Abstraction mechanisms and modeling perspectives in conceptual and enterprise modeling

John Krogstie
Overview over lecture

- Modeling as hierarchical abstractions (Chap 3.1)
- Hierarchical relationships used in modeling (Chap 3.1)
- Modeling perspective (Chap 3.3 + start in 3.5 (not 3.5.1-3))
  - Examples of modeling languages of different perspectives
  - Practical usage of different modeling approaches
Modeling as hierarchical abstraction

- Information systems are complex
- Hierarchical structuring is important for human understanding of complex systems
- It can be claimed that it is natural for humans to structure their perception of reality hierarchically (at least we are trained to think hierarchically)
What is a hierarchy? (Bunge)

- 'H' is a hierarchy if and only if it is an ordered triple \( <S,b,D> \), where ‘S’ is a non-empty set, ‘b’ a distinguished element of ‘S’ and ‘D’ a binary relation in ‘S’ such that
  - ‘S’ has a single beginner ‘b’
  - ‘b’ stands in some power of ‘D’ to every other member of ‘S’.
  - For any given element ‘y’ of ‘S’ except ‘b’, there is exactly one other element ‘x’ of ‘S’ such that ‘Dxy’
  - D is transitive and anti-symmetric
  - D represents (mirrors) domination or power
- \( \Rightarrow \) Strict tree representing domination
Types of hierarchies

- A strictly hierarchical graph
- A weakly hierarchical graph: DAG (directed acyclic graph)
Chap 3.1.2: Standard Hierarchical Abstraction Mechanisms

- 4 relations often used in modeling
  - From instance to class (across meta-levels)
    - Classification
  - From class to class (or instance to instance) (at the same meta-level)
    - Aggregation
    - Generalization
    - Association
Classification

- Instances is looked upon as a higher order object-type (using the `is_instance_of` relation)
  - Examples: John Krogstie `is_instance_of` lecturer
  - 'chair' `is_instance_of` noun
- Orthogonal to the others (different meta-levels)
Principles of classification (Parsons and Wand)

- Abstraction from instances: A class can be defined only if there are instances in the relevant domain possessing all properties of the type that defines the class.

- Maximal abstraction: A property possessed by all instances of a class should be included in the class definition.

- Completeness, requiring that all properties from the relevant domain be used in classification,

- Non-redundancy, which ensures that there are no redundant classes.
Variants of classification (Leppänen)

- Objective vs. subjective classification: Do all classify the instances in the same way at all times?
- Permanent vs. evolving classification: Is the instance always of the same class (es) or not?
- Strict vs. non-strict classification. Do instances exist that are not of a class? Are there instances that can be of more than one class at the same time?
Aggregation

- An object is related to the components it exist of using an *is_part_of* relation
  - Example: A bicycle have wheels, a frame, a seat etc.
  - A sentence can consist of a noun and a verb

- Aggregation corresponds to cartesian product
  - if A is an aggregation of $A_1..A_n$, A is a subset of $A_1 \times ... \times A_n$
  - A relationship class in an ER-model (or an association in UML class diagrams) is an aggregation of this type

- Variants
  - Parts exclusive to one aggregation (a motor is part of one car) or shared (a figure is found in many articles) between aggregations
  - The existence of a part is dependant on the existence of the aggregation (brain part of human) or not (figure part of a certain article).
From the viewpoint of the whole

- A homogeneous whole is a thing that is composed of parts of one class (e.g. a puzzle).
- A heterogeneous whole is a thing composed of parts of several classes.
- A single-part whole is a thing that contains only one thing of a certain class (a book have one table of contents list)
- A multi-part whole is a thing that contains several things of a certain class (a book have many chapters)
- A flexible-structure whole is a thing in which parts of some class can be missing (a book might contain an index or not)
- A fixed structure whole is a thing which must be composed of one or more parts of all the defined part classes
Generalization

- Similar object types are abstracted into a higher level object type via an *is_a* relation
  - Example: A lecturer *is_a* human
  - A noun is _a_ word

- Corresponds to union:
  - If a set $A$ is a generalization of $A_1, A_2, ..., A_n$, $A_1 \cup ... \cup A_n = A$

- Variants
  - Coverage
  - Disjointness
  - Partition
Generalization/Classification as one way of implementing inheritance

- Single or multiple inheritance?
- Share properties or share behaviour (or both)?
- Dynamic or static inheritance?
- Selective or full inheritance?
Association

