures matter, cross-disciplinary, cross-cultural
Requirements impacted from the Application Domain to Social-Technical Systems
Conceptual Model for IT Service System

- Fundamental elements for developing applications;
- Organising the discrete functions contained in (business) applications comprised of underlying business process or workflows into interoperable, (standards-based) services;
- Services abstracted from implementations representing natural fundamental building blocks that can synchronise the functional requirements and IT implementations perspective;
- Services to be combined, evolved and/or reused quickly to meet business needs;
- Represent an abstraction level independent of underlying technology.
Problems of IS Systems Modelling Techniques

(1) Limited scope: mainly specification of data structuring;

(2) Poor separation of concerns: (a) intended properties, (b) assumptions about the environment, (c) properties of the application domain;

(3) Low-level schematology: structured and formalised by modelling in the small;

(4) Isolation: context, companion products, people;

(5) Poor guidance: constructive methods for building correct models;

(6) Cost of holistic development: white-box use of tools;

(7) Poor tool feedback: root of problems, proposing better modelling solutions
Specification of the Problem Space

State space: collection of all those states reachable from the initial state; desirable states, i.e. are goal states; properties such as suitability

Actions: move from one state to another state under certain conditions; effect of the actions is observable; blocked or enabled depending on conditions used at some cost

Goal test: whether a given state or state set satisfies the goals

Problem solution controller: evaluates the actions; preferred; optima; based on evaluators beyond the general Polya frame

Problem: based on abstraction techniques

Context

Forces: Constructors, associations, collections, classification, operations, views, derivatives, ...

Solution
Service W*H-Specification Frame

classical rhetorical frame

Hermagoras of Temnos: Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis

Who, what, when, where, why, in what way, by what means

- $W^4$: wherefore (purpose), whereof (origin), wherewith (carrier, e.g., language), and worthiness ((surplus) value)

- secondary characterisation $W^{17}H$:
  - user or stakeholder or community of practice characteristics: by whom, to whom, whichever;
  - characteristics imposed by the application domain: wherein, where, for what, wherefrom, whence, what;
  - purpose characteristics characterising the solution: how, why, whereto, when, for which reason; and
  - additional context characteristics: whereat, whereabout, whither, when.
Service W*H-Specification Frame: Resulting Tasks

(1) $W^4$ (Purpose, Origin, Carrier, Value): critical review of application portfolio, pre-cut of application domain (necessary, potential, never), languages (environment), profit and motivation

(2) $W^{17}H$ (Community of practice): abilities, views, attention and personality, value of opinion and contribution, reliability, roles and responsibilities

(3) $W^{17}H$ (Specifics of application domain): expectations, quality, foci, variations, future and evolution

(4) $W^{17}H$ (Specifics of solution): programs, service after roll-out, quality and pricing, time and version management, coverage and capacity, integration, future migration

(5) $W^{17}H$ (Context): environment, infrastructure, temporal constraints, integration with other solutions

Application systems as socio-technical systems.
Abstraction Layer Models

Separation by Level of Detail

Application domain layer concerned with description of the application

Requirements acquisition layer concerned with prescription of system requirements

Business user layer concerned with behaviour or users, their demands to the system

Conceptual layer concerned with specifications (schemata) that describe the system

Implementation layer concerned with logical and physical (specifications and) programs

Deployment layer concerned with introduction, usage, maintenance, evolution of the system
Abstraction Layer

Application domain layer

Scoping

Requirements acquisition layer

Variating

Business user layer

Designing

Conceptual layer

Implementing

Implementation layer

Structuring specification

Distribution specification

Dialogue specification

Functionality specification
Summarising

- Services are often considered only at the implementation layer.
- Services provide an added value for the user, for the application.
- Distinguish between application domain **description**, requirements **prescription**, and information system **specification**.
- System solutions include social systems, computation systems, and organisational issues.
- W*H specification frame for the full description.
- All W-questions can be mapped to technology in a way that the systems developer understand the issue.
- Our framework captures all requirements including the users.
Thank you!

thalheim@is.informatik.uni-kiel.de
Exception-Aware (Information) Systems

Exception-Aware (Information) Systems
EJC 2012

7. 6. 2012

Hannu Jaakkola & Bernhard Thalheim
(1) Tampere University of Technology, Pori, Finland
(2) Kiel University, Germany
(*) Kolmogorow Professor h.c. of Lomonossow University Moscov
It’s Not a Bug It’s an Exception

Encyclopedia Britannia

(1) the act of excepting: exclusion
(2) one that is excepted; especially: a case to which a rule does not apply
(3) question, objection (witnesses whose authority is beyond exception T. B. Macaulay)
(4) an oral or written legal objection

Wordnet

(1) (18) exception, exclusion, elision
(a deliberate act of omission; “with the exception of the children, everyone was told the news”)
(2) (8) exception
(an instance that does not conform to a rule or generalization; “all her children were brilliant; the only exception was her last child”; “an exception tests the rule”)
(3) exception
(grounds for adverse criticism; “his authority is beyond exception”)
Exceptions in Programming Languages

- Error conditions
- Unforeseen events
- Tracking and controlling

Treatment
typically parameter-less, name-less, with local variables, sequence of statements

Call of an error handling procedure

Dissemination of exceptions

implicit by the controller or explicit call (raise, try-catch)
similar to execution of subprogrammes with potential return
Exceptions in Programming Languages

Object-Based Languages

at the end of each programming unit:

- named rules for exception handling
- predefined exceptions
- implicit call through operation or explicit call through statement (raise)
- upward inheritance until program unit has defined exception
- after completion of exception: control back to calling unit

Assertions and assurance based on contracts

(Eiffel, C++ makros)

Exception injection from superclasses

the nightmare of Java programming

Functional languages

explicit definition of exception
activation through raise
computation through handle with pattern and execution expression
Classical Information Systems Modelling
Results in Exceptions

- Building blocks (Chen’s notions): attribute type, entity type, relationship type, cluster type, etc.
  - great facility for modelers
  - bad restriction for users

- Strict bottom-up specification
  - inductive type construction

- Basic standardized (DBMS-influenced) type system
  - simplicity of programming
  - nightmare of DBS farm integration

- Mapping (transformation) of structuring and requests to ortho-normal languages (e.g., SQL, HERM)

- Overloaded values and types require
  - exception handling (e.g., NULL, default)
What is Wrong with Inductive Modelling?

Decompositional optimization (normalization) enforces granularity and, thus, leads to concepts and types of very fine granularity instead of building blocks.

Component modelling

Strict construction rule application
  ■ does not match with property inheritance (e.g., antonyms)
  ■ leads to exception hierarchies

Parameterized rules, higher-order rules

Orthogonal dimensions cause large (structural) repetition and redundancy

Many-dimensional modelling
  Meta-concepts, e.g., interaction pattern, meta rules
The Category-Problem-Cause-Solution Template

Category: description of general properties of concept-based macro-states with boundaries

Problem: macro-state space reachable from the current state with improved states, actions for moving from one state to another state goal test, problem solution controller

Cause: description of causes and potential observation points,

Solution: definition of the solution, illustration of the solution, examples, pattern used for the solution;

extended by required behaviour is to be controlled so that it satisfies certain conditions, commanded behaviour is to be controlled in accordance with commands issued by an operator, information display states and behaviour information, imple Workpiece (create/edit/...) workpiece by user) and transformation mainly based on I/O transfer.

Reasoning approach:

■ abductive reasoning (generation of hypotheses which, if true, would explain the collection of observations) +

■ plausible reasoning (conclusions may have been withdrawn because of additional information) +

■ evidential reasoning (relating observed effects and problems to causes)
Generalising Database Techniques

Static and dynamic integrity constraints e.g. structural constraints, semantic constraints, representation constraints, pre- and postconditions, operation clustering through TA’s, run restrictions, features for instantiation, aspects as generic cross concerns

Explicit modelling of exceptions e.g. null values, default values, placeholders, optional structuring, optionality

Explicit modelling of goodness in specific applications depending on context, versions

Inherent constraints of the language e.g. set semantics, compositionality,

Modelling of dynamism through schema layers on the basis of base data, production data, archive data, derived data, log data, data under development
Learning from Constraint Maintenance System

Type specification with constraints, with scoping of constraints
extension of current framework for anticipating and preparing

States of integrity constraints as a state diagram
explicit treatment of constraint context (type environment), their propagation
explicit invalidation detection query
first: restricted class of constraints without soft constraints

Operation set for treatment of violations with precondition and postcondition operations

Integrity manager monitoring static properties (successful operation) and dynamic properties
detection of probably inconsistent parts of the database
ability for diagnosing, resolving exceptions
treatment of exception conformance (consistency) and composition

Truth maintenance system
The Exception Facilitation Model

**Inquire:** Discovering the symptoms.

**Investigate** Defining the current state.

**Vision:** Defining the possibilities.

**Analyse:** Generating a list of potential solutions.

**Qualify:** Narrowing solutions down to those with the greatest leverage.

**Plan:** Securing ownership, commitment, permission.

**Apply:** Managing the realisation of the solution(s).

**Report:** Measuring the final outcome and capturing experience.
Proactive Exception Handling

based on a *general conceptual approach*

- (a) *Conceptualisation of exception handling solutions.*
- (b) *Enhancement of conceptual schemata by exception handling templates.*
- (c) *Development of control and measurement practices.*
- (d) *Development of parameter set reduction and dependence representation techniques.*
- (e) *Substantiation of data mining and statistics techniques for performance analysis.*
- (f) *Development of a exception handling framework.*
Our Approach to Exception Handling

Multi-layering of applications: object in a good state, bad state, exceptional state

State-based separation: states for the example

- normal state such as insured car
- intermediate states such as dealer car, owned car not (yet) insured, lessee car, factory car
- final states such as junkyard cars

Business rule injection into exception handling: extended transaction or trigger models

Automatic exception containers: orthogonal ‘ghost’ schemata based on horizontal decomposition

Multi-shell exception handling beyond open-world for exceptions based on explicit shell modelling
Ideas Driving Our Approach

- **Conceptualisation of solutions to exceptions**: Category-Problem-Cause frame
- **Enhancement of conceptual schemata by exception templates**: schemata and schema elements enhanced by templates that characterise exception problems
- **Development of control and measurement practices**: proactive exception handling

exception monitor
Kinds of Exceptions

Exceptions caused by errors: operation errors, design errors, organizational errors, hardware errors

explicit error treatment

recovery management based on explicit specification of errors

Exceptions caused by randomness or non-determinancy: appear and vanish at any point of time

cannot be eliminated not described

extensions of recovery management?

Exceptions caused by incompleteness: modelling, specification incomplete due to complexity, limitations of languages and abilities

robust system specification?

Exceptions as systems flexibility: exceptions as ‘normal’ states or ‘normal’ reactions

exception handler: exceptional situations and correct treatment
### Occurrence of Exceptions (A)

**Incompleteness**

<table>
<thead>
<tr>
<th>Reasons, causes</th>
<th>Possible resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>incomplete knowledge</td>
<td>negated specifications</td>
</tr>
<tr>
<td>incomplete coverage</td>
<td>robust specifications</td>
</tr>
<tr>
<td>macrodata modelling</td>
<td>redesign to microdata</td>
</tr>
<tr>
<td>inability to represent</td>
<td>approximative specifications</td>
</tr>
</tbody>
</table>

---

Incompleteness of specifications as “modelling gap”

**Exception-Aware (I)Systems**

7. 6. 2012

Jaakkola/Thalheim
The Modelling Gap

"Partial reality"

Part of reality

Things of reality

Observed property

"Topic"

Predicator

Foundation of decisions

Modelling decision

Context

Modeler

under usage

acts within

Revision during the development process

Usage of theory

"Schema" as result and partial point of view of a database development process

Modality

Exactness

Confidence

Content

Information

Concept

Topic

State-of-art
Cy/Pm/Ce/Sn

Our solution

Levels

Integrity Ex.

Handler

Finally

Exception
(Aware
(I)Systems
7. 6. 2012
Jaakkola/Thalheim
Occurrence of Exceptions (B)

<table>
<thead>
<tr>
<th>Reasons, causes</th>
<th>Possible resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>implementation restrictions</td>
<td>extending by theories and languages</td>
</tr>
<tr>
<td>conceptual language restrictions</td>
<td>novel theories, advanced logics</td>
</tr>
<tr>
<td>restricted attention of developers</td>
<td>scope of reference models</td>
</tr>
<tr>
<td>non-axiomatizability</td>
<td>change of logics</td>
</tr>
<tr>
<td>locality of reasoning</td>
<td>interference reasoning</td>
</tr>
</tbody>
</table>

Insufficiency to represent the current knowledge
### Occurrence of Exceptions (C)

Due to evolution

<table>
<thead>
<tr>
<th>Reasons, causes</th>
<th>Possible resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>change-sensitive normalization</td>
<td>change of normal form</td>
</tr>
<tr>
<td>time overload and mingling</td>
<td>separate TA, user, validity time</td>
</tr>
<tr>
<td>non-temporal types</td>
<td>temporal types</td>
</tr>
<tr>
<td>too restrictive models</td>
<td>flexibility</td>
</tr>
<tr>
<td>instability of schema</td>
<td>dynamic schemata</td>
</tr>
<tr>
<td>temporary runtime error</td>
<td>similar to 9 kinds of nulls</td>
</tr>
</tbody>
</table>
Occurrence of Exceptions (D)

<table>
<thead>
<tr>
<th>Hidden cases</th>
</tr>
</thead>
</table>

due to limiting to “normal case”

<table>
<thead>
<tr>
<th>Reasons, causes</th>
<th>Possible resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>pragmatic assumptions</td>
<td><em>explicit modelling</em></td>
</tr>
<tr>
<td>hidden assumptions</td>
<td><em>iterative testing</em></td>
</tr>
<tr>
<td>self-restrictions</td>
<td><em>detection of reasons</em></td>
</tr>
<tr>
<td>restricted scope of users</td>
<td><em>education, sharpening</em></td>
</tr>
<tr>
<td>overlooked cases</td>
<td><em>analysis, verification</em></td>
</tr>
</tbody>
</table>
### Occurrence of Exceptions (E)

#### Operational nondeterminism

<table>
<thead>
<tr>
<th>Problem-Cause</th>
<th>Solution template</th>
</tr>
</thead>
<tbody>
<tr>
<td>hidden environment</td>
<td><em>context-aware programming</em></td>
</tr>
<tr>
<td>automatic optimisation</td>
<td><em>directives for the optimiser</em></td>
</tr>
<tr>
<td>operational freedom</td>
<td><em>dynamic hints for execution</em></td>
</tr>
<tr>
<td>change from test to operational mode</td>
<td><em>analyse tests with defects</em></td>
</tr>
<tr>
<td>consolidation and integration</td>
<td><em>re-engineering and redesign</em></td>
</tr>
</tbody>
</table>
Insufficiency

Treatment of concurrent actions and overcoming locality (mapping to atomic constraint cases)

Introduction of specific default values for exceptional cases in domains
dates misspelled, wrong, not according rules, doubtful
specialization of types for treatment of exceptions

Measuring incompleteness of specification, implementation etc.
correctness of schema #correct types
completeness of schema #types #represented concepts #application world concepts
capability #conflict−free types among application world concepts #types among application world concepts
exception of various types: (e.g. correctness)
incorrectness of type system
incorrectness of constructor system
incorrectness of structuring
incorrectness of static semantics

Introduction of robust schema parts
tolerance and distance to error measures
accuracy and confidence
Dynamic changes

Change of normal form
separation of robust and changing parts within schemata
separation of strong and soft constraints
introduction of “almost valid” constraints

Explicit introduction of time domains
TA, user, validity time
explicit monitoring of environment

Temporal types
explicit volatile types (temporary tables)
explicit virtual types

Flexibility
robust schemata

Dynamic schemata
introduction of an explicit semantics state
state management similar to TA management
Hidden cases

Detection of assumptions
development and programming culture
making restrictions explicit
AI or mining of developers and programmers behavior

Iterative modelling
robustness properties
explicit change and version management

Detection of reasons for self-restriction

Broadening the scope
of developers
abstract programming

Control and correction of completeness problems
development of completeness criteria
analysis and verification tools
predictability of correctness and exceptions
Exception Management by Separation into Levels

‘what’ (level 1) provides a specification;

‘how’ (level 2) defines the way the framework is going to work;

‘do’ (level 3) prescribes the application of the assessment;

‘plan’ (level 4) provides the methodology for the application;

‘manage’ (level 5) allows the governance of the quality framework;

‘coordinate’ (level 6) integrates the framework into the entire development process;

‘optimize’ (level 7) revises the project management.

Finally
Example: Integrity Constraints

Integrity constraint globalisation through globalisation graphs

Integrity constraint handlers with resumption, termination, explicit treatment of status of exceptions, basic primitives, control structures, and specific invocation management of explicit and derived exceptions

Integrity violation patching as small “repairing” action (locally, partially, nested?)

Embedding into Oracle PL/SQL facilities as reference implementation proposal

Abstract specification through ASM framework with correctness and completeness properties and refinement
Integrity Constraint Specification

Integrity Constraint $\varphi$

[Localization: $< \text{unit\_name}>$]
[Partiality: $< \text{validity\_condition}>$]
[Exception: $< \text{exception\_condition}>$]
[In-Context: $< \text{enforcement\_rule, time, granularity}>$]
[Out-Context: $< \text{conditional\_operation, accept\_on}>$].

(Local and global) integrity constraints (environment and behavior)

- Static
  - Structural
  - Semantical
  - Representation constr.
  - Development constr.
- Dynamic
  - Transition constr.
  - Temporal
Constraint Graphs through VisualSQL

(1) Computing constraint environment
(2) Case-base reasoning techniques for graph completion
(3) Derivation of deontic covers within each case
### Elementary Exception Case Specification

<table>
<thead>
<tr>
<th>Exception</th>
<th>referential integrity failure after update in referenced table</th>
</tr>
</thead>
<tbody>
<tr>
<td>For Event</td>
<td>update in referenced table</td>
</tr>
<tr>
<td>Definition</td>
<td>$(DB \models R[X] \subseteq S[Y]) \land (\text{upd}(DB,S,Y) \not\models R[X] \subseteq S[Y])$</td>
</tr>
<tr>
<td>Critically</td>
<td>availability of referencing table</td>
</tr>
<tr>
<td>Anticipated processes</td>
<td>cascading</td>
</tr>
<tr>
<td>Detection processes</td>
<td>checking availability</td>
</tr>
<tr>
<td>Avoidance processes</td>
<td>locking referencing table and channel or deriving a retrying/resumption model</td>
</tr>
<tr>
<td>Resolution processes</td>
<td>call exception handler “cascade2” or “retry3”</td>
</tr>
</tbody>
</table>
**Elementary Exception Case Specification**

<table>
<thead>
<tr>
<th>Exception handler</th>
<th>cascade2: cascading an action throughout scope</th>
</tr>
</thead>
</table>
| Definition         | `optain_lock(R); upd(DB,R,select Y as X from S`  
|                    | `where Y = :change)`                         |
| Precondition       | update scheduled on parent table              |
| Overhead           | locking on child table                        |
| Misleading positives | update blocked in child table                |
| Best for           | direct push of an update                      |
Enforcement of Integrity Constraint Specification

Direct enforcement through declarative constraints with RESTRICT, NO ACTION, CASCADE, SET VALUE (null, default), [INITIALLY] DEFERRED, [INITIALLY] IMMEDIATE [DEFERABLE]

Transactions with three mechanisms on failure:

(1) rollback on inconsistency currently exclusive treatment

(2) erasing effects of TA:

   transaction COMPENSATED_ON_FAILURE_BY transaction

(3) raising an exception:

   transaction CONTINGENTED_ON_EXCEPTION_BY exception

Triggers with the after-before activation time, row-statement granularity,

1-n (SQL:1999, DB2, Informix, SQL Server), n-1 (Sybase) or n-n (Ingres, Oracle) event-trigger pairs
Exception Specification in Oracle PL/SQL

**Declaration Section:** Deklaration (names,...) of user-defined exceptions

```
DECLARE exception EXCEPTION
```

**Exception Section:** Definition of actions raised in case of appearance of an exception

```
WHEN exception THEN pl/sql-statement;
WHEN OTHERS THEN pl/sql-statement;
```

Exceptions can be called at arbitrary places in PL/SQL blocks through `RAISE`:

```
IF condition THEN RAISE exception
```

**Execution sequence**

- call of an exception
- execution of corresponding actions in `WHEN`-clauses
- stop of execution in the calling block (e.g., continuation in anonymous blocks)

```
CREATE OR REPLACE TRIGGER forbidden_action_at_evening
    BEFORE DELETE ON System_Table
    BEGIN
        IF TO_CHAR(SYSDATE,'HH24:MI') BETWEEN '18:00' AND '22:00'
            THEN RAISE_APPLICATION_ERROR (-20101,'Wer zu spaet kommt, den ...');
        END IF;
    END;
```
Exceptions and Processes

ASM specification
universe: signal, action, timePoint
monitored functions: Clock, Signal
shared functions: stateAction(a), stateTA(ta)
rules: Exception(exc)

\[
\text{ACTIONREACTIONTIMEOBSERVER}(a, eV, s, sReact, iReact, excVal, excReact, TA) = \\
\begin{align*}
\text{if } & \text{Clock} = p \text{ and } \text{time}(a) = p \text{ then } \text{stateAction}(a) := \text{active} \\
\text{if } & \text{stateAction}(a) = \text{active} \text{ and } \text{time}(s) \in [\text{time}(a), \text{time}(a) + p] \\
& \text{and } \text{Clock} \in [\text{time}(a), \text{time}(a) + p] \\
& \text{then } \text{stateAction}(a) := \text{receivedSignal} \\
\text{if } & \text{stateAction}(a) = \text{active} \text{ and } \text{time}(s) > \text{time}(a) + p \text{ and } \text{Clock} > \text{time}(a) + p \\
& \text{then } \text{Exception}(\text{excVal}) \text{ par } \text{stateAction}(a) = \text{interrupt} \\
\text{if } & \text{stateAction}(a) = \text{receivedSignal} \text{ and } \text{Clock} = \text{time}(s) + s\text{React} \\
& \text{then } \forall \text{ta} \in T\text{AstateTA}(ta) := \text{enabled} \\
\text{if } & (\forall \text{ta} \in T\text{AstateTA}(ta) = \text{enabled}) \\
& \text{and } \text{Clock} = \text{time}(s) + s\text{React} + i\text{React} + 1 \\
& \text{then } (\forall \text{ta} \in T\text{AstateTA}(ta) = \text{abort}) \text{ par } \text{Exception}(\text{excReact}) \\
\text{for } & \text{a(ction), e(nd)\text{V(aility), s(ignal), s(tart)\text{React(ion),}} \\
& \text{i(nterval)\text{\text{React(ion), exc(ption)\text{Val(idity), exc(ption)\text{React(ion), TA}}} \\
\end{align*}
\]
Exceptions and Processes

ASM specification
universe: signal, action, timePoint
monitored functions: Clock, Signal
shared functions: stateAction(a), stateTA(ta)
rules: Exception(exc)

ACTIONREACTIONTIMEOBSERVER(a, eV, s, sReact, iReact, excVal, excReact, TA) =
  if Clock = p and time(a) = p then stateAction(a) := active
  if stateAction(a) = active and time(s) ∈ [time(a), time(a) + p]
  then stateAction(a) := receivedSignal

  if stateAction(a) = active and time(s) > time(a) + p and Clock > time(a) + p
  then Exception(excVal) par stateAction(a) = interrupt

  if stateAction(a) = receivedSignal and Clock = time(s) + sReact
  thenforall ta ∈ TA stateTA(ta) := enabled

  if (forall ta ∈ TA stateTA(ta) = enabled) and Clock = time(s) + sReact + iReact + 1
  then (forall ta ∈ TA stateTA(ta) = abort) par Exception(excReact)

for a(ction), e(nd) V(alidity), s(signal), s(tart) React(ion), i(ntervall) React(iion),
exc(eption) Val(idity), exc(eption) React(iion), TA
Exception-Aware (I)Systems

7.6.2012 Jaakkola/Thalheim

State-of-art Cy/Pm/Ce/Sn
Our solution
Levels
Integrity Ex.
Handler
Finally

Content
Information
Concept
Topic

State Chart of an Exception Handler

- Exception reasoner
- Collect possibilities
- Instantiate solution
- Specific EH plans
- Pick EH plan
- Execute EH plan
- Actions using advanced stored procedures
Concluding

General model for exception specification, handling and monitoring both for TA’s and processes

Five main pattern for exceptions

This paper: based on signals/events

Open issues

- hierarchically structured exception sets
- general monitors
- tracers and detectors
- exceptions of exceptions
Future Research

Hierarchically structured exception sets

General monitors for exception sets

Tracers and detectors for exceptions of various categories and

Exceptions of exceptions
Thank you!
Pattern-Based Construction of Large Information Systems Schemata

Pori

Feb. 08, 2012

Rene Noack and Bernhard Thalheim

Technologie der Informationssysteme

Institut für Informatik, Christian-Albrechts-Universität zu Kiel, BRD
Kolmogorow-Professor e.h. der Lomonossow-Universität Moskau
Our Background

Our Schema Library

collection of applications since 1989
developed with the database design workbenches (DB)^2, ID^2
more than 9.500 schemata,
more than 4.500 very large (more than 1.000 E/R types),
≈ 50 % in use for at least 5 years
largest schema: ≈ 80.000 attribute, entity, relationship types
analysis of SAP R/3 schema (1999): 16.500 relations, 35.000 views,
150.000 functions (part of more than 350.000)
2002 more than 21.000 + 43.000 + 450.000
e.g., more than 75 relations for storing address information
MySAP more than 28.000 + 53.000 + 3.500.000

superficial similarity, repeating solutions
e.g., production applications can be condensed to 6 large framework schemata (person + organization, product, production, ordering + delivery + invoice, accounting + budgeting, human resources)
Modern Information Systems

Database systems as the kernel
\[ \text{DBS} = \text{DBMS} + \{ \text{DB} \} \]

- Structuring of databases (structure + static integrity)
- Functionality based on programming by generic functions

Information systems based on database systems and coping with the user perspective:

- Users stories, scenarios within the story space
- Users views on content based on media types
- Context (users, content, functionality, environment, provider, history)

Collaborating information systems coping with component systems

- Components providing services over networks
- Collaboration for task solution

Nowadays: Co-Design of Four Dimensions
Towards real software engineering and towards software science
Conceptual Modelling in the Large

Database applications form an integral part of most computational infrastructures everywhere in applications.

Models also depend on the cultural and educational background, coherence and consistency of the many coexisting models challenged by liberation of data from structure and the integration of derived or aggregated data, e.g. in streaming databases, data warehouses.

Conceptual modelling in the small has become state of the art.

Conceptual modelling in the large is mainly developed within companies that handle large and complex applications that cover a large variety of aspects such as models of structures, of business processes, of interaction among applications and with users, of components of systems, and of abstractions or of derived models such as data warehouses and OLAP applications.

Conceptual modelling in the large is typically performed by many modelers and teams. It includes architectural aspects within applications, quality, configuration and versioning of models.
Challenges of Modern Software Engineering

Currently mainly handicraft development in the small starting with requirements

Programming of systems depending on scope and complexity

programming in the small ✓
programming in the large: strategies, architectures, patterns
programming in the world within a collaboration
programming by composition or construction

Engineering as handicraft work of an artisan
or simply as copy and refine
or construction based on components

Trilogy consisting of

• application domain description,
• requirements prescriptions,
• systems specifications
Paradigms of Programming in the Large

Architecture with a variety of viewpoints (application architecture, technical (component/module) architecture, infrastructure architecture)

Component construction via components, interfaces, connectors

Development methodology for teamwork, team chairing, chair, roles, responsibilities, obligations

Integration/Collaboration of components, also dynamic

Distributed and embedded systems via networks

Testing, verification, validation for control and coherence/consistency check

Open source development under guru supervision

Pattern-based refinement
Architecture: Towards an Integration of Architecture into Design & Development

... after the nightmare of 1001 definitions

System-construction orientation: components, interfaces, services

Meta-structure of a system

Viewpoints depending on purpose

Conceptualisation of a system: components, obligations, permissions

Abstraction layers of system development: application domain layer ... implementation layer ... maintenance layer

Architectural pattern for the definition of functioning of a system

Semiotics for a system: structure with syntax, semantics and pragmatics

Concerns for a system: static and dynamic representation

Quality of service as one of the assessment criteria

[Let's learn from architects!]

Architecture is an art and an engineering not only one concern.

