An Introduction to Model-Driven Engineering (MDE)

Bahman Zamani, Ph.D.
bahmanzamani.com

Department of Software Systems Engineering
University of Isfahan

Fall 2013

Overview

- ✓Model & Modeling
- ✓UML & UML Profile
- ✓Model-Driven Approaches
  - ✓MDE
  - ✓MDA
- ✓The dream comes true
  - ✓xUML
  - ✓OOIS UML

Model [Sel 06]

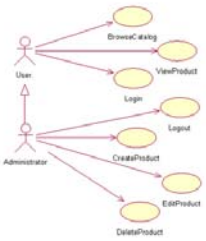
- What is a model?
  - A model is a representation of a system.
  - A representation of a system that hides some of the properties and highlights the ones that are of interest for the user.
  - A model is an abstraction.

Model Forms

- Mathematical, eg Linear Programming
- Physical, eg Aircraft or bridge
- Diagrammatic, eg Use case model

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum a_{ij} x_j \le b_i \quad (i = 1,2,\ldots,m) \\ & x_j \ge 0 \quad (j = 1,2,\ldots,n) \end{aligned}$$

http://www.cise.ufl.edu/~davis/Morgan/image13.gif



http://www.javaworld.com/javaworld/jw-07-2004/images/jw-0719-jxt1.gif

http://yourmilitaryaircraft.com/pics/c130-20.jpg

http://texnrails.com/images/track/tms/1200701.jpg

Why Model?

- In most of the engineering disciplines, it is de rigueur to use models when designing a complex system [Sel 03].
- Since today's software systems are becoming more and more complex, benefiting from using models is inevitable [Sel 03].

Four aims of modeling

1. Visualize a system
   (*visualization*)
1. Specify the structure and behavior of the system
   (*specification*)
2. Give templates that guide in constructing systems
   (*construction*)
3. Document the design decisions
   (*documentation*)

Raising the Level of Abstraction [Mel02]

- The history of software development is a history of raising the level of abstraction:

- Programming languages
  - Structured: Fortran, Cobol, C
  - OO: Smalltalk, C++, Java
- Mnemonics code (assembly)
- Machine code

At first, each higher layer of abstraction was introduced only as a concept.
Then the tools for automatic conversion to lower layers are invented!

The Unified Modeling Language (UML)

- To do modeling we need a modeling language
- UML is a graphical language for
  - visualizing
  - specifying
  - constructing
  - documenting

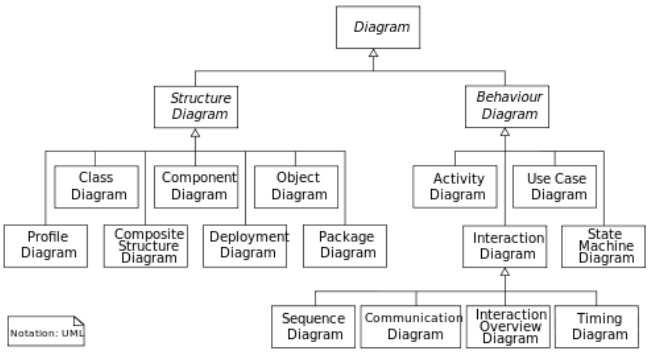
    the software artifacts.

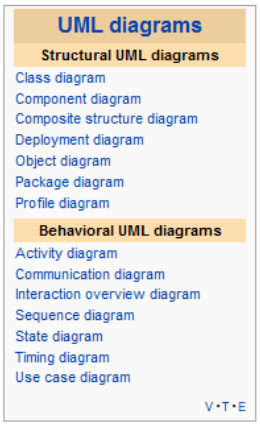
http://www.uml.org/

- In the graphical modeling of OO software systems, UML is the dominant approach.
- UML 1.1 adopted by OMG (www.omg.org) in November 1997
- Current release is UML 2.4.1 (Dec 2013).

