

Metamodelling and Model-Driven Engineering

Department of Informatics
University of Oslo

Tuesday 19th of March 2013

Outline

- Metamodelling
- Model-Driven Engineering (MDE)
- Reasons to use metamodelling and MDE
- Challenges

History

ἀφίστασθαι ἀπὸ τῆς αἰῶνος· καὶ πηγάσαν
τὸς αὐτὸν τοῖς ὀφθαλμοῖς οὗ ταύτην τῶν
λαϊκῶν δαιμονίων ὁμοίαντα αὐτὸν δεξ-
πύματα, ἡ μὲν ἀπὸ τῆς ξυμμαχίας αὐ-
τομόμου οὗ ἀπ' προσφάσεται, οὕτω
δέχονται τὸν φρατὸν· καὶ οὕτω μὲν
ἀφρον καὶ ἀφρον ἀφ' ἄλλων ἀπὸ τῆς
ζωῆς ἀπὸ τῆς· ταῦτα μὲν οὕτω βίβω φέρει
τοῦτο δὲ βίβω· τοῦ δὲ ἀπὸ τῆς γομῆς
χῆμῶνος βίβω ἀρχομῆς οὗ τῶ ἰω-
ποῦρα τῆς καὶ δημοστὸν φρατῆς



亞物瑪利亞滿被額
辣濟亞者主與爾偕
馬女中爾為讚美爾
胎子耶穌併為讚美
天主聖母瑪利亞為
我等罪人今祈天主
及我等死候亞孟

Metamodelling

What is metamodelling?

- **Analysis** and **construction** of artefacts and concepts for modelling a predefined class of problems
- Metamodelling results in a **metamodel**
- A metamodel is an **abstraction** of the properties of conforming models
- Hence, a metamodel reflects a **problem domain**

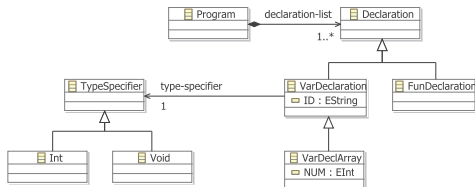
Metamodelling

What is a metamodel?

- A metamodel primarily describes the legal **structure** of models
- Typically defined as a **class model**
- Resembles a **grammar** specification to some extent
- Semantics (both static and behavioural) may be seen as part of the metamodel (broader view)

1. *program* → *declaration-list*
2. *declaration-list* → *declaration-list declaration* | *declaration*
3. *declaration* → *var-declaration* | *fun-declaration*
4. *var-declaration* → *type-specifier ID ;* | *type-specifier ID [NUM] ;*
5. *type-specifier* → **int** | **void**

6. *fun-declaration* → *type-specifier ID (params) compound-stmt*
7. *params* → *param-list* | **void**
8. *param-list* → *param-list , param* | *param*
9. *param* → *type-specifier ID* | *type-specifier ID []*
10. *compound-stmt* → { *local-declarations statement-list* }
11. *local-declarations* → *local-declarations var-declaration* | *empty*
12. *statement-list* → *statement-list statement* | *empty*
13. *statement* → *expression-stmt* | *compound-stmt* | *selection-stmt*
| *iteration-stmt* | *return-stmt*
14. *expression-stmt* → *expression ;* | *;*



Metamodelling

That is the relation between metamodels and models?

- A metamodel describes a **language** - a set of models
- A model **conforms** to its metamodel
- Models are either **executable** or used as source for **code generation** (behavioural semantics/translational semantics)

Domain-Specificity

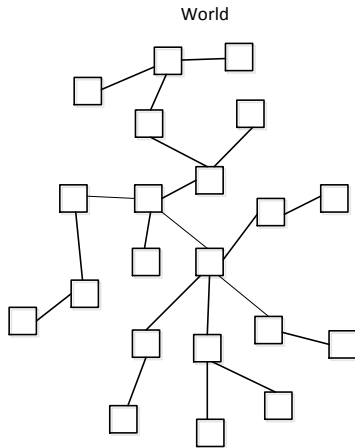
General purpose versus domain-specific (a class of problems)