Think about facility management.
Pattern

Pattern: class of solutions to a problem within a Kripke world space (context)

$$(\mathcal{P}, \{(\mathcal{W}, \{S\})\})$$

Kripke worlds $\mathcal{W}$ are typically given by

a Kripke frame $\mathcal{F} = (S, R)$

that consists of

- a non-empty set $S$ (of worlds / states)
- an accessibility relation $R \subseteq S \times S$

A Kripke structure $\mathcal{K} = (S, R, I)$

consists of

- a Kripke frame $\mathcal{F} = (S, R)$
- an interpretation $I : Var(\mathcal{L}_{modal}) \times S \rightarrow \{1, 0\}$
Problem Space Specification

Allgemeiner Polya-Rahmen

Problem description using abstraction

Context

Carrier/language based on elementary elements, constructors, associations, collections, classification

Solution

more detailed:

state space reachable from the initial state with goal states

actions for moving from one state to another state

goal test

problem solution controller
Solution Space Specification

Definition of the solution

Illustration of the solution

Examples

Pattern used for the solution

extended by

Required behaviour: is to be controlled so that it satisfies certain conditions

Commanded behaviour: is to be controlled in accordance with commands issued by an operator

Information display: states and behaviour information

Simple Workpiece: create/edit/... workpiece by user

Transformation mainly based on I/O transfer
**Elementary Pattern: Elementary Structures**

- **Name pattern**
- **Identifier pattern**
- **Date pattern**
- **Boolean, flag pattern**
- **Units of measure pattern**
- **Description pattern**
- **Comment pattern**

**Association pattern with varying arity (list-of-parameters)**

\[
\text{Component}(R(S),T(\text{Identif}, \text{OtherComponents}))
\]

**Based-on pattern**

\[
\text{BasedOn}(R,T(\text{Identif}, \text{OtherComponents}))
\]

**Constructor pattern**: tree-of constructor, set-of constructor, product constructor
Elementary Pattern: Constructor
Constructor for Bags and Lists

Standardisation results in better constructivity!
Elementary Pattern: Hierarchies in Multidimensional Schemata

Pattern observed for business

- Organisation
- Trade product
- Supplier
- Producer

Concept

Information

Content

Introduction
CM in Large Architectures
Pattern
Construction
Applying
References
Elementary Pattern: Meta-Characterisation Pattern

Quality Constructor quality of data
accuracy (consistency, measures), objectivity (consistency, author, update policy, evolution), believability (source, design, processes) reputation (credibility) relevancy, value-added, timeliness (age, source currency, non-volatility), completeness, and amount of information representational quality parameters: interpretability (design, models and languages, query processing, data and processes), ease of understanding (interpretability, aliases), concise representation, consistent representation, ease of manipulation

Temporality Constructor

The Source Constructor

Terms and Regulations Constructor TermType, association types such as OrderTerm, AdjustmentType
Elementary Pattern: The Multi-Layer Construction Providing Flexible Organisational Structures

Separation of stability

Fixed

Temporal

Organisation structure type

Organisation structure

Construction rule /regulation

Time frame

Organisation
Elementary Pattern: Log of Actions, e.g., Association of Person to Ordering

Recording of actions as a pattern

```
<table>
<thead>
<tr>
<th>Person</th>
<th>Activity Role</th>
<th>Role Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identif</td>
<td>FromDate</td>
<td>[ThruDate]</td>
</tr>
<tr>
<td></td>
<td>[OtherProperties]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Log Type</th>
<th>Identif</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Person</th>
<th>Order Role</th>
<th>Order Role Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identif</td>
<td>FromDate</td>
<td>[ThruDate]</td>
</tr>
<tr>
<td></td>
<td>[PercentContribution]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order</th>
<th>Date</th>
<th>[EntryDate]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identif</td>
<td></td>
<td>[ShippingInstructions]</td>
</tr>
</tbody>
</table>
```
Elementary Pattern: Potential and Actual Objects in a Library Application

Separation of general properties from specific

Derived key for BookCopy: \{ ISBN, CopyNumber \}

with foreign key constraint

Natural for the relational database model
Constructor Pattern: The Bulk Constructor Within The Order Pattern

Generalisation through kind introduction
Construction Pattern: Pattern of Units in Production Applications

Skeleton introduction

- Address
- Person Organization
- Supplier
- Customer
- Product
- Production
- Order
- Delivery
- Billing
- Personnel Management
- Fond
- Budget
- Budget Department

Introduction
CM in Large Architectures
Pattern
Construction
Applying
References
Construction Pattern: Dimensions and Unfoldings for the Order Pattern

Separation of concern

Product → Identif

Order List Item

Order → Identif

Ordering Party → Identif

Sales Order List Item

Sales Order → Identif

Customer → Identif

Purchase Order List Item

Purchase Order → Identif

Supplier → Identif
Introduction
CM in Large Architectures
Pattern
Construction
Applying
References

Content
Information
Concept
Topic

Complex Pattern
Deriving principles of constructing complex structures

Pattern for CoMoL
Feb. 08, 2012
Noack/Thalheim

principles of constructing complex structures

meta
special implementation
star snowflake

context
processing context
quality

utilisation

bulk
utilisation
based on

associations
constructor
component hinge

constructor

hierarchies
type semantics

potential actual
product ...

multiset

classification

history
layers
security performance

log ...
strategic tactical

utilisation

bulk
based on

constructor

component hinge

type semantics

classification

historical associations
Construction Pattern: Person and Organization Pattern

[Standard types used for construction]

Pattern for CoMoL
Feb. 08, 2012
Noack/Thalheim

Introduction
CM in Large Architectures
Pattern
Construction
Applying References

Concept
Topic

Information

Content

Skills profile
Profile
Education profile
Profile
Education institution

Person

Employee

Organization

Role Type

Party

Relation

Kind

Status

From

To

ID

FromDate

ThruDate

Comment

CreditRating

ID

From

To

TaxID#

Relation

Priority(Code, Description)

RelationKind

Description

Property

Code

Description

URL

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModified

LastModi
Construction Pattern: Party Contact Schemata
**Construction Pattern: Log Dimension**

**Recording utilisation**

- **Organization**
- **Role Type**
- **Party**
- **Kind**
- **Relation**
- **Status**
- **Actor**
- **Scenario**
- **Dialogue step**
- **Business process**
- **Scene**
- **Element of**
- **Media object**
- **Address**
- **Acquisition**
- **Source**
- **Priority**
- **Kind (Standard, default)**
- **Occasion**
- **Time**
- **Archived**

**Pattern for CoMoL**
Feb. 08, 2012
Noack/Thalheim

Introduction
CM in Large Architectures
Pattern
Construction
Applying
References

- **Content**
- **Information**
- **Concept**
- **Topic**
Constructors: Meta-Architecture, e.g. Waffle
Constructors: Meta-Architecture, e.g. Evolution

- Approved Application
- Travel Document
- Traveller Application
- Funded Application
- Document Frame
- Document
- IsIn TStage
- Stage Type
Constructors: Meta-Architecture, e.g. Incremental

- Person
- Organization
- Supplier
- Business Sales Rule
- Order
- Party
- Requisition
- Request
- Response
- Quote
- Product

IsA

Governed By

Billing To

In Response To

Answers

On The Basis Of

By

By

By

Creator

For

Of

On

In Response To

For

Creator

Of

On

In Response To

For

Creator

Of

On

In Response To

For

Creator

Of

On

In Response To
Dimensions in the Production Pattern
Layering in Schemata

Knowledge/strategic level

Temporal/tactical level

Regulations

Responsibility type

Party type

for

Responsibility

Party

Action

Time frame

requested_by

responsible

requested_by

responsible

Concept

Topic

Content

Information

Pattern for CoMoL
Feb. 08, 2012
Noack/Thalheim

Introduction
CM in Large Architectures
Pattern Construction
Applying References

Responsibility type

Regulations

Party type

for

Responsibility

Party

Action

Time frame

(requested_by responsible)

(requested_by responsible)

(0,n)

(0,1)
Separation of Aspects

- Protocolling schema
- Quantity schema
- Service area
- Service kind
- Responsibility
- Service
- Product kind
- Resource utilisation
- Spatial schema

Concept T

Topic
Application Domain Layering

Person

Active observation
Hypothesis
Projection

Type of observation

Observation

Type of phenomenon

Phenomenon

Perception

Negative perception
Positive perception

Concept of perception

Measurement

Quantity

Rejected observation
Complex Context-Sensitive Compositions

Pattern for CoMoL
Feb. 08, 2012
Noack/Thalheim

Introduction
CM in Large Architectures
Pattern
Construction
Applying
References

Content
Information
Concept
Topic

Skills profile
Profile
TotalYearsOfExperience
CV
LastModified
Education profile
Education profile
Diploma Issued By
Date
URI
Subject
Specialization

Education institution
Name
Location

Person
Middle Name
Birthdate
Sex
PreviousLastName

Name
LastName
FirstName
Birthdate
Sex

Person Title
Height
MaritalStatus
MothersMaidenName
FathersName

Skillsp
TotalYearsOfExperience

Employee
SocialSecurity#
Position
From
Through

Organization
ID
Name
Lastname

TaxID#
From
Thru

Role Type
ID
FromDate
ThruDate

Kind
Code
Description
Property

Relation
RelationKind
Priority(Code, Description)

Party
ID
From
To

Status
Code
Description

Kind
Description

external
internal

References

Concept Topic

Content Information

Pattern for CoMoL
Feb. 08, 2012
Noack/Thalheim
Let us challenge the idea by an application thus providing a proof of concept!!!
Dimensions
The Specialization and Product-Product-Association Dimensions

Pattern
for CoMoL
Feb. 08, 2012
Noack/Thalheim
Introduction
CM in Large Architectures
Pattern Construction
Applying
References

Pattern for CoMoL
Feb. 08, 2012
Noack/Thalheim

Dimensions (1)

The Characterization Dimension
Dimensions (2)

The Producer, Supplier and Pricing Dimensions
Dimensions (3)

The Product-Product-Association and Categorization Dimensions

Introduction
CM in Large Architectures
Pattern Construction
Applying
References

Pattern for CoMoL
Feb. 08, 2012
Noack/Thalheim

Content
Information
Concept
Topic
Constructing the **Product Schema**

- the central sub-schema **Product\_Intext\_Data**:
  
  \[Product, Item, Service, ProductCharacteristics,\]
  
- **Categorization\_Schema** (\(\alpha,\ Code;\)
  
  \[ProductCategoryClass(\alpha, ProductCategory(Code))\])
  
- **Availability\_Schema** (\(\alpha;\ ProductObsolesence(\alpha))\),
  
- **Product\_Associations\_Schema** (\(\alpha, \beta, \delta;\)
  
  \[Associates(what : \alpha, toWhat : \beta, kind : \delta))\),
  
- **Inventory\_Schema** (\(\gamma, \epsilon;\) Inventory(what : \(\gamma\), store : \(\epsilon\))
  
  labels what, store for binary type InventoryItem,
  
- **Container\_Schema** (\(\zeta, \eta;\) Container(kind : \(\zeta\), partyAddress : \(\eta\))
  
- **Producer\_Schema** (\(\alpha;\) ProducerOf(\(\alpha,\ Organization))\)
  
- **Pricing\_Schema** (\(\alpha;\) ProductPriceComponent(\(\alpha))\)
Compiling the Product Schema

\[
\begin{align*}
\text{Product\_Schema} & \ni_{\alpha} := \text{CategorySet} \quad \text{Categorization\_Schema} \\
& \ni_{\alpha} := \text{Product} \quad \text{Availability\_Schema} \\
& \ni_{\alpha} := \text{Product}, \beta := \text{Product}, \delta = \text{ProductSubstitute} \oplus \text{ConsistsOf} \\
\text{Product\_Association\_Schema} & \\
& \ni_{\zeta} := \text{ContainerType}, \eta := \text{Address} \quad \text{Container\_Schema} \\
& \ni_{\gamma} := \text{Item}, \epsilon := \text{Container} \oplus \text{Address} \quad \text{Inventory\_Schema} \\
& \ni_{\alpha} := \text{Product} \quad \text{Producer\_Schema} \\
& \ni_{\alpha} := \text{AdditionalCharacteristics} \quad \text{Pricing\_Schema}
\end{align*}
\]

Compilation by

- parameter instantiation based on equality theories,
- expression unification based on term rewriting,
- ER-restructuring based on ER-schematology, and
- derivation of view cooperation based on graph grammars.
Publications on Pattern Development


Privacy-Enhanced Information Systems
A Proposal

24. Workshop Grundlagen von Datenbanken (GvD) 2012
Mai 31, 2012

Bernhard Thalheim
Technologie der Informationssysteme
Institut für Informatik, Christian-Albrechts-Universität zu Kiel, BRD
Kolmogorow-Professor e.h. der Lomonossow-Universität Moskau
Worüber ich nicht sprechen werde.

The Privacy Issue
Known Proposals
Our Target
Policies
The Infom
Management
Concluding

Content
Information
Concept
Topic

HERM 'bible'
Handbook
Dependencies

Practical
Database
Design
Methodologies.
Kuwait
University
Press, 1989

ADBIS
ASM
CM
EJC
e-Bus.
ER

FoIKS
MFDBS
NLDB
Semantics
WIS
WISE

i.e., reaching the big triple "3"'s (3, 30, 300)
(3 (habilitation students, profs, books), 30 (PhD students, editorials, projects),
300 (master/ diploma students, papers, presentations))
Zusammenfassung

Current Situation

Market-based values, decision based on health-irrelevant factors

lack of care, lack of consumer information

- German ‘Gesundheitskarte’: personal health condition, medical treatment, physician visits, specific diseases and disorder, full medical record prepared by doctors
  - views supporting pin-based general role-restricted access
  - storage of data, usage of data
  - relevant patient data stored on the SIM of a health identity card
  - all data are centrally accessible depending on the role of the accessor without control of the proprietor
- Marketing patient data (medications) by IMS Health, NDC Health
  - based on patient tracking software (patient id, age, gender, insurance, medication, physician, therapy)
  - currently used for personalized pharmacy and treatment marketing of physicians
  - data mining of therapies, patient behavior, patient history, patient profiling
- Cross exchange of information between banks, insurance companies and creditation tracking institutions (Schufa)
- Customer card data: direct marketing, analysis, combining card data, cards from banks
- Data keeping without trackable deletion: German government security stories
- Bill Gates’ social security number: 539-60-5125
Facebook - der Freund der Schufa


Mitarbeiter: Prof. Dr. Felix Naumann Dr. Gjergji Kasneci Toni Gruetze Dustin Lange

Für Facebook: 200.000 € jährlich von der Schufa

Maximilian Jenders: Identifying viral tweets on Twitter, 2012


Prof. Dr. Christoph Meinel: "Mit der SCHUFA konnten wir ein renommiertes Unternehmen für ein gemeinsames Forschungsprojekt gewinnen, um gesellschaftlich und wirtschaftlich spannende Entwicklungen im Internet zu untersuchen."
Internet Personalisation Techniques

- Google tracks users, their behavior and habits

Google wants your cell-phone number

Mayer (Google): “Right now we can watch a lot more activity on someone’s PC. But it’s easily possible over the next few years that a lot of that will move to the cell phone.”

Raghavan (Yahoo): targeting users according to their relationships with other users

“Our research has to get more deeply embroiled with social networks. Finding out your likes and dislikes is just the beginning of the proposition.”

Experimental search engine on Yahoo: adjust the results to make them more or less oriented towards shopping, and sponsored results could become a part of that mix.

Microsoft: web-based life services that work partly on your network/computer;

Adam Sohn (MSN): “You give us access to more personal information about you and your activities, and we can show you ads that make sense to you.”

- New business spying on customers: need to know who you are, where you are; what you buy, watch and read; who you spend time with; what you say to your friends

plan to nudge and seduce us, literally bit by bit, into agreeing to let them gather the information advertisers need for tailored pitches

- RFID (Radio frequency identification)
Erosion of Privacy

Privacy is currently widely ignored or intensionally neglected

- Changing habits and attitudes
  more comfortable with a lack of privacy;
  companies bribing us with services

  Prabhakar Ragvahan: “Young people think of the internet as a fact of life. They are more open to targeting. ... The ‘sentient network’ magically figures out what you want and provides it at the right time, anywhere.” [to anybody]

- US federal judge: Google must partially comply with a Department of Justice demand for some of the companies search data

- Horror:
  - someone’s cigar-buying habits shared with the insurance company that issued him the nonsmoker’s policy
  - churchgoer who is sitting next to the deacon when her cell phone chirps out a $3-off offer on a bottle of Scotch
  - Boy Scout leader who gets kicked out when he ends up in a gay-rights mailing list after a camera noticed he was standing near by gay-literature section in a bookstore
Issues Alarming People on Privacy

- Unnoticed readout due to ‘invisibility’ of escorting technology and permanent observation
  (“we are currently not completely storing car highway data”)
calm computing of profiles, portfolio, targeted marketing
- Traceability, composition of mobility profiles, RFID in travelling tickets
- Responsibility for history of objects if those are tracked
- Paternalism based on technology, systematic observation and automatic compilation of all minor lapses
- Personalisation and categorization with all negative consequences
- Criminal abuse

Many countries with protection of proprietorship of data but without data protection
Auswüchse: Cookies

Haben Sie auch schon mal erstaunt festgestellt, daß nach Erledigung einer Aufgabe im Netz (z.B. Flugbuchung nach Hyderabad) sich dann plötzlich viele Webseiten für Sie mit spezifischer Information füllen (z.B. Hotels, sightseeing, Impfungen, sowie weiteres recht eindeutiges Material) füllen?

Wie oft habe ich die Seite schon besucht?

Werbefirmen nutzen Cookies, um einzelne Browser auf einem bestimmten Computer zu identifizieren und die Nutzung über mehrere Seiten hinweg verfolgen zu können.

Flash cookies

Aufgedeckte, neuartige Cookies: ETags, pixel cookies, router cookies

Cookie-Blocker-Underminer, cookie placer, cookie revitaliser

Kissmetrics.com

Hulu.com, Spotify

Super-Cookies: kein Kraut dagegen gewachsen

Facebook cookies: ein Aufruf von Facebook genügt.
Facebook - Dein bester Freund

Offiziell Cookies zum Schutz vor Spam-Angriffen mit eindeutiger Identifikationsnummern

Bislang hat Facebook diese Maßnahmen immer mit dem vagen Hinweis auf die Abwehr "böswilliger Aktivitäten" gerechtfertigt. An dieser Aussage zweifelt der hamburgische Datenschutzbeauftragte. Durch seine Anfang November veröffentlichte technische Analyse sah Johannes Caspar den Verdacht erhärtet, dass Facebook "Nutzungsprofile führt, bei denen die gesammelten Daten den Nutzern direkt zugeordnet werden". Wenn der Vorwurf stimmt, könnte Facebook protokollieren, wie Menschen andere Websites nutzen, auf denen Facebook-Dienste wie der Like-Button eingebunden sind.

Facebook wehrt sich gegen diese Bewertung. Arturo Bejar, der Technik-Chef des Unternehmens, versichert auf SPIEGEL ONLINE: "Die Informationen über Seitenbesuche - ganz gleich ob bei eingeloggten oder nicht-eingeloggten Nutzern - werden nicht fürs Werbetargeting verwendet. Wir nutzen diese Informationen nur für die Sicherheit."
Facebook 0.1: Datenbankschema
Mein Schattenprofil bei Facebook

Bis Dez. 2010 Geschäftsbedingung: “We may share your information with third parties, including responsible companies with which we have a relationship.”

Sehr detailliert: emails vieler mit mir verbundener
kein Pattern, wie diese Sammlung entstehen konnte

Riesenphotosammlung z.T. mit recht kompromitierenden Photos

Zusammenarbeit mit dem Heimatschutz-Ministerium (kondensierte Profile), Stimmungszusammenfassungen von Nutzern (Bruttoinlandsglück), posting-Auswertung, Wunder des data mining

Hochgradig spannend zu lesen: Facebook-Darstellung für Börsengang

Viele Anwendungen erzwingen Facebook-Integration: Smartphone-Apps (iPhone-App) mit Generierung von Freund-Vorschlägen, Friend Importer (Facebook), Foto-Tags, Like-Button (wird schon aktiviert, wenn man Webseite nutzt, nicht erst mit Button)

Genau das macht Facebook aber mit allen Messages. Facebook ist deutlich schlimmer als die viel diskutierte Vorratsdatenspeicherung, wo die Daten nach sechs Monaten gelöscht werden müssen und keine Inhalte erfasst sind. Außerdem speichert Facebook die Daten in den USA, wo ich als EU-Bürger keine Möglichkeit habe, zu kontrollieren, ob nicht die dortigen Behörden unter dem Patriot Act darauf zugreifen.

Auskunftsersuchen an Facebook!

Heidelberg Collaboratory for Image Processing (HCI)

👋 Hier kein Facebook-Bashing! Schließlich können sie 500 Mio. nicht irren! 🍂
Schutz

Gebot der Datensparsamkeit

Tracking Blocker: Ghostery gegen eTracker, IVW, opentracker, GoogleAnalytics

Unterbinden von einigen persistenten Cookies und Java-Skript-Schnittstelle

Selektives Deaktivieren von JavaScript (Opera)

Plugin “Better Privacy” zu Firefox

Spionageskripte ???

Auswertung von Serverkommunikationen ???
Andere Länder - andere Sitten
Vertrauen und Sicherheit im Internet

Deutsches Institut für V. und S. im Internet: Milieustudie 2012

Digital natives (stärker Neuorientierung)

- Digital Souveräne 15%; blogs, Foren, streaming, ...
  - adaptive Navi., Autonomie, digital culture, diversity, hedonism
  - ITK selbstverständlich, virtuell, Beschleunigung, Netze

Effizienzorientierte Performer 14%; Arbeit, interneting, Kontakt

- sustainability, regrounding, adapt. Navi., autonomy, digital culture, diversity
- Ich-Vertrauen, Selbstmanagement, Individualität, Unabhängigkeit, Leistungsethos

Unbekümmerte Hedonisten 12%; sharing, peer-to-peer, downloads

- digital culture, diversity, Hedonismus, underdog culture
- Spaß, Unterhaltung, Leben hier und jetzt, gegen Routine und Zwang

Digital Immigrants (mehr Modernisierung/Individualisierung)

- Postmaterielle Skeptiker 10%; emails, Online-Nachrichten, skyping
  - social criticism & resistance, disorientation
  - Vertrauensverlust, Frustration, Pessimismus, Gesellschaftskritik, Technik-Distanz

Verantwortungsbewusste Etablierte 10%; Anwendungen wie online banking

- adaptive Navigation, autonomy, diversity
- Flexibilität, Pragmatismus, Nutzen, Wandel als Chance, Lebensoptimismus

Digital outsiders (mehr traditionell)

- Ordnungsfordernde Laien 12%; Suche (Adressen, Telefon, ...), Angebotsinformation
  - regrounding, sustainability, balance & harmony
  - Balance in Lebenslagen, Schutz, Harmonie, Gesundheits- und Wellnessorientierung

Internetferne Verunsicherte 27% !!!!; keine überdurchschnittliche Internet-Aktivität

- regrounding, balance & harmony, slow down, social criticism, resistance, Disorientation
- Reduktion, Überschaubarkeit, Einfachheit, Sicherheit und Klarheit
### Aus einem 2010-Vortrag zu Web 2.0, Web 3.0

<table>
<thead>
<tr>
<th>Zeit</th>
<th>Aktivitäten</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.24-8.59</td>
<td>Nachrichtenportale</td>
</tr>
<tr>
<td>10.40-11.02</td>
<td>Veranstaltungen, Webmail</td>
</tr>
<tr>
<td>14.03-16.30</td>
<td>Webmail, spez. Parteiseiten, Verkehrsseiten, Wetter, Gesundheitsberatung</td>
</tr>
<tr>
<td></td>
<td>(im wesentlichen männlich), Kieler Seiten (wahrscheinlich beruflich)</td>
</tr>
<tr>
<td>18.21-21.45</td>
<td>GoogleMail, diverse Gesundheitsseiten (wahrscheinlich weiblich), Wetter,</td>
</tr>
<tr>
<td></td>
<td>Veranstaltungen, nichts berufsrelevantes</td>
</tr>
<tr>
<td>22.31-23.45</td>
<td>Männerseiten</td>
</tr>
<tr>
<td>5.24-6.12</td>
<td>YouTube, Facebook, Politically incorrect</td>
</tr>
</tbody>
</table>

*Head-Hunter-Firmen machen dies schon lange und viel besser!*
Lösungen

Ostdeutsch: Verwirrung durch Wirrwarr, Vermüllung

Vorsicht bis zur Paranoia

Überprüfung des Surfverhaltens auch mit mehreren Browsern

Personal Identity Management

Verstecken von Nachrichten: Steganograph z.B.

Bewusste Weitergabe

Separation: echter Freund, Freund, Kumpel, Gesprächs- und Zwitscherpartner

Lebe auch mit/trotz/außerhalb YouTwitFace!
Personal Identity Management

mehrere Identitäten

Verhaltensadaption (bis hin zum Klickverhalten)

Dienste (Anonymisierung, lokale Proxy-Tools, Rewebber, de-Referrer, re-directing, ...

Ausarbeitung einer persönlichen Medienstrategie
Identität und Selbstdarstellung

Virtualität bedeutet auch Identität, ggf. auch gespaltene Teilidentitäten

- **Identität**: Übereinstimmung von Objekt und Subjekt in allen Einzelheiten mit sich selbst und auch relative Konstanz von Verhaltensmustern
- **Selbst**: Ganzheit derjenigen Merkmale, die das Individuum als zu sich gehörig auffaßt
- **Ebenen der Identität**: physisch, beruflich, in der Freizeitaktivität, in der Beziehung, in den Auffassungen;
  - soziale Identität: Selbstidentifikation in Gruppe, Identifikation durch Gruppe
  - personale Identität: Selbstdarstellung
Steganograph als Beispiel versteckter Daten
Das World Wide Web ist voller Daten, deren Inhalte, deren Herkunft, deren Qualität und deren Kontext ein Nutzer oder Nutzerin verstehen muß, ehe er seine Informationen daraus destillieren kann oder gar Wissen daraus entnehmen kann. Und voller Perlen. Daten im Web sind Nachrichten; Daten sind nicht unbedingt Information; Daten sind nicht unbedingt Wissen. Wir versuchen zu klären, wann Daten zu Information werden und wann Daten zu Wissen werden. Anhand von prominenten Webseiten wird untersucht, inwieweit ein Nutzer oder eine Nutzerin zu Informationen kommt, inwieweit die Daten zu Wissen werden können, wie man nach Informationen sucht und man sich auch im weltweiten Wirrwarr (eine bessere Bezeichnung des WWW) zurecht findet. Perlen zu suchen will gelernt sein und ist Arbeit ... ebenso wie dies schon für die Perlentaucher gilt.
“[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.”

**Media privacy:** supported by laws, constitutional rights and other legal frameworks

**Territorial privacy:** supported by laws, constitutional rights and other legal frameworks

**Communication privacy:** supported by laws, constitutional rights and other legal frameworks

**Bodily privacy:** supported by laws, constitutional rights and other legal frameworks

**Information privacy:** not well supported, tools for the “glass box customer”
**Information Privacy**

**Basic Requirements**

**Information privacy:** not well supported, tools for the “glass box customer”

- **Openness and transparency:** no secret record keeping
- **Individual participation:** ability of change by the subject of the record
- **Data quality:** relevant to the purposes and up-to-date
- **Collection limitation:** collection proportional to its purpose
- **Use limitation:** used for their specific purpose by authorized personnel
- **Reasonable security:** adequate security safeguard
- **Accountability:** accountable for the compliance with the other principles
Privacy Principles

Ethics: respecting rights and legitimate interests of others

Proportionality: controls and costs should be commensurate with value and criticality of information

Timeliness: actions should be timely and coordinated to prevent and respond to privacy breaches

Reassessment: privacy should periodically be reassessed and upgraded accordingly

Accountability: responsibility must be explicit

Awareness: principles, standards, conventions, mechanisms should be known

Democracy: weighted against relevant rights of users and other affected individuals
Trends Against Privacy Enhancing Technologies

Ego striptease - I will show you who you are
Data crakes flickr, del.icio.us, upcoming.org and blog: social Web 2.0 applications

Powerpoint Karaoke
The linguistic fingerprint: Silver bullet or mere myth?
In the absence of trust
Transparency and privacy: The 7 laws of identity and the identity metasystem
Know your citizens: State authorities’ access to sensitive information
The 7 laws of identity

(1) User Control and Consent - Technical identity systems must only reveal information identifying a user with the user’s consent.

(2) Minimal Disclosure for a Constrained Use - The solution that discloses the least amount of identifying information and best limits its use is the most stable long-term solution.

(3) Justifiable Parties - Digital identity systems must be designed so the disclosure of identifying information is limited to parties having a necessary and justifiable place in a given identity relationship.

(4) Directed Identity - A universal identity system must support both “omni-directional” identifiers for use by public entities and “unidirectional” identifiers for use by private entities, thus facilitating discovery while preventing unnecessary release of correlation handles.

(5) Pluralism of Operators and Technologies - A universal identity metasystem system must channel and enable the inter-working of multiple identity technologies run by multiple identity providers.

(6) Human Integration - The universal identity metasystem must define the human user to be a component of the distributed system integrated through unambiguous human-machine communication mechanisms offering protection against identity attacks.

(7) Consistent Experience Across Contexts - The unifying identity metasystem must guarantee its users a simple, consistent experience while enabling separation of contexts through multiple operators and technologies.

Microsoft has proposed architectural principles ("7 Laws of Identity") to support convergence towards an inter-operable, secure, and privacy-enhancing plurality of identity systems - an "Identity Metasystem".
Privacy, Identity, and Anonymity in Web 2.0

Web2.0 has created a new rush towards social networking and collaborative applications. This enables new possibilities, but also is a threat to users’ privacy and data. On the surface, many people seem to like giving away their data to others in exchange for building communities or getting their 15 seconds of fame. But below it lie less obvious privacy implications. Some of them are accidental, like publicly marking someone as a “friend” without asking that person before or putting personal data under a creative commons license. But some are more fundamental, as they are based on voluntary surveillance of the users. On the extreme end of the spectrum, the trend towards “identity 2.0″ services - from microformats like OpenID and addressing systems like XDI to infrastructures like Cardspace and Higgins - will have far-reaching impacts on the future of privacy and anonymity on the web.

A number of hidden privacy implications of some web2.0 and identity2.0 services, standards and applications can be observed here.
Research Issues on Privacy Enhancing Technologies

Decentralized architectures and query methodologies for weakly structured and heavily distributed data

Automatic acquisition and integration of context for information supply on real demand (context-sensitive information logistics)

Dynamic orchestration of services, conditioning, optimal service granularity, information asymmetry, payment

Data protection and security preferences of users and automatic alignment with characteristics of services

Adjustment between inspection of technology and minimization of transaction costs

Ubiquitous and calm availability of all relevant data with redesign of business processes, including logistics

Novel cooperation and coordination models based on policies, contracts, arbiters

Support for economy of attention of human users with limited time
Our Restricted Scope to Privacy and Security

Privacy and security as brothers and sisters

- Two sides of the same coin: privacy and security

- Huge variety of conferences, workshops, journals, standards
  - Conferences: Privacy enhanced technology, Information security and privacy, IEEE symposium on security and privacy
  - Books, e.g. D. Salomon, Data privacy and security, Springer 2003

- Large number of papers, proposals, ... on the technological aspects

- Almost no **modelling** research

Our scope: Modelling based on possession and proprietorship

Derivation of privacy enhanced infrastructures
Database Privacy and Security: Highlights

- Security approaches: classifying information, granularity, standards (PICS, P3P, APPEL, ...)
- Protecting sensitive and classified data: access and manipulation restriction; data coding
- Context removal and non-identifiability of information: data cannot be associated to a person, ...; k-anonymization techniques
- Separating networks, extranets: Freenet, Publius, Free Haven, ...
- ‘Hippocratic’ database systems: privacy as central concern
- Privacy and security are different issues of concern
- Many proposals, e.g., Personal Data Access Language (PEDAL)
Security Models

¥
¨
Biskup’s Model
§
¦

Lübbenau 12
Mai 31, 2012
B. Thalheim

Participant ¾

TrustsIn

k

6

k

Requires

^

The Privacy Issue
Known Proposals

ThreatenBy

Known

Our Target

Y
6

-

?
Service

¾

ProvidedBy

6

Policies
The Infon
Management
Concluding
Content

?

-

System

6

+

Environment

Forbidden

¾

Whitelisted

BoundWith

• reliable correct: if all the participants’ requests can be handled by services in
which the participant has trusts

Information
Concept

Topic

• confine: if no illicit services pop up with services used
• trusted: whenever a service pops up with the requested services, it is whitelisted


Database Privacy and Security: Unsolved Issues

Learning from the past

- Traceability of information: inference abilities, deja.news
- Intrusion avoidance: protection schemata problems, protocol problems
- Uninformed users: unawareness of collections, joining
- Carelessness of users: information provision without control
- Portfolio-less and profile-less storage of information: access cannot be separated
- Enterprises and governmental tracking of data: ‘white-box’ taxpayer, mobile phone and phone line connection tracking, biometrical data (Iris data can be used for health tracking)
- Absence of information is information
Our Contribution

Explicit association of pieces of information to proprietors, possessors, ..., enterprises

Systematic methodology: “private information”, privacy-based decomposition and constraints

Framework: informational privacy assertions
- definition of private information and categories
- classification of relationships of “private information pieces” to its possessors and proprietors

Model: restrictions and conditions of possession
- possession is according to consent/no-consent of the proprietor, legal/illegal possession of private information, and awareness/no-awareness

Proof of concept: Framework for closed systems
The Information Privacy Model (IPM)

Towards sound theory and feasibility of management

Infons as the basic unit of chunks of data to be considered

Information: true infon(s)

Possession of infons by agents

Proprietorship of infons by individuals

Logical and procedural treatment of possession, proprietorship and their relations

Constraints limiting the usage of infons

Architectures for information privacy enhanced information systems

Management of possession and proprietorship
Separation of Concern: Proprietor, Possession, Privacy Unit

The Privacy Issue
Known Proposals
Our Target
Policies
The Infon
Management
Concluding

Content
Information
Concept
Topic
The Integrated Privacy Policy

Well-defined policy

Principles on which partners are agreeing

Proper policy: technical (security), administrative, and operational measures
laws and implied threats of legal actions
for discouraging misuse and improper user behavior

Adequately enforced confidentiality
against unauthorized people
assuming that authorized individuals will behave well enough

Contracts between owners, users, providers, keepers, ...

