Metamodeling and Metaprogramming Seminar

1. Introduction

Prof. O. Nierstrasz
Spring Semester 2008
Metamodeling and Metaprogramming Seminar

*Lecturer:* Oscar Nierstrasz  
www.iam.unibe.ch/~oscar

*Assistant:* David Röthlisberger

*WWW:* scg.unibe.ch/Teaching/MM
1. Introduction

Roadmap

> Goals of this seminar
> Seminar topics
> Historical perspective
1. Introduction

Roadmap

> Goals of this seminar
> Seminar topics
> Historical perspective
1. Introduction

**Goals**

*Learn about:*  
> Models and metamodels  
> Metaprogramming  
> Reflection:  
  — introspection and intercession  
  — structural and behavioural reflection

*Get experience with:*  
> Reflective programming languages  
> Manipulating models at runtime  
> Modern model-driven technology  
> Researching a topic and presenting it (in English!)
1. Introduction

Roadmap

- Goals of this seminar
- Seminar topics
- Historical perspective
1. Introduction

Planned lecture topics

I.e., lectures that we will do.

> FAME (AA+TV)
> Traversals (AA)
> Magritte (LR)
> Geppetto and sub-method reflection (MD)
> …
### Seminar topics (suggestions)

- UML OCL (TV)
- MDE Case Study (ON, TV)
- Business Rule Modeling (OG)
- Transformation Languages (LR)
- DSLs (TG, LR)
- CLOS Metaprogramming (TV)
- AOP (OG)
- Business Process Modeling (AA)
- EMF / eCore in eclipse (AA)
- GMF (Graphical Modeling Framework) (LR)
- Template Metaprogramming (ON)
- Naked Objects (ON)
- Self (ON)
- ...
1. Introduction

Deliverables

> Presentation
  — Talk
  — Cheat Sheet

> Demo
  — Presentation
  — Quick Start

> Draft exam questions

Your final grade will be based 50% on your seminar plus 50% on the final exam (all topics).
1. Introduction

Roadmap

> Goals of this seminar
> Seminar topics
> **Historical perspective**
  — What is a model? A meta-model?
  — Reflection and reification
  — Reflection in programming languages
  — Model-driven engineering
1. Introduction

Roadmap

> Goals of this seminar
> Seminar topics
> Historical perspective
  — What is a model? A meta-model?
  — Reflection and reification
  — Reflection in programming languages
  — Model-driven engineering
What is a model?

- Description/abstraction of real world things
- Something with a meta description of how it should be structured
- Objects & relationships (a graph?)
- What's a supermodel?
- Composition of models — cars & traffic
- Could be abstraction of something imaginary
- For reasoning
- Abstract representation that can be manipulated by a program
- Can be easier to modify or work with
- Simulation (cost)
- Abstraction of a process
- Abstraction of something that does not exist yet
What is a meta-model?

This slide intentionally left blank
Example from databases

Meta-meta-model

Meta-model

Model

System

Relational data model: Tables, attributes, tuples

Database schema: Student, Course, Enrolment ...

Database tables of tuples: (andreas, muster, 07-123-123), ...

Real world: You, MMS, ...

«instance-of»

«instance-of»

«represented-by»
Programming is Modeling

Programs are models ... so they should look and behave like models!
1. Introduction

Roadmap

> Goals of this seminar
> Seminar topics
> Historical perspective
  — What is a model? A meta-model?
  — Reflection and reification
  — Reflection in programming languages
  — Model-driven engineering
A metaprogram is a program that manipulates a program \textit{(possibly itself)}
Reflection

“Reflection is the ability of a program to manipulate as data something representing the state of the program during its own execution.

— Introspection is the ability for a program to observe and therefore reason about its own state.

— Intercession is the ability for a program to modify its own execution state or alter its own interpretation or meaning.

Both aspects require a mechanism for encoding execution state as data: this is called reification.”

— Bobrow, Gabriel & White, “CLOS in Context”, 1993
“As a programming language becomes *higher and higher level*, its implementation in terms of underlying machine involves *more and more tradeoffs*, on the part of the implementor, about what cases to optimize at the expense of what other cases. ... the *ability to cleanly integrate* something outside of the language!s scope *becomes more and more limited*”

Kiczales, in Paepcke 1993
Reflection and Reification

Metamodel

Object

Model

Object class

«instance of»

«reification»

«introspection» (reflection)

«intercession» (reflection)

«modification»

anObject
Causal connection

“A system having itself as application domain and that is causally connected with this domain can be qualified as a reflective system”