ἀφίστασθαι ἀπὸ ἡμῶν· καὶ πισώσαν
τὸς αὐτὸν τοῖς ὀφθαλμοῖς οἷοι τὰ ἐξ ἡ τῶν
λαλεῖ δὲ μομίον ὁμοσηται αὐτὸν θεσ-
πόμεται, ἡ μὴν ὄσθαι ξυμμαχοῖς αὐ-
τομομοῖς οἷοι προσεθήται, οὕτω
δέχονται τὸν ἀρατὸν· καὶ ὀυπομνύ-
αθρον καὶ ἀφροσ ἀμδρεων ἀποιλια
ξω ἀπόσθαι· ταῦτα μὲν οὖν ἐγὼ φέρει
τοῦτα ἐγὼ γὰρ· τοῦ δὲ ἀπὸ ἡ γρομῶμου
ἡμῶν οἷοι ἀρχομῶν οἷοι τὸ ἴω
ποῦρα τῶ καὶ δημὸς τῶ ἀρατῆροῖς



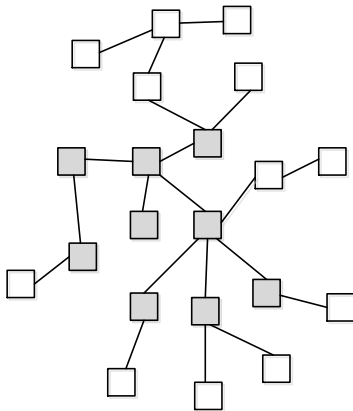
亞物瑪利亞。滿被額
辣濟亞者。主與爾偕
馬。女中爾為讚美。爾
胎子耶穌。併為讚美
天主聖母瑪利亞。為
我等罪人。今祈天主。
及我等死候。亞孟。

