Inference Rules of Semantic Dependencies in the Enterprise Modelling

Remigijus GUSTAS
Department of Information Systems, Karlstad University
Phone: +46-54 700 17 65
E-mail: Remigijus.Gustas@kau.se
http://www.cs.kau.se/~gustas/
Content

✓ Challenges of Service Oriented Analysis and Design (SOAD)
✓ Foundations of Enterprise Modelling Approach
✓ Restrictions of Inference Rules for Object-Oriented Models
✓ Inference Rules of semantic dependencies in the Enterprise Modelling
✓ Conclusions
Reassessment of the existing System Design Principles

- The implementation bias of many system analysis methods is a big problem. The same implementation-oriented foundations are applied for the system analysis and for the system design phase, without rethinking these concepts fundamentally.
- System Analysis/Enterprise Modelling should deal with only conceptually relevant aspects and it cannot be influenced by the possible implementation solutions.
- Graphical specifications that follow the basic conceptualization principle (Griethuisen, 1982) are less complex and, therefore, they are more comprehensible for humans.
- Technology neutral descriptions of Enterprise Architectures should provide integration principles. Separate technical system representations are difficult to maintain.
- More reasonable is to use one type of model for conceptualization of organisational system part, before the supporting technical system is defined. Interdependencies across multiple diagrams should be a critical part of Service Oriented Analysis and Design (SOAD).
Enterprise Architecture – Framework (Zachman, 1996)

<table>
<thead>
<tr>
<th>View</th>
<th>Data (What)</th>
<th>Function (How)</th>
<th>Network (Where)</th>
<th>People (Who)</th>
<th>Time (When)</th>
<th>Motivation (Why)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scope</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Planner view</strong></td>
<td>List of concepts</td>
<td>List of processes</td>
<td>List of locations</td>
<td>List of organizational units</td>
<td>List of business events</td>
<td>List of business goals, vision</td>
</tr>
<tr>
<td><strong>Enterprise Model</strong></td>
<td><strong>Owner view</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Information System</strong></td>
<td><strong>Designer view</strong></td>
<td>Entity relationship diagram</td>
<td>Business Process diagram</td>
<td>Diagram of logistic network</td>
<td>Schedule/State Trans. Diagrams</td>
<td>Business Strategy / Plan</td>
</tr>
<tr>
<td><strong>Technology</strong></td>
<td><strong>Builder view</strong></td>
<td>Logical Data Architecture</td>
<td>Software application architecture</td>
<td>Distributed system architecture</td>
<td>Human interface architecture</td>
<td>Control structure</td>
</tr>
<tr>
<td><strong>Representations</strong></td>
<td><strong>Subcontractor view</strong></td>
<td>Physical Data architecture</td>
<td>Deployment architecture</td>
<td>Technology architecture</td>
<td>Presentation/Layout structure</td>
<td>Component control structure</td>
</tr>
<tr>
<td><strong>Functioning System</strong></td>
<td></td>
<td>Data definition</td>
<td>Process design</td>
<td>Network architecture</td>
<td>Interface architecture</td>
<td>Timing definitions</td>
</tr>
</tbody>
</table>

© R Gustas
Two Fundamental Problems

✔ Graphical Modelling Foundation for Service – Oriented Analysis & Design (SOAD)
  ■ Enterprise Models/Architectures (EM) need to be captured, visualised and agreed across the organisational and technical system boundaries.
  ■ Just as the complex buildings or machines require explicit representations of design structures, so does an overall SOAD process

✔ Integration Principles of EM
Challenges of SOAD

✓ Traditional methods are dividing system specifications into separate parts (they are typically devoted to data architecture, application architecture or technology architecture).

✓ Holistic understanding of Enterprise Architecture is typically not available in traditional approaches.

✓ Integrated representations of EM are necessary to reach consensus among partners involved.