UML 2.2 Diagrams

[http://en.wikipedia.org/wiki/Unified_Modeling_Language]





UML Profile [zam 09]

- A mechanism provided by UML that allows us to
  - Extend it for specific purposes, in other words
  - To define a Domain-Specific Modeling Language (DSML)
- Defining a UML Profile includes
  - Identifying a subset of UML metamodel
  - Specifying the well-formedness rules, preferably by OCL
  - Defining Stereotypes, Tagged Values, and Constraints

Model-Driven Approaches

- MDA: Model-Driven Architecture
- MDD: Model-Driven Development
- MDE: Model-Driven Engineering
- MDSD: Model-Driven Software Development
- MDSE: Model-Driven Software Engineering

http://www.t-one.net/~om/NCTblog/ist2_2771207_dizzy.jpg

Don't worry, just go **Model-Driven**!

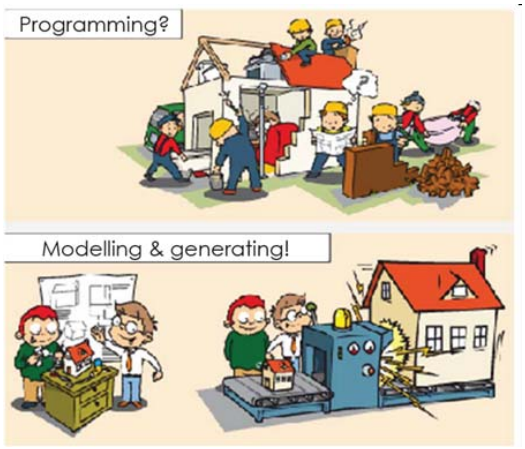
Model-Driven Approach of Software Development

- Despite the processes that are code-centric, in these approaches, models are the main artifacts which drive the development ➔ **Model-Driven** [Zam 09]

http://www.clker.com/clipart-2403.html

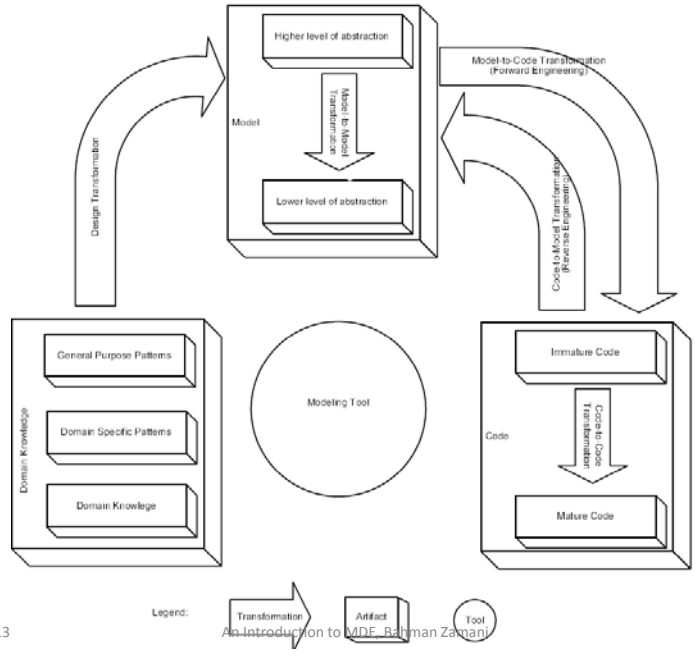
- The ultimate goal is to automatically generate programs from the corresponding models [Sel 03].

A metaphor for MDE



http://www.theenterprisearchitect.eu/archive/2009/08/05/a-metaphor-for-model-driven-engineering

An MDE Roadmap [Zam 09]



An MDE Roadmap [Zam 09]

- MDE is a model-centric approach.
- The software development is indeed a correct application of some transformations.
  - That means developers transform artifacts from one level of abstraction to another level, until they obtain a working code.
- For applying MDE, we need a modeling tool to utilize automatic execution of transformations.

