Model-Driven Software Engineering

Foundations of Model-Driven Software Engineering

Dr. Jochen Küster (jku@zurich.ibm.com)
Contents

- Introduction to Models and Modeling
- Concepts of Model-Driven Software Engineering
- Goals and Roadmap of the Lecture
- Overview of Approaches
- Summary and Literature
Model-Driven Software Engineering in a Nutshell

- Model-Driven Software Engineering (MDSE) is a software engineering paradigm
- Models are considered as primary artifacts from which parts of a software system can be automatically generated.
- Models are usually more abstract representations of the system to be built
- MDSE can improve productivity and communication
- MDSE requires technologies and tools in order to be successfully applied
- Various terms and approaches to MDSE
  - Model-driven architecture, model-driven engineering, model-driven development, …
Introduction to Models and Modeling
What is a Model?

“A model is an abstraction of something for the purpose of understanding it before building it” (J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. Object-Oriented Modeling and Design. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1991)

- Models are widely used in engineering disciplines
Examples of Models in Electrical Engineering
Important Properties of Models

- Abstraction from certain aspects of the real world
- Focus on certain aspects of the real world
- Ability to analyze properties of the system using the model
- Models are usually expressed in a modeling language with a well-defined syntax and semantics
- Many different forms of analysis, depending on the model and the application of modeling
Different kinds of models are used in software engineering
- Models for requirements analysis
- Models for expressing the software architecture of a system
- ...
Usage of Models in Software Engineering

- Models as description of the domain of the system to be built
  - Model focuses on relevant aspects of the domain
  - Example: Class diagram of the domain

- Models as abstract representation of the system to be built
  - Model focuses on aspects that are relevant, leaves other aspects open
  - Example: Component diagram specifies components of a system to be built

- Models for documentation
  - Abstraction of models helps to understand the system faster
  - Example: Class diagram of the key entities in a system are explained in a document

- Models as specification for testing
  - Model focuses on important aspects of the system for testing
Sample Models from Software Engineering

[Source: IBM developerworks, M. Berfeld, UML-to-Java transformation in IBM Rational Software Architect editions and related software, 2008]

[Source: IBM developerworks, F. Xu et al, Reverse engineering UML class and sequence diagrams from Java code with IBM Rational Software Architect, 2008]

[Source: IBM developerworks, D. Sheldon et al, Exploiting use cases to improve test quality, 2008]
Models can be used for many different purposes in software engineering
In different phases of the development lifecycle

[Illustration by B. Rumpe]
Concepts of Model-Driven Software Engineering
Concepts of Model-Driven Software Development

- How to get from requirements to running code satisfying requirements and user expectations?

- Models are used in many development processes
  - requirements for the system (e.g. use case model)
  - software architecture (e.g. component model)
  - behavioral description (e.g. statechart)
Usages of Models and Model Transformations in MDSE - Example

- In each phase, different models are constructed
- In each phase, we reuse parts of models as input for other models
  - Test scenarios are partially derived from Use Case Diagrams
  - Design Class Diagram may reuse parts of Domain Class Diagram
- MDSE focuses on automatically generating parts of models or code from other models
Eclipse Modeling Framework as an Example

Many lines of code are automatically generated!
Important Aspects of MDSE

- In MDSE approaches, the use of models and model transformations is proposed.
- Models are expressed in UML, an extension of UML, or a domain-specific language.
- The syntax and semantics of models used in a MDSE approach has to be clearly defined.
- The software development process is changed when an MDSE approach is adopted.
Questions

So many new concepts and terms…

Modeling Language?

Code generation?

Model Transformation?

Syntax and Semantics?

Domain-specific language?
Goals and Roadmap of the Lecture
Goals of the Lecture

- Understand principles and concepts of Model-Driven Software Engineering (MDSE)
  - Modeling language, meta-modeling, domain-specific language, model transformations, code generation
  - Different approaches to MDSE

- Get familiar with languages and technologies of Model-Driven Software Engineering (MDSE)
  - Eclipse Modeling Framework
  - Technologies for model transformations and code generation

- Apply MDSE in practice and get to know tools
  - Eclipse Modeling Framework Example, Service-Oriented Architecture Example
  - Extensions of Eclipse for model transformations and code generation
  - IBM Rational Software Architect
Roadmap for Model-Driven Software Engineering

- Foundations (1 lecture)
- Metamodels and Domain Specific Languages (2 lectures)
- EMF as Architecture Centric MDSD Environment (2 lectures)
- Model transformations (Model-to-Model, Model-to-code, transformation languages) (2 lectures)
- Code generation (1 lecture)
- MDSE of SOA Applications with IBM Rational Software Architect (2 lectures)
- Models in Software Architecture Design (1 lecture)
- Software Product Lines (1 lecture)
Overview of Approaches
Different Approaches to MDSE

- Model-Driven Architecture (MDA)
  - OMG MDA initiative

- Model-Driven Software Development (MDSD)
  - M. Voelter et al.

- Domain Specific Modeling (DSM)
  - S. Kelly, J. Tolvanen
MDA Concept Overview

- Computation Independent Model (CIM) defines domain vocabulary
- Platform Independent Model (PIM) captures domain-related specifications
- PIM does not contain platform details, independent of a platform
- Platform Specific Model (PSM) captures specifications with platform details
- For expressing PIM and PSM, domain-specific languages are used
  - UML profiles and other techniques for defining DSLs
- Model transformations transform PIMs into PSMs
public interface Account extends EJBOBJECT {...}
public interface AccountHome extends EJBHome {...}
public abstract class AccountBean implements EntityBean {...}
...