Domain-Specificity

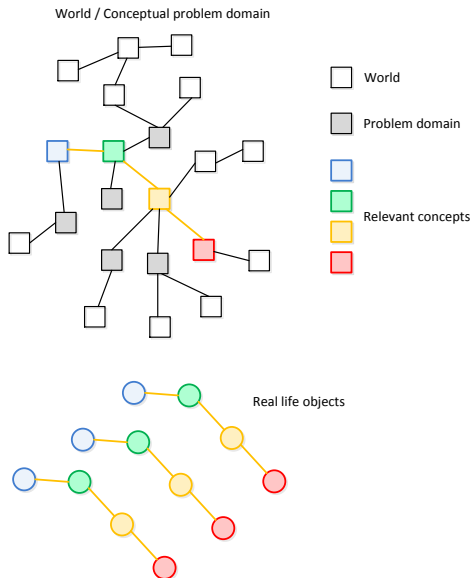


Domain-Specificity

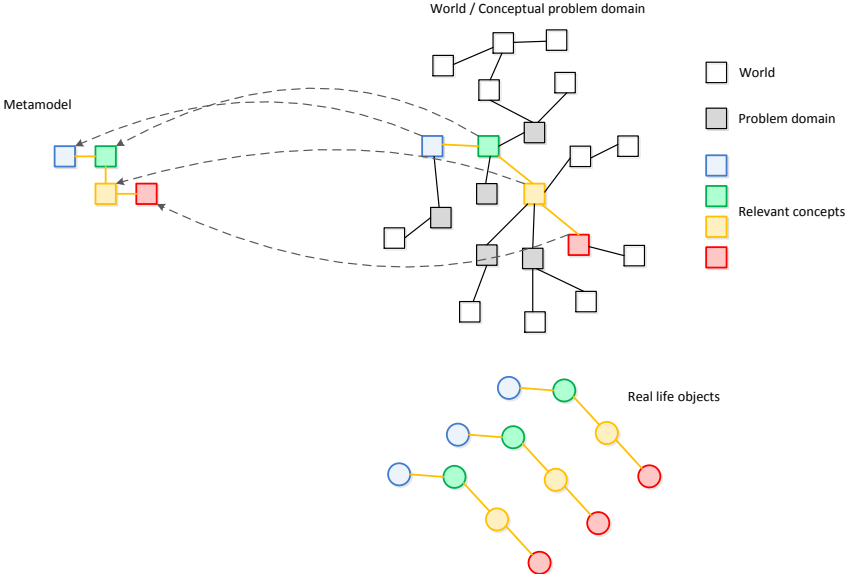
World / Conceptual problem domain



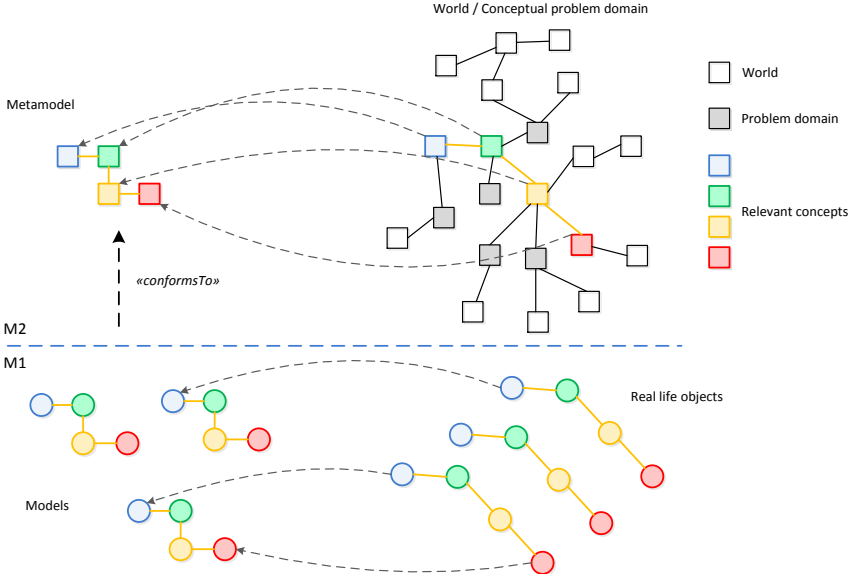
Domain-Specificity



Domain-Specificity

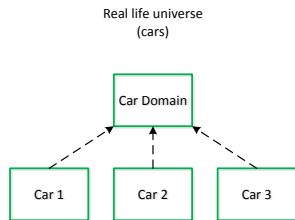


Domain-Specificity



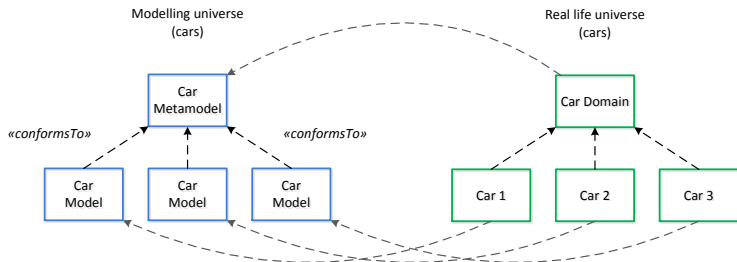
A Simple Example

A Domain-Specific Language (DSL) for modelling of cars



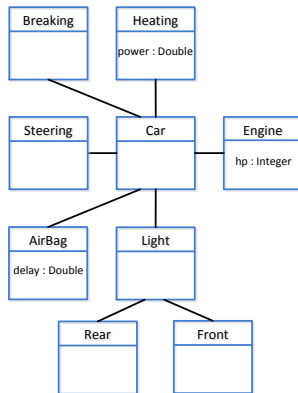
A Simple Example

A Domain-Specific Language (DSL) for modelling of cars



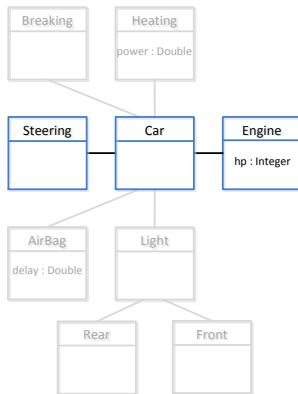
