09 Modeling, Metamodeling, Hybrid Wikis
Vorlesung IT-Unternehmensarchitektur

Dr. Sabine Buckl
Software Engineering for Business Information Systems (sebis)
wwwmatthes.in.tum.de
Learning objectives of this unit

Students

- know the basic principles of conceptual modeling
- can distinguish between describing and designing models and know their corresponding quality criteria
- are able to structure a modeling language into its constituents and know different methods for describing these constituents
- can explain the fundamentals of UML MOF
- are able to derive the information model from a specific viewpoint
- can apply different techniques to develop an organization-specific information model
Outline of this unit

3.1 An introduction to conceptual modeling
  - Models in context
  - Modeling languages and meta-models
3.2 EA Modeling
3.3 Collaborative, emergent EA modeling
Motivating example (1)

- Reality is often too complex to model or comprehend it.
  - Task: How do I get from FMI in Garching to the Marienplatz with the public transport system of the MVV?

Source: Google Earth
Motivating example (2)

- **Questions**
  - Do I have to know where a traffic light is?
  - Do I have to know where a tree stands?
- **Result is** *abstraction* and **reduction**
  - The model has to contain the important information for the user.

- **Model**
  - Plan of the public transport system of the MVV

Source: MVV
Key characteristics of a model

Key characteristics of a (representing) model – according to Stachowiak [St73]:

- **Models are always models of something, namely surrogates or representations of natural or artificial originals, which can be models themselves.** (engl. **Mapping** – dt. **Abbildungsmerkmal**)

- **Models commonly do not capture all attributes of their corresponding original, but only those, which seem to be relevant for the model creator and/or model user.** (engl. **Abstraction** – dt. **Verkürzungsmerkmal**)

- **Models are no 1:1 copies of their originals, they are surrogates for the original**
  - for certain – cognitive and/or acting, model using – subjects,
  - within given time intervals and
  - under constraints to certain mental or real operations.
  (engl. **Pragmatics** – dt. **Pragmatisches Merkmal**)

**But:** Models may refer to yet not built originals, i.e. may be **design models**.

→ Slightly different definition of model
Motivating example (ctd.) – Two more models of the MVV public transport system

Model 2 (Timetable):
- Different selection of attributes – arrival and transport times
- Similar model pragmatics:
  - Users that want to get via MVV from FMI to Marienplatz
  - in the year 2012

Model 3 (Spatial plan):
- Different selection of attributes – spatial information
- Different model pragmatics:
  - Users that want to perform urban planning
  - in the year 2012

→ Make-up of the models depends on its users (stakeholders).
→ Users might combine different models to a view.
A model?

Questions:
- Who is the intended user of the visualization? (Stakeholder)
- What do the rectangles and colors mean? (Viewpoint)

Anecdote:
„These pictures are meant to entertain you. There is no significant meaning to the arrows between the boxes.“

[Cle03]
What makes a (representing) model a good one? Conceptions of model quality (1)

Connecting model and modeled domain – *representation* and *interpretation* [Gu05]:

- **Lucidity**: Every construct in the model must represent at most one object from the modeled domain. Overloaded model constructs are forbidden. *(injective representation)*
- **Soundness**: Every construct in the model must represent at least one object from the modeled domain. Construct excess in the representation is avoided. *(surjective representation)*
- **Laconicity**: Every object from the modeled domain must “interpret” at most one construct in the model. Construct redundancy is forbidden. *(injective interpretation)*
- **Completeness**: Every object in the modeled domain must “interpret” at least one construct in the model. Model completeness is ensured. *(surjective interpretation)*
What makes a (design) model a good one? Conceptions of model quality (2)

Different types of model quality for the model in usage context [Kr02]:

- **Semantic quality**: Does the model cover the modeled domain?
- **Pragmatic quality**: Can the model be interpreted by the model users?
- **Physical quality**: Does the model capture the modeler’s domain knowledge?
- **Perceived semantic quality**: Does the model correspond to the users’ knowledge about the domain?
- **Social quality**: Does the model facilitate user discussions on the domain?
- **Tool quality**: Can the model be “interpreted” by a modeling tool?
- **Syntactic quality**: Does the model conform to a *modeling language*?
Outline of this unit

3.1 An introduction to conceptual modeling
  ▪ Models in context
  ▪ Modeling languages and meta-models

3.2 EA Modeling

3.3 Collaborative, emergent EA modeling
Every model has a modeling language

Main parts of a modeling language [Kü04]:

- **Syntax:** Describes the set of language concepts and their relationships to each other as well as the rules for forming *correct* models.

- **Notation:** Describes the representation of the language concepts (may be graphically or textually).

- **Semantics:** Describes the meaning of the language concepts and of their relationships.

A modeling language

- incorporates *domain knowledge*,
- reifies the *substantial laws* of the domain, and
- determines what a *valid model* is.