Portfolio and profiles of users

Separation of general content to content, concepts, and topics

Derived security ... techniques for support of privacy

Provable properties
Principles Statement

Towards an explicit management

Accountability: (description)
Identifying purposes: (description)
User consent: (description)
Collection of infons: (description)
Limitations of use: (description)
Accuracy: (description)
Safeguards: (description)
Openness: (description)
Individual access: (description)
Compliance: (description)
Confidentiality: (description)
Access for assuring correctness: (description)
Control over use: (description)
Based on: general conditions
Policy Explication

Towards an open management

Policy: ⟨name⟩
Parties: ⟨general description⟩
Role: party role
Modality: obligation, permission, prohibition
Condition: ⟨name⟩
Based on: general conditions

Policies - defined in the context in which they apply, e.g. community combined with policy tracking and monitoring

State of policy: ⟨characterization⟩
Calculation: ⟨general description⟩
Update: OnCondition, UpdateExpression

extension of P3P policies (test, entity, access, disputes, remedies, statement, nonidentifiable, purpose, recipient, retention, data)
Specification of Contracts

Towards enforcement of policies

Contract: ⟨name⟩
Based on: general conditions
Parties: ⟨general description⟩
  Proprietor: ⟨...⟩
  Possessor: ⟨...⟩
  Notary: ⟨...⟩
  Arbiter: ⟨...⟩
Subject matter: ⟨Infon suite⟩
  Exchange: ⟨binding obligations, permissions⟩
  Computation: ⟨obligations, permissions⟩
  Distribution: ⟨obligations, permissions⟩
  Monitoring: ⟨managers: recognizer, states, timer, constraint scanner⟩
  Notification: ⟨obligations, permissions⟩
  Correlation: ⟨protocols, obligations, permissions⟩
Considerations: ⟨legal conditions⟩
Enforcement: ⟨actions, termination⟩
Based on: general conditions
Specification of Portfolio

Explicit scoping of information

Party portfolio: ⟨party portfolio name⟩
Task: ⟨general description⟩
Characterisation: ⟨general description⟩
  Initial state: ⟨characterisation of the initial state⟩
  Target state: ⟨characterisation of the target state⟩
Profile: ⟨profile presupposed for solution⟩
Instruments: ⟨list of instruments for solution⟩
Collaboration: ⟨specification of collaboration required⟩
Auxiliary: ⟨list of auxiliary conditions⟩
Execution: ⟨list of activities, control, data⟩
Result: ⟨final state, satisfied target conditions⟩
Party involvement: ⟨general description⟩
Role: ⟨description of role⟩
Part: ⟨behavioural categories and stereotypes⟩
Collaboration: ⟨general description⟩
Restrictions: ⟨general description⟩
Party restrictions: ⟨general description⟩
Environment: ⟨general description⟩
Based On: ⟨life cases⟩
Specification of Profiles

Explicit user modeling

Party profile: ⟨party profile name⟩
Education profile: ⟨general description⟩
   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 polarity properties⟩
Derivable profiles: ⟨profile description and enforcement⟩
Security profile: ⟨access control and privacy⟩
Privacy profile: ⟨safety requirements⟩
Content, Concepts, and Topics

Explicit modeling of deliverables

Content: direct representation
query for views, add functionality for processing infons, representing infons, restructuring infons

Concepts: reasoning abilities
theories and semantics based on chunks
aggregation and logical functionality for extracting or deriving concepts

Topics: association generation and representation
annotation and referencing by utterances
merging, associating, integrating, refining topics and topic maps

Infons: concept-topic terms

Assets: content-topic pairs

Semantical units: concept-content pairs

Restricting abilities based on policies
Restrictions contracted by contracts
Explicit Collaboration

Towards behavior of people who came of age

Communication via exchange of messages and information
as only one of the perspectives
demands cooperation and
generates commitments that are managed by coordination
choice of media, transmission modes, meta-information,
conversation structure and paths, restriction policy

Coordination via management of individuals, their activities
and resources
as the dominating perspective
generates communication and arranges tasks for cooperation
pre-/post-articulation of tasks; management of tasks, objects, and time;
loosely ... tightly integrated activities, enabled, forced, blocked

Cooperation as production taking place on a shared space
as the workflow or life case perspective
creates opportunities for coordination and demands communication
storyboard-based interaction, mapped to (generic, structured) workflows
production, manipulation, organization of contributions through media objects

Awareness is fostered by each of the aspects and
mediates each of the aspects
Special Case: Participatory Privacy Enhancing Systems

Matured user with informed proprietors within a fully trusted community

Proprietor sovereignty principle: right to sovereignty over his/her proprietary infons

Policy supporting the proprietor sovereignty principle:
possessor in the role of ‘content and topic observer’;
proprietor in the role of ‘informed owner’ and ‘refresher’

Contract: between proprietor and possessor for using content and topics on an ongoing basis in order to monitor, collect information (about conditions of possession), give a warning, and take actions such as use, security, welfare, accuracy, correctness, and maintenance of infons

Faithful collaboration:
portfolio and profile of contracting possessor do not include any forbidden action or ability, all reporting obligations are observed proprietor is observing his/her obligations
Properties of Participatory Privacy Enhancing Systems
towards proving faithfulness of collaborations

conforms with basic principles of the American ”Code of fair Information Practices” (1973)

1. There must be no personal data record-keeping systems whose very existence is secret.
2. There must be a way for an individual to find out what information is in his or her file and how the information is being used.
3. There must be a way for an individual to correct information in his or her records.
4. Any organization creating, maintaining, using, or disseminating records of personally identifiable information must assure the reliability of the data for its intended use and must take precautions to prevent misuse.
5. There must be a way for an individual to prevent personal information obtained for one purpose from being used for another purpose without his or her consent.


Right of access to personally identifiable information, and to correct the information
Data subject’s right to object to having their personally identifiable information processed
Supportable Policies (1/2)

Individual Participation Policy (IPP): An individual should have the right to obtain from a data controller, or otherwise, confirmation of whether or not the data controller has data relating to him; to have communicated to him data relating to him within a reasonable time, at a reasonable charge, in a reasonable manner, and in an intelligible form; to be given reasons a request is denied; to be able to challenge the denial to challenge data relating to him; and, if the challenge is successful to have the data erased, rectified, completed or amended.

based on proprietor/possessor communication,
verification by third party and possessor

Collection Limitation Policy (CLP): There should be limits to the collection of personal data and any such data should be obtained by lawful and fair means and, where appropriate, with the knowledge or consent of the data subject.

based on contracted limitations of functionality and observation injected enforcement commands and arbiter support

Data Quality Policy (DQP): Private data should be relevant to the purposes for which they are to be used, and, to the extent necessary for those purposes, should be accurate, complete and kept up-to-date.

based on portfolio and profiles,
proprietor recharge of data
Supportable Policies (2/2)

Purpose Specification Policy (PSP): The purposes for which private data are collected should be specified not later than at the time of data collection and the subsequent use limited to the fulfillment of those purposes or such others as are not in compatible with those purposes and as are specified on each occasion of change of purpose.

Use Limitation Policy (ULP): Private data should not be disclosed, made available or otherwise used for purposes other than those specified except with the consent of the data subject or by the authority of law.

Security Safeguards Policy (SSP): Private data should be protected by reasonable security safeguards against such risks as loss or unauthorized access, destruction, use, modification or disclosure of data.

Openness Policy (OP): There should be a general policy of openness about developments, practices and policies with respect to private data. Means should be readily available of establishing the existence and nature of private data, and the main purposes of their use, as well as the identity and usual residence of the data controller.
Support for Policies and Principles

media type: views + functionality + adaptation + presentation

Active media types: in parallel

open(content); inform(proprietor, usage)

Self-protecting media types: content based on queries and supported view

protocol: contact(proprietor, possessor, usage);
obtain(proprietor, token); provide(media type, token)

Communication protocols based on service (distributed ADT), signals, shared
variables, sender/reponder ASM (signature (e.g., signal), phases (via small
ASM)), and timer

represented through SDL or message sequence chart or other protocols

Security techniques against passive/active sniffing, trust exploitation, viruses,
downloadables, OS holes, hacking

Control techniques for focusing, access, user authorization, (password)
protection, biometrics, content/concept/topic security, firewalls, hid-
ing/anonymizing/translating, identity management

Privacy enhancing techniques based on virtual private networks, key encryp-
tion, secure transaction, and corporate policy on security, privacy and control,
cookie cooker
Infon: The Concept

[Generalizing Kauppi, Devlin, Seligman, Wille]

Infon definition:

- discrete item of information
- parametric: objects, anchors

Action results in changing many infons

Basic infon set $I$: (temporary, epistemic) subset of the set of all infons

- all dual infons of the basic infons
- expansion of all anchored infons into the basic infons

all others: secondary infon

e.g., predicates $\text{Patient}(\text{Bernhard, Thalheim,}...,100352422728),
\neg \text{Village}(\text{Dresden,Saxony})$
**Infon Logics**

Constraint relation $\sim \subseteq \text{Infons} \times \text{SecondaryInfons}$

generalization/specialization order $A \preceq B$, similarity $\cong$

**Homogeneity of two infons** $A \sqcap B := \exists X (A \not\supseteq X \land B \not\supseteq X)$

**Inhomogeneity of two infons**

$A \not\sqcap B := \neg \exists X (A \not\supseteq X \land B \not\supseteq X)$.

**Compatibility of two infons** $A \sqcup B := \exists X (X \not\supseteq A \land X \not\supseteq B)$

**Incompatibility of two infons**

$A \not\sqcup B := \neg \exists X (X \not\supseteq A \land X \not\supseteq B)$.

**Derived predicates**:

- Divergency $\Upsilon := \exists \sqcap \land \not\sqcup$
- Isolation $\leftrightarrow := \not\sqcap \land \not\sqcup$
- Potential homogenizability $\not\equiv := \exists \sqcap \lor \exists \not\sqcap \land \not\equiv \land \not\equiv$
- Heterogeneity $\triangleleft := \exists \not\sqcup \land \exists \not\sqcap$


Properties of Infon Logics

- $\vdash A \sqcap B \leftrightarrow \exists x \forall y (x \succeq y \rightarrow (A \succeq y \land B \succeq y))$

- $\vdash A \uplus B \leftrightarrow \exists x \forall y (x \preceq y \rightarrow (A \preceq y \land B \preceq y))$

- **Product:** $C = A \boxdot B := \forall Y (C \succeq Y \leftrightarrow (A \succeq Y \land B \succeq Y)))$

  may be: $Ax_{\boxdot} \vdash A \sqcap B \rightarrow \exists x (x = A \boxdot B)$

- **Sum:** $C = A \boxplus B := \forall Y (C \preceq Y \leftrightarrow (A \preceq Y \land B \preceq Y))$

  possibly required $Ax_{\boxplus} \vdash A \uplus B \rightarrow \exists x (x = A \boxplus B)$

  also: $Ax_{\text{Distrib}} \vdash (\left( A \boxdot B \right) \boxplus (A \boxdot C) \succeq A \boxdot (B \boxplus C)) \land (A \boxplus (B \boxdot C) \succeq (A \boxdot B) \boxplus (A \boxdot C))$

- **Negation** $B = \bar{A} := \forall x (x \succeq B \leftrightarrow x \uplus A)$

  be may add the axiom $Ax_{-} \vdash \exists x (x \uplus A) \rightarrow \exists x (x = \bar{A})$

- **Difference** $C = A \obslash B := \forall x (C \succeq x \leftrightarrow (A \succeq x \leftrightarrow B \uplus x))$

- **Quotient** $C = A \oslash B := \forall x (C \succeq x \leftrightarrow (A \succeq x \land B \uplus x))$

  $\vdash A \oslash B = A \boxplus \bar{B}$
Operations of the Infon Logics

**Product** $C = A \square B := \forall Y (C \succcurlyeq Y \leftrightarrow (A \succcurlyeq Y \land B \succcurlyeq Y)))$

potentially valid axiom $Ax_{\square} \vdash A \forall B \rightarrow \exists x (x = A \square B)$

**Sum** $C = A \bigoplus B := \forall Y (C \preccurlyeq Y \leftrightarrow (A \preccurlyeq Y \land B \preccurlyeq Y)))$

potentially valid axiom $Ax_{\bigoplus} \vdash A \exists B \rightarrow \exists x (x = A \bigoplus B)$

typical properties: $\forall x (\vdash A \bigoplus B \succcurlyeq A \square x)$, idempotency, ...

potentially valid axiom

$Ax_{Distrib} \vdash ((A \square B) \bigoplus (A \square C) \succeq A \square (B \bigoplus C)) \land$

$(A \bigoplus (B \square C) \succeq (A \square B) \bigoplus (A \square C))$

**Negation** $B = \bar{A} := \forall x (x \succeq B \leftrightarrow x \not\bigoplus A)$

(Negation of properties; not subtraction of properties)

potentially valid axiom $Ax_{-} \vdash \exists x (x \not\bigoplus A) \rightarrow \exists x (x = \bar{A})$

if there is a most general infon, then it is not negatable

$\vdash (A \not\bigoplus \bar{A}) \land (A \exists B \lor A \exists \bar{B}) \land (A \succeq \bar{A})$

**General symbol and spezial symbols**

**Quotient** $C = A \ominus B := \forall x (C \succcurlyeq x \leftrightarrow (A \succcurlyeq x \land B \mho X))$

$\vdash A \ominus B = A \bigoplus \bar{B}$

**Difference** $C = A \ominus B := \forall x (C \succcurlyeq x \leftrightarrow (A \succcurlyeq x \leftrightarrow B \not\bigoplus x))$

largest infon that is not contained in $A$ enthalten and that is incompatible with $B$
Extended Infon Calculus

Non-infon for contradictions and empty infon

\[ \vdash \emptyset \uplus (A \not\in A \leftrightarrow \emptyset \equiv A) \land (A \not\in A \leftrightarrow W(A)) \]
\[ \vdash (A \supset B \land W(B) \rightarrow W(A)) \land (W(A) \rightarrow A \supset \overline{A}) \]
\[ \vdash W(A) \leftrightarrow \exists x \exists y (A \supset x \land A \supset y \land x \not\in y) \]

Abstractions at the same layer in general not negatable

Abstractions of higher layers for layering of infon worlds

The laws of the general infon algebra must not be valid!
Infon: Decomposition and Hierarchical Building

Compositional collections of infons

Zero infon has no referent

Spare part number 54321 is in slot a-2
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 has a single referent

Compound infon is an infon that has more than one referent

Reduction property: reducible to a set of atomic infons

\[ A(A_1, A_2, \ldots, A_n) \] reducible to \( A(A_1), \ldots, A(A_n) \) and zero infon:

\( \text{Infon}_1 \oplus \text{Infon}_2 \oplus \ldots \oplus \text{Infon}_n \)
Possession and Proprietorship of Infons

Information: infon $A$ that is true

$\Leftrightarrow$ zero information, atomic information, compound information

private information: atomic (compound) information: that refers to identifiable individual(s).

Possession of single piece of atomic private information by many possessors and one proprietor

Infons of individuals

Proprietary infons

Possession (infons of others)

Not known (to the individual)

Known by individual

Not known by other individuals

Shared with others by contracts

Known through consent

Public to others
State of Possession of Infons

Consent of possession

State of possession

Awareness of possession

Legality of possession

Constraints to the state of possession

Awareness of the individual

Consent of the individual

Greedy deletion of information

Restrictions on visibility of information impact
Agents and Infons

Agents $A \in \mathcal{Z}$: individuals $\mathcal{V} \cup$ non-individuals $\mathcal{N}$

Individual: natural person       non-individual: artificial person

- **Possesses** $\subset \mathcal{Z} \times \text{PrimaryInfons}$
  
  $\text{Possesses}(A_{100352422729}, \neg \text{Village(Dresden,Saxony)})$

- **Knows** $\subset \mathcal{Z} \times \text{PrimaryInfons}$
  
  compositional

  $\text{Knows}(A_{100352422728}, \text{Possesses}(A_{100352422729}, \neg \text{Village(Dresden,Saxony)}))$

  $\text{Knows}(A_{180753308154},$

  $\text{Knows}(A_{100352422728}, \text{Possesses}(A_{100352422729}, \neg \text{Village(Dresden,Saxony)})))$

- **Belongs** $\subset \mathcal{Z} \times \text{PrimaryInfons}$
  
  $\text{Belongs}(A_{John}, \text{John developed AIDS})$
**Collaboration Based on Infon Tracking**

Local “databases” bound by collaborating views

Diagram:

languages $L_i$ for representing local infon sets

collaboration transfer relation (or function) $r_{ij}$ extended to $r_{ij}^*$ infon sets translating and representing the ability of $j$ to import $i$

collaboration space $(D, r)$ (local databases, transfer relations)

typed formulas on the collaboration space ($i \in I, \alpha \in L_i$)

$$CF ::= i : \alpha | CF \rightarrow CF | CF \land CF | CF \lor CF | \exists i : x.CF | \forall i : x.CF$$
Collaboration Based on Infon Tracking

Local “databases” bound by collaborating views

Communication of infons among agents

based on lifting and bridge rules expressing the result of communication

\[ i_1 : \alpha_i, \ldots, i_n : \alpha_i \quad \frac{}{i : \alpha} \quad \text{application condition} \]

with rigidity properties, e.g. \( D_j \) only knows foreign infons if this is supported

Cooperation as common exploration of infons

Coordination modelled by the coordination space

collaboration formulas as constraints for the collaboration space

mapped to queries that represent validation conditions

e.g. \( \exists i : x.(i : q_1(x) \land x \in CoDom(r_{ij}(D_i)) \land j : q_2(r_{ij}(x))) \)

(shared existence of infons)
Privacy Policies Based on Deontic Logics

Multi-level security with (completely) ordered sets
\{val(o, t, d)|o \in Obj, t \in Time, d \in Dom\}

Junktoren →, ∧, ∨; ¬

Rollenwissen \( K_{A,t}\phi \)
A weiß \( \phi \) zu t

Erlaubnisse \( R_{A,t}\phi \)
A darf \( \phi \) zu t wissen

Obligation \( O_{A,t}\phi \)
A hat Obl. \( \phi \) zu t zu wissen

Verbot \( F_{A,t}\phi \)
A darf \( \phi \) nicht zu t wissen
Privacy Policies Based on Deontic Logics

\[ S = \langle S, O, D, A, T \rangle \text{ (CSP-Version)} \]

O - Objekte (In(put), Out(put), Intern)

T - Zeitpunkte (z.B. ganze Zahlen)

D - Wertebereich (mit Null)

S Menge der Verläufe \( S \subseteq D^{O \times T} \)

A - Subjekte, Rollenmenge \( A \subseteq \mathcal{P}(O \times T) \)
Privacy Policies Based on Deontic Logics

\[ (S, s) \models \phi \text{ für } S, s \in S \]

- \( (S, s) \models \text{val}(o, t, d) \text{ falls } s(o, t) = d \)
- \( (S, s) \models p \land q \text{ falls } (S, s) \models p \text{ und } (S, s) \models q \)
- \( (S, s) \models \neg p \text{ falls nicht } (S, s) \models p \)
- \( (S, s) \models K_{A,t}p \text{ falls } \forall s' \in S(s|_{A \leq t} = s'|_{A \leq t} \Rightarrow (S, s') \models p \)
  
  A weiß \( p \) in Verlauf \( s \), falls dies der Fall für alle ‘äquivalenten’
- \( (S, s) \models R_{A,t}p \text{ falls } \exists X \in R(A, s, t) : (S, s) \models K_{X,t}p \)
  
  A hat die Erlaubnis für \( p \) in \( s \) zu \( t \), falls A kann \( p \) erhalten durch seine autorisierte Rolle \( X \)
  zu \( t \) in \( s \)
- \( (S, s) \models O_{A,t}p \text{ falls } \forall X \in R(A, s, t) : (S, s) \models K_{X,t}p \)
- \( (S, s) \models F_{A,t}p \text{ falls } \forall X \in R(A, s, t) : (S, s) \models \neg K_{X,t}p \)

\( S \) ist sicher für \( A \) falls \( \forall t \in T, \phi \in \mathcal{L} \)

\( K_{A,t}\phi \rightarrow R_{A,t}\phi \) ist \( S \)-wahr
Privacy Policies Based on Deontic Logics

Sicherheitspolitiken

1. Autorisierte Rollen $R(A, s, t)$

$$R(a, s, t) = \{X \in P(O \times T) | \text{Card}(X) \leq \text{Max}(A)\}$$

Maximale Anzahl von erreichbaren Objekten ist begrenzt

2. Ursächlichkeit

Zusammenhang zu Menge von Objekten, die ein Objekt beobachten darf

3. Unabhängigkeit

B unabhängig von A, falls Wissen von A kompatibel mit dem Wissen, daß jeder Input von B Null ist

2. bzw. 3. sind gdw. gegeben, wenn System sicher für A ist

sicher, falls autorisierte Rollen, d.h.

$$R(A, s, t) = \{X \in P(O \times T) | (S, s) \models \neg K_{X,t} \neg (\land_{(o, t') \in B} val(o, t', Null))\}$$

Erfolge: Spezifikation des Bell-LaPadula-Modelles
Extending PES by Reputation

Fischmann Models

Rights for functionality: owners may not change their data, only in collaboration with trusted partners for this functionality

Distribution strategy: based on a network with rights to distribution to others with(out) notification

Reputation models: tribes (subseting), threshold, highest reputation first
  various models for hostility

Explicit request handling: infon demanded by party from party
  \( i \xrightarrow{\text{request}} j \), infon provided to party from party
  \( i \xleftarrow{\text{request}} j \), infon not delivered to party from party on intention
  \( i \not\xleftarrow{\text{request}} j \), infon supply to other agents, infon demand by parties
Enhancing Infons by Transaction Number

Infon-TANs (I-TAN)

- Issue a set of transaction numbers to infons encode
- Similar to DRM2 (digital rights management) of OMA
- Public-key encryption schemata

Combining keys and rights for usage through I-TANs
The Privacy Issue
Known Proposals
Our Target
Policies
The Infon Management
Concluding

The Infon Management Schema for Privacy Support

- Kind of sharing
  - through
  - Contract

- Known by others
  - Known possession

- Agent = Non-individual ∪ Individual
  - possesses
  - Signifies

- Individual

- Infon

Privacy classification schema

- applies
  - true
  - derivable
- Trivial information

Zero: card = (0,0)
Atomic: card = (1,1)
Compound: card = (2,n)

Information

Concept

Topic
The Privacy Issue
Known Proposals
Our Target
Policies
The Infon
Management
Concluding

Views in Infon Possession Management

Main Data Structures

- Zero infon
- Supported usage
- Basic privacy infon
- Derived information
- Shared privacy infon
- Strategy for privacy orchestration

Stage / epistemic truth
Applicable rules / principles
Proprietorship / identification / truth

Kind of privacy / possession property

Individual
Orchestration of Infon Possession Management

\[ \text{Partially a dream} \]

Insertion workflows providing the correct storage of private infons

Access portfolio depending on role, consent, contract, possession, etc. of agents accessing the infons

Star schema architectures for simple and trackable access

Active maintenance workflows depending on the profile of the user, greedy etc. maintenance procedures

Active usage collector providing data to proprietors on possession of infons and usage by other agents

Reasoning agent tracking infon combination or deduction of information by possessors and reporting to proprietors
Development of Strategies for Infon Possession

Independent databases of each agent (may be on its/his own computer)
AH (agent hospital), AP (agent patient), AD (agent doctor)

- Checking that the private information of P and D in the possession of H are correct, consented, etc.
- Informing AP and AD about any new information collected about him.
- Verifying any releases of private information and informs the ‘owner’ of this information about such action. Etc.

Automatic maintenance based on strategies for each agent
AP and AD interact automatically and with AH

- Accessing the H database to check the accuracy, updateability, etc. of the private information of the ‘owner’
- Accessing the database of each other to check the accuracy, updatability, etc. of the private information of the ‘owner’
- Communicating with each other and with AH regarding any action (e.g., releasing) related to the private information of their owner.
General Architecture of Closed Privacy Supporting Systems

- Recording possession of infons
- Tracking usage of infons
- Enabling usage of infons by profiles, portfolio and contracts
- Development of contracts with known consequences
- Rigid separation of databases
Open Systems based on Strategies used for Enforcement

AP checks his PI in H and AH verifies what it has of PI about P

Patient Database

Hospital Database

P

D

H limits P’s view to about D

H limits D’s view to about P

Doctor Database

AP checks his PI in D and AD verifies what it has of PI about P and vice versa

AD checks his PI in H and AH verifies what it has of PI about D.

PI = private information
Open Systems based on Information Wallets

Personal wallet of private information of one person

My private information

In possession of others

My hospital
My doctor
My bank
My work
My club

Others private information

in my possession

About my doctor
About my employee
About my partner
About my neighbors
About my broker

I collected it my self

My software agent downloaded it from the club's database

Send to me by the hospital software agent

The Privacy Issue
Known Proposals
Our Target
Policies
The Infor Management
Concluding
Public presentation of hidden information

Technology for working in unsafe environments

Information security and privacy

Information data security

Information Systems Security

Information hiding

Cryptography

Steganography Watermarking

Whether information is embedded is not visible

Content association by associative images

Portability of content

Variety in folder access key combination

based on pgp delivery and explicit exchange and negotiation
Separation of Concerns Based on the Semiotic Triangle of Content, Concepts and Topics

- **Content**
  - Syntax
  - Semantical unit
  - Semantics
  - Concept
  - Validation
  - Model theory
- **Computation**
  - Computation theory
- **Pragmatics**
  - Pragmatic
  - Asset
  - Presentation
  - Content delivery
- **Topic**
  - Presentation theory
  - Explanation
  - Publication

The Privacy Issue
Known Proposals
Our Target
Policies
The Infon
Management
Concluding
Faithful, consistent and well-founded user-oriented CMS

Well-foundedness:
\[
\text{interpretation(explanation}(t)\text{)} \subseteq \text{delivery}(t) \quad \text{and} \\
\text{presentation(foundation}(cs)\text{)} \subseteq \text{annotation}(cs)
\]
are valid for any topic \( t \) and any content suite \( cs \).

Faithfulness:
\[
\text{interpretation(explanation}(\text{associate}(m))\text{)} \subseteq \text{delivery}(\text{associate}(m))
\]
for any meme \( m \).

Saturatedness:
\[
\text{interpretation(explanation}(t)\text{)} \supseteq \text{delivery}(t) \quad \text{and} \\
\text{presentation(foundation}(cs)\text{)} \supseteq \text{annotation}(cs)
\]
are valid for any topic \( t \) and any content suite \( cs \).

Consistency:
\[
\text{interpretation(explanation}(\text{associate}(m))\text{)} \supseteq \text{delivery}(\text{associate}(m))
\]
for any meme \( m \).
### The data layers of well-founded and saturated CMS

| Layer 0: | Data and documents of underlying databases as micro-data |
| Layer 1: | Content of content bases as macro-data or aggregations |
| Layer 2: | Concepts of concept bases for foundation/explanation |
| Layer 3-4: | Privacy protection layer |
| Layer 3: | Topics of topic landscapes for annotation/representation |
| Layer 4: | Memes of the users |

- The user understands chunks of concepts.
- The user expresses data needs through utterances based on association to topics.
- The user queries for content or data through views.
Facilities of the top-layer of user-oriented CMS

Utterance interpreter and analyzer: analysis of utterances
generation of the appropriate topic landscape

Portfolio manager: derive, manage, change, retrieve and associate portfolio
specific task glossary that supports utterances

Profile manager: storage, retrieval, change, and introduction specific slang-like vocabularies

Meme manager: storage, manipulation, and retrieval

Privacy preserver: preserving privacy by protecting ownership against possessors
Proposal for an architecture of user-oriented content management systems

Towards this century CMS

Web Playout System
- Story Space
- Stories
- Scenarios
- Actors
- Context

User Management System
- Profile manager
- Portfolio manager
- Association generator
- Natural language engine

Privacy Protection System

Content Management System
- Content types
  - Structure
  - Functionality
  - Container
- Structuring
  - Structure
  - Static IC
  - (Pragmatics)
- Functionality
  - Processes
  - Dynamic IC
  - (Pragmatics)

Topic Management System
- Topic manager
- Community manager
- Asset manager
- Infon representer

Concept Management System
- Concept manager
- Derivation engine
- Unit manager
- Infon representer

Data-base

Content base

Topic landscape

Concept base

The Privacy Issue
Known Proposals
Our Target
Policies
The Infon
Management
Concluding
Summarizing and Concluding

- Privacy management is achievable
- Privacy management is expensive
- The simplest way for privacy management: closed systems
- Formalisms for possession, consent, privacy policy rules, playout rules
- Development of security meta-profiles, proprietor information facilities, infon tracking functions, encryption mechanism
- Development of generic model management facilities for privacy enhanced infrastructures
- Privacy-aware individuals
Next Steps

- Providing a formal foundation for principles, policies, contracts, portfolio, profiles, collaboration (based on ASM)
- Development of a personal information theory based on separation of creation, collection, processing, and disclosing
- Common aspect and differences of privacy and security
- Case studies for health information systems and e-learning systems
- Theory of shared knowledge of infons (bridges and lifting rules)
- Theory of cooperation among privacy enhanced systems
- Development of privacy enhanced systems in partially hostile environments
- Proposal for e-government privacy enhanced systems
Privacy Enhancing System

Questions?
Query and Answer Forms for Sophisticated Database Interfaces

Sophisticated NoSQL Questioning of a Database in Native Form
EJC 2012
5.6.2012

Markus Berg¹ & Antje Düsterhöft¹ & Bernhard Thalheim²,∗
(1) Hochschule Wismar, Department of Electrical Engineering and Computer Science, Germany
(2) Kiel University, Germany
(∗) Kolmogorow Professor h.c. of Lomonossow University Moscov
Plan for this Talk

Observations for the current state-of-art
  Trapped by SQL and database schemata
  Being limited for formulation, understanding, culture

Systematic querying by reconsidering search
  Property-based search is the toughest form!

Query forms as a framed form for query formulation

Answer forms as a way of deriving the format of the answer

Query formulation from questions
  SQL users have to state queries in the SQL form!

Question liquefaction for generation of queries
SQL Query Generation

**Static query interfaces**

No or very restricted NL access

**Simplicity of query interfaces**

**Problematic IR solutions**

ER querying is better than relational but not widely used

**Problematic database evolution**

Classical relational approach

- Relational database schema
- NL utterance
- SQL query

Enhanced relational approach

- Relational database schema
- ontology / thesaurus
- NL utterance
- SQL query
SQL is Easy to Read, to Develop and to Understand? Of Course, for Everybody!!!