A Simple Example

A Domain-Specific Language (DSL) for modelling of cars



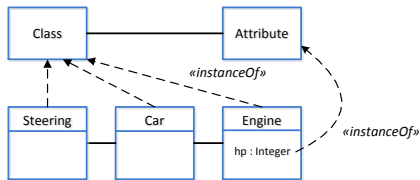
A Simple Example

A Domain-Specific Language (DSL) for modelling of cars



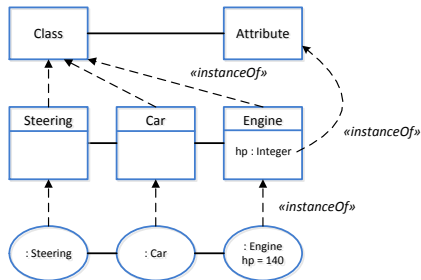
Metamodelling Architecture

MetaObject Facility (MOF)



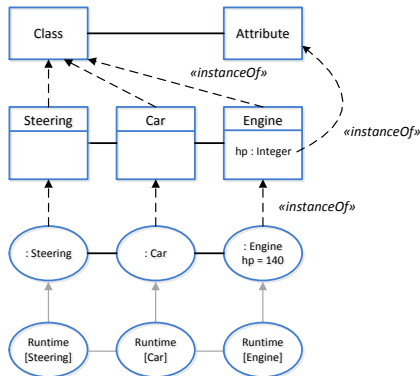
Metamodelling Architecture

MetaObject Facility (MOF)



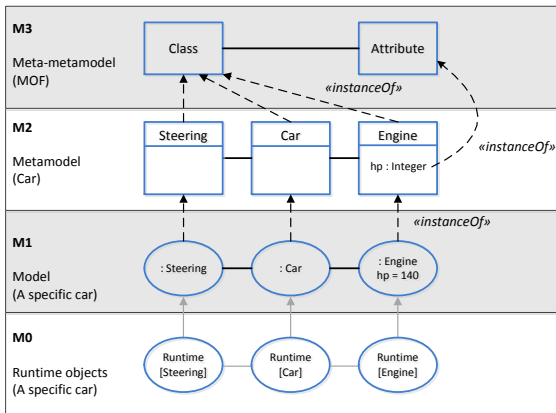
Metamodelling Architecture

MetaObject Facility (MOF)



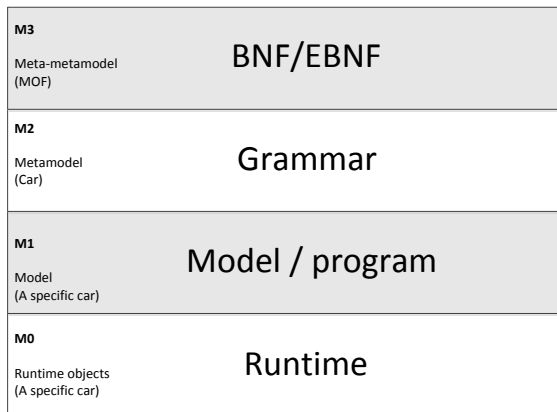
Metamodelling Architecture

MetaObject Facility (MOF)



