Model Driven Engineering (MDE)

Yngve Lamo[1]

[1]Faculty of Engineering, Bergen University College, Norway

26 April 2011
Ålesund

Outline

Software complexity problems leads to new abstractions in software technologies every 10 year

- 1950s Assembly languages. Programs around 10^3 lines of code
- 1960s Imperative programming (Fortran 1954), compilers. Semantics of languages, correctness of programs. Programs 10^4 LOC. (Functional programming)
- 1970s increasing project size, modularity and information hiding, object orientation (Simula1967). 1968 first NATO conference in SE. Programs 10^5 LOC
- 1980s increasing project size changing requirements, OO mature, formal specifications introduced. Programs 10^6 LOC
- 1990s Distributed systems, internet, component models, quality of services, generic programming. Programs 10^7 LOC
- 2000s Multi-platform applications, mobile, cloud computing, **Model Driven Engineering?** Programs 10^8 LOC

MDE aims to

- Increase the level of abstraction and automation in program development by:
  - Using models as the major development artefact
  - Using executable model transformations for:
    - Program development steps, higher (abstract) level of models are transformed to lower (more concrete) level until the model is executable by code generation or model interpretation
    - Technology migration e.g. from a C++ Windows desktop application to a Android mobile application
    - Modeling of cross platform applications (SOA)

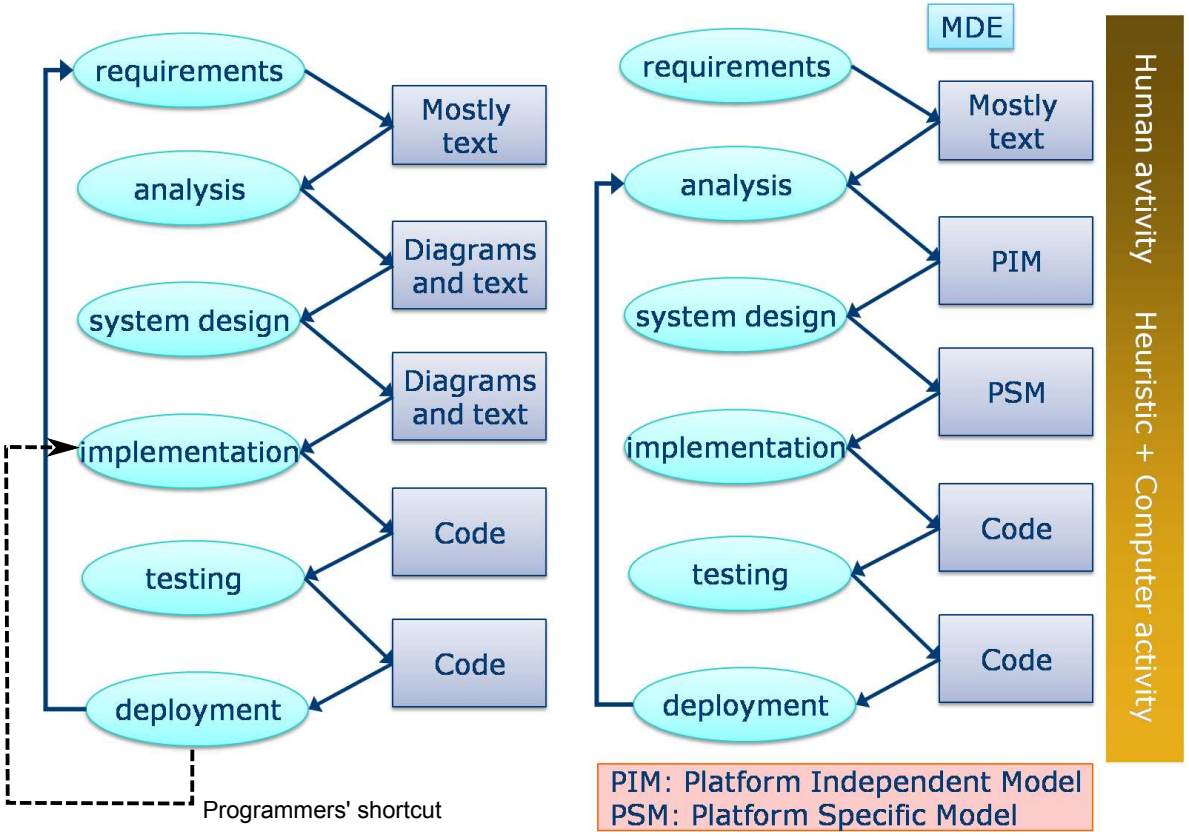
MDE

- Engineering techniques where models are first-class entities
- Evolved from the popularity of diagrammatic languages such as UML and ER(1976) and their popularity for specification and documentation of software systems
- Developed from CASE tools from the 80ies
- Aims to raise the abstraction level of software development from text to models
- The development process is based on software models, not on plain text

