Modeling I

Class diagrams

Ketil Stølen

Partly based on slides prepared by Prof. Øystein Haugen, HiØ & SINTEF
Overview of lecture

- Modeling
  - What is it?
  - Why do we do it?
  - Modeling and Programming – sides of the same coin?

- UML Class modelling

- Tooling
  - Papyrus
What's a Model?

Term

Phenomenon

Concept

Model

Symbol

Thought or Reference

Stands for
(an imputed relation)

TRUE

Correct
Symbolises
(a causal relation)

Adequate
Refers to
(other causal relations)

Referent
Exercise: Explain *class* in the setting of the previous slide

- In which corner does *class* belong?
- What would you put in the two other corners?
Exercise: Explain *threat* in the setting of the previous slide

- In which corner does *threat* belong?
- What would you put in the two other corners?
Artefacts in Informatics

Abstraction
- Models
- Frameworks
- Patterns
- Algorithms

Languages
- Programming
- GPL
- DSL
- Formal
- Visual ....

Tools
- Editors
- Compilers
- Verifiers
- Simulators
- Apps ....
Exercise: How do the other languages you have been thought fit in the previous picture?
Modeling a system

- A system is a part of the world
  - which we choose to regard as a whole, separated from the rest of the world during some period of consideration, a whole which we choose to consider as containing a collection of components, each characterized by a selected set of associated data items and patterns, and by actions which may involve itself and other components

- Mental systems
  - Systems existing in the human mind, physically materialized as states of the cells of our brains

- Mental and manifest models
  - when a limited set of properties is selected from a system

- These definitions are from K. Nygaard and his DELTA team (in 1977)
What language(s) to use?

- Must have good mechanisms for abstraction
- Must have adequate tooling
- Must scale to "real systems"
Why make a language?
UML Class modelling

- Concepts
- Identity
- Generation
- Meta
- Aggregate
## Concepts

<table>
<thead>
<tr>
<th>Class</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Instance</td>
</tr>
<tr>
<td>Pattern</td>
<td>Entity</td>
</tr>
<tr>
<td>Method</td>
<td>Method call</td>
</tr>
<tr>
<td>Function</td>
<td>Function call</td>
</tr>
<tr>
<td>Datatype</td>
<td>Variable</td>
</tr>
<tr>
<td></td>
<td>Prototype</td>
</tr>
<tr>
<td></td>
<td>Clone</td>
</tr>
</tbody>
</table>
A small story about Courses

- The Software Engineering Course is a special Course
- Courses contain Lectures
- The lectures may generate questions
A small Story with Boxes and Arrows

What are the boxes (Relata)?
What are the arrows (Relations)?

course

is-a-special

SWEcourse

consists-of

lectures

generates

questions
A small Story with UML class diagram
Exercise:

- Can Software Engineering course (SWEcourse) be held without lectures?
- Can there be lectures without questions asked?
- Can the very same lecture be given in two different courses?
- Can the very same question be posed to several lectures?
- If a course is cancelled, will all remaining lectures also be cancelled? (or "terminated")
### Identity modifiers:

<table>
<thead>
<tr>
<th></th>
<th>Languages:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalization</td>
<td>UML</td>
</tr>
<tr>
<td>Subclass</td>
<td>Simula</td>
</tr>
<tr>
<td>Derived Classes</td>
<td>C++</td>
</tr>
<tr>
<td>Extension</td>
<td>Java</td>
</tr>
<tr>
<td>Interface</td>
<td>UML, Java</td>
</tr>
<tr>
<td>Parameters</td>
<td>FORTRAN, Pascal, Algol, ...</td>
</tr>
<tr>
<td>Overloading</td>
<td>C++, Java</td>
</tr>
<tr>
<td>Redefined operations</td>
<td>UML</td>
</tr>
<tr>
<td>Virtual procedures</td>
<td>Simula, Smalltalk</td>
</tr>
<tr>
<td>Virtual functions</td>
<td>C++</td>
</tr>
<tr>
<td>Overriding methods</td>
<td>Java</td>
</tr>
<tr>
<td>Pointers to functions</td>
<td>C, C++</td>
</tr>
</tbody>
</table>
Subclassing or Inheritance

- SWEcourse inherits from Course
- Operation redefined
- Generalization

Diagram showing the relationships and operations between different classes.
Generation

Questions are not owned by Lecture

Navigable association
Aggregation

Composition

Concept aggregate relation
Metamodel

Modeling Language

Model

Program Execution

Data Structure

Change Observe

Define

Describes

Modeling Environment

- Editor
- Compiler
- Interpreter
- Debugger
- Model database
- Simulator
- Validator
- Text generation

Meta
Exercise: Explain the previous slide wrt the language

**English**

- What is the meta-model?
- What is the modeling-language?
- What is a model?
- What is program execution?
- What is the data structure?
The 4-level meta hierarchy

<table>
<thead>
<tr>
<th>Lev</th>
<th>UML model</th>
<th>Language</th>
<th>Programming</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>M3</td>
<td>MOF metamodel</td>
<td>MOF</td>
<td>Grammar of BNF</td>
<td>BNF?</td>
</tr>
<tr>
<td>M2</td>
<td>UML metamodel</td>
<td>MOF</td>
<td>Grammar of Java</td>
<td>BNF</td>
</tr>
<tr>
<td>M1</td>
<td>UML user model</td>
<td>UML</td>
<td>Java user program</td>
<td>Java</td>
</tr>
<tr>
<td>M0</td>
<td>Execution of user model</td>
<td>Execution of java program</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A piece of the UML Metamodel
Exercise: Represent the drawing below in UML

- **Vulnerability**
- **Threat**
- **Asset**
- **Party**
- **Unwanted incident**
- **Likelihood**
- **Consequence**
- **Risk**
- **Treatment**
Modeling tool used for the UML part of this course

You may use the tool of your preference

Some alternatives:
- [https://www.eclipse.org/papyrus/](https://www.eclipse.org/papyrus/) (powerful but involves a lot to install and use)
- [https://www.draw.io](https://www.draw.io) app (light weight)