[Example from T. Stahl et al]
Advantages and Disadvantages of MDA

- Advantages:
  - Separation of PIM and PSM enables better reuse
  - Improved interoperability due to standards (e.g. UML)

- Disadvantages:
  - Code generation is only partial and requires manual completion of code
  - Semi-automatic generation of one model from another model leads to maintenance problems if a model is changed
MDSD is based on the following observations:

- Generic code is identical for all applications.
- Schematic code possesses the same systematics (e.g., based on an architectural pattern).
- Individual code is application specific.
Generate generic code for the platform instead of writing it

Generate schematic code using transformations based on an application model

Write individual code that is application specific
Architecture-Centric MDSD

- Application
  - Architecture-Centric Design Model
  - Infrastructure code
    - Business Logic Code manually written
- Generative Architecture
  - Domain-specific Language expressed in
  - Generator Templates
  - Infrastructure Components

- Generator
Example for Architecture-Centric Design Model

- Domain related meaning is expressed in the architecture-centric design (using stereotypes), can be considered as PIM
- Depending on the platform, the PIM is translated differently to code

[Example from T. Stahl et al]
Example Translation into Code (Sketch)

- **EJB-based architecture with HTML clients**
  - Activity classes are stateless session beans
  - Entity classes are beans
  - Attributes of type key constitute the primary key classes
  - For public attributes, getter and setter methods are applied
  - Presentation classes specify JSP models that are used to fill JSP/HTML pages
  - …

- **C++/CORBA-based client-server architecture**
  - Activity classes are IDL interfaces, all attributes are mapped to IDL types
  - Entity classes are non-distributable C++ classes
  - Presentation classes are Java Swing GUIs
  - …
Comparison to MDA

- MDSD does not focus on iterative model refinement by transformations, no intermediate models are created
- Transformations are primarily used for translating models into code
- A PIM model contains all necessary details to be translated into code which is then platform specific
- Roundtrip engineering is avoided, design changes have to be made to the model
- Focus on software architecture
- No 100 per cent generation, rather 60 to 80 percent
Advantages and Disadvantages of MDSD

Advantages:
- Increased development speed
- Increased software quality
- Better maintainability
- Better reusability
- Increased manageability of complexity
- Better portability and interoperability

Disadvantages:
- MDSD has to be tailored to the domain, no off-the-shelf solution
- No platform-independence of models
Domain Specific Modeling in a Nutshell

- Raise level of abstraction by specifying solution in a domain specific language
- Generate final products from these high-level specifications
- Model is expressed in the concepts of the domain
- Code is fully generated
Generator can be considered as a compiler
Modification of the generated code is not needed
Generated code accesses a domain framework
Value of Domain Specific Modeling

- Productivity within software development
  - Higher level of abstraction leads to higher productivity
  - Common defects when coding are avoided due to generation

- Quality of the produced solution
  - Early validation with the customers
  - Risk reduction of code not meeting the requirements

- Improved testing approaches
  - Testing of the generator vs testing of the model
Comparison of DSM to MDSD and MDA

- DSM puts a lot of emphasis on the domain-specific modeling language
- DSM does not favor to use UML or UML extensions as a DSM (in comparison to MDSD)
- DSM proposes to generate the solution from the model, without intermediate models (similar to MDSD)
- Generators as well as the DSM itself are developed by domain experts
Common Aspects of MDSE

- In MDSE approaches, the use of models and model transformations is proposed
- Models are expressed in UML, an extension of UML, or a domain-specific language
- The syntax and semantics of models used in a MDSE approach has to be clearly defined
- The software development process is changed when an MDSE approach is adopted
Basic Conceptual Architecture of MDSE

Modeling

Application

Model

Modeling language

Application domain

Transformation / Code generation

Abstraction (bottom-up)

Transformation definition

Meta-Level

Reuse

Construction (top-down)

Transformation language

Artifacts (e.g. code)

Platform

[Slide by G. Kappel]
Advantages of MDSE

- **Abstraction** from specific realization technologies
  - Improved **portability** of software to new/changing technologies – model once, build everywhere
  - **Interoperability** between different technologies can be automated
  - Requires modeling languages, which do not hold specific concepts of realization technologies (e.g., Java EJB)

- **Automated code generation** from abstract models
  - e.g., generation of Java-APIs, XML Schemas, etc. from UML
  - Requires expressive und precise models
  - Increased **productivity** and **efficiency** (models stay up-to-date)

- **Separate development** of application and infrastructure
  - Separation of application-code and infrastructure-code (e.g. Application Framework) increases **reusability**
  - **Flexible** development cycles as well as **different development roles** possible

[Slide adapted from G. Kappel]
General Requirements for MDSE

- Models used for generating other models have to contain all details that are needed
  - Model quality
  - Models must be precise with well-defined syntax and semantics (if used for e.g. code generation)
  - Model must be appropriate to express concepts of the domain

- Technology
  - For defining model transformations from model to code as well as model to model
  - For keeping models consistent if changes occur in one model
  - For supporting versions of models and multi-user modeling

- Development process
  - Has to take into account how to generate models
Summary and Literature
Summary of Lecture

- Models provide an abstraction from the real world
- Models are expressed in a modeling language
- Model-driven software engineering uses models to generate other models or code
  - Domain-specific models
  - Model transformations
- MDA, AC-MDSD and DSM represent different approaches to model-driven software engineering, however many common aspects exist
- MDSD requires skills and understanding of concepts, techniques and tools to be successfully applied
Literature