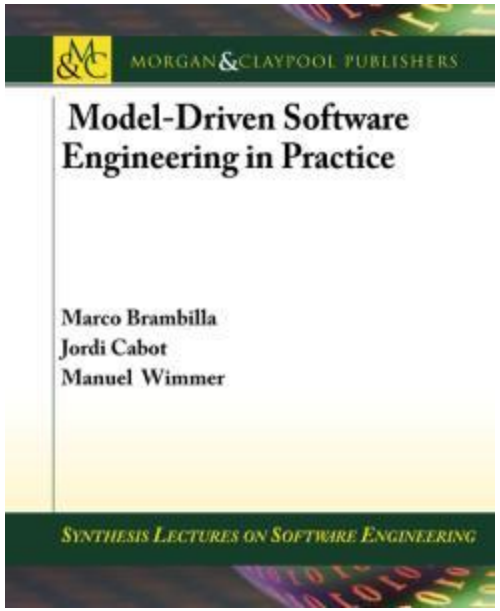
Chapter #1

INTRODUCTION

Teaching material for the book
Model-Driven Software Engineering in Practice
by Marco Brambilla, Jordi Cabot, Manuel Wimmer.
Morgan & Claypool, USA, 2012.



Introduction

Contents

- Human cognitive processes
- Models
- Structure of the book

Abstraction and human mind

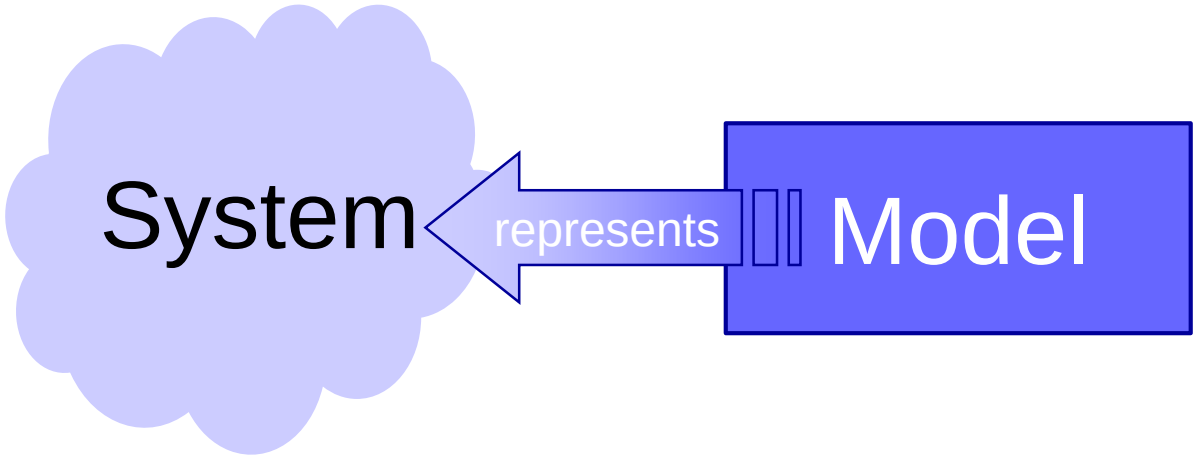
- The human mind continuously re-works reality by applying cognitive processes

- **Abstraction:** capability of finding the commonality in many different observations:
  - generalize specific features of real objects (generalization)
  - classify the objects into coherent clusters (classification)
  - aggregate objects into more complex ones (aggregation)

- **Model:** a simplified or partial representation of reality, defined in order to accomplish a task or to reach an agreement

Models

What is a model?



Mapping Feature	A model is based on an original (=system)
Reduction Feature	A model only reflects a (relevant) selection of the original‘s properties
Pragmatic Feature	A model needs to be usable in place of an original with respect to some purpose