What does this query?

```
SELECT P1.Name, P2.Name
FROM Person P1, Person P2, Student S1, Student S2, Enrol H1, Enrol H2
WHERE P1.Name = S1.Name AND P1.DateOfBirth = S1.DateOfBirth
AND S1.StudNo = H1.StudNo AND H1.Grade IS NOT NULL
AND P2.Name = S2.Name AND P2.DateOfBirth = S2.DateOfBirth
AND S2.StudNo = H2.StudNo AND H2.Grade IS NOT NULL
AND NOT EXISTS
  (SELECT *
   FROM Enrol H3
   WHERE H3.Grade IS NOT NULL AND
     H3.StudNo NOT IN
     (SELECT H4.StudNo
      FROM Enrol H4
      WHERE H4.StudNo = H2.StudNo
        AND H4.Grade IS NOT NULL)
AND NOT EXISTS
  (SELECT *
   FROM Enrol H5
   WHERE H5.Grade IS NOT NULL AND
     H5.StudNo NOT IN
     (SELECT H6.StudNo
      FROM Enrol H6
      WHERE H6.StudNo = H1.StudNo
        AND H6.Grade IS NOT NULL)
     AND H2.StudNo = H5.StudNo)
AND S1.StudNo < S2.StudNo
GROUP BY P1.Name, P2.Name;
```
We are Humans!!

and thus we are

limited in our formulation capabilities what hampers the user,

limited in abstraction skills what limits jumping into somebodies context,

not keen to learn database schemata which have been created by somebody whom we do not know or understand,

not able to read exhaustive result sets what requires sophisticated presentation, visualisation and compactification,

finite and bounded what means that we need support for parsimony of our memory, and

not able to guess meta-data such as quality, timeliness, actuality, ...
Querying with a topic-based, concept-backed, user-oriented CMS

Not trapped in the SQL trap

State-of-art
Systematics
Query forms
Answer forms
Question2Query
Liquefaction
Conclusion

Tina Musterfrau, casual user

Search request

topic welt concepts

search concept

result concept

parametric HERM expressions

query form

answer form

relational database schema

SQL query

SQL query set

answer for search

DBS

DBMS query representation

user in the DBMS trap

help !! help !!

Query & Answer Forms
5.6.2012
Berg/Düsterh./β

Content
Information
Concept Topic
Systematical Querying

Traditional database querying

<table>
<thead>
<tr>
<th>input</th>
<th>SQL query</th>
</tr>
</thead>
<tbody>
<tr>
<td>process</td>
<td>SQL answer set</td>
</tr>
<tr>
<td>output</td>
<td>DBMS answer representation</td>
</tr>
</tbody>
</table>

Linguistic search facilities

<table>
<thead>
<tr>
<th>map</th>
<th>query form</th>
</tr>
</thead>
<tbody>
<tr>
<td>compile</td>
<td>SQL query</td>
</tr>
<tr>
<td>map</td>
<td>answer form</td>
</tr>
<tr>
<td>process</td>
<td>SQL answer set</td>
</tr>
<tr>
<td>output</td>
<td>answer to search</td>
</tr>
</tbody>
</table>
Kinds of Search Features Applicable to Types

Search by main properties: weighted high in the star schema

Fuzzy search generalization of domain values and similarity values. *SoundEx*

Search by associations: step-wise scoping, refinement and narrowing; its context

Search by meta-properties: space, time, history of objects and database, profiles of actors, specific data types, specific constraints

Search on the basis of the utilization record: The search engine can record results of previous search request, the story space of a group of users or log file

Search through browsing: the entire set of objects is scanned on the basis of some main properties
Search Combined with Control Approaches

- *depth-first search* (developing each type completely before moving to the next type),

- *hill-climbing search* (using a selection function and a heuristic function in order to determine the next best local step),

- *breadth-first search* (developing all types to a certain extent before moving to the next reification),

- *beam-search strategy* (same procedure as breadth-first-search but with the use of heuristic functions to select the next types), or

- *best-first search* (developing the best unexpanded type as far as possible using a general control function and a general selection function).
Deriving The Navigation Search From Associations

- Traverse uni-directional associations
- Traverse qualified association
- Traverse generalizations/specializations
  - Upwards traversal
  - Downwards traversal
  - Obtain the XML object
- Traverse from link to object
- Traverse from object to link
  - Link collection
  - Traversal by roles
- Filter objects
- Filter links
- Traverse from object to value
- Traverse from link to value
Context-based Retrieval from the Web

Context capturing performed at the client side software. It is based on correlation-basic metrics for similarity and may use advanced dictionaries, e.g., WordNet.

Keyword extraction from the captured text and context based on clustering algorithms.

High-level classification of the query to a small set of predefined domains. The ontology object may be applied to a set of search engines, may be ranked by their relevance and coverage depending on the keyword set.

Ordering and adhesion of query results is obtained from different search engines by reranking with distance measures, adhesion, and cohesion functions.

Context-based retrieval is a variant of ‘blind’, non-informed search. It may be enhanced by search algorithms, e.g. the A*-algorithm.
The $W^7(+W^4+W^{17}H)$ Question Frame

- **matter** (what, concepts, in what way)
- **situational context** (when, where, in what means).
- **user profile** (who) and **user portfolio** (wherefore, wherein, where, for what, wherefrom, whence, what)
- **carrier language** (wherewith) within a **namespace** (whereto, by what means).
- **answer solution characteristics** (how, why, whereto, when, for which reason),
- **solution context embedding** (whereat, whereabout, whither, when)
- **surplus value** (worthiness) of the answer.
Query Forms

The more general and far simpler form of queries

(question content, matter (concepts, situation),
user(profile, portfolio), carrier).

parametric view expression \( expr(T_1, \ldots, T_n, x_1, \ldots x_m) \)

graph of query notions
graph can be extended to the given DB schema through homomorphic embedding

definable by Visual SQL
see example below

embedding through graph grammar formalism
with integrity constraints
Answer Forms

Any question contains also the expected answer format.

(\textit{answer content, solution (characteristics, context, value)})

Visual SQL schema

query form

SQL query

SQL answer set

answer form

answer to search

DBMS answer representation

SQL database schema

DBMS query interface
Six Steps From Question to Query

1. Extension of the Search Question
2. Orthonormalisation and Extension of the Search Question and Mapping to Query Forms and Answer Forms
3. Rephrasing of the Question into an Existential Form
4. Mapping of the Query Form to Database Schema Notions
5. Derivation of the Extended Answer Form
6. Derivation of the Database Query
1: Extension of the Search Question

Which students occur only together?

- extend by issuer’s context,
- extend by community of practice common sense,
- resolve ambiguities,
- use issuer semantics, e.g. for connectives,
- resolve ellipses, and
- add scope and issuers.

... doing same things at the same time and with the same success?
2: Orthonormalisation and Extension of the Search Question and Mapping to Query Forms and Answer Forms

Which students occur only together?

- categorise by the $W^7(+W^4+W^{17}H)$ frame
- orthonormalisation
- connectives interpretation
- abbreviations
- matter, own concepts, aggregates
- profile and portfolio of the issuer, data on demand as information demand
- query form graph
- answer forms graph

Which students complete the same courses in the same term?
Connectives and Quantifiers in Reality?

Different truth definitions

Material, logical, and normative connectives, e.g. implication

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi \rightarrow \phi$</td>
<td>$\phi$</td>
<td>$\psi \rightarrow \phi$, $\psi \supset \phi$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>??</td>
</tr>
</tbody>
</table>

Generalisation operators e.g. $(t,f)$-quantifier $Q_{t,f}$ with validity dependence of $Q_{r,s}\alpha(x)$ in structure $A$ such that

$$|\{o \in \pi_x(dom(A)) | I_x^o(\alpha) = 1\}| = t$$

$$|\{o \in \pi_x(dom(A)) | I_x^o(\alpha) = 0\}| = f$$

classical $\forall \equiv Q_{s,0}$, $\exists \equiv Q_{t,*}$ for $t \geq 1$, $Majority \equiv Q_{n+k,n}$, $k, n \in \mathbb{N}^+, k \geq 1$

Models for the knowledge operator $K_A$ for actors $A$
Query and Answer Forms

Which students occur only together?

Query form:
- Student \(\rightarrow\) completes \(\rightarrow\) Course
  - Name etc.
  - ALL

Answer form:
- Student \(\rightarrow\) Name etc.
  - Term
  - ALL \(\neq\) ALL

Query and answer forms as potentially homomorphically embedable graph.
3: Rephrasing of the Question into an Existential Form

Which students occur only together?

- $\forall$-sentence transformation
  
  $\neg \exists v ((\neg Enrol(a, v) \lor \neg Enrol(b, v)) \land (Enrol(a, v) \lor Enrol(b, v)))$
  
  =
  
  $\forall v ((Enrol(a, v) \land Enrol(b, v)) \lor (\neg Enrol(a, v) \land \neg Enrol(b, v)))$

- connectives transformation
- handling negation
- canonical set representation
- injection into query and answer forms
- null value resolution

... so that does not exist a course that is not taken by the other?
4: Mapping of the Query Form to Database Schema Notions.

Which students occur only together?

far simpler and easier to formulate, to capture, to understand without the SQL burden
5: Derivation of the Extended Answer Form

Which students occur only together?

- parameterisation
- storage alternatives
- answer representation style, e.g., Venetian blind for XML
- add context
- map tu question issuer’s language
- extend by features for visualisation, representation
- provide functions for marking, drill-down, roll-up, slice, dice, rotate, refinement, new query issuing, export, session storage, and reuse

... nonsymmetric name pairs ordered by corresponding StudNo...
6: Derivation of the Database Query

Which students occur only together?

- correct formulation
- consider the kind of SQL
- adapt to DBMS profile, facilities
- provide query hints
- derive query using integrity constraints
- consider DBMS and user-defined types
- decompose - if necessary - to views and combination query
- handle NULL
- consider materialisation of sub-results for answer form instantiation

... see next slide ...
The Resulting Query

SELECT P1.Name, P2.Name
FROM Person P1, Person P2, Student S1, Student S2, Enrol H1, Enrol H2
WHERE P1.Name = S1.Name AND P1.DateOfBirth = S1.DateOfBirth AND
S1.StudNo = H1.StudNo AND H1.Grade IS NOT NULL AND
P2.Name = S2.Name AND P2.DateOfBirth = S2.DateOfBirth AND
S2.StudNo = H2.StudNo AND H2.Grade IS NOT NULL
AND NOT EXISTS
  (SELECT *
   FROM Enrol H3
   WHERE H3.Grade IS NOT NULL AND
     H3.StudNo NOT IN
     (SELECT H4.StudNo
      FROM Enrol H4
      WHERE H4.StudNo = H2.StudNo
      AND H4.Grade IS NOT NULL)
AND NOT EXISTS
  (SELECT *
   FROM Enrol H5
   WHERE H5.Grade IS NOT NULL AND
     H5.StudNo NOT IN
     (SELECT H6.StudNo
      FROM Enrol H6
     AND H2.StudNo = H5.StudNo)
AND S1.StudNo < S2.StudNo
GROUP BY P1.Name, P2.Name;

How long would it take you to formulate this query?

Is the first query in this talk equivalent to this query?
Question Liquefaction

The Three-Step Approach to Automatic SQL Query Generation

Generation of SQL query candidates based on full information

- Ontology / WordNet / thesaurus
- NL utterance
- Relational database content
- Database schema in extended ER model
- Enriched syntax tree
- Proper name candidates
- Priority-ordered paths in the extended ER schema
- Translation style used for compilation of relational schemata from HERM schemata
- Priority-ordered set of SQL query candidates
The Cottbus Intelligent NL Request Transformer

Query & Answer
Forms
5.6.2012
Berg/Düsterh./β

State-of-art
Systematics
Query forms
Answer forms
Question2Query
Liquefaction
Conclusion

Content
Information
Concept
Topic

Query
Liquefaction

NL query
Web Input

Ontology
WordNet

RADD DB Design Tool
DB Thesaurus Manager

DB Schema Manager
(e)ER Schema

ER2R Translation Style
ISL

DBMain DB Design Tool

Database Management System
DB2Web System

Web Presenter

Syntax tree
Intelligent Path Extractor

Paths in ER Schema
Relational Query Melting-Pot

Syntax Analysis

Paths and SQL queries

Database
Example: Lecture Scheduling

The full schema

Query
& Answer
Forms
5.6.2012
Berg/Düsterh./β

State-of-art
Systematics
Query forms
Answer forms
Question2Query
Liquefaction
Conclusion

Content
Information
Concept Topic
Example NL Analysis

Which lectures are given by Vierhaus and Thalheim?

```
s [praes]
  np [akk,plural,3,noun]
    quant [akk,plural,fem,finit]
    welch [akk,plural,fem,finit]
  n [akk,plural,fem,finit,3,noun]
    noun [akk,plural,fem,finit,3]
    Veranstaltung [akk,plural,fem,finit,3]
  vp [[np,[nom,plural,3,noun]],plural,,3,praes]
    v [vf,[noaux,haben],[finit,nopsp],plural,3,praes,noprae]
      les [vf,[noaux,haben],[finit,nopsp],plural,3,praes,noprae]
  connp []
    np [gen,sing,3,noun]
      n [gen,sing,mas,infinit,3,noun]
      noun [gen,sing,mas,infinit,3]
      tktktk [gen,sing,mas,infinit,3]
  conn []
    und []
  np [akk,plural,mas,infinit,3,noun]
    n [akk,plural,mas,infinit,3,noun]
    noun [akk,plural,mas,infinit,3]
    tktktk [akk,plural,mas,infinit,3]
```
The Resulting Query

Which lectures are given by Vierhaus and Thalheim?

```sql
au tk 12 (.../diplom/SQL-Generator) ; echo "Welche Veranstaltungen lesen Thalheim und Vierhaus." 1
". /src/syntax-analyse/sql-gen
Anfrage: Welche Veranstaltungen lesen Thalheim und Vierhaus.
1. Pfad: Veranstaltung les Thalheim

2. Pfad: Veranstaltung les Vierhaus
```
Plan and Achievements for this Talk

Observations for the current state-of-art
- Trapped by SQL and database schemata
- Being limited for formulation, understanding, culture

Systematic querying by reconsidering search
- Property-based search is the toughest form!
- Extension of search forms

Query forms as a framed form for query formulation
- Questions are anyway stereotyped.
- Use the stereotype for query generation.

Answer forms as a way of deriving the format of the answer
- Questions contain partially the answer format.
- Use the answer format for answer stereotypes.

Query formulation from questions
- SQL users have to state queries in the SQL form!
- Why we should not support the user?

Question liquefaction for generation of queries
- Automatic query decomposition, liquefaction and composition.
- Natural language approaches to generation.
Summarising

Systematic question transformation +
  automatic query generation +
  automatic answer delivery

Query formulation
  as a six-step procedure

Query and answer forms
  for orthonormalised questions and
  for any kind of question

Tools as a proof-of-concept
  with applications in everyday life

VisualSQL
  as the better form for query formulation
  without the SQL burden

http://www.informatik.uni-kiel.de/en/information-systems-engineering/miscellaneous/visualsql/
Thank you!
There is no common understanding of model.
Thus there is no notion of a model.
Conceptual modelling cannot have a theory.
Entity-relationship modelling language is sufficiently general.

thalheim@is.informatik.uni-kiel.de
Syntax, Semantics and Pragmatics of Conceptual Modelling

17th International conference on Applications of Natural Language Processing to Information Systems
NLDB 2012, Groningen

27.6.2012

Bernhard Thalheim
Technologie der Informationssysteme
Institut für Informatik, Christian-Albrechts-Universität zu Kiel, BRD
Kolmogorow-Professor e.h. der Lomonossow-Universität Moskau
Conceptual Modelling of Information Systems - What is That?

Conceptual Modelling Survey

Conceptual modelling with the aspects structuring + functionality + interactivity models + concepts (from the application domain with semantics and pragmatics) compositionality incremental building

Extended ER modelling language: as language-based modelling being aware of \( \frac{5}{6} \)-cognitive completeness (what is better than \( \frac{1}{3} \)-completeness of classical ER or UML but ...) strictly hierarchically typed schemata

... but what is now a model?
Models we are Not Going to Consider

Combinatoric-regular polyeder (Dycks map)

the usual joke

Model used as a form for production

Models for replication: Nofretete
**Brief Survey**

More than 50 different notions of a model already in Computer Science.

There is not a chance to find a notion of a model.

Models do not exist on their own!

Language-based modelling inherits limitations of languages

Limited expressiveness and living with Sapir-Whorf.

Language-base modelling can be based on semiotics.

Models are used in different settings with different goals.

They are different in scope, focus, validity, time, ...

Differentiate goal, purpose, function of a model.

Each user perceives a model differently and thus interprets somehow.

There cannot be a common culture for modelling.

Provide adaptation facilities.

Too many different model deployments due to variety of disciplines.

There is no systematic way.

Learn from humans how they master description.
Central Property: Invariance of Purpose

Deep Understanding of the Purpose

<table>
<thead>
<tr>
<th>application</th>
<th>origin</th>
<th>model 1</th>
<th>model 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>application domain with problem</td>
<td>city Königsberg with bridges</td>
<td>rough topographical model with nodes (area) and edges (bridges)</td>
<td>graph with degree of incidence</td>
</tr>
<tr>
<td>application domain with problem</td>
<td>city Königsberg with bridges</td>
<td>rough topographical model with nodes (area) and edges (bridges)</td>
<td>tree scanning</td>
</tr>
</tbody>
</table>

Model 2^A: an Euler path exists iff \( \text{card} \{ n \mid \text{degree}(n) \text{ is odd} \} \leq 2 \)
And what about the general case? Write a program and verify!!!
Separation of Goal, Purpose and Function of Models

Goal: current and target state for a community of practice
Purpose: realisable goals
Function of models within (intellectual) processes

A model is simply

- a material or virtual *artifact*
- which is called model within a community of practice
- based on a judgement of appropriateness for representation of other artifacts (things in reality, systems, ...) and
- either adequate for the goals or fit for its *purpose*
  or useful for the function within this community.
Spectrum of Notions of a Model: IS Case

Stachowiak’s general properties: mapping truncation, pragmatic properties
enhanced by amplification, distortion, idealisation
restricted by carrier and driven by added value

Axiomatic approach: formal system with specific foci and scope, reality-driven

Mapping-oriented approach: reality as origin, model system as mediator, homomorphic interpretation, goals of the subjects (CoP)

Construction-oriented approach: state transformation from current to improved, construction of the CoP engaged, CoP language, CoP focus and scope

Goal, purpose and function is underestimated
An Environment for the Notion of a MODEL

Grounding $G$: concepts, foundations, language as carrier, cargo

(Meta-)Basis $B$: basement, paradigms, theories; status in application; context; paradigmatic evolution; abstraction, scale

Deployment $D$: goal, purpose, function; reason

Community of practice $P$: stakeholder with their roles and plays, with their interests, portfolio and profiles

Context $C$: time, space, scope

Quality $Q$: correctness (...), generality (...), usefulness (...), comprehensibility (...), parsimony, ..., robustness, novelty (...)

Viability $V$: corroboration, coherence, falsifiability, stability, assurance(restrictions, modality, confidence)
A Notion of a MODEL

Given a collection of artifacts $\mathcal{M}^*, \mathcal{M}_1, ..., \mathcal{M}_k$, a community of practice $\mathcal{P}$ ($\subseteq \mathcal{P}$), a grounding $\mathcal{G}$ ($\subseteq \mathcal{G}$), viability $\mathcal{V}$ ($\subseteq \mathcal{V}$), bases $\mathcal{B}$ ($\subseteq \mathcal{B}$), context $\mathcal{C}$ ($\subseteq \mathcal{C}$), and deployment $\mathcal{D}$ ($\subseteq \mathcal{D}$).

The artifact $\mathcal{M}^*$ is called a \textbf{model} for $\mathcal{M}_1, ..., \mathcal{M}_k$ by $\mathcal{P}$ for $\mathcal{D}$ with $\mathcal{V}$ if it is appropriate with $\mathcal{Q}_a \cup \mathcal{Q}_f \cup \mathcal{Q}_u$ and within $\mathcal{C}$ based on $\mathcal{B}$ using $\mathcal{G}$, i.e.,

- it is adequate (has potential for goals) [similar + regular + fruitful + simple] according to the relation between $\mathcal{M}^*$ and $\mathcal{M}_1, ..., \mathcal{M}_k$ at level of $\mathcal{Q}_a$ for $\mathcal{D}$ with $\mathcal{V}$ within $\mathcal{B}$ and grounded by $\mathcal{G}$,
- it is fit for $\mathcal{D}$ with $\mathcal{Q}_f$ within $\mathcal{C}$ and compliant with $\mathcal{G}$ and $\mathcal{B}$ and
- it is useful for $\mathcal{P}$ within their $\mathcal{D}$ and at level of $\mathcal{Q}_u$.

Model suite: $\mathcal{M}_1^*, ..., \mathcal{M}_n^*$ instead of $\mathcal{M}^*$
...out of scope of the definition ...

to model as the construction process; artifact can be constructed constructive models versus immaterial models versus virtual models versus imaginations

recognition of a stakeholder see however assessment as more objective element

homomorphic image as specific quality criterion see however similarity as element of adequacy
Quality of a Model and Modelling

Assessment

Correctness

- Probable $\Omega_f$
- Validatable $\Omega_a$
- Testability $\Omega_f$
- Provability $\Omega_u$
- Conditions of validity $\Omega_f$
- Range of validity $\Omega_f$
- Power for reasoning $\Omega_f$
- Relevance $\Omega_f$
- Realisability $\Omega_u$

Generality

- Simplicity $\Omega_f$
- Coherence $\Omega_a$
- Clarity $\Omega_f$
- Parsimony $\Omega_f$

Usefulness

- Not deductable $\Omega_u$
- Unexpected $\Omega_u$
- Previously unknown $\Omega_u$

Comprehensibility

Novelty

- parsimony: the quality of being careful with resources

Why
Model
Semiotics
Syntax
Semantics
Pragmatics
$W^*H$
Finally

Content

Information

Concept
Topic
Stereotypes for Disciplines

kind of model $M^*$

artifacts $M_1, \ldots, M_k$:

deployment $\mathcal{D}$

CoP $\mathcal{P}$:

viability $\mathcal{V}$:

basis $\mathcal{B}$:

grounding $\mathcal{G}$:

quality $\mathcal{Q}$:

context $\mathcal{C}$:
Stereotypes for Disciplines: Information Systems

kind of model $M^*$ as a mediator that is later refined, extended, corrected

artifacts $M_1, M_2$: reality, augmented targeted reality

deployment $\mathfrak{D}$ for construction of a solution to application domain problems
(either as business system or as embedded system)

CoP $\mathcal{P}$: modeller, programmer, application specialist

viability $\mathfrak{V}$: empirical corroboration by realisability, rational coherence as observable property, falsifiability by validation, often neglected stability

basis $\mathfrak{B}$: von-Neumann-Machine, DBMS architectures, type systems, object-relational systems, model as intermediate and later neglected mediator, finiteness, constructivity, compositionality

grounding $\mathfrak{G}$: ER/UML/.../programming language, application domain concepts, biases of modellers

quality $\mathfrak{Q}$: correct (validatable, testable), partially complete, general (within time/space scope), useful (realisable), novel (previously unknown), comprehensible (simplicity, coherent, clear)
Stereotypes for Disciplines: Mathematics of Demography

kind of model $M^*$ evolution functions/matrices

artifacts $M_1, ..., M_k$: population, observed data

deployment $\mathcal{D}$ within the Mathematical modelling workflow, prognosis

CoP $\mathcal{P}$: application specialist, mathematician

viability $\mathcal{V}$: depending on level of detail,

basis $\mathcal{B}$: ‘religious’ beliefs/ideology, many assumptions (nothing changes anymore, static, behaviour, homogeneous population, linear dependence, human limitation, medical progress, reactive behaviour) black/grey/white-box approach, background theories (e.g., Kolmogorow), representability through variables

grounding $\mathcal{G}$: Eigenvalue theory for matrices, Leslie processes, estimated parameters, Gauss/.. distribution, approximation, specifics of application concepts, causality

quality $\mathcal{Q}$: mathematically correct (without contradictions), true, suitable, similar, invariant problem/solution mappings, general (also to other applications), as simple as possible and necessary
Stereotypes for Disciplines: Metaphors as Models

kind of model $\mathcal{M}^*$ transformational, partiality, typicality
with internal, predicative, heuristic, emotional, social, rhetoric and esthetic functions

artifacts $\mathcal{M}_1, \ldots, \mathcal{M}_k$: difficult to describe variety of complex and partially known things under consideration

deployment $\mathcal{D}$ brief annotation in combination of a combination of metaphors; fuzzy description of some properties

CoP $\mathfrak{P}$: speaker/hearer mode; personalisable, allegoric, symbolic, synoptic transfer of meaning

viability $\mathcal{V}$: partial correctness, ideomatic summarisation

basis $\mathfrak{B}$: lexical semantics; ordered compositional combination; user-centred abstraction

grounding $\mathfrak{G}$: mental theories, abilities of hearer/speaker, common sense of meaning

quality $\mathfrak{Q}$: simple interpretation, parsimony, substitution,
# Parameters for being a model

<table>
<thead>
<tr>
<th>kind of model model</th>
<th>artifacts ( M_1, \ldots, M_k )</th>
<th>deployment ( D )</th>
<th>CoP ( \mathfrak{P} )</th>
<th>viability ( \mathfrak{V} )</th>
<th>basis ( \mathfrak{B} )</th>
<th>grounding ( \mathfrak{g} )</th>
<th>quality ( \mathfrak{Q} )</th>
<th>context ( \mathfrak{C} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>conceptual DB model (DB schema)</td>
<td>k=2: application domain, realisation</td>
<td>G+F: construction of augmented reality</td>
<td>modeler, implementer</td>
<td>*</td>
<td>CS &amp; IS paradigms</td>
<td>concepts, CM languages</td>
<td>similarity, ...</td>
<td>R-DBMS, data, current, branch</td>
</tr>
<tr>
<td>IS/DB mediator model</td>
<td>reality under consider.</td>
<td>IS construction</td>
<td>developer, ...</td>
<td>realisable</td>
<td>CS assumptions</td>
<td>concepts, IS languages</td>
<td>correct</td>
<td>application domain</td>
</tr>
<tr>
<td>mathematica evolution model</td>
<td>observed dynamic datasets</td>
<td>Math workflow</td>
<td>expert, mathematician</td>
<td>depends</td>
<td>1001 backgrounds</td>
<td>math. lang. causality theory</td>
<td>true, suitable</td>
<td>application domain</td>
</tr>
<tr>
<td>metaphor injection</td>
<td>reality</td>
<td>annotation</td>
<td>speaker/hearer ideomatics</td>
<td>lexical semantics</td>
<td>mental properties</td>
<td>parsimonic</td>
<td>current li-festream</td>
<td></td>
</tr>
<tr>
<td>biological function model</td>
<td>human, manipulated mouse as substitute</td>
<td>evolution simulation for diseases</td>
<td>biologist, pharmacists</td>
<td>observation, falsifiability, assurance</td>
<td>disease development, experimentation</td>
<td>gen injection, genetic belief, biology, human-mouse similarity</td>
<td>robust, main effect</td>
<td>instruments</td>
</tr>
</tbody>
</table>
Language Matters

Global versus local: Petri nets as a local representation pitfalls in Michael Jackson’s crossroad example

State-oriented versus event-oriented representation

Conceptual completeness of the language

Users, users, users are different in their (visual, oral, literacy) communication skills, (visual, oral, literacy) cognition and thus require different design

Hearer, speaker, archiver views
A Picture Might Replace a Full Subsection

The database aims in supporting lecture and course scheduling within a university application. A course is typically proposed. If the proposal becomes accepted then the course is going to be planned. Typically planned course may be also held. The course proposal, planning and organisation is bound to a semester. Some university employees may be responsible for a course. Typically a course has one responsible person. Responsibilities may vary by semesters. Courses are taught by professors. Professors are specialisations of a person.

Courses may be given for various programs. The proposal also includes the assignment of a course kind. The course proposal typically also requests for a room and for a time at which the course preferably could be given. Additionally time slots may be in conflict with other proposals. Therefore, conflicting time slots are given as well. The room and time preference may be overwritten during the planning phase. The same opportunity exists for proposals for the kind of the course to be given. If the time is assigned then typically a time slot is assigned. A course may have several non-overlapping time slots.

The proposal for a course should be recorded. A person may act in the role of somebody who inserted the course.

Finally, courses may also be held at a different location than originally planned.
Compare Visual Representation with Textual One

Is this better to read for users of ER diagrams?

For more information: http://www.is.informatik.uni-kiel.de/~thalheim/slides.htm
Modelling + Concepts $\iff$ Conceptual Modelling

1. Typicality of the feature: (typical, moderately typical, atypical, borderline)
   - necessary feature
   - sufficient feature (in relation to other features), commonality of features
   - measures of typicality, weights
   - goodness of prototypes

2. Relevance of a feature

3. Importance of a feature:
   - recognised
   - used
   - frequency of occurrence, number of individuals, effect of ideals

Constructing expressions: features with certain ordering
   - hierarchical structuring
   - containment relations (concept contains concept/knowledge): is_a, has_a_component, contains_another_concept

   driver of car is a person, car contains engine, car is red

G.L. Murphy, The big book of concepts. MIT press
Views the Concept of Concept

- **Prototype view:** representation through best example or examples with some summary representation, partially with special hint on importance of special properties
  
  prototype and computational explosion
  
  Schema: KL-ONE-alike \((frame(slot,filler \in Dom(slot)))\)

- **Exemplar view:** concept of a person = collection of things a person remembers

  Characterisation through similarity relation with measures, weights, stimuli for their acceptance

  multiplicative rules for typicality

  see also Concept lattice (Ganter/Wille)

- **Knowledge view:** concepts are the basic items of knowledge assuming that concepts are consistent

  coexistence and co-evolution of concepts and knowledge

  concepts learning through knowledge and concepts obtained before build a language, a theory, a reasoning system
Semiotics of Models

**Syntax**
- well built
  - easy to express

**Semantics**
- well defined
  - easy to understand

**Usage**
- well applicable
  - easy to apply

**Application**
- well supported
  - of high value

It is written: “In the beginning was the Word!”

Even now I balk. Can no one help?
I truly cannot rate the word so high.
I must translate it otherwise.
I believe the Spirit has inspired me
And I must write: “In the beginning there was Mind.”

Think thoroughly on this first line,
Hold back your pen from undue haste!
Is it mind that stirs and makes all things?
The text should state: “In the beginning there was Power!”

Yet while I am about to write this down,
Something warns me I will not adhere to this.
The Spirit’s on my side! The answer is at hand:
I write, assured, “In the beginning was the Deed.”

Goethe, *Faust I*, Faust’s Study
Separation of Concerns Based on the Semiotic Triangle of Content, Concepts and Topics

Conceptual Modelling
Semiotics
27.6.2012
B. Thalheim

Why Model
Semiotics
Syntax
Semantics
Pragmatics
W*H
Finally

Content
Information

Computation theory
Computation
Syntax

Semantics
Pragmatics

Asset
color: #333333
content delivery

presentation
explanation

Topic

Model theory

Interpretation

Validation

Foundation
Components of a (Meta-Semiotic) Logics

- **Syntactic constituent:**
  which symbols are going to be used in the language, which combinations (words) of these symbols are allowed, which constructors can be (partially) used if there is a chance that expressions can be constructed inductively

- **Semantic constituent:**
  what is the purpose of the language, which structures are of interest, what is going to be expressed

- **Constituent that relates syntax and semantics,** e.g. definition of truth (or appropriatedness):
  which structures or expressions are true or meaningful or potentially meaningful?