**But:** Not all *valid models* are *sensible* models, too.
Different ways of defining the syntax (1)

Grammar-based: a grammar describes how to get from a correct simpler language element to a more complex one – examples:

For textual languages: semi-Thue system and term rewriting systems, e.g. (Extended) Backus-Naur-Form (BNF)
- For graphical languages: graph rewriting systems

- **Advantages:**
  - easy to use
  - easy to implement in a tool

- **Disadvantages:**
  - grammar rules do not necessarily reflect domain concepts
  - hardly used and taught for conceptual models

For our example:
Different ways of defining the syntax (2)

*Meta model-based*: a model of higher abstractness, the meta model, describes the language elements and their intended relationships

- For object-oriented languages: MOF, UML
- For general knowledge representations: RDF, OWL

**Advantages**: 
- meta model concepts reflect domain concepts
- widely used and taught in conceptual modeling

**Disadvantages**: 
- meta model is expressed in (another) modeling language → infinite regress
- meta modeling language influences conceptualization of domain

For our example:

```
Station
name: String
```

```
Line
name: String
```

```
2..* has 1..*
```
Modeling language syntax and model

Syntax has two main functions:
- Specify the admissible model constructs
- Impose rules how the constructs can be combined

A model can comply with a syntax on different levels:
- “Nonsense” – does not (only) use the admissible constructs
- “Gibberish” – uses the admissible constructs but does not comply with the rules
- “Unintended models” – uses the constructs, complies with the rules, but does not correspond to a sensible reality
- “Intended models” – uses the constructs, complies with the rules, and is sensible

Language expressiveness may not be sufficient to avoid unintended models:
- **Contextual grammar rules** in grammar-based language specifications
- **Constraints** on meta-level in meta-model based language specifications
Different ways of defining semantics

- **Textually**: language concepts are provided informal descriptions of their meanings
- **Denotational**: language concepts are mapped to mathematical concepts, e.g. sets or groups, with well-founded semantics
- **Algebraic**: language concepts form elements and operators in an algebraic structure
- (**Operational**: language concepts are operationalized via code-fragments)
- (**Axiomatic**: language concepts are complemented with logical pre- and post-conditions)

→ For enterprise architecture modeling the first three ways are applicable
→ Different ways are helpful for different utilization contexts
Different ways of defining notations

Definition by *example*
- exemplary graphical symbols representing the modeling concepts
- rules for adapting the symbols according to concept’s properties are either
  - not given (*static symbols*) or
  - given textually (*dynamic symbols*).

Definition by *transformation*
- transformation rules translate from modeling concepts to graphical symbols
- strongly dependent on the expressiveness of the graphical language
  - nodes and edges visualizations (see e.g. [DV02])
  - charts and diagrams visualizations (see e.g. eclipse BIRT)
  - hierarchies, nodes and edges visualizations (see e.g. eclipse GMF)
  - visualizations with complex relative positioning (see e.g. [Er06])
Development of MOF (Meta Object Facility) by the OMG was heavily influenced by the evolution of UML and the appearance of MDA (Model Driven Architecture)

- **4-layer architecture**
  - Instantiation is used repeatedly ➞ M3-, M2-, M1-, M0-layer
  - MOF on M3 layer ➞ “hard-wired” meta-metamodel
- **MOF** does not “only” define the syntax
  - Possible forms of notations: MOF-Notation (~class diagram)
  - Restrictions define guidelines for the models
- **Notation** is defined by example
  - Through notation tables
  - Possible notation options with natural language
- **Semantics** is described in natural language
  - Additional semantic variations are defined
Language architecture of UML 2.4
4 layer architecture

M3 (MOF)

M2 (UML)

M1 (a Model)

M0 (runtime instances)
The UML and MOF support the utilization of constraints

Constraints are specified textually

- using natural language
- using mathematical terms
- using the Object Constraint Language (OCL)

**Example (M1):** any project must start before it ends

**Example (M2):** all properties must have unique names
Conceptual modeling beyond UML – Challenges of EA modeling

Relevant meta-properties for types:

- Notion of rigidity: *rigid*, *anti-rigid*, and *semi-rigid*:
  - any instance of a rigid type remains an instance of that type over its entire lifetime – example rigid type *human*
  - any instance of an anti-rigid type has not always been or will not forever be an instance of that type – example anti-rigid type *baby*
  - some instances of a semi-rigid type may forever be or have always been an instance of that type, while others not – example semi-rigid type *rich person*

- Versioning
- Ordering
- Hierarchical
Multiple EA modeling languages – example

Process owner
- View:

![Diagram of process flow]

Project manager
- View:

<table>
<thead>
<tr>
<th></th>
<th>SAP v3.58</th>
<th>SAP v4.05</th>
<th>L&amp;L 4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsidiary Munich</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Subsidiary London</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Legend:
- A: Business Process „A“
- B (1): Application „B“ with Id 1
- C: Org. Unit „C“
- „A“ is predecessor of „B“
- „B (1)“ supports „A“ at „C“
An information model can be derived from a view