Transformations [Zam 09]

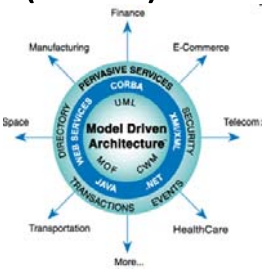
- Modes of transformation
  - Manual
  - Automatic
- Types of transformation
  - Model-to-Model
    - Refinement
    - Refactoring
  - Model-to-Code = Forward Engineering
  - Code-to-Model = Backward Engineering

Round-Trip Engineering

  - Code-to-Code

Model-Driven Architecture (MDA)

- A primary example of MDD
- An example of MDE vision
- A framework for MDSD [Sen 03]
- The realization of MDE principles around a set of OMG standards [Bez 05]

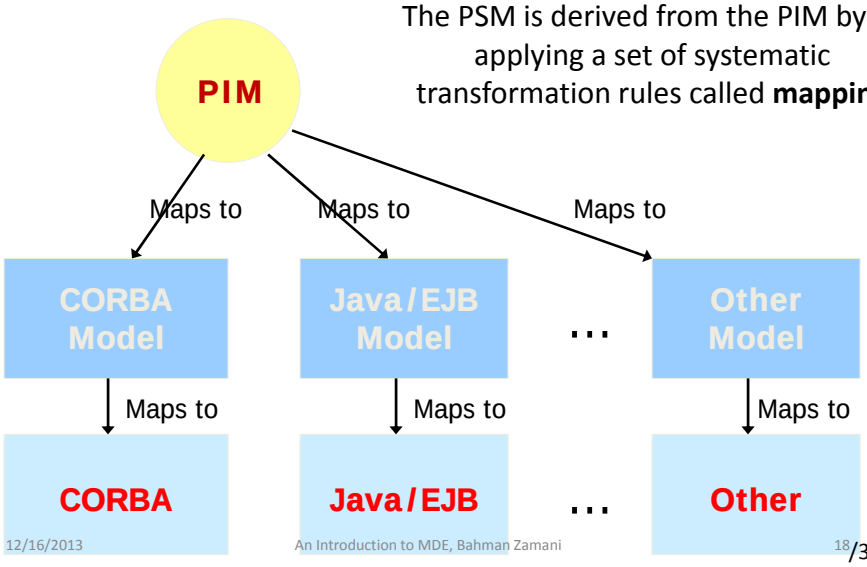


http://www.omg.org/mda/

- MDA Separates
  - the specification of functionality from
  - the specification of the implementation
- There are two types of models
  - Platform-Independent Model (PIM)
  - Platform-Specific Model (PSM)
- Design once, build it on any platform (.NET, J2EE, CORBA,…)

MDA: Mapping from PIM to PSM

The PSM is derived from the PIM by applying a set of systematic transformation rules called **mapping**



Executable UML (xUML) [Mel02]

- xUML is a profile of UML that allow defining the behavior of a single subject matter in sufficient detail that it can be executed.
- To build a system, we build an executable UML of each subject matter.
- Typically, the system includes subject matters such as
  - the application
  - a user interface
  - some general services
- The xUML models for each of these subject matters are then woven together by an xUML model compiler.

xUML [Mel02]

- Uses three basic types of models:
  - UML class diagrams
    - abstract common real world objects into classes
  - UML state charts
    - each active object has a lifecycle which is abstracted as state machines
  - Action Language
    - specifies the behavior of the object as it goes through each state of its lifecycle
- Other UML diagrams can be used to support the construction of xUML models

xUML [Mel02]

- An xUML specification comprises a set of models.

Concept	Called	Modeled As	Expressed As
the world is full of things	data	classes attributes associations constraints	UML class diagram
things have lifecycles	control	states events transitions procedures	UML statechart diagram
things do things at each stage	algorithm	actions	action language

Concepts in an Executable UML Model

Model Compilers [Mel02]

- An xUML model completely specifies the semantics of a single subject matter.
- An xUML model compiler turns an xUML model into an implementation using a set of decisions about the target hardware and software environment.
- There exist many compilers that can translate an xUML model into, for example
  - Multi-tasking C++ optimized for embedded systems, targeting Windows, Solaris, etc
  - Multi-processing C++ with transaction safety and rollback
  - Fault-tolerant, multi-processing C++ with persistence supporting three processor types and two operating systems
- Example of off the shelf compilers
  - Example: BridgePoint, iUML, Rational Rose
- Do not modify generated code!