- **Pragmatic constituent:**
  which meaning can be canonically assigned to words; which restrictions must be considered, which closure operators are applicable
typically finiteness assumption
The “Playground” of Logics

see too: Semantics in DB & KB

Signature of a language and structures: for description
assumption: signature embedding of structures

Construction of the language based on constructors for words
assumption: inductive construction
special assumptions for inherently cyclic language constructs

Conceptionalisation of structures with quality properties, e.g.
“true”
assumption: observability of properties
assumption: canonical truth values

Association of language and structures by interpretation of language constructs with structural properties
assumption: canonical association
**Syntax, Semantics, Pragmatics**

**Morphology:** construction of words, lexem-based, pre-/post-association, adds

**Syntax:** construction of expressions, structural description
subject, predicat, objects, schemes and pattern

*What is the green in the boil of soup?*

**Semantics:** formal description of the logical meaning in dependence on propositional content

**Semantik:** (Bedeutungslehre) formale Beschreibung des propositionalen Gehalts ((green(x), boil(Soup(y), in(x,y)) , ?)

**Pragmatics** depending on circumstances
utilisation depending on communication situation and context
intended function: information, praise, critics, ...

potential answer: *If you don't like it, then goto MacDonalds!*
**Special Features of Natural Languages**

- Semantically fixed base elements (vocabulary), general purpose means for expression (grammar, reference)
- Manifoldness of expression forms and speech acts
- Primarily oriented to actual association among things, situations and characteristics
- Objectified abstraction and generalisation
- Metalanguage on its own
- High error tolerance
- No disction between abstraction levels (true\textsuperscript{content}, true\textsuperscript{concept} (metalanguage), true\textsuperscript{symbol} (intensional))
- Variable interpretation of connectives and quantifiers

\begin{align*}
\text{‘Julia found a unicorn’} & = \text{true} \Rightarrow \exists \text{ unicorn} \\
\text{‘Julia searches for a unicorn’} & = \text{true} \not\Rightarrow \exists \text{ unicorn}
\end{align*}
Specifics of Natural Languages

Ambiguities lexical (bank), structural (word or sentence structure), referential (flats of landlords and their addresses), pragmatical (Could you please ...? Yes!), semantical (IC, die height of hills in Groningen)

Indefiniteness as a concept for later focusing (many, some, (almost) all, in, with, have, from)

Ellipses for shortening: intra-sentence ellipses (forward/backward (Most cheapest and expensive flats)), compositional ellipses (mentioning before, 1. flats with shower; 2. with bath (substitution); 3. at main streets (add-on))

Tempus, modality, metaphoric with embedment of context, coding, ... (The corridor looks like a tube.)
The Place of Semiotics

D. Nauta, Jr. The Meaning of Information, Mouton, 1972, p. 170

Why Model
Semiotics
Syntax
Semantics
Pragmatics
W*H
Finally
Semiotics of Models: Syntax: Morphology

- science of word form structure.
- classified according to their categories and roles within a model and according to their specific expression within a model.
- ruled by inflection, deviation, and composition. Lemmatisation (reduction of words to their base form) and characterisation by the (morpho-syntactic) role within a model.
- morphological features:
  - full or partial specification,
  - layering within a model,
  - integrity constraints,
  - cyclic or acyclic structuring,
  - complete set of schemata for cognitive semantics, open or closed context, and kind of data types.
Basic Categories

- Person
- Thing
- Event
- Action
- State
- Time
- Place
- Direction
- Attribute
- Manner
Morphological Lexicon

How does a parser know which feature is used by which word?

The lexicon has for each word of a language

- the language particle (syntactical category)
- potentially no/default values for features
- information on the meaning
- reference to the lexem, root, ...

Morphological lexicon against lexicons of full forms

love, loved, loving, beloved, unbelieved

The lexicon has for each word of a language

- the language particle (syntactical category)
- potentially no/default values for features
- information on the meaning
- reference to the lexem, root, ...

<table>
<thead>
<tr>
<th>Word</th>
<th>Morphological Lexicon</th>
</tr>
</thead>
<tbody>
<tr>
<td>boys</td>
<td>boy + (n-number = 3p)</td>
</tr>
<tr>
<td>dishes</td>
<td>dish + (n-number = 3p)</td>
</tr>
<tr>
<td>playing</td>
<td>play + (tense = progressive)</td>
</tr>
<tr>
<td></td>
<td>+ (v-number = 1s 2s 3s 1p 2p 3p)</td>
</tr>
<tr>
<td>taken</td>
<td>take + (tense = pastp)</td>
</tr>
<tr>
<td></td>
<td>+ (v-number = 1s 2s 3s 1p 2p 3p)</td>
</tr>
<tr>
<td>loves</td>
<td>love + (tense = present) + (v-number = 3s)</td>
</tr>
<tr>
<td>played</td>
<td>play + (tense = past)</td>
</tr>
<tr>
<td></td>
<td>+ (v-number = 1s 2s 3s 1p 2p 3p)</td>
</tr>
</tbody>
</table>
Classical Lexical Entries for the Parser

- development of a lexicon, e.g. with a function “dictionary”

- An entry may use the following form:
  \[(word \ part-of-speech \ feature-assignments)\]
  \[(word \ root-form \ part-of-speech \ feature-assignments)\]
  ↑
  optional

(dictionary
  (a det)
  (be auxverb (tense = tenseless))
  (is be auxverb (tense = present)
    (v-number = 3s))
  (block noun)
  (block verb)
  (can modal
    (v-number = 1s 2s 3s 1p 2p 3p))
  (do modal)
  (did do modal (tense = past)
    (v-number = 1s 2s 3s 1p 2p 3p))
  (fish noun (n-number = 3s 3p))
  (frog noun)
  (jack proper-noun)
  ... \)
Semiotics of Models: Syntax: Namespace and Lexigraphy

- coding and structuring lexical elements based on the lexicon
  'name', 'description', 'identifier'

- general and an application-dependent namespace

- model as product of a community of practice with its needs, its common-speak, its specific functions of words, its specific phrases and abbreviations, and its specific vocabulary.

Namespace enhanced by specific wordings
from the application domain
specific CoP slang
Semiotics of Models: Semantics: Lexicology

- ontologies

- linguistic relations such as homonym, antonym, paronym, synonym, polysemy, hyponym, etc.

- meanings in the namespace:
  - referential meaning establishes an interdependence between elements and the origin (‘what’);
  - functional meaning is based on the function of an element in the model (‘how’)

intext (within the model), the general, the part-of-model, and the differential (homonym-separating) meaning

- change of meaning for legacy models
Ontology

Ontology is a classification with a set of constraints representing subtyping, partition, disjointness, covering and incoherence, more specifically

- set of terms (objects) of interest in a particular domain \( O = \{ o_1, \ldots, o_n \} \) and relationships \( R = \{ R_1, \ldots, R_m \} \) among them (ontological commitment) relating concepts with kinds, valuation (value, modality, existency) and actors (worlds)
- \( o_i = (k, id_i, v_i) \) for \( k \in \text{Kind} \), \( v_i \) value form \( \text{DOM}(O) \), \( id_i \in ID \),
- \( R_i = \{ r_j = (tr_j, o_{j,1}, \ldots, o_{j,k}, o_j) | tr_j \in TR \}, o_{j,l} \) parameters of \( r_j \)
- \( \text{Kind} = \{ \text{predicator thing, action, actor, rule} \} \) (predicator/structural view; actor+action/dynamic view; rules/deontic view)
- \( \text{TR} = \{ \text{execute, actand, use, extend, ...} \} \) (dynamic view, e.g., actand)

Shared ontology of two communities \( G_1 \) and \( G_2 \) with \( A_1 \) and \( A_2 \) defined by:

Common generic extensible ontology \( A \) that can be mapped by infomorphisms \( (f_1, g_1) \) to \( A_1 \) and by \( f_2, g_2 \) to \( A_2 \)
greatest consistent classification that is finer (or equal) than \( A_1 \) and \( A_2 \)

Core classification \( A^* \) of the communities is defined by the fusion of the classification lattice theories of \( A_1 \) and \( A_2 \) modulo synonyms of \( A_1 \) and \( A_2 \), respectively

with a local classification theory (coincide on common classification, local on non-shared classifications)
General Semantics

Goal: Clarifying semantics beyond CS

- Semantics as one constituent of semiotics
- Semantics with a variety of notions and approaches
- Resulting misunderstanding among communities
- Be aware of hidden semantics of languages, e.g., well-formedness constraints of the ER model
- Models and schemes are restricted by their languages and developed with some intentional incompleteness
- Variety of notions of implication and consequence
- Axiomatisation as one issue, complexity of derivations as another one

Take home: integrating logics, linguistics, philosophy, CS
General Notions of Semantics

Semantics is the study of meaning, i.e. how meaning is constructed, interpreted, clarified, obscured, illustrated, simplified, negotiated, contradicted and paraphrased.

Scientific community: ‘always valid’ semantics based on the mathematical logic

Database modellers: ‘strong’ semantics for constraints

Database analysers and miners: ‘may be valid’ semantics

Users: ‘in most cases valid’ semantics based on prototypes or exemplars remembered

Groups of users: ‘epistemic’ semantics depending on the group

Intuition-driven: ‘hidden’ experience-based semantics

K.-D. Schewe, B. Thalheim: Semantics in Data and Knowledge Bases. SDKB’08, LNCS 4925
Six (or more) Kinds of Semantics

*Lexical semantics:* words in a language
word meaning and of association of words
“Semantic” Web: rudimentary form of word semantics for meta-characterisation

*Grammatical semantics:* categories ‘noun’, ‘verb’ ‘adjectives’
combinatorial semantics - specific form of grammatical semantics

*Statistical semantics:* meaning on co-occurrence of words, on pattern of words in phrases, and on frequency and order of recurrence

*Logical semantics:* relation between the formal language of logics and structures or worlds

*Prototype semantics:* meaning through users evolving experience

*Program and dynamic semantics:* semantic memory, i.e. the memory of meanings, understandings, and other concept-based knowledge unrelated to specific experiences of agents
Formal Semantics

is typically given

- by an *interpreter* that maps syntactic types to semantic types,
- by a *context abstraction* that is based on an aggregation of values which remain fixed in certain temporal and spatial intervals,
- by *states* that provide a means of representing changes over time and space,
- by an *configuration* that is based on an association of contexts and states,
- by an *interpretation function* that yields state results based on certain computation,
- by an *evaluation function* that yield some value results for the syntactic types, and
- by an *elaboration function* that yield both state and some other value results.
## Variations of Semantics

<table>
<thead>
<tr>
<th></th>
<th>mappings</th>
<th>logics</th>
<th>closed world logics</th>
<th>logics on finite worlds</th>
<th>logics on natural worlds</th>
</tr>
</thead>
<tbody>
<tr>
<td>matching of syntactic language $L$ and semantic structure worlds $W$ on signature $\tau$</td>
<td>exact or coincidence embedding</td>
<td>exact</td>
<td>exact</td>
<td>partial depending on interest and meaning in use</td>
<td></td>
</tr>
<tr>
<td>considering context</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>depending on use and user</td>
</tr>
<tr>
<td>considering states</td>
<td>any</td>
<td>any</td>
<td>only finite structures</td>
<td>states in scope</td>
<td></td>
</tr>
<tr>
<td>restricting states and context</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>depending on interest and demand</td>
<td></td>
</tr>
<tr>
<td>interpretation for alphabets</td>
<td>exact for alphabet</td>
<td>exact</td>
<td>potentially restricted</td>
<td>multiple interpretations</td>
<td></td>
</tr>
<tr>
<td>evaluation of variables</td>
<td>full</td>
<td>full</td>
<td>full</td>
<td>partial evaluation</td>
<td></td>
</tr>
<tr>
<td>elaboration</td>
<td>full</td>
<td>negation as failure</td>
<td>derivable structures</td>
<td>extrapolation</td>
<td></td>
</tr>
</tbody>
</table>
Semantics in Computer Science

Operational semantics interprets syntactic types by computational types of a machine.

Denotational semantics associates mathematical functions to syntactic types.

Axiomatic semantics uses a calculus with axioms and proof rules.

Transformational semantics uses mappings to other syntactic types.

categorical or functorial semantics: translation to category theory

Algebraic semantics uses a set of abstract basic syntactic systems with their semantics and a set of rules for construction of more complex systems based on these systems.

Macro-expansion semantics is based on static or dynamic inductive rewriting of syntactic types and allows to introduce abstractions such as the types of the λ calculus.

Grammar semantics uses a state consisting of semantic category variables and on instantiations for atomic non-terminal syntactic types.
Reduction (/Database) Semantics (Indoeuropean Variant)

Deductive normal form with generation based on knowledge and context

Concept T
opic

Information

Content

Topic

Noun (0,1) Verb (1,∞) Adjective (1,1)

fnc/arg

sem_function

is-a

core_value

prole

pc

nc

pnr

sur

in

pnr

idy

mdd/mdr

categories

semantics

Why
Model
Semiotics
Syntax
Semantics
Pragmatics
W+H
Finally

Copyright © P. Broman: Survey on R. Hausser database semantics
Overuse and Misuse of Semantics

The word ‘semantics’ has still a positive co-notation.

semantic web, semantics as a metamodel, semantics as context, semantics as behaviour, semantics as being executable, semantics as the meaning of individual constructs, semantics as mathematical notation, semantics as mappings, ....

Semantic web: micro-semantics of wordings, vocabulary of name spaces or of ontologies

*semantification of semantic web*

Separation of semantics and behaviour: confusion of semantics and behaviour,

Semantics illustration through diagrams: UML 100+x types of diagrams, spatial restriction, consistency and coherence of diagrams
Semantical Interpretation

Principles of compositional semantics (theory of understanding the meaning of language)

(1) syntactical analysis

(2) derivation of the interpretation of the whole by interpreting components.

There are many other forms of semantic interpretation!

- assignment of interpretation to singleton words (e.g., based on a concept lexicon):
  - red --> (color ?x red)
  - block --> (inst ?x block)
  - the --> retrieve-val

- combination of individual interpretation for derivation of interpretation of expressions:
  - red block --> (and(inst ?x block)(color ?x red))
  - the red block --> (retrieve-val '?x (and (inst ?x block) (color ?x red)))

- binding of these processes to the syntactical parser
Ambiguity Problems

Lexical, structural, semantical

- Millers saw the Alps as they were flying to Italy.
  (Who flies here?)
- I took the meal with forks and knifes.
- I took the meal with Gaby and Erwin.
- Can companies litter the environment
  (?)
- I saw the man in the park with the telescope
  ... only schematic since the lens was dirty.
  ... and had the impression that we was observing the moon.
- I saw her duck.
- Flying planes made her duck.
  Planes: aircraft or geometry       flying: active or I am currently flying.
Connections, Ellipses and Ambiguities

(Natural language ... is a wide field - not only for the computer.)

Nimmer diesen Monitor legen, wo der Schnur von Personen darauf spazieren gehen grausam behandelt wird.

a description from the manual for a Japanese TV monitor (Source: Spiegel 10/92)


Scharping (10.95): “... und damit der Proporz gewahrt bleibt ...”
Scharping (10.95): “... Diese Runde war eine Niederlage, aber es werden weitere folgen. Meine Damen und Herren ... Runden, Runden!!”

Lebende Karpfen - auch geteilt!
Montague’s intensional Semantics

Extended signature with categories $\Omega' \subseteq \Omega$ for sorts and connections $\omega_1/\omega_2$, $\omega_1//\omega_2$ (truncation)

(in)transitive verbs, terms, modified adverbs, appelatives, ...

Type calculus for expressions with assignment of categories and sorts

Ground functions $F_0(\beta) \doteq$ every $\beta$, $F_1(\beta) \doteq$ the $\beta$, $F_2(\beta) \doteq$ a(n) $\beta$ (with vowel rule), $F_3,n(\alpha, \beta) \doteq \alpha$ such that $\beta$

Function application $F_4, F_5, F_6, F_7$

Conjunction and disjunction $F_8, F_9$

Quantification $F_{10}$

Tempus and presage rules $F_{11}, F_{12}, F_{13}, F_{14}, F_{15}$ (z.B. negative 3. person singular praesens perfect)
Montague’s intensional Semantics

Example of a function application

Every man loves a woman such that she loves him

\[
F_{10,0}(F_0(\text{man}), F_{4}(\text{he}_0, F_{5}(\text{love, } F_{2}(F_{3,1}(\text{woman, } F_{4}(\text{he}_1, F_{5}(\text{love, } \text{he}_0)))))))
\]
Intensional Logics

distinguishes between

- value (through interpretation, reference to extension (semantical value))
- meaning (through sense, reference to intension)

Inductive introduction of a type logics

- entities and truth are types
- elementary types (types and situations (world, time, context))
- \((a, b), (s, a)\) are types for types \(a, b\), situations \(s\)

with connectives and quantifiers

interpretation through \(Mod(\alpha, M, W, t, g)\) with structure \(M\) in the world \(W\) at time point \(t\) and variables assignment \(g\) \((Content(\alpha) \cup Extension(\alpha))\)

presentation through \(Sym(\alpha, M, g)\) with structure \(M\) and variables assignment \(g\) \((Symbol(\alpha) \cup Intension(\alpha))\)

harmonisation for \(GetCont(Sym(\alpha, M, g), w, t) \equiv Mod(\alpha, M, W, t, g)\) with \(Extension(Symbol(\alpha)) = Content(\alpha)\) for mappings

\(Extension : Symbol \rightarrow Content\) and \(Intension : Content \rightarrow Symbol\)

where in general it is not required that \(Intension(Content(\alpha)) = Sym(\alpha)\)
Semiotics of Models: Pragmatics: Community of Practice

study how languages are used for intended deployment functions in dependence on the purposes and goals within a community of practice

- **descriptive-explanatory** and **persuasive-normative** functions of a model: (1) acting (2) within a community, especially the modeller and have (3) different truth or more generally quality

- **far-side** versus **near-side** pragmatics separating the ‘why’ from the ‘what’ side of a model

- development and deployment modes, model surface compositionality (methodological principle), model presentation order’s strict linearity relative to space (empirical principle), model interpretation and production analysed as cognitive processes (ontological principle), reference modelled in terms of matching an model’s meaning with context (functional principle)

- methodologically valid, support subjective deductive (paradeducutive) inference with an open world interpretation, allow context-dependent reasoning (implicature), provide means for collaborative interaction, weaken connectives and quantifications, and integrate deductive, inductive, abductive and paradeducutive reasoning.
Connectives and Quantifiers in Reality?

**Different truth definitions**

Material, logical, and normative connectives, e.g. implication

- $\psi \rightarrow \phi$ means $\phi$ necessarily if $\psi$ (strict, logical)
- $\psi \Rightarrow \phi$ means 'It is the case that if $\psi$ (can be observed) then also $\phi$.' (material)
- $\psi \supset \phi$ means 'In situations for which there exists a dependence then $\phi$ follows from $\psi$ (norms) (counter-example-based)

<table>
<thead>
<tr>
<th>$\psi$</th>
<th>$\phi$</th>
<th>$\psi \Rightarrow \phi$</th>
<th>$\psi \supset \phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>??</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>??</td>
</tr>
</tbody>
</table>

Generalisation operators e.g. $(t,f)$-quantifier $Q_{t,f}$ with validity dependence of $Q_{r,s,\alpha}(x)$ in structure $A$ such that

$|\{o \in \pi_x(dom(A)) | I_x^o(\alpha) = 1\}| = t$ and

$|\{o \in \pi_x(dom(A)) | I_x^o(\alpha) = 0\}| = f$

classical $\forall \equiv Q_{*,0}, \ \exists \equiv Q_{t,*}$ for $t \geq 1$, $\text{Majority} \equiv Q_{n+k,n}$, $k, n \in \mathbb{N}^+, k \geq 1$

Models for the knowledge, ... operator $K_A$ for actors $A$
Semiotics of Models: Pragmatics: Visualisation (Phonology)

*Principles of visual communication*: Vision, cognition, and processing and memorizing characteristics.
visual features: contrast, visual analogies, presentation dramaturgy, reading direction, visual closeness, symmetric presentation and space and movement

*Principles of visual cognition*: ordering, effect delivery, and visualisation model organisation, model economy, skills of users, and standards.

*Principles of visual design*: optical vicinity, similarity, closeness, symmetry, conciseness, reading direction.
Semiotics of Models: W*H-Specification Frame

classical rhetorical frame
Hermagoras of Temnos: Quis, quid, quando, ubi, cur, quem ad modum, quibus adminiculis

Who, what, when, where, why, in what way, by what means

- \( W^4 \): wherefore (purpose), whereof (origin), wherewith (carrier, e.g., language), and worthiness ((surplus) value)

- secondary characterisation \( W^{17} H \):

  - user or stakeholder or community of practice characteristics: by whom, to whom, whichever;
  
  - characteristics imposed by the application domain: wherein, where, for what, wherefrom, whence, what;
  
  - purpose characteristics characterising the solution: how, why, whereto, when, for which reason; and
  
  - additional context characteristics: whereat, whereabout, whither, when.

©A. Dahanayake
### The Zachman Framework

(Reusing the Rhetoric Frame without Mentioning It)

**Living in a world full of different models**

<table>
<thead>
<tr>
<th>Model</th>
<th>Motivation (Why)</th>
<th>People (Who)</th>
<th>Function (How)</th>
<th>Data (What)</th>
<th>Time (When)</th>
<th>Network (Where)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scope View</strong></td>
<td>Vision Needs</td>
<td>Vision Stakeholders</td>
<td>Vision Features</td>
<td>Business entities</td>
<td>Business workflows</td>
<td></td>
</tr>
<tr>
<td><strong>Owner's View</strong></td>
<td>Business rules</td>
<td>Actors</td>
<td>Use cases</td>
<td>Business object model</td>
<td>Use case flow of events</td>
<td>Network configurations</td>
</tr>
<tr>
<td><strong>Designer's View</strong></td>
<td>Constraints, multiplicities, workflow activities</td>
<td>Boundary classes</td>
<td>Use case realizations</td>
<td>Persistent classes</td>
<td>Interaction diagrams</td>
<td>Deployment model</td>
</tr>
<tr>
<td><strong>Builder's View</strong></td>
<td>End user support material</td>
<td>Components</td>
<td>Data model</td>
<td>Process model</td>
<td>Process-to-node mapping</td>
<td></td>
</tr>
<tr>
<td><strong>Detailed View</strong></td>
<td>UI design classes</td>
<td>Design classes</td>
<td>Columns, types, keys, indexes</td>
<td>State machines</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Coherence, calibration, mapping problems**
Summarising

- Modelling and senseful speaking/writing are related to each other!
- Language-backed modelling inherits semiotics!
- Conceptual modelling is modelling by deploying concepts, i.e., basing semantics on concepts.
- Models are specific forms of expressions: goal-oriented, purpose-driven, and providing added value through their function within application.
- Models are the third dimension of scientific reasoning.
- Models are ridden by communities (of practice).

There is a general notion of a model!

There are hundreds of open research issues!
Thank you!

thalheim@is.informatik.uni-kiel.de
Publications on Science and Art of Conceptual Modelling


Publications on Model Suites, Evolution, Migration


Publications on Tool-Based Development


Publications on Pattern Development


Publications on Component Development


Publications on Genericity


Publications on Co-Design


- Schewe, K.-D.; Thalheim, B.: Towards a theory of consistency enforcement. Acta Informatica, 36, 1999, 97-141. *Instead of falling into the traps of rule triggering systems*


The Extended Entity-Relationship Modelling Language HERM

Eötvös Loránd University of Sciences
Budapest

3.7.2012

Bernhard Thalheim
Technologie der Informationssysteme
Institut für Informatik, Christian-Albrechts-Universität zu Kiel, BRD
Kolmogorow-Professor e.h. der Lomonossow-Universität Moskau
Why Structuring
Null marker
CoMoL
Functionality Views
OLAP
References

Content
Information
Concept Topic

My Background

HERM 'bible' Handbook Dependencies

ADBIS ASM CM EJC e-Bus. ER

FoIKS MFDBS NLDB Semantics WIS WISE

i.e., reaching the big triple "3"s (3, 30, 300)
(3 (habilitation students, profs, books), 30 (PhD students, editorials, projects),
300 (master/diploma students, papers, presentations))
Instead of a Personal Profile: General
Instead of a Personal Profile: My Co-Authors
Instead of a Personal Profile: Most Cited Papers (h ≥ 25)
Language Matters

Global versus local: Petri nets as a local representation pitfalls in Michael Jackson’s crossroad example

State-oriented versus event-oriented representation

Conceptual completeness of the language

Users, users, users are different in their (visual, oral, literacy) communication skills, (visual, oral, literacy) cognition and thus require different design

Hearer, speaker, archiver views
The database aims in supporting lecture and course scheduling within a university application. A course is typically proposed. If the proposal becomes accepted then the course is going to be planned. Typically planned course may be also held. The course proposal, planning and organisation is bound to a semester. Some university employees may be responsible for a course. Typically a course has one responsible person. Responsibilities may vary by semesters. Courses are taught by professors. Professors are specialisations of a person.

Courses may be given for various programs. The proposal also includes the assignment of a course kind. The course proposal typically also requests for a room and for a time at which the course preferably could be given. Additionally time slots may be in conflict with other proposals. Therefore, conflicting time slots are given as well. The room and time preference may be overwritten during the planning phase. The same opportunity exists for proposals for the kind of the course to be given. If the time is assigned then typically a time slot is assigned. A course may have several non-overlapping time slots.

The proposal for a course should be recorded. A person may act in the role of somebody who inserted the course.

Finally, courses may also be held at a different location than originally planned.
Compare Visual Representation with Textual One

Is this better to read for users of ER diagrams?

Course

Program

Kind

Semester

Professor

Person

proposed Course

planned Course

Room

TimeFrame

Request

Teacher

inserted

Responsible4Course

Time(Proposal, SideCondition)

Proposal

For more information: http://www.is.informatik.uni-kiel.de/~thalheim/slides.htm
Cognition Completeness

“Principle of linguistic relativity”: actors skilled in a language may not have a (deep) understanding of some concepts of other languages.

Six Lakoff schemata

- container schema (class-building, aggregating, composing)
- part-whole schema (specialisation/generalisation, categorisation, ...)
- link schema (within many kinds)
- centre-periphery schema (oil stain, ...)
- source-path-goal schema
- ordering schemata, e.g., up-down, front-back and linear ordering

ER is cognition incomplete ($\frac{2}{6}$ criteria are satisfied)
HERM is $\frac{5}{6}$-cognition complete (centre-periphery 😊)
HERM with kernel entity types is cognition complete
### Indoeuropean Languages and HERM

<table>
<thead>
<tr>
<th>English sentence concept</th>
<th>HERM feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>transitive verb</td>
<td>relationship type</td>
</tr>
<tr>
<td>common noun</td>
<td>component of relationship type</td>
</tr>
<tr>
<td>adjective</td>
<td>attribute of component</td>
</tr>
<tr>
<td>adverb</td>
<td>attribute of relationship type</td>
</tr>
<tr>
<td>numerical expression</td>
<td>attribute of object type</td>
</tr>
<tr>
<td>preposition</td>
<td>role name of component</td>
</tr>
<tr>
<td>gerund</td>
<td>relationship type that is component of another relationship type</td>
</tr>
<tr>
<td>clause</td>
<td>relationship type with components</td>
</tr>
<tr>
<td>complex sentence</td>
<td>relationship type of order higher than 1</td>
</tr>
<tr>
<td>alternative phrase</td>
<td>cluster type</td>
</tr>
<tr>
<td>plural collection</td>
<td>type/nested attribute</td>
</tr>
<tr>
<td>“IsA” sentence</td>
<td>specialisation</td>
</tr>
</tbody>
</table>
# Comparison to Chen’s Original Correspondences


<table>
<thead>
<tr>
<th>English sentence concept</th>
<th>ER feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>transitive verb</td>
<td>relationship type</td>
</tr>
<tr>
<td>common noun</td>
<td>entity type</td>
</tr>
<tr>
<td>adjective</td>
<td>attribute of entity type</td>
</tr>
<tr>
<td>adverb</td>
<td>attribute of relationship type</td>
</tr>
<tr>
<td>numerical expression</td>
<td>attribute of entity or relationship type</td>
</tr>
<tr>
<td>gerund</td>
<td>relationship-converted entity type</td>
</tr>
<tr>
<td>clause</td>
<td>high-level entity type abstracted from group of interconnected low-level entity and relationship types</td>
</tr>
<tr>
<td>complex sentence</td>
<td>one or more entity types connected by relationship type in which each entity type can be decomposed recursively into low-level entity types interconnected by relationship types</td>
</tr>
</tbody>
</table>

**Why**
Structuring
Null marker
CoMoL
Functionality
Views
OLAP
References
Conclusions for ERM versus HERM


⇝ EER reflects (English) sentence structures more soundly and naturally
⇝ higher-order object types reflect dependence between sentences
⇝ this provides justification for introduction of new ER features
⇝ ER model does not just provide safe constructs that result in good database design, but also features that enable good communication between designer and user
⇝ essential to best approximate requirements
⇝ additional EER features justified in the sense that modelling becomes more natural
⇝ provides also a justification why the EER features exist
⇝ higher-order object types reminiscent of nested sentence structure in natural language text
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
Type Constructor

- with a selector for retrieval (like Select) with a retrieval expression and update functions (like Insert, Delete, and Update) for value mapping from the new type to the component types or to the new type,
- with correctness criteria and rules for validation,
- with default rules,
- with one or several user representations, and
- with a physical representation or properties of the physical representation.
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

\[
\text{integer} := (\text{IntegerSet}, \{0, s, +, -, *, \div\}, \{ =, \leq \})
\]

Attribute type induced on basic data type system

\[
\text{Name} : (\text{FirstNames} < (\text{FirstName}, \text{use} > , \text{FamName},
\text{[NameOfBirth,] Title:}\{\text{AcadTitle} \cup \text{FamTitle})
\]

\text{Contact} : (\text{Phone}\{\{\text{AtWork}\}, \text{private}, \text{email}, ...\})

\text{DateOfBirth :: date}

\text{AcadTitel :: titleType}

Entity type - product of attribute types with at least one direct identification

\[
\text{Person} : (\text{Name, Address, Contact, DateOfBirth,}
\text{PersNo: StudNo} \cup \text{EmplNo, ...})
\]
Structuring in HERM

Relationship type via hierarchical construction

unary type: Role, specialisation

\[\text{InGroup} : (\text{Person}, \text{Group}, \text{Time}(\text{From}[\text{,To}]), \text{Position})\]

\[\text{DirectPrerequisite} : (\text{hasPrerequisite} : \text{Course}, \text{isPrerequisite} : \text{Course})\]

\[\text{Professor} : (\text{Person}, \text{Specialization})\]

Cluster type - disjoint (labelled) union
especially for generalization

\[\text{JuristPerson} : \text{Person} \cup \text{Company} \cup \text{Association}\]

\[\text{Group} : \text{Senat} \cup \text{WorkingGroup} \cup \text{Association}\]

Constructors \(\times, \cup, \{}\) construction complete

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

with underlying type system

and generic operations

plus construction of operations through structural recursion
Why Structuring
Null marker
CoMoL
Functionality Views
OLAP
References
Reasons for Higher-Order Relationship Types

- they are naturally occurring
- awful integrity constraints otherwise
- schema complexity decreasing via compactification
- unary relationship types are sometime IsA-constraints
- unary relationship types with additional identification allow also general-special specialisation
  - e.g., item list or book-coyp
- natural identification inheritance
- relational and network translation creates them anyway
Implicit Assumptions

Set semantics: default semantics

Identifiability: each entity type is identifiable

Partial Unique Name Assumption: attribute names are unique

Referential integrity

Monotonicity of semantics: if $\Phi$ are added to $\Sigma$, then the set of possible instances which satisfy the extended set of constraints $\Sigma \cup \Phi$ is a subset of the set of instances which satisfy $\Sigma$.