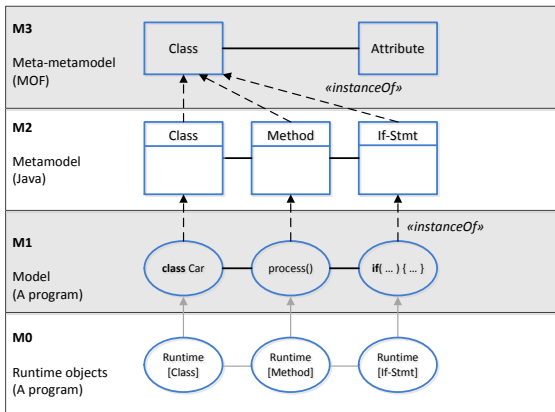
Metamodelling Architecture

MOF versus BNF/EBNF



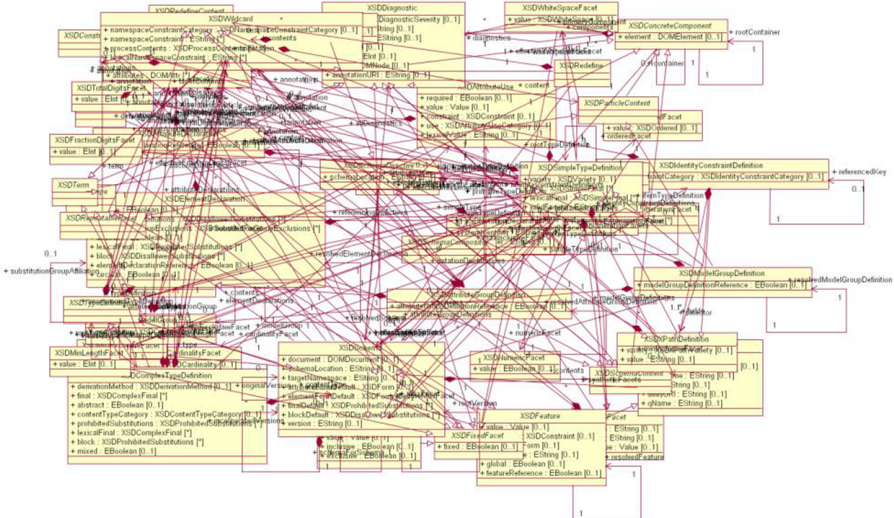
Metamodelling Architecture

MOF GPL example



Metamodel for XSD

XSD Metamodel

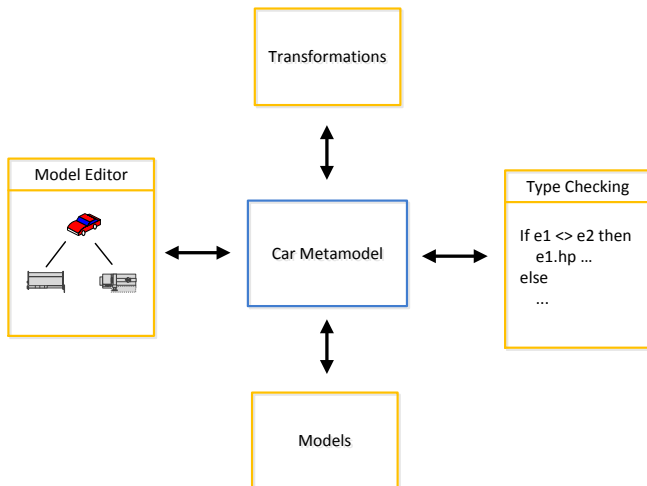


Model-Driven Engineering (MDE)

- Models and metamodels are **central artefacts**
- Metamodels used to, e.g. **formalise** languages (modelling and programming) and domain knowledge
- **Tools** and **editors** are defined relative to a metamodel
- **Transformations** are defined relative to one or more metamodels
- Important principles are automatic **code generation**, **abstraction** of details and **robustness** through reuse
- **Composition** of metamodels is required for code generation, consistency checking, addressing evolution and reuse

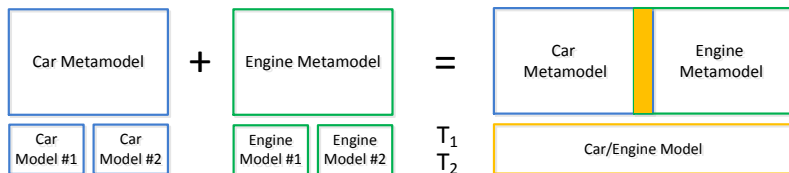
Artefacts in MDE

Metamodels, models and tools



Operations in MDE

Composition/weaving and transformations, etc.