Note:

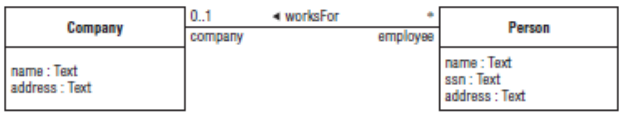
- following slides are based on the book:

Dragan Milicev, "Model-Driven Development with Executable UML," Wrox Programmer to Programmer, 2009.

✓ The book proposes a new executable profile of UML
for the domain of information systems
referred to as the *OOIS UML profile*

Modeling Languages

- The concrete notation of modeling and implementation languages can be textual (sequential) or visual (diagrammatic)
- Both textual and diagrammatic languages have their advantages and drawbacks
- Usually, a combination of both is most efficient in modeling



(a)

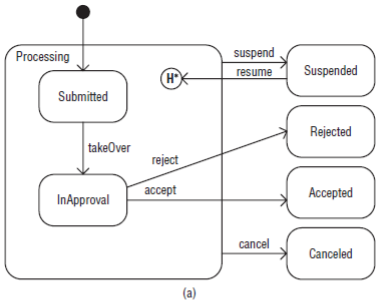
```
class Company {
public:
  String name;
  String address;
  Collection<Person*> employee;
};
```

```
class Person {
public:
  String name;
  String ssn;
  String address;
  Company* company;
};
```

(b)

Figure 1-2: Abstract vs. low-level modeling and visual vs. textual modeling languages. (a) A model in a highly abstract, visual language. (b) A model in a low-level, textual language. Both examples model the same simple fact from the problem domain that a person (described with name, social security number, and address) may work for at most one company (described with name and address), and a company may employ many persons.

Modeling Languages



(a)

```
switch (state) {
  case Submitted:
    switch (event) {
      case takeOver: state = InApproval; break;
      case suspend:  prevState = state; state = Suspended; break;
      case cancel:   state = Canceled; break;
    };
    break;
  case InApproval:
    switch (event) {
      case reject: state = Rejected; break;
      case accept: state = Accepted; break;
      case suspend: prevState = state; state = Suspended; break;
      case cancel:  state = Canceled; break;
    };
    break;
  case Suspended:
    if (event==resume) state = prevState; break;
}
```

(b)

Figure 1-3: Abstract vs. low-level modeling and visual vs. textual modeling languages. (a) A model in a highly abstract, visual language. (b) A model in a low-level, textual language. Both examples model the lifecycle of an application for a course.

Modeling Tools

- Tools are key factors to successful production
- Different kinds of software systems are used as modeling tools:
  - *Model editor:* is used for creating and editing models, responsible for their specification, visualization, and consistency checking
  - *Transformers, translators, generators, or compilers:* transform models into lower-level forms
  - *Code generator:* sometimes called model compiler

Executable UML Foundation

- A recent initiative of OMG is the work on the *Executable UML Foundation*
- The approach described in this book is a result of an independent and simultaneous work that is similar with the Executable UML Foundation in many aspects, most notably in its idea of the formalization of the run-time semantics of a selected part of UML
- However, the approach presented in this book is tailored for one particular, yet very broad, domain of applications (information systems), and brings other concepts that are suitable for modeling in that domain

Profiling UML

- Making a profile of UML for modeling information systems is actually the core goal of this book
- It is a profile of UML for object-oriented information systems (OOIS), called the *OOIS profile of UML (OOIS UML for short)*

Key Features of OOIS UML

- A quick look at *what* can be done with OOIS UML, not precisely *how*
- Using a simple example of a small information system for an imaginary school named Easylearn