Compositionality

Destination of define and use
A Sample Schema

- Course
- Semester
- Professor
- Person
- Program
- Kind
- Room
- proposed Course
- planned Course
- Course held
- Time(Proposal, SideCondition)
- TimeFrame
- Teacher
- Responsible4Course
- Request
- Inserted
- Content
- Information
- Concept
- Topic

Why
Structuring
Null marker
CoMoL
Functionality
Views
OLAP
References
Basic Data Types

*Precision and accuracy:* precision is the degree of refinement in the calculations
accuracy is a measure of how repeatable the assignment of values for properties is.

*Granularity:* scales

*Ordering*

*Classification*

*Presentation*

*Implementation*

*Default values*

*Casting functions*
# Data types and their main canonical assumptions

<table>
<thead>
<tr>
<th>Kind of data type</th>
<th>Natural order</th>
<th>Natural zero</th>
<th>Predefined functions</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>extension based</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>absolute</td>
<td>+</td>
<td>+/-</td>
<td>+/-</td>
<td>number of boxes</td>
</tr>
<tr>
<td>ratio</td>
<td>+</td>
<td>+/-</td>
<td>+(type dependent)</td>
<td>length, weight</td>
</tr>
<tr>
<td><strong>intension based</strong></td>
<td>-</td>
<td>-</td>
<td>(-) (except concatenation)</td>
<td>names of cities</td>
</tr>
<tr>
<td>nominal</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ordinal</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>preferences</td>
</tr>
<tr>
<td>range</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>competitions</td>
</tr>
<tr>
<td>interval</td>
<td>+</td>
<td>-</td>
<td>(+)(e.g., concatenation)</td>
<td>time, space</td>
</tr>
</tbody>
</table>
Graphical Notions Might Vary

null marker
CoMoL
Functionality
Views
OLAP
References

Diploma student
Student
Person
Project member
University employee
Professor

Content
Information
Concept
Topic
Unary Relationship Types

- Person
  - Professor

- Person
  - Professor

- Person
  - IsA
  - Professor

Why Structuring
Null marker
CoMoL
Functionality
Views
OLAP
References

Concept Topic

Information

Content
Generalising Cluster Types

The simple HERM schema

\[
\text{Examines}.\text{Enrolls}.\text{Course} = \text{Examines}.\text{GivenBy}.\text{Course}
\]

The sophisticated HERM schema

\[
\Theta := \text{Enrolls}.\text{Course} \sqcup \text{Provides}.\text{Course}
\]

The representational conceptual schema

\[
\begin{align*}
\text{Student} & = (\{\text{StudId}, \ldots\}, \ldots), \\
\text{Course} & = (\{\text{CourseID}, \ldots\}, \ldots), \\
\text{Lecturer} & = (\{\text{LecturerID}, \ldots\}, \ldots), \\
\text{Enrolls} & = (\{\text{StudId}, \text{CourseID}, \ldots\}, \ldots), \\
\text{Provides} & = (\{\text{CourseID}, \text{LecturerID}, \ldots\}, \ldots), \\
\text{Examines} & = (\{\text{StudId}, \text{LecturerID}, \text{CourseID}, \ldots\}, \ldots).
\end{align*}
\]

\[
\text{Examines}[	ext{StudId, CourseID}] \subseteq \text{Enrolls}[	ext{StudId, CourseID}]
\]

\[
\text{Examines}[	ext{CourseID, LecturerID}] \subseteq \text{Provides}[	ext{CourseID, LecturerID}]
\]

The logical relational schema

The association between the “optimised” schema and the relational schema
Constraints in Brief

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)

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

Relationship types with attributes

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

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

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

Keys defined by the component construction

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

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

Functional dependencies for functional associations among groups of attributes

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

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

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

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

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

\[\text{card(DirektVoraussetz, isPrequisite)} = (3,4) \quad \text{not satisfiable}\]

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

\[\text{card(proposedCourse, Sem Prof)} = (3,5) (??) \quad \text{or} \quad (0,5) (!!!)\]
Classes and Instances

Classes are sets of (markable) objects of corresponding type
Entity class is the basic class of objects

β: ((⟨(Karl,add),(Bernhard,call)⟩, Thalheim, {Prof., Dr.rer.nat.habil, Dipl.-Math.}), BTU Cottbus,

((+49 355 692700, +49 355 692397}, +49 355 824054),
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)

Profβ: (637861, Database and information systems)


Senator5β: (637861, Senat, (2000), Chair)

PrerequDBIVMain: (DBIV, DBI),

CourseDBIVSS02: (DBIV, SS2002, 637861, SR1, Mo. first pair)

...
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

\{ Senator2π: (889721, Senat, (1998,2000), Chair), 
Senator5β: (637861, Senat, (2000), Chair), 
DBIS: (Database and information systems, 637861, IfI), 
CBnetβ: (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
Frame for Handling Constraints

generalising the SPICE framework to constraints:
declaration, control, managed, optimised, predicted

Defined (Specification level - Description layer)

- General description, context embedding, implementation abstraction

Managed (Application level - Technology layer)

- Application modi, reaction on fault, log

Performed & executed (Control level - Technical layer)

- Enforcement pattern, validity through phrases

Established & controlled (Saturated level - Organisation layer)

- Understood, predictable, performed (Holistic level - Economical layer)

Optimised (Systematic improvement level - Evolution layer)

- declaration/formation specification b a mathematical formula

~\Rightarrow\text{ resulting properties (e.g., decomposition property)}
Kinds of Constraints

kind_of_constraint ∈ \{ identification, independence, existence, redundancy \}

- Materialisation
- Partial Identification
- Inner Unit Coupling
- Specific

Syn.: FD as stored function
Sem.: Pragm.: structural association among units
Syn.: Partial identification
Sem.: Functional coupling
Syn.: Tight functional coupling
Sem.: Valid only in given application
Kinds of Constraints

kind_of_constraint ∈ \{ identification, independence, existence, redundancy \}

(1)-(6): see theory lecture
Kinds of Constraints

inclusion dependency

- materialisation
  - syn.:...,
- existence constraint
  - syn.:...
- redundancy constraint
- inner unit coupling
  - sem.: specialisation hierarchy
- specific
  - syn.:...,

completeness constraint, e.g., no-null constraint

- materialisation
  - syn.:...
- existence constraint
  - syn.:...
- redundancy constraint
  - syn.:...
- inner unit coupling
  - syn.:...
- specific
  - syn.:...,
Constraints in Detail

see my theory lecture

Why Structuring
Null marker
CoMoL
Functionality
Views
OLAP
References
Brute-Force Utilisation of the Predicate Calculus

The tuple calculus and the predicate calculus are two big aberrations of current database systems teaching and of historic database research!! 😄

Correct attribute calculus

- \( P_{T_i}(v_{i,1}, \ldots, v_{i,n_i}) \), for variables \( v_{i,j} \) of type \( A_j \) for \( T_i = \{ A_1, \ldots, A_{n_i} \} \) and order over \( T_i \) abbreviated: \( T_i(v_{i,1}, \ldots, v_{i,n_i}) \)
- \( \alpha \circ \beta, \circ \in \{ \lor, \land \} \)
- \( \neg \alpha \)
- \( \exists v \alpha, \ \forall v \alpha \)
Brute-Force Utilisation of the Predicate Calculus

Atzeni/Ceri/Paraboschi/Torlone definition schema

\{target\ structure \mid context \mid conditions\}

Correct tuple calculus over a schema $$\mathcal{S}$$

\[
\{ t \mid T \mid \bot \} \quad \text{for each type } T \text{ in schema } \mathcal{S}
\]

\[
\{ t \mid C \mid \beta \} \quad \text{for selection } \sigma_\alpha
\]

\[
\{ t_1 \mid C_1 \mid \alpha_1 \}, \{ t_2 \mid C_2 \mid \alpha_2 \} \quad \text{for join } \Join
\]

\[
\{ t_1 \Join t_2 \mid C_1 \cup C_2 \mid \alpha_1 \land \alpha_2 \}
\]

\[
\{ t \mid C \mid \alpha_1 \}, \{ t \mid C \mid \alpha_2 \} \quad \text{for union } \cup
\]

\[
\{ t \mid C \mid \alpha_1 \lor \alpha_2 \}
\]

\[
\{ t \mid C \mid \alpha_1 \land \neg \alpha_2 \} \quad \text{for set difference } \setminus
\]

\[
\{ \rho(t) \mid C \mid \alpha \} \quad \text{for renaming } \rho
\]

\[
\{ t \mid C \mid \alpha \} \quad \text{for projection } \pi_X
\]
Null Markers instead of Null “Values”

- state of the art: wide utilisation of misinterpretation of NULLs, wrong implementation support, confusing logics
- null marker logics

C.J. Date (Logic and Databases - The roots of relational theory (2007), 117): “I apologize for the wording “contains a null”; as I’ve written elsewhere, to talk about anything “containing a null” actually makes no logical sense. Indeed one of the problems with nulls is precisely that you can’t talk about them sensibly! ... the entire topic is a perfect illustration of The Principle of Incoherence ...”

(228): “… nulls are ipso facto nonsense ...

E.F. Codd (The Relational Model for Data Management, Version 2 (1990), 172): The basic principle of the relational approach to missing information is that of recording the fact that a db-value is missing by means of a mark (originally called a null or null value). There is nothing imprecise about a mark: a db-value is either present or absent in a column of a relation in the database.

(197) ... the way the relational model deals with missing values appears to be one of its least understood part.
### Arithmetics with Null “Values”

Missing theory resulted in confusing implementations

be careful with proposal: \( \text{CAST (NULL AS INTEGER)} \)

\[
\frac{\text{NULL}}{0} = ?
\]

<table>
<thead>
<tr>
<th>Database</th>
<th>Result</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingres</td>
<td>NULL</td>
<td>float point err no data</td>
</tr>
<tr>
<td>Oracle</td>
<td>NULL</td>
<td>divide by 0 err no data</td>
</tr>
<tr>
<td>Progress</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>R:BASE</td>
<td>NULL</td>
<td>divide by 0 err no data</td>
</tr>
<tr>
<td>Rdb</td>
<td>truncation at runtime</td>
<td>divide by 0 err no data</td>
</tr>
<tr>
<td>SQL Server</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>SQL Base</td>
<td>NULL</td>
<td>NULL &amp; error</td>
</tr>
<tr>
<td>Sybase</td>
<td>NULL</td>
<td>plus infinity</td>
</tr>
<tr>
<td>WATCOMSQL</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>XDD</td>
<td>NULL</td>
<td>divide by 0 err no data</td>
</tr>
</tbody>
</table>

**Example SQL:***

```sql
CREATE TABLE Got Null (test INTEGER);
    INSERT INTO GotNull VALUES (NULL);
CREATE TABLE GotOne (test INTEGER);
    INSERT INTO GotOne VALUES (1);
CREATE TABLE GotZero (test INTEGER);
    INSERT INTO GotZero VALUES (0);
SELECT test/0 FROM Got One | Got Zero | Got NULL
```

**Results:**

- Ingres: NULL
- Oracle: NULL
- Progress: NULL
- R:BASE: NULL
- Rdb: NULL
- SQL Server: NULL
- SQL Base: NULL
- Sybase: NULL
- WATCOMSQL: NULL
- XDD: NULL

**Notes:**

- Ingres: NULL
- Oracle: divide by 0 err no data
- Progress: NULL
- R:BASE: NULL
- Rdb: truncation at runtime
- SQL Server: NULL
- SQL Base: NULL
- Sybase: NULL
- WATCOMSQL: NULL
- XDD: NULL

**Summary:**

When using arithmetic operations with NULL values, it is important to be aware of the specific behavior of each database system. The results can vary significantly, and it is crucial to carefully consider the implications of NULL values in your queries and calculations.
Problems with SQL NULL

Overloading is a bad business: ad-hoc polymorphism

Semantically different aspects are represented in the same form.

Predicates with NULL

\[
\text{SELECT } * \quad \text{SELECT } * \quad \text{SELECT } *
\]

\[
\text{FROM Table} \quad \text{FROM Table} \quad \text{FROM Table}
\]

\[
\text{WHERE Column} = 2 \quad \text{WHERE Column} \neq 2 \quad \text{WHERE Column} \text{ IS NULL}
\]

SQL-92 decision is [NOT] TRUE | FALSE | UNKNOWN

e.g., z.B. ((Age <18) AND (Gender = 'female')) is NOT FALSE

is true is NULL in one of the columns.

NULL value representation in host languages

Important advice: testing which treatment of NULL is supported in host languages

e.g. cursor

e.g. INDICATOR variables ( NULL ≜ -1)
Modelling Advices

- NOT NULL - wherever applicable
- NULLs need additional implementation effort, e.g. an extra bit
- NULLs require specific storage formats, indexes, search support
  \(\Rightarrow\) in any case better: DEFAULT
- Usage of conventions, e.g. ISO for gender
  0 - unknown; 1 - male; 2 - female; 9 - not applicable
- Special support for arithmetic functions: explicit assignment (unknown, not applicable (-1, ..., nonsense ... 0)
- NULLs are also used if domain types do not support special values
  \(\Rightarrow\) extend the domain type
A Simplified But Almost Real Application

CREATE TABLE Student (  
  MatriculationNo CHAR(6) could be NULL  
  StudNo CHAR(6) our internalisation  
  MainProgram VARCHAR(20) could be NULL, could be more than one  
  Name VARCHAR(50) far too simple  
  SecondaryProgram VARCHAR(20) might be more than one  
  PRIMARY KEY (StudNo)  
  FOREIGN KEY (MainProgram) REFERENCES ProgramAtCAU (ProgName)  
  ON DELETE RESTRICT  
... );

CREATE TABLE Enroll (  
  StudNo CHAR(6) our internalisation  
  Course CHAR(8) full of beans  
  Term VarCHAR(7) that is a devils invention  
  EnrollmentCondition VARCHAR(10)  
  Grade VARCHAR(3)  
  PRIMARY KEY (StudNo, Course, Term)  
... );
### A Simplified But Almost Real Application

**CREATE TABLE Enroll (**

| StudNo     | CHAR(6) |
| EnrollmentCondition | VARCHAR(10) |
| Grade       | VARCHAR(3) |
| ...         |          |

**Who can handle the mess?**

<table>
<thead>
<tr>
<th>Diploma student</th>
<th>Certificate Diploma</th>
<th>Graduated Certificate Diploma</th>
<th>Bachelor &amp; Master</th>
<th>Guest Student</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>During course</strong></td>
<td>never exists</td>
<td>not yet decided</td>
<td>unknown</td>
<td>unknown</td>
</tr>
<tr>
<td><strong>After course</strong></td>
<td>never exists</td>
<td>not exists</td>
<td>known</td>
<td>known</td>
</tr>
<tr>
<td><strong>Special condition</strong></td>
<td>not yet decided</td>
<td>not exists</td>
<td>under change</td>
<td>under change</td>
</tr>
</tbody>
</table>
A Simplified But Almost Real Application

CREATE TABLE Enroll (  
    StudNo CHAR(6)  
    EnrollmentCondition VARCHAR(10)  
    Grade VARCHAR(3)  
    ... );

<table>
<thead>
<tr>
<th>During course</th>
<th>Diploma student</th>
<th>Certificate Diploma</th>
<th>Graded Certificate Diploma</th>
<th>Bachelor &amp; Master</th>
<th>Guest Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>never exists</td>
<td>not exists</td>
<td>unknown</td>
<td>unknown</td>
<td>not applicable</td>
<td></td>
</tr>
<tr>
<td>After course</td>
<td>never exists</td>
<td>not exists</td>
<td>known</td>
<td>known</td>
<td>not applicable</td>
</tr>
<tr>
<td>Special condition</td>
<td>not yet decided</td>
<td>not exists</td>
<td>under change</td>
<td>under change</td>
<td>potentially known</td>
</tr>
</tbody>
</table>
### A Simplified But Almost Real Application

`CREATE TABLE Enroll (`  
| StudNo | CHAR(6) | EnrollmentCondition | VARCHAR(10) | Grade | VARCHAR(3) | ... |)

#### Who can handle the mess?

<table>
<thead>
<tr>
<th>Diploma student</th>
<th>Certificate Diploma</th>
<th>Graded Certificate Diploma</th>
<th>Bachelor &amp; Master</th>
<th>Guest Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>never exists</td>
<td>not yet decided</td>
<td>unknown</td>
<td>unknown</td>
<td>not applicable</td>
</tr>
<tr>
<td>never exists</td>
<td>not exists</td>
<td>known</td>
<td>known</td>
<td>not applicable</td>
</tr>
<tr>
<td>not yet decided</td>
<td>not exists</td>
<td>under change</td>
<td>under change</td>
<td>potentially known</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>During course</th>
<th>After course</th>
<th>Special condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>never exists</td>
<td>never exists</td>
<td>not yet decided</td>
</tr>
<tr>
<td>not yet decided</td>
<td>not exists</td>
<td>not exists</td>
</tr>
<tr>
<td>unknown</td>
<td>known</td>
<td>under change</td>
</tr>
<tr>
<td>not applicable</td>
<td>not applicable</td>
<td>potentially known</td>
</tr>
</tbody>
</table>

---

**Why Structuring**

- Null marker
- CoMoL
- Functionality
- Views
- OLAP

**References**
A Simplified But Almost Real Application

CREATE TABLE Enroll (  
    StudNo CHAR(6)  
    EnrollmentCondition VARCHAR(10)  
    Grade VARCHAR(3)  
    ... );

<table>
<thead>
<tr>
<th></th>
<th>Diploma student</th>
<th>Certificate Diploma</th>
<th>Graded Certificate Diploma</th>
<th>Bachelor &amp; Master</th>
<th>Guest Student</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>During course</strong></td>
<td>never exists</td>
<td>not yet decided</td>
<td>unknown</td>
<td>unknown</td>
<td>not applicable</td>
</tr>
<tr>
<td><strong>After course</strong></td>
<td>never exists</td>
<td>not exists</td>
<td>known</td>
<td>known</td>
<td>not applicable</td>
</tr>
<tr>
<td><strong>Special condition</strong></td>
<td>not yet decided</td>
<td>not exists</td>
<td>under change</td>
<td>under change</td>
<td>potentially known</td>
</tr>
</tbody>
</table>

Who can handle the mess?
# A Simplified But Almost Real Application

CREATE TABLE Enroll (
    StudNo CHAR(6),
    EnrollmentCondition VARCHAR(10),
    Grade VARCHAR(3),
    ...
);

<table>
<thead>
<tr>
<th></th>
<th>Diploma student</th>
<th>Certificate Diploma</th>
<th>Graded Certificate Diploma</th>
<th>Bachelor &amp; Master</th>
<th>Guest Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>During course</td>
<td>never exists</td>
<td>not yet decided</td>
<td>unknown</td>
<td>unknown</td>
<td>not applicable</td>
</tr>
<tr>
<td>After course</td>
<td>never exists</td>
<td>not exists</td>
<td>known</td>
<td>known</td>
<td>not applicable</td>
</tr>
<tr>
<td>Special condition</td>
<td>not yet decided</td>
<td>not exists</td>
<td>under change</td>
<td>under change</td>
<td>potentially known</td>
</tr>
</tbody>
</table>

Who can handle the mess?
**A Simplified But Almost Real Application**

Who can handle the mess?

```sql
CREATE TABLE Enroll (  
    StudNo CHAR(6)  
    EnrollmentCondition VARCHAR(10)  
    Grade VARCHAR(3)  
    ...  
);  
```

<table>
<thead>
<tr>
<th></th>
<th>Diploma student</th>
<th>Certificate Diploma</th>
<th>Graded Certificate Diploma</th>
<th>Bachelor &amp; Master</th>
<th>Guest Student</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>During course</strong></td>
<td>never exists</td>
<td>not yet decided</td>
<td>unknown</td>
<td>unknown</td>
<td>not applicable</td>
</tr>
<tr>
<td><strong>After course</strong></td>
<td>never exists</td>
<td>not exists</td>
<td>known</td>
<td>known</td>
<td>not applicable</td>
</tr>
<tr>
<td><strong>Special condition</strong></td>
<td>not yet decided</td>
<td>not exists</td>
<td>under change</td>
<td>under change</td>
<td>potentially known</td>
</tr>
</tbody>
</table>

Practical solution: No programmer cares; let’s educate the user.
14 (or 20) Kinds of NULL ‘Values’ in Databases (1/2)

1. The property is not applicable for this object but belongs to this class of objects.
   1.1. Independently from the point of time \( t \). "not applicable"
   1.2. At the current point of time \( t \). "currently not applicable"

2. The property does not belong to the object.
   2.1. The property is not representable in the schema.
       2.1.1. Due to changes of value type (temporarily, fuzzy, ...). "many-typed"
   2.2. The property is representable in the schema.
       2.2.1. But there is no value for the object. "unknown"
           2.2.1.1. Because it has not been transferred from another database.
           2.2.1.2. Because is has not yet inserted into the database. "existential null"
       2.2.2. The value for the property exists but is "under change".
           2.2.2.1. However the value is trackable.
               2.2.2.1.1. But is at the moment forbidden.
               2.2.2.1.2. At the moment permitted.
                   2.2.2.1.2.1. But not defined for the database.
                   2.2.2.1.2.2. The value is defined for the system.
                       2.2.2.1.2.2.1. But is currently incorrect.
14 (or 20) Kinds of NULL ‘Values’ in Databases (2/2)

2. The property does not belong to the object.

2.2. The property is representable in the schema.

2.2.2. The value for the property exists but is “under change”.

2.2.2.2. The value is not trackable.

2.2.2.2.1. Because of changes.

2.2.2.2.2. Because of reachability. “place-holder null”

2.2.3. There are several values for the property of this object. “partial null” (2.2.3.1., 2.2.3.2.1, 2.2.3.2.2. similarly to 2.2.2.) “nondeterministic”

2.2.4. There is no value for the property of this object. “not exists”

2.2.5. There is never a value for the property of this object. “never exists”

3. The property is may-be applicable for this object but it unknown whether it is true for the object in this case.

3.1. It is not known whether the property is applicable to the given object. If it is applicable then its value for this property is taken from certain domain. “partial may-be null”

Who can reason with this variety?
**NULL Marker Logics**

\[ \text{Dom}(\text{NULL}) = \{ \text{UNKNOWN}, \text{NOTAPPLICABLE}, \text{NOTEXISTS}, \text{NEVEREXISTS} \} \]

Correspondence of the NULL domain and logical values

<table>
<thead>
<tr>
<th>NULL domain value</th>
<th>Logical value for values</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNKNOWN</td>
<td>\textit{unk}</td>
</tr>
<tr>
<td>NOTAPPLICABLE</td>
<td>\not\in</td>
</tr>
<tr>
<td>NOTEXISTS</td>
<td>\neg !</td>
</tr>
<tr>
<td>NEVEREXISTS</td>
<td>\not\not</td>
</tr>
</tbody>
</table>

Observation: \( \not\not < 0 < 1 \)
# NULL Marker Logics

<table>
<thead>
<tr>
<th>NULL domain value</th>
<th>Logical value for values</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNKNOWN</td>
<td>unk</td>
</tr>
<tr>
<td>NotApplicable</td>
<td>¤</td>
</tr>
<tr>
<td>NotExists</td>
<td>¬!</td>
</tr>
<tr>
<td>NeverExists</td>
<td>赀</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(x \land y)</th>
<th>1</th>
<th>¬!</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>¬!</td>
<td>0</td>
</tr>
<tr>
<td>¬!</td>
<td>¬!</td>
<td>¬!</td>
<td>¬!</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>¬!</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(x \lor y)</th>
<th>1</th>
<th>¬!</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>¬!</td>
<td>1</td>
<td>¬!</td>
<td>¬!</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>¬!</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(x \land y)</th>
<th>1</th>
<th>¬!</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>¬!</td>
<td>0</td>
</tr>
<tr>
<td>¬!</td>
<td>¬!</td>
<td>¬!</td>
<td>¬!</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(x \lor y)</th>
<th>1</th>
<th>¬!</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>¬!</td>
<td>1</td>
<td>¬!</td>
<td>¬!</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>¬!</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(x \land y)</th>
<th>1</th>
<th>unk</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>unk</td>
<td>0</td>
</tr>
<tr>
<td>unk</td>
<td>unk</td>
<td>unk</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(x \lor y)</th>
<th>1</th>
<th>unk</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>unk</td>
<td>0</td>
</tr>
<tr>
<td>unk</td>
<td>unk</td>
<td>unk</td>
<td>unk</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>unk</td>
</tr>
</tbody>
</table>
Negation for NULL Marker Logics

<table>
<thead>
<tr>
<th>NULL domain value</th>
<th>Logical value for values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown</td>
<td>unk</td>
</tr>
<tr>
<td>NotApplicable</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>NotExists</td>
<td>$\neg!$</td>
</tr>
<tr>
<td>NeverExists</td>
<td>$\nexists$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOT</th>
<th>$\neg$</th>
<th>Slupezki</th>
<th>$\sim_3$ $x$</th>
<th>$\sim_3^2$ $x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>L</td>
<td>1</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>L</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

We need at least three kinds of negation!!!

$\wedge, \neg$ is not complete for 3-valued logics

$\wedge, \neg \sim$ is not complete for 3-valued logics

$\wedge, \neg \sim_3$ is not complete for 3-valued logics

$\wedge, \neg, \sim, \sim_3$ is complete for 3-valued logics
Introduction of a Combined Logical Value Lattice

<table>
<thead>
<tr>
<th>NULL domain value</th>
<th>Logical value for values</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNKNOWN</td>
<td>unk</td>
</tr>
<tr>
<td>NOT_APPLICABLE</td>
<td>≠</td>
</tr>
<tr>
<td>NOT_EXISTS</td>
<td>¬!</td>
</tr>
<tr>
<td>NEVER_EXISTS</td>
<td>¬</td>
</tr>
</tbody>
</table>

- information order (infological) with $0 < \text{unk} < 1$
  and $0 < \not\exists < 1$
- Dunn/Belnap’s 4-valued system
  problem 1: $\text{unk} \land \not\exists = 0$
  problem 2: $\text{unk} \lor \not\exists = 1$
  resolution 1: $0 < \not\exists < \text{unk} < 1$
  resolution 2: redefine $\land, \lor$
  resolution 3: introduce two more truth values for $\text{unk} \land \not\exists$ and $\text{unk} \lor \not\exists$
- strict order for $\not\exists$ and $\neg!$
- paraconsistent logics
# Truth Value Table

<table>
<thead>
<tr>
<th>NULL domain value</th>
<th>Logical value for values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unknown</td>
<td>unk</td>
</tr>
<tr>
<td>NotApplicable</td>
<td>( \notin )</td>
</tr>
<tr>
<td>NotExists</td>
<td>( \neg )</td>
</tr>
<tr>
<td>NeverExists</td>
<td>( \nexists )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( x \land / \lor y )</th>
<th>( x )</th>
<th>( \neg y )</th>
<th>0</th>
<th>( \nexists )</th>
<th>unk</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
<tr>
<td>( \neg y )</td>
<td>( \neg y )</td>
<td>( \min / \text{weakmax} )</td>
<td>( \min / \text{weakmax} )</td>
<td>( \min / \text{weakmax} )</td>
<td>( \min / \text{weakmax} )</td>
<td>( \min / \text{weakmax} )</td>
</tr>
<tr>
<td>0</td>
<td>( \min / \text{weakmax} )</td>
<td>0</td>
<td>( \min / \text{max} )</td>
<td>( \min / \text{max} )</td>
<td>( \min / \text{max} )</td>
<td>( \min / \text{max} )</td>
</tr>
<tr>
<td>( \nexists )</td>
<td>( \min / \text{weakmax} )</td>
<td>( \min / \text{max} )</td>
<td>( \nexists )</td>
<td>( \min / \text{max} )</td>
<td>( \min / \text{max} )</td>
<td>( \min / \text{max} )</td>
</tr>
<tr>
<td>unk</td>
<td>( \min / \text{weakmax} )</td>
<td>( \min / \text{max} )</td>
<td>( \min / \text{max} )</td>
<td>( \min / \text{max} )</td>
<td>unk</td>
<td>( \min / \text{max} )</td>
</tr>
<tr>
<td>1</td>
<td>( \min / \text{weakmax} )</td>
<td>( \min / \text{max} )</td>
<td>( \min / \text{max} )</td>
<td>( \min / \text{max} )</td>
<td>( \min / \text{max} )</td>
<td>1</td>
</tr>
</tbody>
</table>

Based on resolution 1 but not on resolution 2 of the connectives problems.

Weakmax: contraction operator for paraconsistent values.
User-Defined Domain Types and Functions for NULL Markers

SQL:1999 already supports flexible management

CREATE DOMAIN BOOLExt AS VARSTRING(14)
  [ DEFAULT value ]
  CHECK (VALUE = 'TRUE' OR VALUE = 'FALSE'
          OR VALUE IS 'UnknownNULL'
          OR VALUE IS 'NotExistNULL'
          OR VALUE IS 'NeverExistNULL'
          OR VALUE IS 'NotAppIINULL');

GO;

CREATE FUNCTION [dbo.]OrExtend (@FirstBool BOOLExt, @SecondBool BOOLExt)
  RETURNS BOOLExt WITH ENCRYPTION AS
  BEGIN
    DECLARE @ResultBool BOOLExt
    ....
    RESULT @ResultBool
  END;

GO;
Treatment in Predicates

SQL:1999 already supports flexible management

- **NOT IN** predicate
  - **KNOWN**: treat as value
  - **UNKNOWN**: treat as NULL marker
  - **NOT APPLICABLE**: remove object from result
  - **NOT EXISTS**: questionable query
  - **NEVER EXISTS**: remove object

- **NOT EXISTS** predicate
  - **KNOWN**: treat as value
  - **UNKNOWN**: treat as NULL marker
  - **NOT APPLICABLE**: treat as empty object
  - **NOT EXISTS**: remove object from result
  - **NEVER EXISTS**: questionable query

### Why Structuring
- **Null marker**
- **CoMoL**
- **Functionality**
- **Views**
- **OLAP**
- **References**

### IS Modelling by HERM
3.7.2012
B. Thalheim
Support for Aggregation

SQL:1999 already supports flexible management

![Diagram of SUM aggregation](image)

- Known
  - use value
- Unknown
  - use expectation value
- NotApplicable
  - use 0
- NotExists
  - questionable sum
- Never Exists
  - questionable sum

Similar for:
- min/max/count
- average
- other distributive, algebraic and holistic aggregation functions
Health Care Information Systems

- People and organisations that are concerned with patients, health care provider organisations, individual practitioners, insurance companies;
- Relationships between parties such as patient relationships and practitioners relationships;
- Types of services and goods available from the health care providers;
- Types of agreements that exist between the various parties;
- Records of health care services performed;
- Claims submitted and the status of the claim;
- Amounts directly owned from the patients as well as payments made by the patients;
- Other supporting information such as accounting information to create the financial statements and human resource information to track personnel.
People and Organisations in Health Care

1/2

Inherent complexity because of the application area ...

Roles: patients, insured individuals, individual health care practitioners, administrators, provider staff support, contact people (insurance company, pharmaceutical company), ...

Other parties: Employer, Supplier, Household, Regulatory Agency, Organisational Unit (Parent Organisation, Subsidiary, Division, Department, and Other Organisation), Internal Organisation, ...

Generic dimensions: Contact, Employee, Party Qualification, Party Skill, Skill Type, Qualification Type

Inherent complexity because of the application area ...


