The Conceptual Framework to Multi-Layered Database Modelling

EJC’09, Maribor

June 2, 2009

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

Visions, results, projects

(1) Modelling and Babylonian language chaos
   Order and revise

(2) Model suites
   Seriosity instead of hype jumping

(3) Foundations
   Mathematics, engineering, application domains

(4) Multi-layered modelling
   Abstraction layers and mapping

(5) Information systems modelling
   Separation of concern
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
Deployment Diagram
Component Diagram
Composite Structure Diagram

Interaction Diagrams

Sequence Diagram
Communication Diagram
Timing Diagram
Interaction Overview Diagram

Overview
Multi-Modelling
Model Suite
Foundation
Layering
Database Modelling
Concluding

Content
Information
Concept
Topic

©B. Thalheim
Coherence of UML Diagram Clusters

inherent integrity constraints in diagrams

\[ EC_1^{states(SC,CT)} : StateChart(States) \subseteq ClassDiagram(\pi_X(RulingClass)) \]

\[ State' \not\in ClassDiagram(\pi_X(RulingClass)) \rightarrow F \text{ modify}(StateChart(State, State')) \]

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

\[ \text{do}(Agent_1, \text{modify(ClassDiagram(RulingClass, X))) \bowtie} \]

\[ \text{do(notify(Agent_2, modify(ClassDiagram(RulingClass, X)))} \]

\[ StateChart(\{IsBorrowed, IsReturned\}) \subseteq ClassDiagram(\pi_{LendingState}(Book)) \]
# Languages-Based Information Systems Modelling: State Of The Art So Far

<table>
<thead>
<tr>
<th>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 &amp; 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
Varieties and Dimensions of Models

Aspects

Behavior
Function
Structure

Abstraction level

Simulation Decision

Aim

Overview
Multi-Modelling
Model Suite
Foundation
Layering
Database Modelling
Concluding

Concept Topic

Content Information

Varieties and Dimensions of Models

- Abstraction level
- Theory
- Formal
- Graphical
- Physical
- Statements
- Demo
- Experiment
- Aim
- Decision
- Simulation
Separation of Concern: The Zachman Framework

Coherence, calibration, mapping problems
The Solution: Model suites

- Definitions (at the first glance)
  Example for achieveability
- Refinement (for dynamic systems (case: computer systems))
- Example: Structural and functional integration (case: database and information systems)
- Hierarchical model suites (towards a handling strategy)
- Collaboration of models (towards handling consistency)
- Explicit contracts
- Co-evolution

and a resulting theory with its challenges
Model suites: The Program

Handling abstraction

Explicit collaboration of models based on
- constructors
- mappings
- contracts among models

Dimensions of models based on the minimalisation of models and constructors

Abstractions of models among mappings

Constructors for construction of new models
- shuffle product, reduct, scope, integration

Theory extension for model context representation and context integration
Success Story: Model Suites for Heart Modelling

5-layer model of the heart

Genes layer: networks, based on molecular functions

Proteins: elementary units, chemistry, and their composition

Cell structure: functions, key organisational unit; with biological processes; pathway models

Tissue: structure and function, with cellular components

Body: myocardian activation

time range: \(10^{15}\)

space range: \(10^9\)

©Peter Hunter, see too: International Institute for Theoretical Cardiology
http://www.bioeng.auckland.ac.nz/People/people_display.php?people_id=353
Success Story: Model Suites for Heart Modelling

Multi-scale modelling: 5-layer model of the heart

Tissue: structure and function, with cellular components
based on mathematical models for structures and geometry
  • mechanics,
  • kinematics,
  • equations (equilibrium, constitutive),
  • conditions (e.g. boundary),
  • factors (e.g. stress)
myocardian activation
coupled with electro-mechanics, e.g. energy flow
modelling cell processes
  • electro-physical
  • proton and biocarbonat
  • calcium
  • myofilament

ⒸPeter Hunter, see too: International Institute for Theoretical Cardiology
30 years of heart modelling conference July 2009
Success Story: Model Suites for Heart Modelling

Multi-scale modelling: 5-layer model of the heart

• Biophysics of nerve and muscle: cable theory, ionic currents, Hodgkin-Huxley equations, muscle models (anatomy, contraction, sliding filament theory, energetics), fading memory model (finite duration length step, force step response)

• Cardiac electrophysiology: cardiac cells, units, diFrancesco-Noble model, membrane models, bidomain model

• Electrocardiography: cardiac anatomy and function, activation, body surface potential mapping, transfer matrices, myocardial inverse procedure, normal and abnormal ECG

