Conceptual Modeling for Information Systems

2. Information Systems Development and the Role of Conceptual Models

Marco Montali

KRDB Research Centre
Faculty of Computer Science
Free University of Bozen-Bolzano

A.Y. 2012/2013
Development of an IS: problem-solving.

1. **Analysis** (includes conceptual modeling): what the IS is about, what are the requirements.

2. Design (includes logical modeling): how to accomplish the requirements.

3. Implementation: coding of the design under specific architectural/technological choices.

4. Testing: check if the implementation works and meets the requirements.

5. Maintenance: assist users after release, keep the IS working.
IS Engineering Processes

Waterfall

Analysis
Design
Implementation
Test
Maintenance

Iterative

Analysis
Design
Implementation
Test
Maintenance

Analysis
Design
Implementation
Test
Maintenance

Analysis
Design
Implementation
Test
Maintenance

XP

Analysis
Design
Implementation
Test
Maintenance

Analysis
Design
Implementation
Test
Maintenance

Analysis
Design
Implementation
Test
Maintenance
Incremental Iterative Approach

Divide et impera.
Refined Steps

1. Feasibility study: is the idea implementable?
2. **Requirements analysis**: what should the system do?
3. **Conceptual design - data, processes**: what is the conceptual schema modeling the UoD?
4. **Logical design - data, processes**: how can the conceptual schema be translated into a logical schema?
5. Basic physical design - data and processes: how can the logical schema be represented in a concrete management system?
6. Basic external design - data and processes: which information can be accessed by users, and how?
7. Prototyping: how does the IS look like?
8. Completion of design.
10. Testing and validation: does the IS work well and satisfy the requirements?
12. Maintenance.
Requirements analysis

First delineation of the IS to be.

- Relevant documentation examined.
- Meetings with domain experts, intended users, policy makers, stakeholders.
- Prioritization of the next steps.
- Output: requirements specifications document that clearly describes functional/non-functional requirements, and sketches an initial conceptual schema of the IS to be.
  - Contract!

To conceptual design...

Requirements specifications document

Initial conceptual schema

- Initial UoD analysis
- Partial conceptual schema
- Requirements specifications document
- Use cases, scenarios,

Joint sessions with domain experts and stakeholders

Requirements elicitation & analysis

Domain (UoD)
Interaction with Domain Experts

M. C. Escher - Up and down

Umberto Boccioni - Visioni simultanee
Development of the structural conceptual schema.

- Continuous involvement of domain experts.
  - Definition of a glossary of terms.
- Incremental iterative approach.
  - Start from the initial structural conceptual schema.
  - Iterate...

1. Split UoD into (overlapping) sub-areas, with priority.
2. Generate/refine the conceptual schema of each area.
3. Integrate the sub-schemas into a global conceptual schema.
Logical/Physical Design without Explicit Processes

Development of a typical 4-tier architecture.

- Mirrors in the physical design.
- Processes embedded in the application logic.
- Two similar logical information schemas, two similar physical databases:
  - transient - data logic of the application (typically: OO).
  - persistent - data logic of the persistence layer (typically: relational).
Object-Relational Mapping

- Maps the object-oriented logical schema to the relational one.
  - Thanks to mapping meta-data (established by the modeler).
- Acts as a mediator between the application and the persistence layer.
- Mappings used to automatically synchronize the transient OO database with the persistent relational one.
  - Changes on the OO databases rewritten as update statements over the underlying relational database.
  - High-level queries over the OO schema rewritten as queries over the relational schema.
  - Query execution over the OO database materialize the data by taking them from the physical relational database.
Zachman’s Framework
Partitioning of the IS abstract architecture.

- Vertical partitioning: levels of abstraction.
- Horizontal partitioning: aspects/concerns.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>goal list</td>
<td>material list</td>
<td>process list</td>
<td>event list</td>
<td>organizational unit&amp;role list</td>
<td>geographical locations list</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Conceptual</th>
<th>goal relationship</th>
<th>entity relationship model</th>
<th>process model</th>
<th>event model</th>
<th>organizational unit&amp;role rel. model</th>
<th>locations model</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Logical</th>
<th>rules diagram</th>
<th>data model diagram</th>
<th>process diagram</th>
<th>event diagram</th>
<th>role relationship diagram</th>
<th>locations diagram</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Physical</th>
<th>rules specification</th>
<th>data entity specification</th>
<th>process function spec.</th>
<th>role specification</th>
<th>location specification</th>
<th>event specification</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Detailed</th>
<th>rules details</th>
<th>data details</th>
<th>process details</th>
<th>event details</th>
<th>role details</th>
<th>location details</th>
</tr>
</thead>
</table>

- Logical and physical layers can be split into transient and persistent.
- Mappings between levels to be considered.
Modeling Languages???