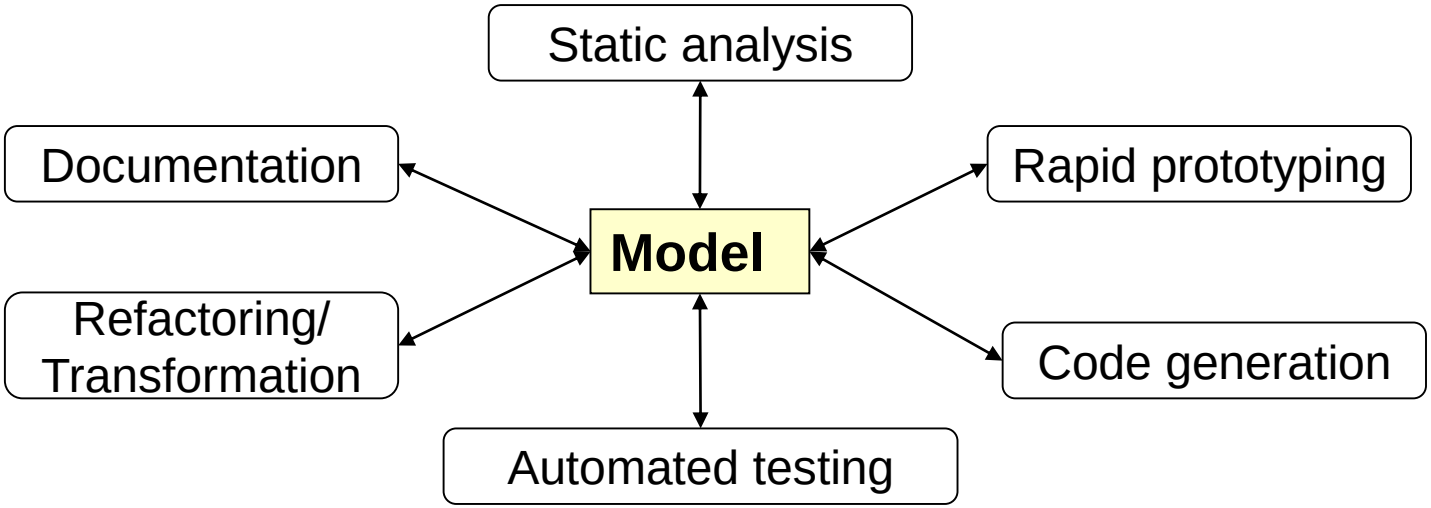
Purposes:

- descriptive purposes
- prescriptive purposes

Motivation

What is Model Engineering?

- Model as the **central artifact** of software development



- Related terms
  - Model Driven Engineering (MDE),
  - Model Driven [Software] Development (MDD/MDSD),
  - Model Driven Architecture (MDA)
  - Model Integrated Computing (MIC)

[Illustration by Bernhard Rumpe]

Motivation

Why Model Engineering?

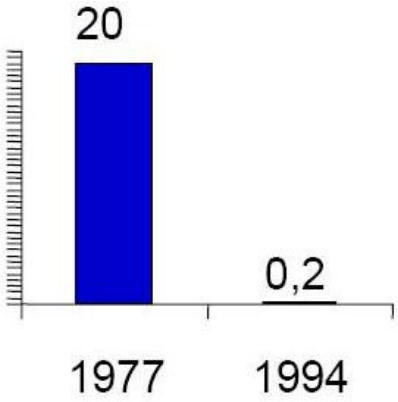
- Increasing **complexity** of software
  - Increasing basic requirements, e.g., adaptable GUIs, security, network capabilities, …
  - Complex infrastructures, e.g., operating system APIs, language libraries, application frameworks
- Software for **specific devices**
  - Web browser, mobile phone, navigation system, video player, etc.
- **Technological progress …**
  - Integration of different technologies and legacy systems, migration to new technologies
- … leads to **problems** with software development
  - Software finished too late
  - Wrong functionality realized
  - Software is poorly documented/commented
  - and can not be further developed, e.g., when the technical environment changes, business model/ requirements change, etc.

Motivation

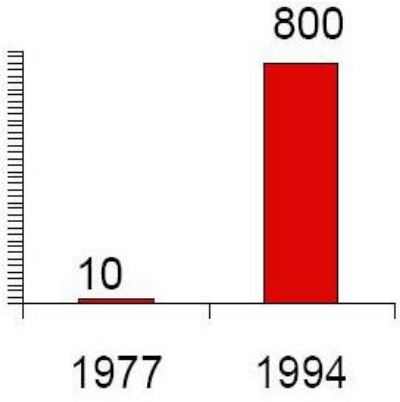
Why Model Engineering?

[Balzert, H.: Lehrbuch der Softwaretechnik: Software-Entwicklung, Spektrum, Akad. Verlag, 1996]

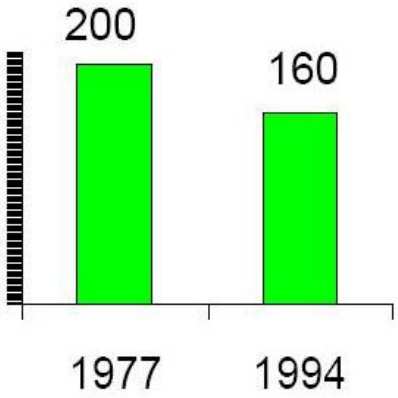
- **Quality problems** in software development



Number of bugs per 1000 LOC



Program size (1000 LOC)



Resulting absolute bug count

Real quality improvements are only possible if the increase in program complexity is **overcompensated** !