Example cell.ml and field.ml

• ontologies, content models

• molecular function, biological process, cellular component

Lesson Learned: Model suites

Handling abstraction

Structure and association based on
- multi-layered models
- constructors and languages

Topology and geometry based on
- topological space
- (homo-)morphisms and equivalence

Algebra and evolution based on equations, development rules, and restrictions
- mass balance
- charge balance
- osmotic balance
- thermodynamic balance (e.g. feasibility)

Abstraction and refinement based on
- mappings
- informorphisms
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 $\mathcal{C}$
$\Sigma_{\mathcal{S},\mathcal{C}}$ well-formedness conditions

Model type $\mathcal{T}_{\mathcal{L}_S} = (\mathcal{L}_S, \Sigma_{\mathcal{L}_S})$
language of the model and
constraints $\Sigma_{\mathcal{L}_S} \in \mathcal{L}(\Sigma_{\text{WellFormed}})$

Partial mappings $\mathcal{R}_{i,j} : \mathcal{L}_{S_i} \rightarrow \mathcal{L}_{S_j}$ 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 \( ST = (T_{L_{S_1}}, ..., T_{L_{S_n}}, \Sigma_{L_{S_1}}, ..., \Sigma_{L_{S_n}}) \)

model types \( T_{L_{S_i}} \) defined on a set \( L_{S_i}, ..., L_{S_n} \)
\( \Sigma_{S_1}, ..., \Sigma_{S_n} \) constraints

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

Contract on \( C \):

- constraints \( \Sigma_{L_{S_1}} \cup ... \cup \Sigma_{L_{S_n}} \cup \Sigma_{L_{S_1}}, ..., \Sigma_{L_{S_n}}, \)
- 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

Overview
Multi-Modelling
Model Suite
Foundation
Layering
Database Modelling
Concluding

Concept
Information
Topic

©B. Thalheim
Collaboration within model suites

Binding, integration, and calibration of models

Structural and functional integration among layers with the

- bindings for export and import interfaces of models
- provisioning of data and schemata
- exchange and mapping facility

Coordination with model suites for consistency management responsibilities, obligations, permissions and protection

Co-evolution of model suites with evolution choreography, configuration, versioning, orchestration
Collaboration of models

Explicit and eager bindings for models in suites

Export/import interfaces \( S = (I, F, \Sigma_S) \) specifying

Infon models \( I = (V, M, \Sigma_T) \) specifying
the content \( V \) based on media types (“what”),
the collaboration manager \( M \) (“how”), and
the competence \( \Sigma_T \) through a set of tasks (“for what”)

Collaboration characteristics \( F \) specifying the organization frame (“how”),
the parties (“who”) and the context (“whereby”)

Quality of collaboration \( \Sigma_S \) 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
Computational Refinement

Given two abstractions $M, M^*$, refinement is based on refinement of states
states of interest $S, S^*$, correspondence between the states of interest
abstract computation segments $\tau_1, \ldots, \tau_m$ on $M$ and $\sigma_1, \ldots, \sigma_n$ on $M^*
(m,n)$-refinement
locations of interest
equivalence relation $\equiv$ on locations of interest

$M^*$ is a correct refinement of $M$ if
there for each $M^*$-run $S_0^*, \ldots, S_k^*$, ... there is an $M$-run and sequences $i_0 < i_1 < \ldots$
and $j_0 < j_1 < \ldots$ such that $i_0 = j_0 = 0 S_{i_k} \equiv S_{j_k}^*$ for each $k$ and either
- both runs terminate and their final states are the last pair of equivalent states,
or
- both runs and both sequences are infinite.

Complete refinement: $M$ correct refinement of $M^*$ and $M^*$ correct refinement of $M$
Constituents of Modelling

Development of decisions and strategies

Aims of the integrated model and target outcome
  criteria of fitting
  similarity models

Integration and mapping of models depending
  on their correlation and cohesion

Detection, description, experimentation of laws with main
  players, stability factors, entailers

Evolution of ecosystems with explicit extraction of control parameters that effect or drive evolution
# Layers of Models for IS