- Several object types are considered as a higher level object type via `is_a_member_of` relation
  - Example: The sets `verb`, `noun`, `adjective`… are members of the set `word-classes`
  - The sets `men` and `women` are members of the set `sex-groups`

- Association corresponds to membership:
  - If A is an association of $A_1$ … $A_n$, it means that
    
    \[ A = \{ A_1, \ldots, A_n \} \]

- Other terms used are membership, grouping and collection
Types of associations (class and instance level)

- In homogeneous association there is only one member class (group of persons)
- In heterogeneous association, several member classes can be involved
- In a categorical association, a member class is related to one association class at the time
- In shared associations, a thing of a certain class can be member of several associations at the same time

- Disjoint association: An instance is only member of one association.
- Overlapping association. An instance can be member of several associations
- Mandatory association: All instances are members of one (or more) groups
- Optional association: There might be instances not being member of an association
Positive aspects with standard relations

- Positive experiences in the use in modeling, implemented in a number of modeling approaches (vs. Chapter 3.3)
- General, can be used in many connections
- Are used in many other contexts (such as language, biology, mathematics)
Weaknesses of relations (and their set-theoretical definitions)

- Relations between instances
  - Possible to use association
  - Can threat instances as sets with one member

- Masses : (water, money etc)
  - Can use definitions based on types rather than on sets
  - Can use the set-theoretical definitions in special ways

- Can be difficult to use correctly in practice

- The different mechanisms are defined using different terms in different modelling languages (and at time with different semantics)
Classification levels often used in modeling

- Instance: John Krogstie as lecturer in TDT4252 in 2011
- Model: Lecturer in course
- Meta-model: Entity class, relationship class
- Meta-metamodel: Node, edge
Levels of models – OMG-MOF

M3 level
Meta-metamodel

instance-of

M2 level
Metamodel

instance-of

M1 level
User-defined model

instance-of

M0 level
Object diagram
Perspectives to conceptual and enterprise modeling (Chap 3.3)

John Krogstie
An overview of conceptual modeling languages

- More than 500 different modeling languages described in literature
- Are classifying languages according to modeling perspectives to get an overview
- Today: Overview of perspectives, with examples of languages of each perspective, will go in more detail on the languages in some of the perspectives later in the course.
Discussion

- Why are there so many modeling languages?
Modeling perspectives

- A modeling perspective: What is important, what is emphasized, and what is ignored
  - What are the fundamental concepts
  - What aspects are explicitly represented
    - As nodes, relationships, or properties
  - What is visualized?
  - What is modeled first?

- Different perspectives can use the same concepts
- Can also potentially represent the same
8 perspectives to conceptual modeling

- Behavioral
- Functional
- Structural
- Goal and rule-oriented
- Object-oriented
- (Social) communication
- Actor/role-oriented
- Topological
Behavioral perspective

- Description of the bahavioural dynamics of a system
- Main concept: States and transitions between states

**Examples**

- State Transition Diagrams (STD/STM).
- Statecharts
- Petri-nets (1961)
- SDL
- System Dynamics

- Different types of state transition diagrams used particularly within real-time systems and telecommunications systems. Petri-net semantics is the base semantics for a number of modern process modeling languages (see below), much applied in BPM
Example: STD/statecharts

- Transition
- Action
- State
- Event
- Condition

XOR decomposition

(Generalization)

AND decomposition

(Aggregation)
States in a paper process

Non-existent → Received → Confirmed → Distributed

Received CRC → Received CRC → Conditionally accepted

Conditionally accepted → Accepted

Accepted → Received review

Received review → Reviewed

Reviewed → Reject paper

Reject paper → Rejected

Conditionally accepted → Deadline expire

Deadline expire → Reject paper

Reject paper → Rejected

Receive CRC (satisfactory) → Published

Published → Published

Receive CRC (not satisfactory) → Distributed

Distributed → Distributed

Receive paper → Received

Send confirmation → Confirmed

Distribute paper → Distributed

Receive review → Reviewed
Example of use of statecharts
Example: Petri-nett
Improving expressiveness through Petri-nets
Functional perspective

- Often called process-oriented (although process modeling can be done according to a number of perspectives)
- Main concept: Transformation (function, process, activity, action, task)
- Description of dynamic processing
- Paradigm example: DFD

- Many variants and extensions (SA/RT, IDEF/0)
- Aggregation through decomposition
- Many newer languages also include also behavioral modeling e.g. BPMN
- Used a lot in process modeling, both for business processes and processes within system applications and workflow modeling
Example on extension of DFD: SA/RT