MDA implementation of MDE

- Model Driven Architecture (MDA) introduced in 2001 by Object Management Group (OMG), where:
  - First abstraction level is to specify a Computation independent domain model, CIM
  - Next step is building platform independent models, PIM's that focus on the functionality of the system without specific technologies
  - PIM's are refined to platform specific models PSM's
  - Refinement process more or less automatized as model transformations
  - Code generation and model refinement can be seen as model transformations
  - Reuse of software models by using model transformations

MDD vs Traditional Development Processes



Key point for success for MDE

Models are not longer only for documentation

- Models should be **both**:
  - intuitive for **software engineers** and **domain experts** to intercept and work with
  - formal such that **machines** could have precise understanding and reason about the models

Key point for success for MDE

Models are not longer only for documentation

- Models should be **both**:
  - intuitive for **software engineers** and **domain experts** to intercept and work with
  - formal such that **machines** could have precise understanding and reason about the models

$\Rightarrow$ Diagrammatic approach enhance human understanding

$\Rightarrow$ Challenge to combine intuition with formalization

Modeling

Model comes from Latin *modulus* meaning measure/standard

- A model is a pattern, plan, representation (especially in miniature), or description designed to show the main object or workings of an object, system, or concept
- A model need to meet the following 3 criteria

Reflection	A model reflects some properties of an original (system). The original system could be a construction that exists or is planned to be built, or a conceptual abstract idea
Abstraction	The model describes only some of the "interesting" properties of the original
Substitute	The model can be used instead of the original for some purposes

Some modeling languages

Behavior Trees formal, graphical modeling language used to unambiguously represent natural language requirements for large-scale software-integrated system

Business Process Modeling Notation (BPMN, it's XML form BPML and executabel form BPEL) examples of a Process Modeling language

Object Role Modeling (ORM) conceptual DB modeling language

Petri nets For model checking, graphically-oriented simulation, and software verification

Unified Modeling Language (UML)

Formal languages Algebraic, logic, categorical, set based, ...

Some facts about UML

- UML is defacto industry standard for modeling
- UML can only express constraints on binary relations, basically UML only gives cardinality constraints
- UML has serious issues regarding:
  - Semantics; UML models may be ambiguous and have semi-formal semantics
  - Complexity; UML uses 13 different types of diagrams
  - Expressibility; To express constraints over higher order relations one need to use string based logic, (Object Constraint Language, OCL)

Formal modeling languages

- Nice semantics, several fundamental Software Engineering problems solved by use of formal methods
- Set based semantics (logic, algebras, type theory, . . .)
- No concept of meta-modeling
- Hard for software engineers to apply in practice
- Lack of good software development tools based on formal approaches
- Only used by well trained experts
- High cost low productivity

Meta Modeling

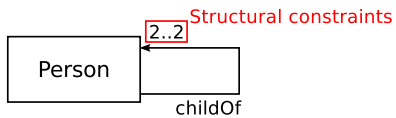
- A meta-model is a model of models
- Meta-models are used for defining modeling languages
- E.g. in OO modeling a person is an instance of class
- One can also use meta-modeling to create Domain Specific Languages DSL, i.e. we make language constructs for important concept, e.g. a student and a teacher is instances of persons