Generic associations: Party Role, Party Relationship, ...

Health care facilities: Hospital, Office, Room, Clinic, ...

Generic facility types: Facility (Medical Building, Ambulatory Surgery Center, Floor, ... Bed, Room), ...

Specific types: Licence within a Geographic Boundary, ...

Generic characterisations: Medical Condition, Physical Characteristic, ...
Reporting Schemata

One of many

Financial analysis: Balance sheets and statement trends allow to determine trends on the income and the profitability over time, incident types, patient types, health care practitioner types etc.

Human resource analysis: Employees can be classified regarding age, gender, marital status, position and other demographic information.

Claims analysis: History of claims and settlements can be classified regarding service codes, types of diagnosis, episode types, geographic areas, dates, and payors. Trend analysis allows an insight what types of health care deliveries have been reimbursed and allows to predict what to expect regarding insurance receipts.

Health care delivery outcome analysis: The outcome of health care deliveries can be analysed under various circumstances.

Health care episode outcome analysis: The outcome of different kinds of health care episodes is of specific interest depending on various circumstances.
Schemata Easily Get Large
## Generic Entity Types

<table>
<thead>
<tr>
<th>Entity Name</th>
<th>Entity Type</th>
<th>Entity Attribute Name</th>
<th>Entity Attribute Required</th>
<th>Entity Attribute Is PK FK</th>
<th>Entity Attribute Is FK</th>
<th>Entity Attribute Domain Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CERTIFICATION REQUIREMENT</td>
<td>Dependent</td>
<td>HEALTH CARE OFFERING ID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>CERTIFICATION REQUIREMENT</td>
<td>Dependent</td>
<td>CERTIFICATION TYPE ID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>CERTIFICATION STATUS</td>
<td>Dependent</td>
<td>STATUS TYPE ID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>CERTIFICATION STATUS</td>
<td>Dependent</td>
<td>HEALTH CARE DELIVERY ID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>CERTIFICATION STATUS</td>
<td>Dependent</td>
<td>STATUS DATE</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Date/time</td>
</tr>
<tr>
<td>CERTIFICATION STATUS TYPE</td>
<td>Dependent</td>
<td>STATUS TYPE ID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>CERTIFICATION TYPE</td>
<td>Independent</td>
<td>CERTIFICATION TYPE ID</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>ID</td>
</tr>
<tr>
<td>CERTIFICATION TYPE</td>
<td>Independent</td>
<td>DESCRIPTION</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Description</td>
</tr>
<tr>
<td>DELIVERY OUTCOME</td>
<td>Dependent</td>
<td>HEALTH CARE DELIVERY ID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>DELIVERY OUTCOME</td>
<td>Dependent</td>
<td>DELIVERY OUTCOME ID</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>ID</td>
</tr>
<tr>
<td>DELIVERY OUTCOME</td>
<td>Dependent</td>
<td>OUTCOME TYPE ID</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>DELIVERY OUTCOME</td>
<td>Dependent</td>
<td>OUTCOME EXPLANATION</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Long varchar</td>
</tr>
<tr>
<td>DIAGNOSIS</td>
<td>Independent</td>
<td>DIAGNOSIS ID</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>ID</td>
</tr>
<tr>
<td>DIAGNOSIS</td>
<td>Independent</td>
<td>HEALTH CARE EPISODE ID</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>DIAGNOSIS</td>
<td>Independent</td>
<td>DIAGNOSIS TYPE ID</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>DIAGNOSIS</td>
<td>Independent</td>
<td>DIAGNOSIS DATE</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Date/time</td>
</tr>
<tr>
<td>DIAGNOSIS TREATMENT</td>
<td>Dependent</td>
<td>DIAGNOSIS ID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>DIAGNOSIS TREATMENT</td>
<td>Dependent</td>
<td>HEALTH CARE DELIVERY ID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>DIAGNOSIS TYPE</td>
<td>Independent</td>
<td>DIAGNOSIS TYPE ID</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>ID</td>
</tr>
<tr>
<td>DIAGNOSIS TYPE</td>
<td>Independent</td>
<td>DESCRIPTION</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Description</td>
</tr>
<tr>
<td>DIAGNOSIS TYPE</td>
<td>Independent</td>
<td>ABBREVIATION</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Short varchar</td>
</tr>
<tr>
<td>DRUG DELIVERY</td>
<td>Dependent</td>
<td>HEALTH CARE DELIVERY ID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>DRUG DELIVERY</td>
<td>Dependent</td>
<td>EPISODE EPISODE ID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>EPISODE OUTCOME</td>
<td>Dependent</td>
<td>HEALTH CARE EPISODE ID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>EPISODE OUTCOME</td>
<td>Dependent</td>
<td>EPISODE OUTCOME ID</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>ID</td>
</tr>
<tr>
<td>EPISODE OUTCOME</td>
<td>Dependent</td>
<td>OUTCOME TYPE ID</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>ID</td>
</tr>
<tr>
<td>EPISODE OUTCOME</td>
<td>Dependent</td>
<td>OUTCOME EXPLANATION</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Long varchar</td>
</tr>
<tr>
<td>EXAMINATION</td>
<td>Dependent</td>
<td>HEALTH CARE DELIVERY ID</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>ID</td>
</tr>
</tbody>
</table>
Star Schema for Person

CREATE TABLE person (  
    person_id       NUMBER NOT NULL,  
    first_name      VARCHAR2(20) NULL,  
    mi             VARCHAR2(20) NULL,  
    gender         VARCHAR2(20) NULL,  
    date_of_birth  DATE NULL,  
    date_of_death  DATE NULL  
);  
ALTER TABLE person  
    ADD ( PRIMARY KEY (person_id) ) ;  
CREATE TABLE person_language (  
    person_language_seq_id NUMBER NOT NULL,  
    person_id            NUMBER NOT NULL,  
    language_code       VARCHAR2(20) NULL  
);  
ALTER TABLE person_language  
    ADD ( PRIMARY KEY (person_language_seq_id, person_id) ) ;  
CREATE TABLE person_address (  
    person_address_seq_id NUMBER NOT NULL,  
    person_id             NUMBER NOT NULL,  
    address_type         VARCHAR2(20) NULL,  
    address_line_1       VARCHAR2(20) NULL,  
    address_line_2       VARCHAR2(20) NULL,  
    city                 VARCHAR2(20) NULL,  
    state                VARCHAR2(20) NULL,  
    zip                  VARCHAR2(20) NULL  
);  
ALTER TABLE person_address  
    ADD ( PRIMARY KEY (person_address_seq_id, person_id) ) ;  
CREATE TABLE person_phone (  
    person_phone_seq_id NUMBER NOT NULL,  
    person_id           NUMBER NOT NULL,  
    phone_type          VARCHAR2(20) NULL,  
    area_code           NUMBER NULL,  
    phone_number        NUMBER NULL,  
    extension           NUMBER NULL  
);  
ALTER TABLE person_phone  
    ADD ( PRIMARY KEY (person_phone_seq_id, person_id) ) ;  
ALTER TABLE person_language  
    ADD ( FOREIGN KEY (person_id)  
          REFERENCES person ) ;  
ALTER TABLE person_address  
    ADD ( FOREIGN KEY (person_id)  
          REFERENCES person ) ;  
ALTER TABLE person_phone  
    ADD ( FOREIGN KEY (person_id)  
          REFERENCES person ) ;

separation of concern
deep normalisation
componentisation
view towers instead of redundant storage
The Star Schema for Health Care Episode Outcome Analysis

- **Episodes**
  - ID, Description, ...
  - (0,1)
  - (0,n)

- **Diagnoses**
  - ID, Description, ...
  - (0,1)
  - (0,n)

- **Incidents**
  - ID, Description, ...
  - (0,1)
  - (0,n)

- **Times**
  - ID, Description, Day, Weak, Month, Quarter, Year
  - (0,1)
  - (0,n)

- **Outcomes**
  - ID, Description, ...
  - (0,1)
  - (0,n)

- **Practitioners**
  - ID, Description, Name(Last, First),
  - (0,1)
  - (0,n)

- **Providers**
  - ID, Description, Name,
  - (0,1)
  - (0,n)

- **Patients**
  - ID, Description,
  - (0,1)
  - (0,n)

- **CareEpisode Fact**
  - Total Charges
  - # Of Episodes
  - # Of Visits
  - # Of Deliveries
  - Avg Length Of Episode
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

http://www.informatik.uni-kiel.de/~thalheim/slides.htm
## 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 for:

1. **Entry**
   - Responsible person of chair
   - Main information entry;
   - *(Classification || categorize || input of other data || input of side conditions);*

2. **Confirmation step**
   - Responsible person and members of chair
   - Proofreading, correction, extension for requests, main and other data

3. **Submission step**
   - Responsible person of the chair
   - Archive in chair folder; send data to caller; publish at chairs internal page
HERM Query Algebra

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

- $src[e, g, \sqcup](\emptyset) = e$
- $src[e, g, \sqcup](\{x\}) = g(x)$ for $x$ of type $t$,
- $src[e, g, \sqcup](X \cup Y) = src[e, g, \sqcup](X) \cup src[e, g, \sqcup](Y)$ for $X, Y$ of type $\{t\}$,

e.g., $\text{filter}(\phi) = src[\emptyset, \text{if}_\phi \text{then}_\phi \text{else} \star (\phi \times \text{single} \times (\text{empty} \star \text{triv})), \cup]$,
- $\text{sum} = src[0, \text{id}, +]$

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 $ev : (t_1 \rightarrow t_2) \times t_1 \rightarrow t_2$
- 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

$$t = b | l | t_1 \times \ldots \times t_n | \{t\} | [t] | l : t$$

that is restricted to rational trees (number of different subtrees is finite)
HERM-Algebra via Structural Recursion

- **Selection**: \( \sigma_\alpha = \text{srec}_\emptyset,\iota_\alpha,\cup \)
  \[ \iota_\alpha(\{o\}) = \begin{cases} \{o\} & \text{if } \{o\} \models \alpha \\ \emptyset & \text{otherwise} \end{cases} \]

- **Projection**: \( \pi_X = T[X] = \text{srec}_\emptyset,\pi_X,\cup, X \in T \)

- **(Natural) Join**: \( \Join = \text{srec}_\emptyset,\Join_T,\cup \)
  \[ \Join_T (\{(o_1, o_2)\}) = \{ o \in \text{Dom}(T_1 \cup T_2) | o|_{T_1} = o_1 \land o|_{T_2} = o_2 \} \]

*Cartesian product and intersection* as special cases

- **Renaming**: \( \rho_{X,Y} = \text{srec}_\emptyset,\rho_{X,Y},\cup \)
  \[ \rho_{X,Y}(\{(o)\}) = \{ o' \in \text{Dom}((T\setminus X)\circ Y) | o|_{T\setminus X} = o'|_{T\setminus X} \land o|_X = o'|_Y \} \]

- **Nesting**: \( \nu_X = \text{srec}_\emptyset,\nu_{X,\{X\}},h_2 \) for \( X \in T = R \),
  \[ T' = C(R\setminus X)\cup_R \{X\}, o'|_X \in \pi_X(T'^\mathcal{C}) \]
  \[ h_2(\{o'\}, T'^\mathcal{C}) = \{ o' \} \cup T'^\mathcal{C} \text{ if } o'|_X \notin \pi_X(T'^\mathcal{C}) \]
  \[ h_2(\{o'\}, T'^\mathcal{C}) = \{ o \in \text{Dom}(T') | \exists o' \in T'^\mathcal{C} : o|_{R\setminus X} = o'|_{R\setminus X} \land o(X) = \{ o''[X] | o'' \in T'^\mathcal{C} \land o'|_{R\setminus X} = o''|_{R\setminus X} \} \} \]

- **Unnesting**: \( \mu_X = \text{srec}_\emptyset,\mu_{X,\{X\}},h_2 \)
  \[ T' = C(R\setminus \{X\})\circ_X \]
  \[ h_2(\{o'\}, T'^\mathcal{C}) = \{ o' \} \cup \{ o \in \text{Dom}(T') | \exists o'' \in R^\mathcal{C} : o[R\setminus \{X\}] = o''[R\setminus \{X\}] \land o|_X \in o''|_X \} \]
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?
VisualSQL versus SQL

SELECT P1.Name, P1.BirthDate, P1.Address, 
P2.Name AS ‘Name of supervisor’
FROM Person P1, Professor P2, Student S1, Supervisor, Lecture L, Enroll E
WHERE P1.Name = Student.Name AND P1.BirthDate = Student.BirthDate
AND S1.StudNo = E.StudNo
AND E.Result NOT NULL
AND S1.StudNo = Supervisor.StudNo
AND Supervisor.Name = Professor.Name
AND Supervisor.BirthDate = Professor.BirthDate
AND P2.Name = Professor.Name AND P2.BirthDate = P2.BirthDate
AND L.Name = Professor.Name AND L.BirthDate = Professor.BirthDate
AND L.CourseNo
IN (SELECT E2.CourseNo
       FROM Enroll E2
       WHERE S1.StudNo = E2.StudNo AND
           E2.Result NOT NULL )
AND E.CourseNo
IN (SELECT L2.CourseNo
       FROM Lecture L2
       WHERE L2.Name = P2.Name AND
           L2.BirthDate = P2.BirthDate );
SQL is Easy to Read, to Develop and to Understand? Of Course, for Everybody!!!

What does this query?

```
SELECT P1.Name, P2.Name
FROM Person P1, Person P2, Student S1, Student S2, Enrol H1, Enrol H2
WHERE P1.Name = S1.Name AND P1.DateOfBirth = S1.DateOfBirth AND
S1.StudNo = H1.StudNo AND H1.Grade IS NOT NULL AND
P2.Name = S2.Name AND P2.DateOfBirth = S2.DateOfBirth AND
S2.StudNo = H2.StudNo AND H2.Grade IS NOT NULL
AND NOT EXISTS
  (SELECT *
   FROM Enrol H3
   WHERE H3.Grade IS NOT NULL AND
     H3.StudNo NOT IN
     (SELECT H4.StudNo
      FROM Enrol H4
      WHERE H4.StudNo = H2.StudNo
        AND H4.Grade IS NOT NULL)
AND NOT EXISTS
  (SELECT *
   FROM Enrol H5
   WHERE H5.Grade IS NOT NULL AND
     H5.StudNo NOT IN
     (SELECT H6.StudNo
      FROM Enrol H6
      WHERE H6.StudNo = H1.StudNo
        AND H4.Grade IS NOT NULL)
     AND H2.StudNo = H5.StudNo)
AND S1.StudNo < S2.StudNo
GROUP BY P1.Name, P2.Name;
```
Which students complete courses all the time together?

Person P1
- Name
- DateOfBirth

Student S1
- Name
- DateOfBirth

Enrol
- StudNo
- Semester
- CourseNo
- Grade

Student S2
- Name
- DateOfBirth

Enrol
- StudNo
- Semester
- CourseNo
- Grade

Which students complete courses all the time together?

- Person P1 (Name, DateOfBirth)
- Student S1 (Name, DateOfBirth)
- Enrol (StudNo, Semester, CourseNo, Grade)
- Student S2 (Name, DateOfBirth)
- Enrol (StudNo, Semester, CourseNo, Grade)

- Person P1 (Name, DateOfBirth) < Student S1 (Name, DateOfBirth)
- Student S2 (Name, DateOfBirth) = Student S1 (Name, DateOfBirth)
- Student S2 (Name, DateOfBirth) = Student S1 (Name, DateOfBirth)

far simpler and easier to formulate, to capture, to understand without the SQL burden

http://www.informatik.uni-kiel.de/en/information-systems-engineering/miscellaneous/visualsql/
Yet Another Resulting Query

SELECT P1.Name, P2.Name
FROM Person P1, Person P2, Student S1, Student S2, Enrol H1, Enrol H2
WHERE P1.Name = S1.Name AND P1.DateOfBirth = S1.DateOfBirth AND
S1.StudNo = H1.StudNo AND H1.Grade IS NOT NULL AND
P2.Name = S2.Name AND P2.DateOfBirth = S2.DateOfBirth AND
S2.StudNo = H2.StudNo AND H2.Grade IS NOT NULL
AND NOT EXISTS
  (SELECT *
   FROM Enrol H3
   WHERE H3.Grade IS NOT NULL AND
     H3.StudNo NOT IN
     (SELECT H4.StudNo
      FROM Enrol H4
      WHERE H4.StudNo = H2.StudNo
      AND H4.Grade IS NOT NULL)
AND NOT EXISTS
  (SELECT *
   FROM Enrol H5
   WHERE H5.Grade IS NOT NULL AND
     H5.StudNo NOT IN
     (SELECT H6.StudNo
      FROM Enrol H6
      WHERE H6.StudNo = H1.StudNo
      AND H4.Grade IS NOT NULL)
     AND H2.StudNo = H5.StudNo)
AND S1.StudNo < S2.StudNo
GROUP BY P1.Name, P2.Name;

How long would it take you to formulate this query?

Is the first query in this talk equivalent to this query?
The VisualSQL Tool
The VisualSQL Tool
Abstracting from HERM Programs to Diagrams in the Business Process Modelling & Notation

Why Structuring
Null marker
CoMoL
Functionality
Views
OLAP
References

Content
Information
Concept
Topic
Ex.: Paper Submission Review System

[Diagram showing a process flow for paper submission review system]
Ex.: Paper Submission Review System
Yet an Example of a Workflow Diagram

A BPMN diagram taken from literature with many problems
BPMN Assumptions

- separation of the specification into workflow process and workflow process instance
- singleton isolatable process instance bound to its token
- inter-process collaboration only through messages and events
- hidden resource dependences
- swimlanes for different roles of users
- pools for views on process sets
- separation of nodes

\[ \text{Node} = \text{Activity} \cup \text{Event} \cup \text{Gateway} \]

- separation of events

\[ \text{Event} = \text{StartEvent} \cup \text{IntermEvent} \cup \text{EndEvent} \]

- tasks comprehend only some of possible executions

\[ \text{TaskType} = \{ \text{Service, User, Receive, Send, Script, Manual, Reference, None} \} \]

- rigid localisation \(\Rightarrow\) avalanches of side constraints
  \[ \downarrow \downarrow\text{context-sensitiv functions}, \]
  \[ \downarrow \downarrow\text{none-incremental semantics}, \]
  \[ \downarrow \downarrow\text{goto jumps, token progression may become “arbitrary”} \]
Resulting Orchestration Problems

Communication through explicit specification of protocols and services: interprocess communication exclusively by messages and links.

Coordination only implicitly by message protocols or links.

typical coordination problems:
- opportunities, obligations, permissions and forbids must intentionally be taken into consideration by the modeller
- contracts among parties
- exceptional cases, timeouts

Cooperation between processes only through messages.

typical cooperation problems:
- Zachman enactment: *who*, *what*, *when*, *where*, *how*, *why*
- rights, obligations and roles

Data dependencies among processes become a sideway task for the modeller.
Implicit BPMN Assumptions

Petri net illustration for token flow results in messy problems

- limitations of Petri nets
- negative token with runtime overwriting of semantics of constructs
- token colouring
- token with history transfer for sub-processes

Process instance duplication for each call of a process with optimisation by elimination of dead paths

Incorporation of other standards in rather fuzzy way: interchange of messages, SOAP, UDDI, web services, web transaction, XML (XPath, XPDL, XSchema, ..), ...

Concentration on workflow processes leaving out control among process instance and data flow among them

Partial BPEL transformation with reference to BPEL meaning
Formalisation of the Business Process Modelling & Notation

see

my theory talk

Event Process Chains can be handled in a similar way

Activity and other dynamic UML diagram languages ...
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** \( \text{GENERAL PRESENTATION STYLE} \)

& **Abstraction (Granularity, measure, precision)**

& **Orders within the presentation**

& **Hierarchical representations**

& **Points of view**

& **Separation**

**browsing definition** \( \text{CONDITION} \)

& **Navigation**

**functions** \( \text{SEARCH FUNCTIONS} \)

& **Export functions**

& **Input functions**

& **Session functions**

& **Marking functions**
Views: Archiv view

XML as the display medium

XML specifications can be automatically generated out of this view
Algebraic expressions for views

Views for internet presentation as Read-Only-View

Archiv view as materialized Read-Only-View

Slice SS00/01 with Archiv.Semester := e(Semester)

for \( e = \sigma_{Bezeichn="SS00/01"}

Archiv.Course := e(CourseHeld [Course])

Archiv.Person := e(CourseHeld[plannedCourse[

proposedCourse[Responsible4Course : Person]])

Program, Kind, Professor analogous

Archiv.CourseHeld := e(CourseHeld[plannedCourse[

proposedCourse [ Course, Program, Teacher:Professor,

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

- **required Course**: retrieve
- **Course**: retrieve, select, input
- **Semester**: retrieve
- **Professor**: retrieve, select (default = Profβ)
- **Person**: retrieve
- **Room**: retrieve, select
- **Program**: retrieve, select
- **Kind**: retrieve, select
- **Proposal**: input
- **Wish**: insertedBy = "SecrKK"
- **Chair**: retrieve

*Description = "SS01/02"*
*ShortDescr = "DBIS"*
*Responsible4Course = "β"*
*Time(Proposal, SideConditions)*
Outlook: Media Types, Media Object Suites

- Raw media types = \((cont(M), sup(M), view(M), op(M))\)
  content type \(cont(M)\), set of supertypes \(sup(M)\),
  \(view(M) = Q(S_{inp}, S_{outp})\) HERM view
  generic functions \(op(M)\) for changing the database
- Attached operations: (signature, selection type, body)
  selection type - supertype of \(cont(M)\)
  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 = \((cont(C), layout(C), kind(C))\)
  for shipping and representation
Pitfalls and Misunderstandings of Micro-/Meso-/Macrodata

Snodgrass Example for Temporality Problems

Awful queries, overwhelming complexity

Microdata of the Temporal Database
Micro-data versus Macro-data

The right level of data granularity

Typical Snodgrass query: “Give the History of Lots being co-resident in a Pen”

```sql
select L1.Lot_Id_num, L2.Lot_Id_Num, L1.Pen_Id, L1.From_Date, L1.To_Date
from Lot_Loc as L1, Lot_Loc as L2
where L1.Lot_Id_num < L2.Lot_Id_num and L1.Fdyd_Id = L2.Fdyd_Id
and L1.Pen_Id = L2.Pen_Id and L1.From_Date = L2.From_Date
and L1.To_Date <= L2.To_Date
union
select L1.Lot_Id_num, L2.Lot_Id_Num, L1.Pen_Id, L1.From_Date, L1.To_Date
from Lot_Loc as L1, Lot_Loc as L2
where L1.Lot_Id_num < L2.Lot_Id_num and L1.Fdyd_Id = L2.Fdyd_Id
union
select L1.Lot_Id_num, L2.Lot_Id_Num, L1.Pen_Id, L2.From_Date, L1.To_Date
from Lot_Loc as L1, Lot_Loc as L2
where L1.Lot_Id_num < L2.Lot_Id_num and L1.Fdyd_Id = L2.Fdyd_Id
and L1.Pen_Id = L2.Pen_Id and L1.From_Date > L1.From_Date
and L2.To_Date <= L1.To_Date;

select distinct L1.Lot_ID, L2.Lot_ID, R1.Pen_ID, R2.From, min(R1.To, R2.To)
from Cattle C1, Cattle C2, Resides R1, Resides R2, Lot L1, Lot L2
where L1.Lot_ID = C1.BelongsTo and L2.Lot_ID = C2.BelongsTo and
R1.Pen_ID = R2.Pen_ID and R1.From <= R2.From and
R2.From < R1.To and L1.Lot_ID <> L2.Lot_ID;
```
Paradoxes like for Cantor set theory
but this time statistical and structural paradoxes

Aggregation against semantics through roll-up, drill-down, dicing and slicing
independence of properties is assumed but not true

There are three kinds of lie: lies, damned lies and statistics.
attributed to Benjamin Disraeli (1804-81) by Mark Twain (Autobiography, 1924)

Forgetful schema transformation due to properties of aggregation functions

Ignoring properties of type systems e.g. granularity, nulls, defaults, interpretations
The Hierarchy Paradox

Cube A: Participants per Lecture and Day

Cube B: Usage of Rooms per Day

Scheduling Schema on University and Evening Lectures
Hierarchies in the Example 2: 1

- day
  - day inside term
    - morning
      - date of university lectures
  - day outside term
    - evening
      - date of evening lectures
Hierarchies in the Example 2: 2

- Working day
  - Morning
  - Day inside term
    - Date of university lectures
  - Day outside term
    - Date of evening lectures at working days
- Non-working day
  - Evening
  - Day

References

- Why
- Structuring
- Null marker
- CoMoL
- Functionality
- Views
- OLAP
## Cube B Data in the Example 2

<table>
<thead>
<tr>
<th>Day category</th>
<th>Room utilization</th>
<th>Daytime</th>
<th># of rooms</th>
</tr>
</thead>
<tbody>
<tr>
<td>working</td>
<td>occupied</td>
<td>morning</td>
<td>20</td>
</tr>
<tr>
<td>working</td>
<td>occupied</td>
<td>evening</td>
<td>12</td>
</tr>
<tr>
<td>working</td>
<td>free</td>
<td>morning</td>
<td>10</td>
</tr>
<tr>
<td>working</td>
<td>free</td>
<td>evening</td>
<td>4</td>
</tr>
<tr>
<td>non-working</td>
<td>occupied</td>
<td>morning</td>
<td>2</td>
</tr>
<tr>
<td>non-working</td>
<td>occupied</td>
<td>evening</td>
<td>8</td>
</tr>
<tr>
<td>non-working</td>
<td>free</td>
<td>morning</td>
<td>6</td>
</tr>
<tr>
<td>non-working</td>
<td>free</td>
<td>evening</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Daytime</th>
<th>Working day</th>
<th>Non-working day</th>
<th>All days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evening</td>
<td>75 %</td>
<td>33 %</td>
<td>50 %</td>
</tr>
<tr>
<td>Morning</td>
<td>67 %</td>
<td>25 %</td>
<td>58 %</td>
</tr>
</tbody>
</table>
# Cube B Data in the Example 2

<table>
<thead>
<tr>
<th>Day category</th>
<th>Room utilization</th>
<th>Daytime</th>
<th># of rooms</th>
</tr>
</thead>
<tbody>
<tr>
<td>working</td>
<td>occupied</td>
<td>morning</td>
<td>20</td>
</tr>
<tr>
<td>working</td>
<td>occupied</td>
<td>evening</td>
<td>12</td>
</tr>
<tr>
<td>working</td>
<td>free</td>
<td>morning</td>
<td>10</td>
</tr>
<tr>
<td>working</td>
<td>free</td>
<td>evening</td>
<td>4</td>
</tr>
<tr>
<td>non-working</td>
<td>occupied</td>
<td>morning</td>
<td>2</td>
</tr>
<tr>
<td>non-working</td>
<td>occupied</td>
<td>evening</td>
<td>8</td>
</tr>
<tr>
<td>non-working</td>
<td>free</td>
<td>morning</td>
<td>6</td>
</tr>
<tr>
<td>non-working</td>
<td>free</td>
<td>evening</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Daytime</th>
<th>Working day</th>
<th>Non-working day</th>
<th>All days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evening</td>
<td>75 %</td>
<td>33 %</td>
<td>50 %</td>
</tr>
<tr>
<td>Morning</td>
<td>67 %</td>
<td>25 %</td>
<td>58 %</td>
</tr>
</tbody>
</table>
### Weighted Cube B Data in the Example 2

<table>
<thead>
<tr>
<th>Day category</th>
<th>Room utilization</th>
<th>Daytime</th>
<th># of rooms</th>
</tr>
</thead>
<tbody>
<tr>
<td>working</td>
<td>occupied</td>
<td>morning</td>
<td>20</td>
</tr>
<tr>
<td>working</td>
<td>occupied</td>
<td>evening</td>
<td>12</td>
</tr>
<tr>
<td>working</td>
<td>free</td>
<td>morning</td>
<td>10</td>
</tr>
<tr>
<td>working</td>
<td>free</td>
<td>evening</td>
<td>4</td>
</tr>
<tr>
<td>non-working</td>
<td>occupied</td>
<td>morning</td>
<td>2</td>
</tr>
<tr>
<td>non-working</td>
<td>occupied</td>
<td>evening</td>
<td>8</td>
</tr>
<tr>
<td>non-working</td>
<td>free</td>
<td>morning</td>
<td>6</td>
</tr>
<tr>
<td>non-working</td>
<td>free</td>
<td>evening</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Daytime</th>
<th>Working day</th>
<th>Non-working day</th>
<th>All days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evening</td>
<td>(75 %, $\frac{16}{30}$)</td>
<td>(33 %, $\frac{24}{30}$)</td>
<td>(50 %, $\frac{24}{30}$)</td>
</tr>
<tr>
<td>Morning</td>
<td>(67 %, $\frac{30}{30}$)</td>
<td>(25 %, $\frac{8}{30}$)</td>
<td>(58 %, $\frac{30}{30}$)</td>
</tr>
</tbody>
</table>

The hierarchy paradox
The Simpson Paradox

Revisited but based on a classical OLAP example

<table>
<thead>
<tr>
<th>model</th>
<th>Chevy</th>
<th>Ford</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>color</td>
<td>blue</td>
<td>white</td>
<td>blue</td>
</tr>
<tr>
<td>year</td>
<td>90</td>
<td>91</td>
<td>90</td>
</tr>
<tr>
<td>count</td>
<td>187</td>
<td>155</td>
<td>131</td>
</tr>
</tbody>
</table>

count(*) by model, color, year (data under independence constraint)

Ratios 'Market Share Percentages of Chevy'

\[
p(\text{chevy}|\text{blue}, 90) = \frac{187}{187 + 217} \cdot 100 = \frac{187}{304} \cdot 100 = 46%
\]

\[
p(\text{chevy}|\text{blue}, 91) = \frac{155}{335} \cdot 100 = \frac{155}{335} \cdot 100 = 46%
\]

\[
p(\text{chevy}|\text{white}, 90) = \frac{131}{282} \cdot 100 = \frac{131}{282} \cdot 100 = 46%
\]

\[
p(\text{chevy}|\text{white}, 90) = \frac{108}{233} \cdot 100 = \frac{108}{233} \cdot 100 = 46%
\]

Market share of sold units where model = ‘chevy’
constant for both colors over years 90-91
The Simpson Paradox

Revisited but based on a classical OLAP example

<table>
<thead>
<tr>
<th>model</th>
<th>Chevy</th>
<th>Ford</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>90</td>
<td>91</td>
<td>90</td>
</tr>
<tr>
<td>count</td>
<td>187</td>
<td>155</td>
<td>217</td>
</tr>
</tbody>
</table>

\[
p(\text{chevy, 90}) = \frac{308}{686} \cdot 100 \approx 46\%
\]

\[
p(\text{chevy, 91}) = \frac{263}{568} \cdot 100 \approx 46\%
\]

cohere:nt with the results before

global independence assumption (integrity constraint) on the data space spanned by the attributes ‘model’, ‘year’ and ‘color’, cite
The Simpson Paradox
Revisited but based on a classical OLAP example

<table>
<thead>
<tr>
<th></th>
<th>model</th>
<th>Chevy</th>
<th>Ford</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tm:</td>
<td>count</td>
<td>581</td>
<td>673</td>
<td>1254</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ty:</td>
<td>year</td>
<td>90</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>count</td>
<td>739</td>
<td>515</td>
<td>1254</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tc:</td>
<td>color</td>
<td>blue</td>
<td>white</td>
</tr>
<tr>
<td></td>
<td>count</td>
<td>687</td>
<td>567</td>
<td>1234</td>
</tr>
</tbody>
</table>

Tabelle 1: Marginal tables Tm, Ty, Tc used to generate ??

insert and delete operations

<table>
<thead>
<tr>
<th></th>
<th>model</th>
<th>Chevy</th>
<th>Ford</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>color</td>
<td>blue</td>
<td>white</td>
<td>blue</td>
</tr>
<tr>
<td></td>
<td>year</td>
<td>90</td>
<td>91</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>count</td>
<td>255</td>
<td>156</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>222</td>
<td>175</td>
<td>1254</td>
</tr>
</tbody>
</table>

count (*) by model, color, year
The Simpson Paradox

Revisited but based on a classical OLAP example

\[
p(\text{chevy}|\text{blue}, 90) = \frac{255}{429} \cdot 100 \approx 59\%
\]
\[
p(\text{chevy}|\text{blue}, 91) = \frac{156}{258} \cdot 100 \approx 60\%
\]

Market share of a blue car of type ‘chevy’ increases slightly over years.
Increase of the share is stronger for white cars:

\[
p(\text{chevy}|\text{white}, 90) = \frac{88}{310} \cdot 100 \approx 28\%
\]
\[
p(\text{chevy}|\text{white}, 91) = \frac{82}{257} \cdot 100 \approx 32\%
\]
The Simpson Paradox

*Revisited but based on a classical OLAP example*

Slice now over ‘color’, \( \pi_{\text{year, count}} \)

<table>
<thead>
<tr>
<th>model</th>
<th>Chevy</th>
<th>Ford</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>90</td>
<td>91</td>
<td>90</td>
</tr>
<tr>
<td>count</td>
<td>324</td>
<td>238</td>
<td>396</td>
</tr>
</tbody>
</table>

\[ \text{count (\* ) by model, year} \]

\[ p(\text{chevy, 90}) = \frac{308}{686} \cdot 100 \approx 46\% \]

\[ p(\text{chevy, 91}) = \frac{263}{568} \cdot 100 \approx 46\% \]

**The bad guy:**

\[ Z \text{ (color)} \]

\[ X \text{ (year)} \quad Y \text{ (model)} \]

Dependency graph XYZ with separator (influential) attribute Z
Explaining the Simpson Paradox

- Changing meaning of attributes: Chevys has in 90 and 91 an equal ratio
  
  but there are more white cars

- Problems of slicing:
  
  - Blue slice: Chevys ratio is in 91 better than 90
  - White slice: Chevys ratio is in 91 better than 90

Based on both slices: Chevys ratio is in 91 better than 90

- Not told (but valid):
  
  Overall selling is decreasing from 90 to 91

  Reason why Chevys are better: lower relative decrease
Other Problematic OLAP Applications

Non-commutative operators: Let $O$ be a given set of numeric operators. Let $o_1 \in O$ be a linear operator and $o_2 \in O$ a non-linear operator. Then it is generally not true that $o_1 \circ o_2 = o_2 \circ o_1$.