— Maes, OOPSLA 1987

— A reflective system has an internal representation of itself.
— A reflective system is able to act on itself with the assurance that its representation will be causally connected (up to date).
— A reflective system has some static capacity of self-representation and dynamic self-modification in constant synchronization
1. Introduction

Roadmap

- Goals of this seminar
- Seminar topics
- Historical perspective
  - What is a model? A meta-model?
  - Reflection and reification
  - Reflection in programming languages
  - Model-driven engineering
Reflection in programming languages

> Assembler
> Lisp
> Scheme
> Smalltalk
> CLOS
> Java
> C++
> Generative programming
1. Introduction

Structural and behavioral reflection

> **Structural reflection** lets you reify and reflect on
  — the *program* currently executed
  — its *abstract data types*.

> **Behavioral reflection** lets you reify and reflect on
  — the language *semantics* and *implementation* (processor)
  — the data and implementation of the *run-time system*.

Malenfant et al., *A Tutorial on Behavioral Reflection and its Implementation*, 1996
1. Introduction

**Introspection in Java**

```java
// Without introspection
World world = new World();
world.hello();

// With introspection
Class cls = Class.forName("World");
Method method = cls.getMethod("hello", null);
method.invoke(cls.newInstance(), null);
```
Reflection in Smalltalk

```
HelloWorld new hello 'hello'
```

```
without reflection

(Object subclass: #HelloWorld
     instanceVariableNames: ''
     classVariableNames: ''
     poolDictionaries: ''
     category: 'HelloWorld'.
(Smalltalk at: #HelloWorld) compile: 'hello + "hello"'.
((Smalltalk at: #HelloWorld) perform: #new) perform: #hello.
'hello'
```
Meta Programming in Programming Languages

> The meta-language and the language can be different:
  — Scheme and an OO language

> The meta-language and the language can be same:
  — Smalltalk, CLOS
  — In such a case this is a *metacircular architecture*
Three approaches

1. Tower of meta-circular interpreters
2. Reflective languages
3. Open implementation
1. Tower of meta-circular interpreters

> Each level interprets and controls the next
   — 3-Lisp, Scheme

> “Turtles all the way down” [up]
   — In practice, levels are reified on-demand
2. Reflective languages

> Meta-entities control base entities
  — Smalltalk, Self
  — Language is written in itself
3. Open implementation

> Meta-object protocols provide an interface to access and modify the implementation and semantics of a language — CLOS

> More efficient, less expressive than infinite towers
1. Introduction

Roadmap

- Goals of this seminar
- Seminar topics
- Historical perspective
  - What is a model? A meta-model?
  - Reflection and reification
  - Reflection in programming languages
  - Model-driven engineering
Models and metamodels in software

- Databases
- Model-driven engineering (MDE/MDA)
- XML
- Domain specific languages
- Round-trip engineering
1. Introduction

**MDA in a nutshell**

- One unique Metametamodel (the MOF)
- An important library of compatible Metamodels, each defining a DSL
- Each of the models is defined in the language of its unique metamodel
1. Introduction

The OMG/MDA Stack

The MOF

The UML metamodel ++

Some UML Models ++

Various usages of these models

"the real world"

© Oscar Nierstrasz
The Vision of MDA

software developer

Platform Independent Model

automatic translation
PyPy — Model-driven language implementation
What you should know!

- What is the relationship between a **model** and its **meta-model**?
- How is a **meta-model** also a model?
- What is the difference between **descriptive** and **prescriptive** models?
- Do we need **meta-meta-models**?
- How is **programming** like **modeling**?
- What is the difference between **introspection** and **intercession**?
- What is **reification** and what is it for?
- What is the difference between **structural** and **behavioural reflection**?
- What are M0, M1, M2 and M3 in **MDA**?
Can you answer these questions?

- What kind of reflection does Java support? C++?
- What would it mean to turn Pascal into a reflective language?
- What exactly is “meta-circular” about a “meta-circular architecture” mean?
- In practice, how would you implement a programming language as an infinite tower of meta-circular interpreters?
- What are M1, M2 and M3 in relational databases?
- When does MDA/MDE pay off in practice?
License

> http://creativecommons.org/licenses/by-sa/2.5/

You are free:
• to copy, distribute, display, and perform the work
• to make derivative works
• to make commercial use of the work

Under the following conditions:

BY: Attribution. You must attribute the work in the manner specified by the author or licensor.

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

• For any reuse or distribution, you must make clear to others the license terms of this work.
• Any of these conditions can be waived if you get permission from the copyright holder.

Your fair use and other rights are in no way affected by the above.