Meta-model example

Pattern

OMG

Example

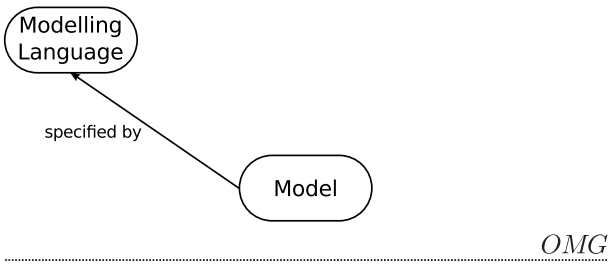


OMG

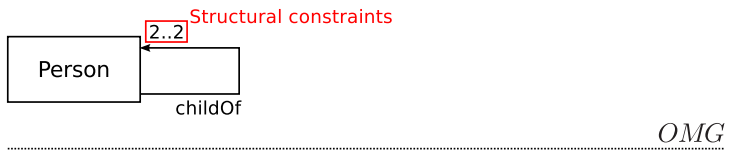
- Models: first class entities

Meta-model example

Pattern



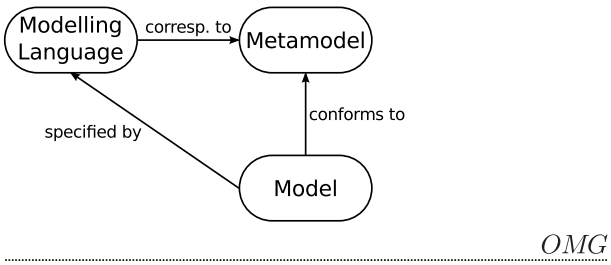
Example



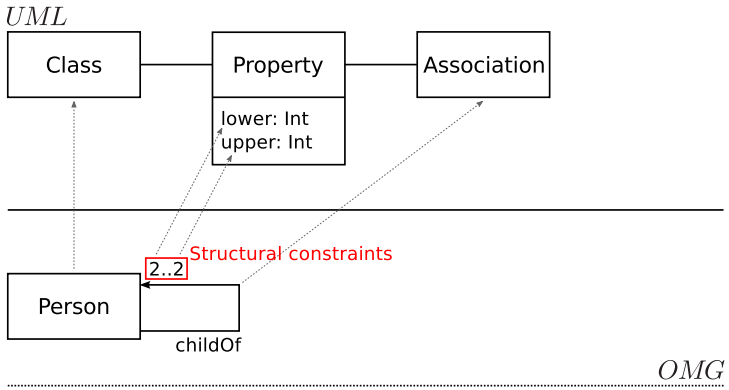
- Models: specified by means of a modelling language

Meta-model example

Pattern

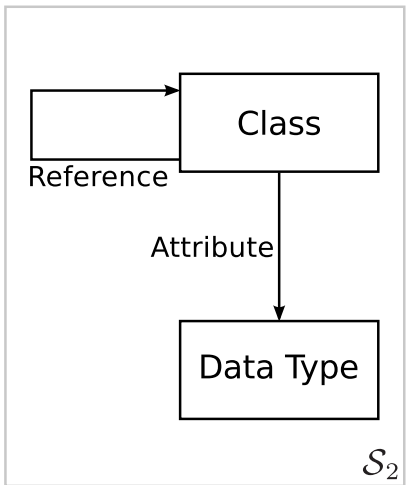


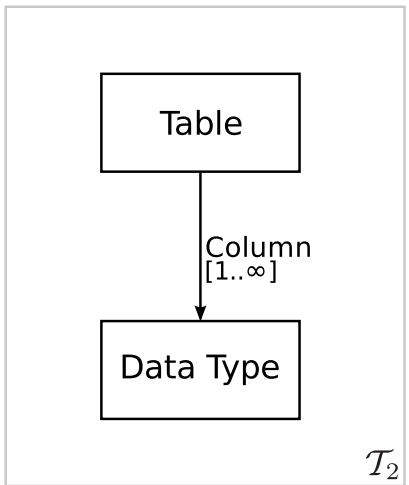
Example



- Modelling language: corresponding metamodel + semi-formal semantics

Meta-model for object orientation and databases





OMG Metamodel Levels

OMG levels	**OMG Standards/examples**
M_3: Meta-meta-model	MOF
M_2: Meta-model	UML language
M_1: Model	A UML model: Class "Person" with attributes "name" and "address"
M_0: Instance	An instance of "Person": "Ola Nordmann" living in "Sotraveien 1, Bergen"

Domain Specific Language, DSL

- A DSL is a modeling language made to model a particular problem domain
- The DSL is based on a meta model that captures essential concept of the domain

Model transformation

- A model transformation is a mapping that takes a model as input and creates a new model (program) as output
- Model transformations can be classified in:

  Homogenous transformation the metamodel of the input and the target model are equal

  Heterogenous transformations the metamodel of the input and output model are different