✓ Interdependencies between different views are crucial to glue the static and dynamic aspects.
## Design Layers

<table>
<thead>
<tr>
<th>Business Layer</th>
<th>Business Process modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Layer</td>
<td>Service-Oriented Analysis</td>
</tr>
<tr>
<td>Component Layer</td>
<td>Component-Oriented Design</td>
</tr>
<tr>
<td>Class Layer</td>
<td>Object-Oriented Design</td>
</tr>
</tbody>
</table>

**Example of design Hierarchy**

- Human Resource Management
  - Recruitment
  - Career
  - Person

**Diagram:**

- Person
  - Example of design Hierarchy
Basic Semantic Dependencies

Dynamic (Semantic) Dependencies

Communication

Transition

Static (Semantic) Dependencies

Inheritance

Association

Composition

Classification
Basic Events in SOAD

- Creation,
- Termination,
- Reclassification (consists of termination and creation events).

Events are triggered by a Communication Action.

Use Cases (UML) can be viewed as Actions.
Reclassification (Compound) Event

1. Communication **Flow** is sent by **Agent**
2. Removal of object in a precondition **Class**
3. Communication **Flow** is accepted by **Recipient**
4. Creation of object in a postcondition **Class**

**Note:** Flows can be **Material**, **Information** or **Control**
(control flow is represented just by communication dependency link)
Example of Reclassification

Note: Reclassification of an object from one concept (could be a class) to another without relating it to any other concepts is not useful. Analysts must to identify a noteworthy semantic difference between two concepts (Applicant and Employee). Otherwise, the action is not useful.
Semantic Difference can be specified by using the Static Dependencies

- Every A is composition of exactly one instance of B
- Every A is composition of one or more instances of B
- A is specialisation of B
- A and B are synonyms
- A is an instance of B
- B is a state of A
- B and C is an exclusive specialisation of A
Basic Associations can be used to define the semantic difference

(0,1;1,1)

(0,*;1,1)

(1,1;1,1)

(0,1;1,*)

(0,1;1,*)

Not Basic Associations (the remaining 5 types of static relations) are not recommended for the final SOAD phase. They give rise to the semantic holes. Since other association types are used in OO analysis and design, they are not forbidden in early SOAD phases.
Creation Event Notation
Termination Event Notation
Communication Loop with Alternative Action
Basic Events in OO Analysis

✓ Creation
✓ Termination
✓ Connection
✓ Disconnection
✓ Classification
✓ Declassification

Source: (Martin and Odell, 1998)

Create, Access (Get, Search), Update and Delete operations are used for definition of design solutions
Inference Rules for a Use Case Diagram
Inference Rules for a Class Diagram

Applicant
- EmployActive : bool = 0
+ CreateApplicant()
+ GetApplicant()
Inheritance of Transition Dependency

[Diagram showing relationships between entities such as Manager, Employee, Company, and Termination Date.]
Inference Rules for Elimination
Semantic Ambiguity
Inheritance and Composition Dependencies are Transitive
Communication Dependencies are Inherited and Propagated along the Composition (Single) Dependency
Inference Rules of Classification

- **Company** → **Organisation**
  - IBM

- **Organisation** → **Company**
  - IBM
  - Karlstad University

- **Company** → **Organisation**
  - IBM

- **Organisation**
  - **Company** → **NOT Company**
    - IBM
    - Karlstad University
Corollary of Classification
View Integration (example)
View Integration (cont.)
Conclusions

✓ Graphical Models that are used for SOAD follow the **basic conceptualization principle** (implementation independent graphical representations).

✓ Semantic dependencies are represented by **one diagram type** (one basic model).

✓ Inference rules facilitate view **integration**, and therefore reuse and ease of reconfiguration through loose coupling.

✓ Integrated static and dynamic representations are necessary to reach **consensus** and to develop **holistic** understanding.

✓ Since models are **implementation agnostic**, they are understandable for domain experts without deep technical expertise.