(Average values, from Balzert 96)

Motivation

Why Model Engineering?

- **Traditional** usage of models in software development
  - **Communication** with customers and users (requirement specification, prototypes)
  - Support for software design, capturing of the **intention**
  - **Task specification** for programming
  - **Code visualization** for understanding

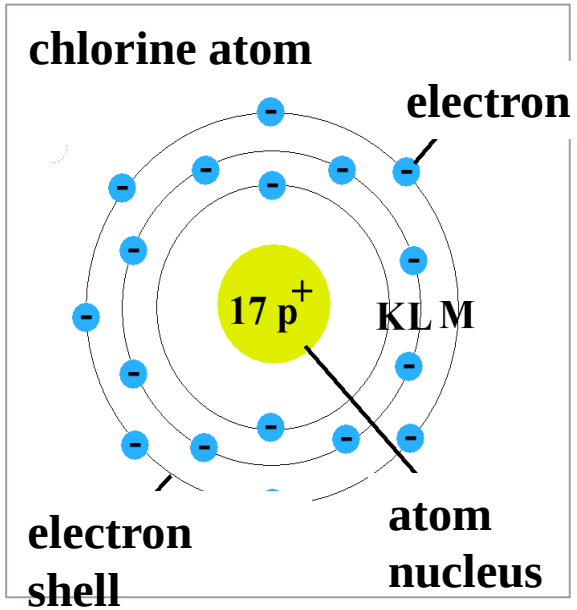
- What is the **difference** to Model Engineering?

Motivation

Usage of models

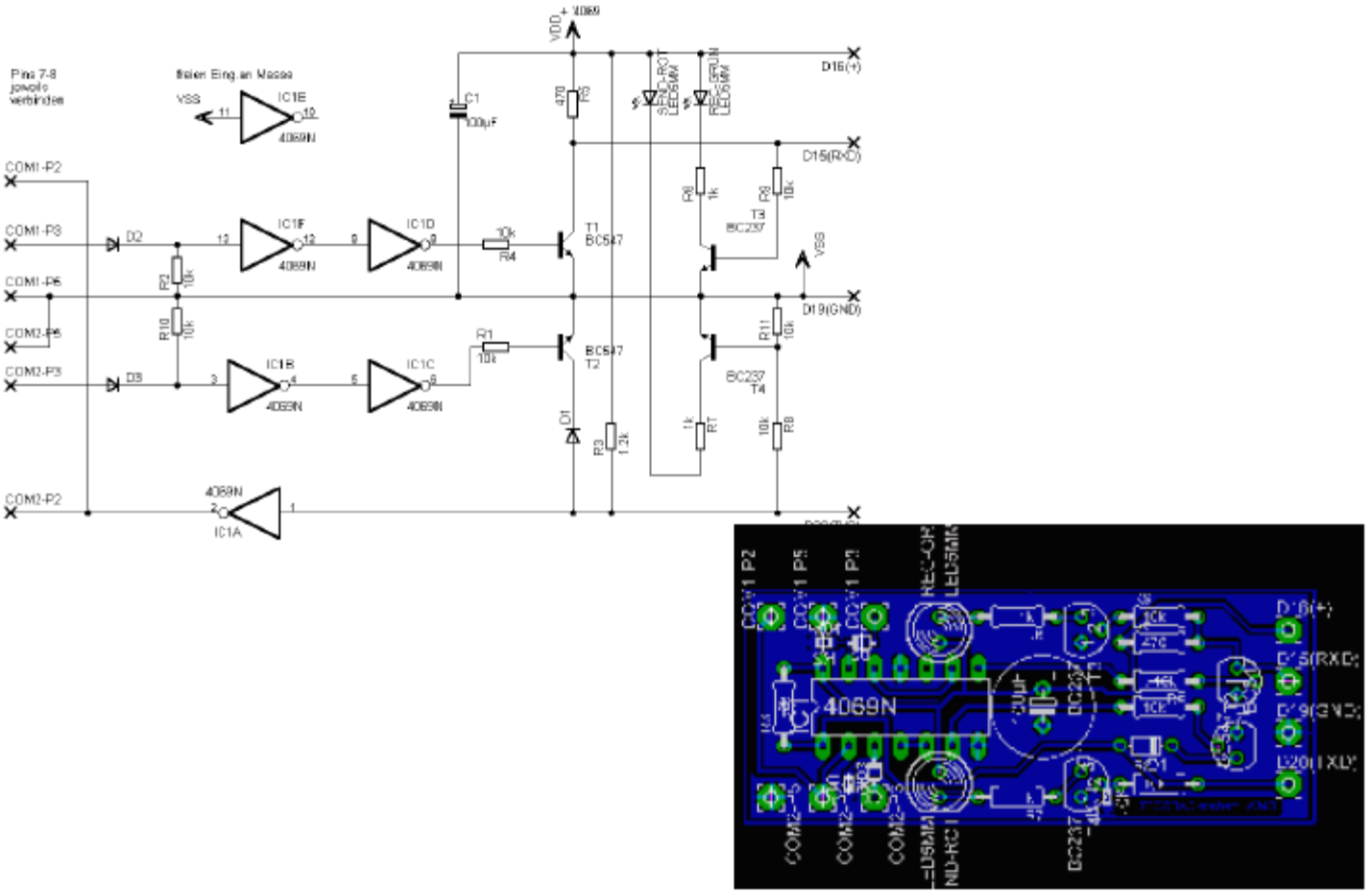
- Do not apply models as long as you have not checked the underlying **simplifications** and evaluated its **practicability**.