- **View:**
  - Acquisition
  - Purchase
  - Subsidiary Munich: SAP (358), L&L (40)
  - Subsidiary London: SAP (405)

- **Information model:**
  <to be completed in the lecture>
An information model can be derived from a view

View:
- Subsidiary Munich: SAP (358)
- Subsidiary London: SAP (405)
- Acquisiton: L&L (40)

Information model:

Legend:
- Business Process "A"
- Application "B" with Id 1
- Org. Unit "C"

Support Relationship:
- "A" is predecessor of "B"
- "B (1)" supports "A" at "C"

Business Process
- name: String
  - 1 for *

Support Relationship
- with
  - 1

Business Application
- name: String
  - id: String
Discussion of information model variants

Can this information model be used for a process support map?
If not, why?
If yes, what would be advantages/disadvantages of this map?

Can this information model be used for a process support map?
If not, why?
If yes, what would be advantages/disadvantages of this map?
Outline of this unit

3.1 An introduction to conceptual modeling
   ▪ Models in context
   ▪ Modeling languages and meta-models

3.2 EA Modeling

3.3 Collaborative, emergent EA modeling
Emerging EA management initiatives often start informal using spreadsheets or text documents since

- the development of an information model is a labor intensive task and
- no widely-accepted standard information model exists.

With the growing complexity of the management body and the rising number of stakeholders involved, problems arise regarding

- scalability and
- collaborative work.

Introducing an EA management tool is often regarded to solve these problems.

How to support an evolutionary approach to EA development (esp. regarding the design of an enterprise-specific information model)?

How to avoid the ivory tower syndrome?
Extending wikis with templates to support structured content

- Automated data processing and visualization, which are essential in an EA management context impose additional requirements on data representation.
  - capture data in a structured form

- Existing wikis rely on text formatting conventions to express structure (e.g. www.wikipedia.org, cf. Figure), but do not offer native support of automated data processing.

- Semantic wikis (e.g. http://semantic-mediawiki.org), try to exploit complex semantic web technologies but often lack usability.

- Our approach: templates provide a simple extendable table containing attributes, textual values, and links.
Capture non-structured and structured information in a unified way.

Data Warehouse

Tags: todo edittags

Description of the application goes here. It may include
- formatted text
- formatted tables
- hyperlinks (Subsidiary Munich)
- graphics (PNG, JPG, ...) and
- editable and linked diagrams (Oryx).

Arbitrary many files can be linked as attachments and are full-text indexed.

0 Comments

Leave a comment:

Non-structured information

Types (0..m)

- Non-rigid attribute list
- Attributes defined for this type
- Attribute suggestions
- Inverse links

Types:
- business application
- edit tags

- criticality: high
- responsible unit: Headquarter
- used technology: Oracle 9i

References

"used applications" of
- Subsidiary London

[Ne12]
Change the information and its structure at any time.

Multi-valued & ordered
Suggestions based on content

Suggestions based on type(s)

[Ne12]
Manage the evolution of the information structures to match changing business needs.
Define the information model and its constraints incrementally (top-down or bottom up).
Identify, understand and cooperatively resolve constraint violations.

Last editor Max Mustermann, 24 minutes ago

---

Types: business application

responsibility unit

used technology

At least one value should be defined.

[Ne12]
Search by full text, tags, attributes and other relevant facets in combination.

Contents matching 'mysql'

- Last modification
  - Any Date
- Content type
  - Wiki Page (9)
  - Space
  - IT-Landschaft (9)
- Type Tags
  - business application (8)
  - technology (1)
- Special
  - Contains Invalid Links

- Search for 'mysql'
- Go
- sort by Relevance
- Tag Filter
- Attribute Filter

**business application**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value contains</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td></td>
</tr>
<tr>
<td>responsible unit</td>
<td></td>
</tr>
<tr>
<td>used technology</td>
<td></td>
</tr>
</tbody>
</table>

Results 1 - 9 of 9

- **MySQL 2.1**
  - Text...
  - IT-Landschaft [Last edited by Max Mustermann, Jan 23]
  - technology edit tags

- **Document Management System**
  - Text...
  - business application business application used technology **MySQL 2.1** responsible unit
  - IT-Landschaft [Last edited by Max Mustermann, Jan 27]
  - business application edit tags

- **POS System (Germany/Munich)**
  - Text...
  - business application business application used technology **MySQL 2.1** responsible unit
  - IT-Landschaft [Last edited by Max Mustermann, Jan 27]
  - business application edit tags
Use generated lists, tables and diagrams to provide stakeholder-specific views.

Which organizational unit is responsible for which business application?

Which business application uses which technology?

Link to detailed information
Use generated lists, tables and diagrams to provide stakeholder-specific views.

What are our domains, subdomains and business applications?

What information dependencies exist for the data warehouse?
The principle behind hybrid wikis – Data first, schema second

For more details see [www.infoasset.de](http://www.infoasset.de)
Bibliography