<table>
<thead>
<tr>
<th>1) TRANSFORMATIONS</th>
<th>2) DATA FLOWS</th>
<th>3) EVENT FLOWS</th>
<th>4) STORES</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>DISCRETE DATA</td>
<td>SIGNAL</td>
<td>DATA STORE</td>
</tr>
<tr>
<td>CONTROL</td>
<td>CONTINUOUS DATA</td>
<td>ACTIVATION</td>
<td>BUFFER</td>
</tr>
</tbody>
</table>
Combinations of functional and behavioural perspectives

- Functional modelling + control flow enables modeling many of the same mechanisms as e.g. Petri-nets
- UML Activity Diagram, YAWL, EPC, EEML, BPMN…
Structural perspective

- Static description of the structure in a domain
- Main concept: Entity (Object, phenomena, concept, thing, referent....)

Examples
  - Data-modelling (e.g. ER and variants of this e.g. NIAM, ORM…)
  - Semantic nets
  - Conceptual graphs

- Especially data modeling is used a lot in practice. Often also used for meta-modeling (logical language modeling).
Example: GSM
Goal and Rule perspective

- Description of goals/means connections
- Main concept: Rule (goal, constraint)
- A rule is something that influences the actions of a set of actors.
- Standard form: IF condition THEN action

Examples:
- Rule hierarchies (goal-oriented modeling) see e.g. i* and EEML
- Business rules systems
- Ontology languages (e.g. OWL)
- Tempora: Combination of structural, functional/behavioural and goal orientation

Usage: Goal hierarchies in RE, more extensive use of rule-based systems “behind the scenes” and in some semantic web application (structuring of (web)-data)
Goal-hierachies (example EEML)

- A research community should spread scientific result
  - and
  - A research community should arrange scientific conferences
    - and
    - A research community should publish journals
  - and
    - The conference should not loose money
      - Obligate
        - A member of the research community should periodically volunteer to arrange a conference
  - and
  - The conference must attract sufficient number of participants
    - Necessitate
      - A budget must be created
Leaf goals relative to tasks (EEML example)

- All reviews should include comments to help the author in improving the paper
- A scientific conferences must have a scientific review process
- It is not allowed to submit the same article to several areas
- A paper should have more than two reviews
- The author should take review comments into account when making the CRC
- The accepted paper should follow the style guide of the publisher

Tasks:
- Review paper
- Distribute results
- Develop CRC
Ontologies: Data + logical rules

- E.g. OWL:
  - *OWL Lite* supports those users primarily needing a classification hierarchy and simple constraint features.
  - *OWL DL* supports those users who want the maximum expressiveness without losing computational completeness (all entailments are guaranteed to be computed) and decidability (all computations will finish in finite time) of reasoning systems.
  - *OWL Full* is meant for users who want maximum expressiveness with no computational guarantees.
Object perspectives

- Description of the world as autonomous, communicating objects
- Main concept: Object. An entity with a unique identity with a local state only accessible by sending messages to the interface of the object.
- Examples: UML, OML, OMT, Shlaer/Mellor, OOA...
- Basis from design and programming of object oriented systems
- Increasing use as object-oriented development is getting more and more common (particularly UML and extensions of UML)
Structural aspects in object-orientation
Dynamic aspects in object-orientation
Social Communication perspective

- Description of the (social) communication structure,
- Also called the language action perspective (LAP)
- Main concepts: Speech Acts and Conversations
- Based on philosophy of language (Austin, Searle, Habermas)
- Different types of speech acts which are put together into conversations
- Specifically used in workflow modeling and CSCW/Groupware (but diminishing popularity)
Speech acts - Illocution

- Three distinct acts
  - **Locution**: The physical expression
  - **Illocution**: The intention of the locution
  - **Perlocution**: Acts following the illocution
Categories of Speech Acts (Searle)

- Directive: "Close the window"
- Commissive: "I shall close the window"
- Assertive: "The window is closed"
- Declarative: "I hereby Pronounce you Man and Wife"
- Expressive: "Congratulations"
Winograd & Flores
Conversation for Action

1. A: Request
2. B: Promise
3. A: Declare
4. B: Assert
5. A: Declare
6. B: Counter
7. B: Renege
8. A: Accept
9. A: Withdraw