- Conceptual structural schema
- Logical OO schema
- Logical relational schema
- O-R mapping

Diagram:

- Conceptual structural schema
- Logical OO schema
- Logical relational schema

Question marks indicate open questions or areas of discussion.
Conceptual Modeling Languages: Criteria

Expressibility: the measure of what can be modeled.

100% principle

The language should be able to express all the relevant static and dynamic aspects of the UoD.

- Remember: the conceptual schema is the knowledge component of the IS.
- 100% principle → completeness: the conceptual schema captures all the required knowledge.
- Completeness → quality.
- Correctness:
  - Syntactic: conformance to the language meta-model.
  - Semantic: knowledge of conceptual schema is relevant and true in the domain.
- Completeness is a goal, correctness is a requirement!
Conceptual Modeling Languages: Criteria

Clarity: how easy the language can be understood and used (by different stakeholders).

- **Graphical** vs textual notations.
- The language must be unambiguous: **formal foundation**.
- The more expressive the language, the more difficult is to retain clarity.
- Less expressive languages require complex combinations of their few constructs.

- **Abstraction**: remove unnecessary details. Use requirements to drive abstraction.
- **Simplicity**: Prefer simple schemas. Follow *Occam’s razor* with a critical approach.
- **Orthogonality**: minimization of the overlapping of language constructs. Their (in)dependence must reflect the one of the corresponding domain aspects.
Conceptual Modeling Languages: Criteria

**Semantic relevance**: modeling of conceptually relevant aspects only.

**Conceptualization principle**

A conceptual model should only include conceptually relevant aspects of the UoD, excluding all aspects of external/internal data representation, physical data organization and access as well as aspects of particular external user representation such as message formats, data structures, etc.

- Again, simplicity!
- **Semantic stability**: how well the model retains its original intent in the face of domain or requirements changes.
- **Design-independence**.
  - Design aspects are tackled during the design phase!
  - No architectural/design patterns.
Trade-Offs

Trade-offs between contrasting desiderata.

- **Expressivity vs tractability**: the more expressive the language, the harder it is to make it efficiently executable.
- **Parsimony vs convenience**: fewer concepts vs compact models.
  - Would you use Assembler to implement a web server?

Remember: we are doing conceptual modeling!

- No design issues, no implementation concerns, no fixed programming languages and architectural commitments.
- First: understand reality.
- Only the UoD + requirements.
Modeling Languages and Frameworks

- Conceptual Structural Schema
- Logical OO Schema
- Logical Relational Schema
- O-R Mapping
- UML (Unified Modeling Language)
- ORM (Object-Role Modeling)
- E-R (Entity-Relationship Modeling)
- JAVA
- Hibernate
- SQL DB

Marco Montali (FUB)
E-R: Abstract Representation of Data

- Introduced by Peter Chen (1976).
- The most widely used approach to data modeling.
- Key notions:
  - entities, relationships, attributes;
  - identification and multiplicity constraints.
- Independent from the target software platform.
- Lack of dynamic modeling.
- Close to relational database schemas → logical relational modeling!
- Different notations/dialects:
  - Chen, Barker, IE, IDEF1X, EXPRESS …
UML: Modeling Standard for OO Software Engineering

- Born from:
  - 3 amigos:
    - Rumbaugh’s Object-modeling technique;
    - Booch’s OO design;
    - Jacobson’s OO software engineering method.
  - Harel’s state-charts.
- OMG standard since 1997.

- Family of notations:
  - Structure diagrams: class/object diagram, component, composite structure, deployment, package, profile.
  - Dynamic diagrams:
    - Behavior: use case, state machine, activity.
    - Interaction: communication, interaction overview, sequence, timing.
UML Class/Object Diagrams

Structural modeling, especially for OO design $\rightarrow$ logical OO schemas.

- Behavioral aspects (operations/methods).
- Encapsulation policies (OO paradigm).
- Key notions:
  - object (class) as entity (type);
  - attributes (with visibility), relationships (basic, generalization, aggregation, composition);
  - multiplicity constraints, OCL;
  - behavioral aspects (operations, parameters, visibility);
  - no mandatory identification for objects (implicit reference, object ids).