Language Artefacts

- Language structure (constructs)
 - ▶ **Abstract syntax**
 - ▶ **Static semantics**, e.g. OCL constraints
- Presentation (interface)
 - ▶ **Graphical** symbols (concrete syntax)
 - ▶ **Textual** concrete syntax
- Meaning (semantics)
 - ▶ **Behavioural** semantics (e.g. in the form of methods or operations)
 - ▶ **Translational** semantics (using transformations to, e.g. a GPL)
 - ▶ **Denotational** semantics (mapping to mathematical objects)

Why use metamodeling, DSLs and MDE?

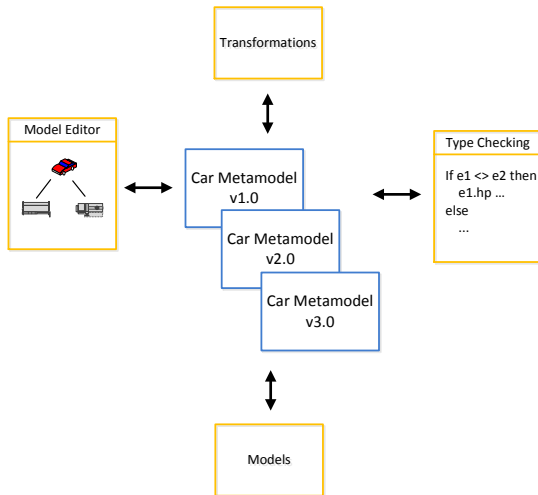
- Small languages - possible to focus on **aspects** or system **concerns separately**
- **Higher abstraction** level
- High degree of **automation**, e.g. code generation
- Rise in **productivity**
- Easier **communication** between stakeholders (model as a communication device)
- **Object-oriented** definition of structure
- **Reuse** through inheritance and composition/transformation mechanisms
- Possible to create **generic** tools (e.g. model editors)

Challenges

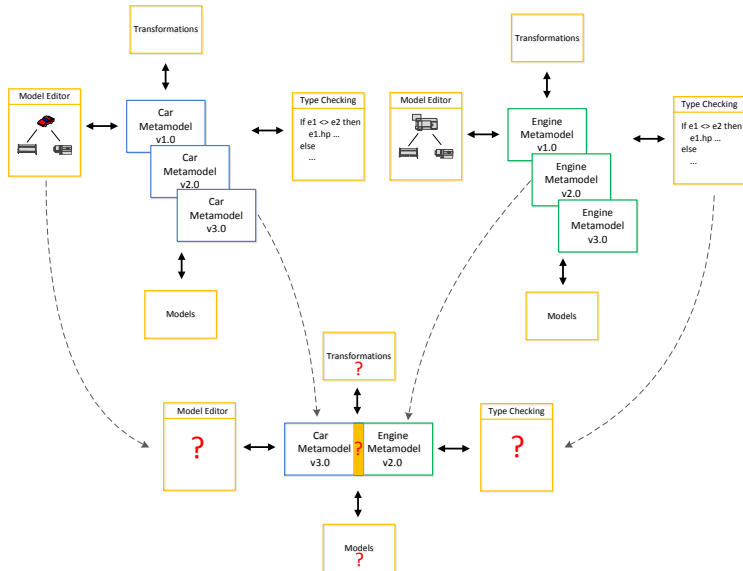
- Metamodels **evolve** due to changing requirements and domains
- Changing metamodels (including composition) compromises **compatibility** with existing models, tools and transformations
- Challenge: how to **reuse** and **compose** (integrate) metamodels and models
- Challenge: how to **address evolution**, where existing models, tools and transformations can be reused (e.g. by automatic revision)
- Challenge: how to **improve** metamodelling (how metamodels, languages and systems are created and modelled)

Challenge: changing metamodels

Including composition and weaving



Challenge: addressing evolution and tool reuse



Summary

- A metamodel **describes** or **formalises** a language (broader view: semantics included as part of metamodel)
- A model **conforms** to its metamodel
- Models are either **executable** or **transformed/mapped** to entities defining the semantics
- Metamodels and models are key **artefacts** in MDE together with **tools**
- Metamodelling and MDE support describing problems at **higher abstraction** levels - using models
- Advantages are increased **productivity** and **reuse**