Perfect aggregation: The perfect homomorphism $H : \hat{y} = y$ exists if and only if $\hat{y} = S^T A T \vee A = S^+ A T^+$ where $S^+(T^+)$ is the Moore-Penrose inverse of $S(T)$.

Summarizability and cover properties

Extensions of These Cases

Non-confluence of aggregations

Identifiability and inclusion/exclusion computation
Generalising the Unique Name Assumption

<table>
<thead>
<tr>
<th>population</th>
<th>Place</th>
<th>Inhabitants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Municipality</td>
<td>Region</td>
</tr>
</tbody>
</table>

### Why Structuring

Null marker

CoMoL

Functionality

Views

OLAP

References

---

π_{Region, #Inhabitants}(R^C) \models \{ Region \} \rightarrow \{ #Inhabitants \}

σ_{Municipality=′Raisdorf′}(R^C) \models \{ Region \} \rightarrow \{ #Inhabitants \}

meaning of attributes is different:

- #Inhabitants as the number of inhabitants of certain municipality
- #Inhabitant as the number of inhabitants in a certain region

---

courtesy by A. Molnar

π_{Kreis, Einwohner}(K) \rightarrow \{ Region \} \rightarrow \{ #Inhabitants \}

σ_{Municipality=′Raisdorf′}(P) \rightarrow \{ Region \} \rightarrow \{ #Inhabitants \}

---

courtesy by A. Molnar
**Scoping and Hierarchies Mismatches**

*Discoveries of A. Molnar*

- **Why** Structuring
- **Null marker**
- **CoMoL**
- **Functionality Views**
- **OLAP**
- **References**

**Solution:** Dependency bases and existence hierarchies
Lessons Learned

(1) Explicit treatment of basic data types
   - domains and equivalences
   - basic operations and their semantics
   - basic predicates and their semantics

(2) Explicit definition of aggregation functions
   - behaviour and properties of aggregation functions
   - combinations of aggregation functions
   - distributive, algebraic and holistic aggregation functions

(3) Definition of cubes
   - explicit definition of aggregations
   - explicit handling of equivalence classes
   - definition of operations
**Basic Data Types**

Extended basic data type $B = (\text{Dom}(B), \text{Op}(B), \text{Pred}(B), \Upsilon)$

algebraic system (Malzev!) (homomorphism are different)

Equivalence relations $eq$ on $\text{Dom}(B)$

Each of these equivalence relations defines a

**partition** $\Pi_{eq}$ of the domain into equivalence classes

Equivalence classes $c$ may be named by $n_c$

Named partitions by $\Pi^*$

- trivial named partition $(\bot^*, \text{fine})$
- top named partition: $(\top^*, B)$

Equivalence relations and partitions may be ordered

$\bot^* \leq \Pi^* \leq \top^*$
The Lattice of One Basic Data Type

\[ B = (\text{Dom}(B), \text{Op}(B), \text{Pred}(B), \Upsilon) \]

Canonical order of partitions on \( \text{DOM}(B) \): \( \Pi^* \leq \Pi'^* \)

for all \( (c, n_c) \) from \( \Pi^* \) there exists one and only one element \( (c', n_{c'}) \in \Pi'^* \) such that \( c \subseteq c' \).

Majority order \( (c, n_c) \leq_m^{\text{choice}} (c', n_{c'}) \):

either \( |c \cap c'| > \max \{|c \cap c''| \mid (c'', n_{c''}) \in \Pi'^*, c'' \neq c'\} \)

or \( (c', n_{c'}) \in \Pi'^* \) is determined by

a (deterministic) choice operator among

\[ \{c^+ \in \Pi'^* \mid |c \cap c^+| = \max \{|c \cap c''| \mid (c'', n_{c''}) \in \Pi'^*\}\} \].

Omit the choice operator in \( \leq_m^{\text{choice}} \) if context determines

\[ \text{DateTime: } eq_{\text{hour}}, eq_{\text{day}} \]

partitions \( \bot^*, \text{Days, Weeks, Months, Quarters, Years, and } \top^* \)

\[ \text{Days } \leq \text{Months } \leq \text{Quarters } \leq \text{Years.} \]

\[ \text{Weeks } \leq_m \text{Months} \]
Combination of Lattices

Is Application-Dependent

Combination of

- characterisations (time, region, ...)
- characterisations and entity types (sales within a region)

- Germany: Municipality - Region - County - Country  
  versus Store - Chain area - Chain subregion - Sales region
- Hungary: Telespülés - Kistérség - Megye - Régió - Country versus Store
- US: City - State -Country versus Store - Sale region
- Mexico: City - State versus Store - Sale region (States are parts)
- Canada: City - Province versus Store - Sale region (Provinces are parts)
- different meanings of characterisations, e.g. time, business time, 
  academic time, accounting time

overlay, multiway partitioning, naming conventions, attributes suffering by aggregation, 
structuring, navigation facilities, constraints

errors in union and intersection: without derived attributes (unless they are already algebraic)
join must by redefined
Aggregation

Aims and assumptions

Grouping of data, records, objects based on selections

Summarisability depending on completeness and disjointness assumptions

Detection of misleading aggregations and prevention from misapplication

Invariance properties of aggregation operators

Harmonisation of aggregation and abstraction without forgetful handling

Preservation or transformation of semantics or explicit semantics transformation

Derivation of repair functions in the case of damages
Aggregation Functions: General Definition

Family $\mathcal{F} = \{f_0, \ldots, f_k, \ldots, f_\omega\}$

$f_k : \text{Bag}_k \rightarrow \text{Num}$ (maps a bag on $\text{dom}(M_j)$ with $k$ elements to a numerical range $\text{Num}$)

Minimum preserving: $f_k(\text{min}, \ldots, \text{min}) = \text{min}_{\text{Num}} \text{min} \in \text{dom}(M_j)$

Maximum preserving: $f_k(\text{max}, \ldots, \text{max}) = \text{max}_{\text{Num}} \text{max} \in \text{dom}(M_j)$

Monotone according to the order of $\text{dom}(M_j)$ and $T$ ($k \geq 1$)

Idempotent: $f_k(x, \ldots, x) = x$ for all $x \in \text{dom}(M_j)$

Continuous: $\lim_{x_i \rightarrow x} f(x_i) = f(x)$ for all sequences $x_i$ of size $k$

Lipschitz property: $|f_k(x_1, \ldots, x_k) - f_k(y_1, \ldots, y_k)|$

$\leq c \sum_{i=1}^{n} |x_i - y_j|$ for some constant $c$

Symmetric: $f_k(x_1, \ldots, x_k) = f_k(x_{\rho(1)}, \ldots, x_{\rho(k)})$ for any $k$-permutation $\rho$

Self-identical: $f_k(x_1, \ldots, x_k) = f_{k+1}(x_1, \ldots, x_k, f_k(x_1, \ldots, x_k))$

Shift-invariant: $f_k(x_1 + b, \ldots, x_k + b) = f_k(x_1, \ldots, x_k) + b$

Homogeneous: $f_k(bx_1, \ldots, bx_k) = bf_k(x_1, \ldots, x_k)$

Additive: $f_k(x_1 + y_1, \ldots, x_k + y_k) = f_k(x_1, \ldots, x_k) + f_k(y_1, \ldots, y_k)$

Associative: $f_r(f_{k_1}(\overline{x_1}), \ldots, f_{k_r}(\overline{x_r})) = f_{k_1+\ldots+k_r}(\overline{x_1}, \ldots, \overline{x_r})$
Aggregation Functions: Properties

Be careful with the average functions used for OLAP

- max, min are minimum preserving, maximum preserving, idempotent, continuous, symmetric, self-identical, additive, homogeneous, associative, and obey the Lipschitz property,
- sum is 1-minimum preserving, 1-maximum preserving, 1-monotone, continuous, symmetric, homogeneous, additive, associative, obeys the Lipschitz property, and
  not idempotent, not self-identical, not shift-invariant,
- avg is minimum preserving, maximum preserving, 1-monotone, idempotent, continuous, symmetric, shift-invariant, homogeneous, additive, obeys the Lipschitz property,
  is not self-identical, not associative,
- count is continuous, symmetric, associative, obeys the Lipschitz property,
  not idempotent, not self-identical, not shift-invariant, not homogeneous, not additive.
Classes of Aggregation Functions: Distributive (Inductive) Functions

preserving partitions of sets

\( \forall f \exists g \ f(X) = g(f(X_1), \ldots, f(X_n)) \)

\( X = X_1 \cup X_2 \cup \ldots \cup X_n, \ X_i \cap X_j = \emptyset, \ i \neq j \)

types \( T, T' \), collection type \( C^T \) on \( T \), operations \( \cup_{C^T}, \cap_{C^T}, \emptyset_{C^T} \) on \( C^T \)

\( h_0 \in T' \quad h_1 : T \rightarrow T' \quad h_2 : T' \times T' \rightarrow T' \)

\( srec_{h_0, h_1, h_2}(\emptyset_{C^T}) = h_0 \)

\( srec_{h_0, h_1, h_2}(\{s\}) = h_1(s) \quad \text{for} \quad |\{s\}| \)

\( srec_{h_0, h_1, h_2}(R_1^C \cup_{C^T} R_2^C) = h_2(srec_{h_0, h_1, h_2}(R_1^C), srec_{h_0, h_1, h_2}(R_2^C)) \)

\( \text{iff } R_1^C \cap_{C^T} R_2^C = \emptyset_{C^T} \)

- \( \text{sum} = srec_{0, Id, +} \quad \text{for relations without nulls} \)

\( \text{sum}^{null}_0 = srec_{0, h_{Id}^0, +} \quad h_{f}^0(s) = \begin{cases} 0 & \text{if } s = \text{NULL} \\ f(s) & \text{if } s \neq \text{NULL} \end{cases} \)

\( \text{sum}^{null}_\text{undef} = srec_{0, h_{Id}^{\text{undef}}, +} \quad h_{f}^{\text{undef}}(s) = \begin{cases} \text{undef} & \text{if } s = \text{NULL} \\ f(s) & \text{if } s \neq \text{NULL} \end{cases} \)

- \( \text{count} = srec_{0, 1, +} \quad \text{count}^{null}_1 = srec_{0, h_1^0, +,} \quad \text{count}^{null}_{\text{undef}} = srec_{0, h_1^{\text{undef}}, +} \)

\( \text{max} = srec_{\text{NULL}, Id, max} \quad \text{max}_{\text{null}} = srec_{0, h_1^0, +,} \quad \text{max}_{\text{undef}} = srec_{0, h_1^{\text{undef}}, +} \quad \text{min} = srec_{\text{NULL}, Id, min} \quad \text{for sets, bags} \)

max, max_{undef} are different functions
Classes of Aggregation Functions: Algebraic Functions

Finite algebraic expressions based on distributive functions

- **average**:

  $\text{average} = \frac{\text{sum}}{\text{count}}$

- **variance** ($\sum_k (x_k - \text{EX})^2 \cdot p_k$, where $\text{EX} = \sum_k x_k \cdot p_k$)

- **ratio values** (e.g. consumption)

  Most of them can be extended to distributive functions

  - e.g. $\text{avg}_{0,1}^{\text{null}}$ based on ordering on pairs ($value$, $occ#$)
Classes of Aggregation Functions: Holistic Functions

the vast majority of functions
no storage bound for computing sub-aggregates

mostFrequent, n-mostFrequent
statistical and financial median
rank
averageDeviation

geometric key-area computations
computable over temporal views (on temporal views ...)
distinction between raw data and ‘data’
maintenance of identification becomes the main issue
data warehouse data
versions of data, temporality
Invariance of Functions

Given: aggregation function \( \psi \)

database function \( f \), e.g., view function, selection function

Problem: does the application of \( f \) change the aggregation?

Can we delay computation of aggregation to the view?

Is there a need to compute results based on raw data only?

\[
\psi(D) = \psi(f(D))
\]

\( f \) is \( \psi \)-invariant in \( D \) if
\[
\psi(f(D)) = \psi(D)
\]
Invariance Examples

- **Bag2Set**: (DISTINCT)
  - min-, max-invariant
  - not sum-invariant
    - sum-invariant for sets
  - not avg-invariant
    - avg-invariant for sets without NULL’s

- **Project**: invariant for all distributive functions
- **Select, Join**: not invariant for most aggregation functions
- **Union, Difference, Intersection**: mostly not invariant
- **Renaming**: invariant for all aggregation functions
Repairing Approaches

1. Aggregation Transformation

Problem: find the $f$-transformation $\Psi_f$ of $\psi$

\[
\Psi_f (f(D)) = F(\psi(D))
\]
1. Aggregation Transformation

Problem: find the $f$-transformation $\Psi_f$ of $\psi$

infeasible
Repairing Approaches

2. Preparing Explicit Aggregation

\[ g - \psi\text{-supplement of } f \]

\[ \exists \psi^* \forall D \quad \psi^*(g(f(D), D)) = \psi(D) \]

Supposition: \( g \) is based on the mapping of algebraic functions to distributive functions
Observation Supporting the Supposition

supplement function which generates the occurrence number for each value
for instance for
\( f = \text{Bag2Set} \)

- \( \text{sum}^* = srec_{0,h,+} \) for \( h((v,n)) = n \cdot v \)
- \( \text{count}^* = srec_{0,h',+} \)
  for \( h((v,n)) = n \), \( \text{count} = srec_{0,1,+} \)
- \( \text{avg}^{null}_{0,1} = \frac{\text{sum}^{null}_{0}}{\text{count}^{null}_{1}} \)
  extended to
- \( \text{avg}^{null*}_{0,1} = \frac{\text{sum}^{null*}_{0}}{\text{count}^{null*}_{1}} \)
Behavior of Aggregation Functions

Subset sensitivity: For given $\mathcal{D}$, $\psi$ and a selection condition $\alpha$:

$$\psi(\mathcal{D}) \neq \psi(\sigma_\alpha(\mathcal{D}))$$

All classical aggregation functions are subset-sensitive.

Critical operations: selection, join, intersection, difference

Object invariance: All objects are residing after application of the operation.

Object-invariant functions are $\min$- and $\max$-invariant.

Value-set invariance: All values of the domain considered are remaining.

Set-value invariant functions are $\min$- and $\max$-invariant.
Behavior within Hierarchies

Poly-Hierarchy Lattices

generalizing the characteristic function $h$ to Boolean lattices of classes $C_1, \ldots, C_m$:

$L(C_{i_1}, \ldots, C_{i_n})$ - intersection $C_{i_1}, \ldots, C_{i_n}$

counting based on the exclusion/inclusion property:

$$
\psi(D) = \sum_{i=1}^{m} \psi(C_i) - \sum_{j=1}^{m-1} \sum_{k=j+1}^{n} \psi(L(C_j, C_k)) + \ldots \\
+ (-1)^{r-1} \sum_{1 \leq j_1 < \ldots < j_r \leq m} \psi(L(C_{j_1}, \ldots, C_{j_r})) + \ldots \\
+ (-1)^{m-1} \psi(L(C_1, \ldots, C_m))
$$

Distributive aggregation functions can be computed based on the exclusion/inclusion principle.

If an abstraction function $f$ is distributive with the exclusion/inclusion principle then $f$ is $\psi$-invariant for distributive aggregation functions.
Algebraic Aggregation Functions

Handling Through Distributive Functions

\[ D \xrightarrow{\text{Aggregation function}} \psi(D) \]

Extended aggregation function

\[ \psi^*(D) \]

Forgetful functor

Example: average

\[ h_1(\{s\}) = (1, s) \]

\[ h_2((i, k)(j, l)) = (i + j, \frac{i \cdot k + j \cdot l}{i + j}) \]
Supplements for Algebraic Functions

Given a database abstraction $f$ and an algebraic aggregation function $\psi$. If a $\psi$-supplement $g$ of $f$ exists then the function $\psi^*$ is distributive.

Collapseability along MVD Trees

Collapsing along MVD trees is invariant for distributive functions. Collapsing along MVD trees is invariant for algebraic functions.

Term algebras might not exist

Distributive functions can be concatenated. Algebraic functions are not concatenatable.
Data Cube Definition (1)

Grounding schema of a cube: (cube) relationship type
\[ R = (R_1, \ldots, R_n, \{(A_1, q_1, f_1), \ldots, (A_m, q_m, f_m)\}) \]

- hierarchical types \( R_1, \ldots, R_n \)
  forming component (or dimension) types
- ("fact") attributes \( A_1, \ldots, A_m \)
  defined over extension based data types
  instantiated by singleton-value queries \( q_1, \ldots, q_m \)
- aggregation functions \( f_1, \ldots, f_m \) defined for \( A_1, \ldots, A_m \)
  defined over \( B_1, \ldots, B_n \)

typically defined by a view over a database schema

Notice: also hidden attributes
Data Cube Definition (2)

grounding schema $R = (R_1, ..., R_n, \{ (A_1, q_1, f_1), ..., (A_m, q_m, f_m) \})$
class $R^C$,
partitions $\Pi_i$ on $\text{DOM}(R_i)$ for any component $R_1, ..., R_n$

cell of $R^C$: non-empty set $\sigma_{R_1 \in c_1, ..., R_n \in c_n}(R^C)$
for $c_i \in \Pi_i$ and $\sigma_\alpha$

named partitions $\Pi^*_1, ..., \Pi^*_n$ for all component types

cube $\text{cube}^ {\Pi^*_1, ..., \Pi^*_n}(R^C)$ on $R^C$ and on $\Pi^*_i$, $1 \leq i \leq n$:

$\{ \sigma_{R_1 \in c_1, ..., R_n \in c_n}(R^C) \neq \emptyset \mid c_1 \in \Pi_1, ..., c_n \in \Pi_n \}$
of cells of $R^C$
for named partitions $\Pi^*_i$, $1 \leq i \leq n$

If $\Pi^*_i = \top^*$ then we may omit the partition $\Pi^*_i$
Essentials and Differences of the Cube

OLAP cube definition

- flat hierarchies (e.g. time instead of time equivalence)
  instead of heterogeneous, application dependent deep hierarchies
- attributes
  - hidden (invisible in the cube) attributes
  - aggregation functions \( f_{B_1}^{e_{j_1}},...,f_{B_n}^{e_{j_n}}(R^C) \)
  - semantics and meaning changes after OLAP operations
- partiality of data existence and cube breaks through
- dependency preserving operations
Operations of the Data Cube

cube \( \text{cube}^{\Pi_1^*, \ldots, \Pi_n^*}(R^C) \)

on \( R = (R_1, \ldots, R_n, \{(A_1, q_1, f_1), \ldots, (A_m, q_m, f_m)\}) \)
given for a dimension \( i \)
and partitions \( \Pi_i^* \preceq \Pi_i' \preceq \top_i^* \)

Basic drill-down functions map \( \text{cube}^{\Pi_1^*, \ldots, \Pi_i^*, \ldots, \Pi_n^*}(R^C) \) to 
\( \text{cube}^{\Pi_1^*, \ldots, \Pi_i^*, \ldots, \Pi_n^*}(R^C) \).

Basic roll-up functions map \( \text{cube}^{\Pi_1^*, \ldots, \Pi_i^*, \ldots, \Pi_n^*}(R^C) \) to 
\( \text{cube}^{\Pi_1^*, \ldots, \Pi_i^{\prime*}, \ldots, \Pi_n^*}(R^C) \).

Basic slice functions map \( \text{cube}^{\Pi_1^*, \ldots, \Pi_n^*}(R^C) \) to 
\( \sigma_\alpha(\text{cube}^{\Pi_1^*, \ldots, \Pi_n^*}(R^C)) \).

Basic dice functions map \( \text{cube}^{\Pi_1^*, \ldots, \Pi_n^*}(R^C) \) to 
\( \text{cube}^{\Pi_1^*, \ldots, \top_i^*, \ldots, \Pi_n^*}(R^C) \).
Operations of the Data Cube

cube \( \Pi_1^\ast, \ldots, \Pi_n^\ast (R^C) \) on \( R = (R_1, \ldots, R_n, \{(A_1, q_1, f_1), \ldots, (A_m, q_m, f_m)\}) \)
a dimension \( i \), \( \Pi_i^\ast \leq \Pi_i^\ast \leq T_i^\ast \)

Roll-up functions are the inverse of drill-down functions.

Basic slice functions are definable through cells:

\[
\text{dimension}(\alpha) \colon \text{set of all dimensions that are restricted by } \alpha
\]

\[
\sigma_{\alpha}^\bigcap(c_i) = \begin{cases} 
\emptyset & \text{if } R_i \in \text{dimension}(\alpha) \land \sigma_{\alpha}(c_i) \neq c_i \\
c_i & \text{otherwise}
\end{cases}
\]

Eager slice functions restrict cube cells to those that entirely fulfill the selection criterion \( \alpha \),

\[
\{\sigma_{R_1 \in \sigma_{\alpha}^\bigcap(c_1), \ldots, R_n \in \sigma_{\alpha}^\bigcap(c_n)}(R^C) \neq \emptyset \mid c_1 \in \Pi_1, \ldots, c_n \in \Pi_n\}
\]

Liberal slice functions restrict cells to those that partially fulfill the selection criterion \( \alpha \),

\[
\{\sigma_{R_1 \in \sigma_{\alpha}(c_1), \ldots, R_n \in \sigma_{\alpha}(c_n)}(R^C) \neq \emptyset \mid c_1 \in \Pi_1, \ldots, c_n \in \Pi_n\}.
\]

Basic dice functions similar to projection

full roll-up functions
### Constraints and Operations on the Cube

<table>
<thead>
<tr>
<th>Student</th>
<th>Identity</th>
<th>Personal</th>
<th>Study characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>StudNo</td>
<td>Address</td>
<td>Major</td>
<td>Semester registered</td>
</tr>
</tbody>
</table>

... ... .... .... ....

different meaning of constraints and thus different behaviour:

\[
\{ \text{StudNo} \} \rightarrow \{ \text{Address} \}
\]

\[
\{ \text{StudNo} \} \rightarrow \{ \text{Major} \}
\]

\[
\{ \text{StudNo} \} \rightarrow \{ \text{Semester\_registered} \}
\]

\[
\text{count(dice}^{\text{Semester\_registered}}(R^C))
\]

semester overall for students

\[
\text{count(dice}^{\text{StudNo}}(R^C))
\]

# of students per semester, address(?) major

\[
\text{count(dice}^{\text{Semester}}(\text{dice}^{\text{StudNo}}(R^C)))
\]

# for address, major combinations

134
Extended Cube Schemata

given cube $cube^{\Pi_1^*, \ldots, \Pi_n^*}(R^C)$ on

$$R = (R_1, \ldots, R_n, \{(A_1, q_1, f_1), \ldots, (A_m, q_m, f_m)\})$$

the cube operation $O$

maps $cube^{\Pi_1^*, \ldots, \Pi_n^*}(R^C)$ to

$$O(cube^{\Pi_1^*, \ldots, \Pi_n^*}(R^C))$$

new cube schema

$$R = (R_1, \ldots, R_n, \{(A_1, q_1, f_1), \ldots, (A_i, q_i, f_i, O(A_m, q_m, f_m))\})$$

recording the history of computation through expressions
Special Issues

Multi-layered semantics depending on operations and changing the meaning of attributes

Forgetting semantics: projection results in independence instead of hidden dependence

Hidden structures are necessary for correctness
are also essential for correct computation
e.g. algebraic functions
roll-up then by approximation functions
rough values

Hidden auxiliary functions and structures e.g. for repair
Some General Results

Roll-up functions are neither sum-invariant nor avg-invariant in general.

Roll-up functions are not min- or max-invariant in general.

Rearrangement functions are min-, max-, sum- and avg-invariant.

Summarizability cannot be obtained for relations after application of any subset-generating function.

Summarizing is correct for union of classes if an exclusion constraint is valid for classes to which union is applied.

Summarizability cannot be maintained for functions after application of which identification of objects is lost.

Summarizability cannot be obtained through collapsing hierarchies.
Some General Extensions

Summarizability cannot be obtained moving up to generalizations of classes.

Summarizability is maintained moving downwards through functional dependencies.

Summarizability is correct on separating collapsing hierarchies if the portion is used for the supplement.

Summarizability can be maintained on the basis of identification properties of values.
The Good Message

EER Functions Enable Complete OLAP

- operations on views (selection as *dice*, projection as *slice*, generalized relational algebra)
- calculation functions, visualization functions
- group-by as *roll-up* (drill-up), un-nest as *drill-down*
- schema restructuring operation for unfold (e.g., drill-down), fold (nest), classification

Modelling Assumptions: Drill-Down Functions

Decomposing groups of data along a hierarchy, refine grouping

Observation 1.
*Drill-down functions are well defined if the cube construction is based on disjointness and completeness modelling assumptions.*

Observation 2.
*Drill-down functions are well defined if data granularity is guaranteed at leaf level $L_1$ and no structural null are used at any level $L_i$ ($i > 1$) in between.*
Modelling Assumptions: Roll-Up Functions

used for combining groups along a hierarchy, i.e., for fusion of groups.
Problematic for collapsing hierarchies especially in the case of algebraic and holistic aggregation functions

Observation 3. Roll-up functions are only well-defined if data granularity is guaranteed at leaf level $L_1$ and no structural null are used at any level $L_i$ ($i > 1$) in between.

Observation 4. Roll-up functions must be query-invariant, i.e. for the roll-up function $f$ and the query function $\psi$:
$$\psi(\bar{x}_1, \ldots, \bar{x}_n) = \psi(f(\bar{x}_1), \ldots, f(\bar{x}_n)) .$$

Observation 5. The Simpson paradox is observed if for groups at roll-up level $L_p \leq L_r$
$$\forall j (f(G_{ij}) < f(G_{kj})) \not\Rightarrow f(\sum_{j=1}^{n} G_{ij}) < f(\sum_{j=1}^{n} G_{kj})$$
Modelling Assumptions: Dice Functions

projection, marginalization (statistics), \( \sum \) of values
unproblematic for distributive functions combinable with repairing functions

Observation 6.
The Simpson paradox is observed if for groups at roll-up level \( L_p \leq L_r \)
\( (\forall j (f(G_{ij}) < f(G_{kj}))) \nRightarrow f(\sum_{j=1}^{n} G_{ij}) < f(\sum_{j=1}^{n} G_{kj}) \)

Observation 7.
Dice functions can only correctly be applied if the cube construction is based on union invariance, i.e. \( \text{aggr}(\bigcup^*_{o \in G_i} \text{value}(o)) = \bigcup^*_{o \in G_i} (\text{aggr}(\text{value}(o))) \) for groups \( G_i \) along dimensions.
If \( f \) is distributive then the \( \bigcup^* \equiv f \).
If \( f \) is algebraic then repair functions must be applied.

Observation 8.
Dice functions can only be used along dimensions for which constraints among cube dimensions are not lost, i.e. \( \Sigma|_{\pi(X)} = \pi(X)(\Sigma) \).

Observation 9.
Dice functions must be based on disjointness and completeness modelling assumptions.
Modelling Assumptions: Slice Functions

Requirement for applicability: query-invariance, for the roll-up function $f$ and the query function $\psi$:

$$\psi(\bar{x}_1, ... \bar{x}_n) = \psi(f(\bar{x}_1), ..., f(\bar{x}_n)).$$

Observation 10.

Slice functions must be subset invariant.

Constraints invalidated by subset construction are those integrity constraints that have to be expressed through $\forall\exists$-constraints, e.g., inclusion dependencies, multivalued dependencies, tuple-generating constraints.
Modelling Assumptions: OLTP-OLAP Transformations

Modelling pragmatism, e.g.,
- instead of arithmetic average function: mean average, weighted average, root mean square, average mean, moving average, weighted average
  very sensitive to extreme values such as outliers and may be distorted by them
- arithmetic averages may be normalized by skew functions

Averages are measures of central tendency. The median is the middle-most value in an ordered set.

Observation 11.
Application of median instead of mean average functions for aggregation leads to more stable OLAP query operations.

Observation 12.
Harmonic mean functions \( \sum_{i=1}^{n} \frac{1}{x_i} \) are shift-invariant, additive, symmetric, continuous, and homogeneous.

Observation 13.
Geometric mean functions \( n \sqrt{x_1 \cdot x_2 \cdot \ldots \cdot x_n} \) provide a better picture for relative scales among values and are OLAP query invariant.

Observation 14.
The sampling frequency must be based on the Sampling Theorem and the generalizations of Nyquist.
Publications on Science and Art of Conceptual Modelling

Publications on Model Suites, Evolution, Migration

Publications on Tool-Based Development

Publications on Pattern Development

Publications on Component Development