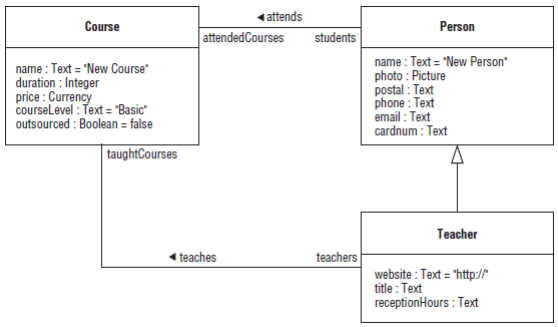


Figure 4-1: A sample UML model for the Easylearn school system

Key Features of OOIS UML

A running application is resulted, without writing a single line of code, from the model (→ MDD)

Figure 4-2: The appearance of the prototype application obtained from the sample UML model for the Easylearn school system

The organization of OOIS UML

OOIS UML is an MDD method consists of:

- **Language**
  - is defined as an executable profile of UML
- **Model library**
  - some reusable, general-purpose models, built in the OOIS UML language, available to modelers
- **Run-time environment**
  - the execution engine for maintaining the object space to the running application
- **Design process**
  - a set of guidelines and hints of how to apply the language

The organization of OOIS UML

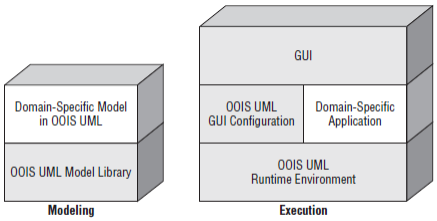


Figure 4-3: A block diagram of the components of the model and of the running information system build in OOIS UML.

- The (major part of) models built in OOIS UML are formal and executable
- When a model is developed, it can be executed within the run-time environment, after a possibly needed compilation, such as a standard computer program
- The generic GUI of the execution environment enables *rapid prototyping* of the system and a kind of *model debugging*

References

- [Bez 05] Jean B´ezivin. On the unification power of models. *Software and System Modeling, 4:171–188, May. 2005.*
- [Bez 06a] Jean B´ezivin and Fr´ed´eric Jouault. Using ATL for checking models. *ENTCS, 152:69–81, Mar. 2006.*
- [Bez 06b] Jean B´ezivin. Model driven engineering: An emerging technical space, *LNCS 4143, pages 36–64. Springer, 2006.*
- [Dug 07] Asif Dugar. *Model Driven Deevelopment for Enterprise Applications,* MSc thesis, Concordia University, Canada, 2007.
- [Jou 08] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. ATL: A model transformation tool, *Science of Computer Programming, 77(1-2):31-39, Jun. 2008.*
- [Mel02] Stephen J. Mellor and Marc J. Balcer, "Executable UML: A Foundation for Model-Driven Architecture," Addison Wesley, 2002.
- [Mil09] Dragan Milicev, "Model-Driven Development with Executable UML," Wrox Programmer to Programmer, 2009.
- [MOF2] OMG. MOF Core Specification, v2.0. OMG document: formal/06-01-01, 2006.
- [Obj 05] Object Management Group (OMG). Unified Modeling Language (UML): Infrastructure, v2.0. OMG document: formal/05-07-05, 2005.
- [Pre 10] Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, Seventh Edition, McGraw-Hill, 2010.
- [Sch 06] Douglas C. Schmidt. Guest editor's introduction: Model-driven engineering. *IEEE Computer, 39(2):25–31, Feb. 2006.*
- [Sel 03] Bran Selic. The pragmatics of model-driven development. *IEEE Software*, 20(5):19–25, Sep. 2003.
- [Sel 06] Bran Selic. Model-driven development: Its essence and opportunities. In *Proc. ISORC'06*, pages 313–319, Los Alamitos, CA, USA, 2006. IEEE Computer Society Press.
- [UML2] OMG. UML Infrastructure, v2.0. OMG document: formal/05-07-04, 2005.
- [Zam 09] Bahman Zamani. *On Verifying the Use of a Pattern Language in Model Driven Design.* PhD thesis, Concordia University, Canada, 2009.

Thank You!

Questions?