- A: Withdraw
- B: Ignore
- B: Counter
- A: Request
- B: Promise
- B: Declare
- A: Assert
- B: Counter
- A: Accept
- A: Reject
- B: Withdraw
ActionWorkflow

Preparation → Negotiation

Conditions of Satisfaction

Negotiation → Performance

Acceptance

Customer

Performer
Speech acts relative to paper process

- CFP
  - dir(paper) & ass(event)

- Letter of intent
  - com(paper)

- Paper
  - ass(paper)
  - com(presentation)

- Withdrawal
  - decl(~paper)

- Rejection letter
  - decl(~paper) & expr(paper)

- Conditional accept letter
  - decl(paper) & expr(paper) & dir(CRC) & dir(copyright) & dir(presentation)

- Acceptance letter
  - decl(paper) & expr(paper) & dir(CRC) & dir(copyright) & dir(presentation)

- Ass(confirmation)

- Ass(reminder)
Actor and role perspective

- Description of organizational or system structure
- Main concepts: Actors and roles
- Possible definition
  - An actor is a phenomena that influence the history of another actor
  - A role is the behavior that is expected by an actor (by other actors) when filling the role
- Based originally on
  - work in AI (Intelligent agents)
  - object-oriented programming
  - Modeling of organizational structures, group dynamics, social network analysis etc.
  - Later also economic theory
- More on this perspective later in the course (i*, e³value)
I* dependencies

- Task Dependency
- Resource Dependency
- Goal Dependency
- Soft-goal Dependency
Dependencies between actors in conference arrangements
e³value
Example from conference case in e³value
Topological perspective

- Relates to the topological ordering between the different concepts (where)
- Place (type of space e.g. Meeting room) vs Space (e.g. MA21).
- Limited support of modelling this aspect in conceptual modelling. ‘Space’ traditionally represented in maps
- ’Where’ is increasingly relevant
  - Outsourcing
  - Supply Chain Management/Logistics
  - Virtual organization
  - Mobile applications and information systems
- And it is possible to utilize ’where’ to a larger degree (also real time) to know where users, equipment and goods should be, are or where at a certain time
  - Tracking (RFID, UWB, GPS, GSM, WiFi, Ultrasound…)
  - Internet of Things (IoT)
Place-oriented modelling

Police Leader
- Distribute control locations
- Plan Control locations
- Get info about locations
- Control Vehicles
- Log control info
- Next Location?
- Make daily report

Police Assistant Personnel
- Plan Control locations
- Get info about locations
- Control Vehicles
- Log control info
- Next Location?
- Make daily report

Police Control Room
- Contact control room
- Processing needs
- Provides further info / action

Legend
- In Office
- Driving car
- Parked car
- At location
Space-oriented modelling
No language perspective covers all situations/domains in itself

- Different approaches for combining different perspectives
  1. Use many different languages in parallel without explicit integration (early CASE-tools, Visio)
  2. Integrate existing languages (UML, EEML)
  3. Develop completely new approaches (i*, e³Value)
  4. Develop framework to be able to create and adapt languages on an as-needed basis (Metaedit, METIS/Troux Architect)

- Perspective-less modeling
  - Facet modeling (Opdahl and Sindre 1997)
  - GEMAL
Integration of existing languages, e.g. UML

- **Structural**
  - Class diagram (without methods)

- **Object-oriented**
  - Object and Class diagrams (with methods)
  - Sequence and collaboration diagrams

- **Functional**
  - Use cases
  - Activity diagrams

- **Behavioral**
  - State diagrams

- **Rule oriented**
  - OCL, no goal-hiearchies, difficult to represent rules across classes

- **Actor/role oriented**
  - Limited coverage
  - Actors in use case diagrams, swimlanes in activity diagrams

- **Social communication**
  - Not covered

- **Topological**
  - Not covered
Central problems of multi-perspective approaches

- How to see the connection between different sub-models and diagrams?
  - Tool support

- Alignment of models in different languages
  - Integrated meta-model

- Consistent notation

- Does it make the language more difficult to learn?
  - ”core” UML vs. extensions and profiles
  - Viewing mechanisms in tools
Summary

- It is possible to model one's perception of the reality from many different perspectives.
- A model expressed in a language of a specific perspective emphasizes a certain way of structuring.
- There is no perspective that is best in all cases.
- Need often to combine perspectives in an integrated manner.
- What to include depends on the domain of modeling and the stakeholders of the modeling.
Abstraction mechanisms and modeling perspectives in conceptual and enterprise modeling

Final questions?

John Krogstie