  Inplace transformation the input model are copied to the target model

  Outplace transformation the target model are created from scratch

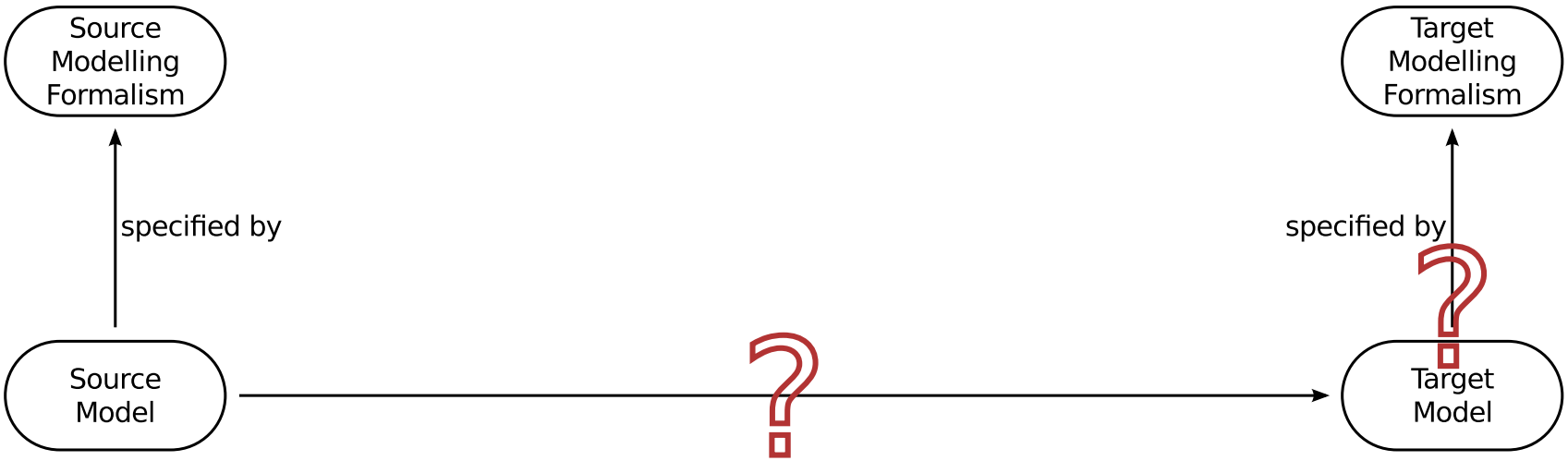
Model transformation

- Given source and target modeling formalisms and a source model ...

Model transformation

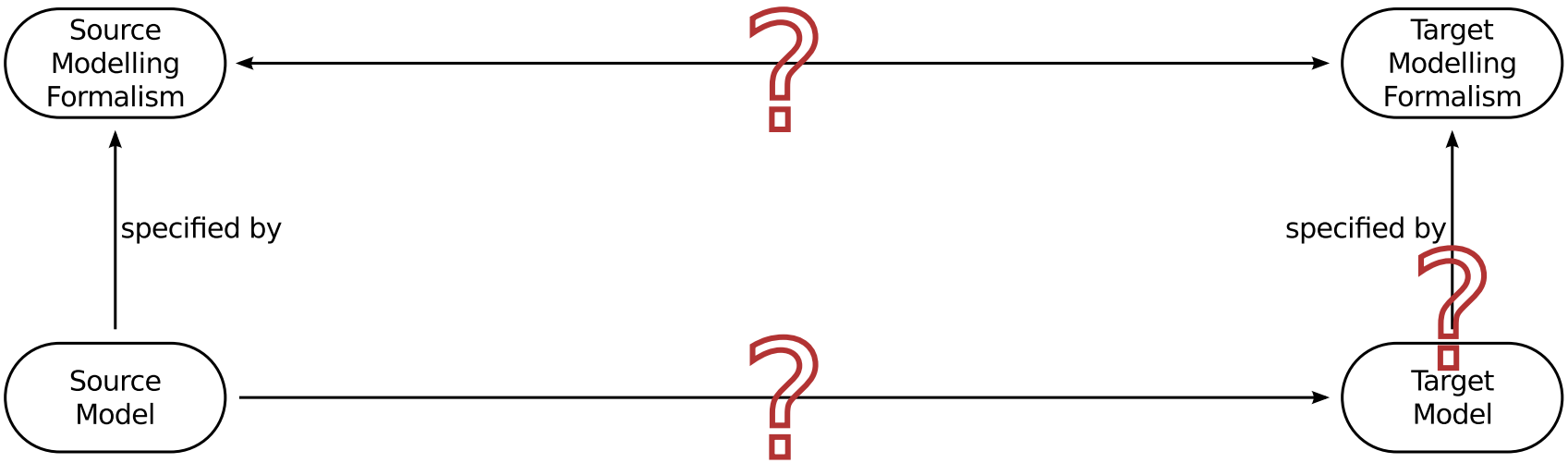
- ... we want to generate a target model

Model transformation



- How can we get the target model from the source model?

Model transformation



- We have to relate the modeling formalisms

State of the art in MDE

Modeling UML or EMF used as modeling language

Model transformations Rule based (Atlas) or add hoc transformations are used

Meta modeling Only support for 2 levels of meta-modeling

Tool support Eclipse based (EMF, GMF) tools

Software constraints Specified in text based language (OCL)

Links to resources

- mde-model-driven-engineering-reference-guide
- model-driven-engineering
- mda-model-driven-architecture-basic-concepts
- Diagram-Predicate-Framework
- Eclipse-Modeling-Technologies

Summary

- The industry have a lot of hidden knowledge, they see no need for formalization
- Modeling community work with typing (fibrations), not pointers (index) as we are trained to in mathematics
- Practical problems seems easy in theory but you really get dirt on your fingers
- MDE is a promising but one need formal understanding of modeling, meta modeling and model transformation
- MDE still need better tool support to be used in software industry
- New challenges regarding serialization and versioning of models arising in MDE