Managing model suites by stratification

<table>
<thead>
<tr>
<th>Collaboration specification layers</th>
<th>Infrastructure</th>
<th>Collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application domain layer</td>
<td>Application environment</td>
<td>Collaboration policy, principles, acts</td>
</tr>
<tr>
<td>Requirements layer</td>
<td>System sketches, requirements, system decisions</td>
<td>Collaboration tasks, contracts, style, pattern</td>
</tr>
<tr>
<td>Business user layer</td>
<td>System view, parties, portfolio</td>
<td>Collaboration stories</td>
</tr>
<tr>
<td>Conceptual layer</td>
<td>Information system specification, context support</td>
<td>3C-C schemata, informational processes, exchange frames</td>
</tr>
<tr>
<td>Logical layer</td>
<td>Information system - logical view</td>
<td>Collaboration supporting system</td>
</tr>
<tr>
<td>Physical layer</td>
<td>IS programs</td>
<td>Collaboration programs</td>
</tr>
<tr>
<td>Deployment layer</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Hierarchical Layered Model Suites

Typical example for models at the same abstraction level

---

**Content**

- explicit inheritance of underlying data
- ownership principle
- explicit explanation based on underlying data
- agreed stratification of data and schemata

---

**Overview**

Multi-Modelling

**Model Suite**

Foundation

Layering

Database Modelling

Concluding

---

© B. Thalheim
Layers as a Building Block of Multi-Layered Architectures

Generalising architectures

Architecture of a multi-layer model suite

- Operators: $O_{i+1,1}, ...$
- Data objects: $t_{i+1,1}, ...$

Layer $i$
- Operators: $O_{i,1}, ...$
- Data objects: $t_{i,1}, ...$

Layer $i+1$
- "uses"
- $O_{i+1,p}(t_{i+1,q})$
- $O_{i,r}(t_{i,1}, ..., t_{i,k}), ..., O_{i,s}(t_{i,1}, ..., t_{i,m})$

+ Layer language
Dimensions of Data Modelling: Profile and Portfolio, Abstraction and Extension, and Quality of Data

Data aggregation, abstraction and enrichment

- founded annotated aggregated macro-data
- annotated aggregated macro-data
- macro-data
- micro-data

Application A

Data profile and portfolio

Application B

Gossip/raw/sensor/source data
Staged/cleansed data with data profiles
Consolidated and transformed data with hooks for data change capture
Integratable data ready for on-demand use in federations
Coherent data enhanced by services for in-line delivery in enterprise data farms

Data quality improvement
Principles of Multi-Layered Modelling

Downward-dependency principle: The main data dependency structure is top-down. Objects at a higher level depend on objects at a lower level.

Upward-notification principle: Objects at a higher level act as subscribers to database changes at lower level. They may decide whether they eagerly or lazily enforce observed changes at lower level. Objects at lower level report however their changes.

Neighbor-communication principle: Objects may exchange data only at the same layer with other objects. The neighborhood may also require that neighboring databases should be synchronised.

Explicit association principle: The data exchange between databases is explicitly documented and recorded. Whenever a database at a higher level perceives data from a lower level then this exchange is logged.

Cycle elimination principle: Cyclic data exchange between layers is broken based on the log information.

Layer naming principle: Data belong to their level and can be identified at their level.
Constituents of Modelling

Development of decisions and strategies

Aims of the integrated model and target outcome
criteria of fitting
similarity models

Integration and mapping of models depending
on their correlation and cohesion

Detection, description, experimentation of laws with main
players, stability factors, entailers

Evolution of ecosystems with explicit extraction of control parameters that effect or drive evolution
Establishing a Model

Phases

- Clarification phase
- Model construction phase
- Model experimentation phase, e.g. based on simulation methods
- Model optimisation phase
- Model validation phase
- Model application phase e.g. for decision making
Structural and Functional Integration of Models

Mappings and actual data mappings (synonyms, homonyms, model suite hypernyms and hyponyms)

- partiality of data mapping, abstractions
  - Microdata as the starting point for collaboration
  - Mesodata through abstractions, filtering, scoping, summarisation
  - Macrodata for model injection

Handling missing, intentionally not available, not applicable, biased and low quality data

Informorphism among different equivalent presentations

Function integration as generic functions depending on the model

Constructor integration depending on the data profile and on the task portfolio
Realisability?!: Life Cases

Multi-Layered Modelling By Model Suites
June 2, 2009
B. Thalheim

Overview
Multi-Modelling
Model Suite
Foundation
Layering
Database Modelling
Concluding

Concept
Topic

Content
Information
Realisability?!!: Life Cases
Concluding and Summarising

Messages and Results

- Modelling and Babylonian language chaos
  Order and revise

- Model suites
  Seriosity instead of hype jumping

- Foundations
  Mathematics, engineering, application domains

- Multi-layered modelling
  Abstraction layers and mapping

- Information systems modelling
  Separation of concern