- Never mistake the **model** for the **reality**.
  - Attention: abstraction, abbreviation, approximation, visualization, …



Motivation

Constructive models (Example: Electrical Engineering)

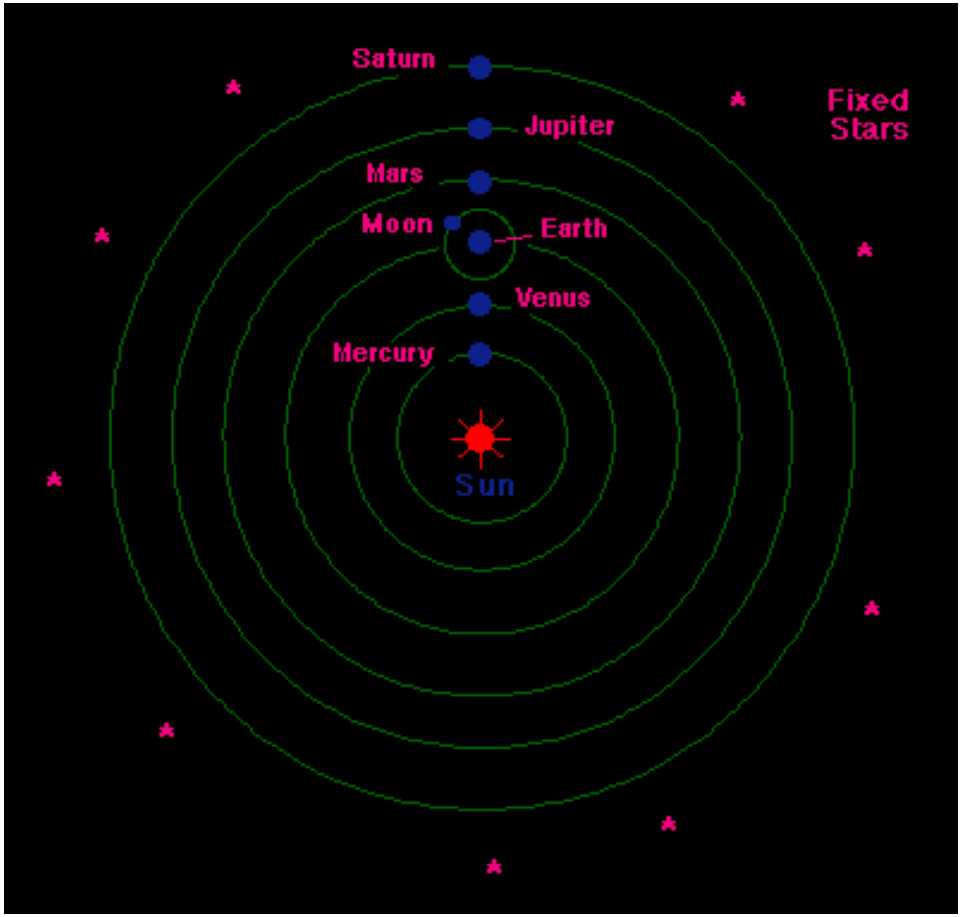


[Slide by Bernhard Rumpe]

Motivation

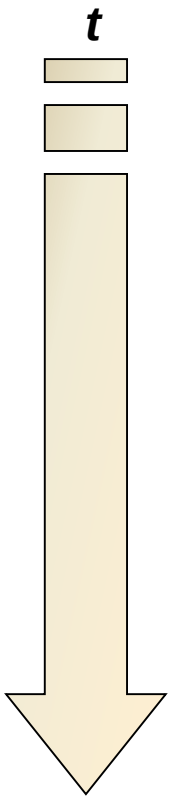
Declarative models (Example: Astronomy)

- Heliocentric model by Kopernikus



Motivation

Application area of modeling

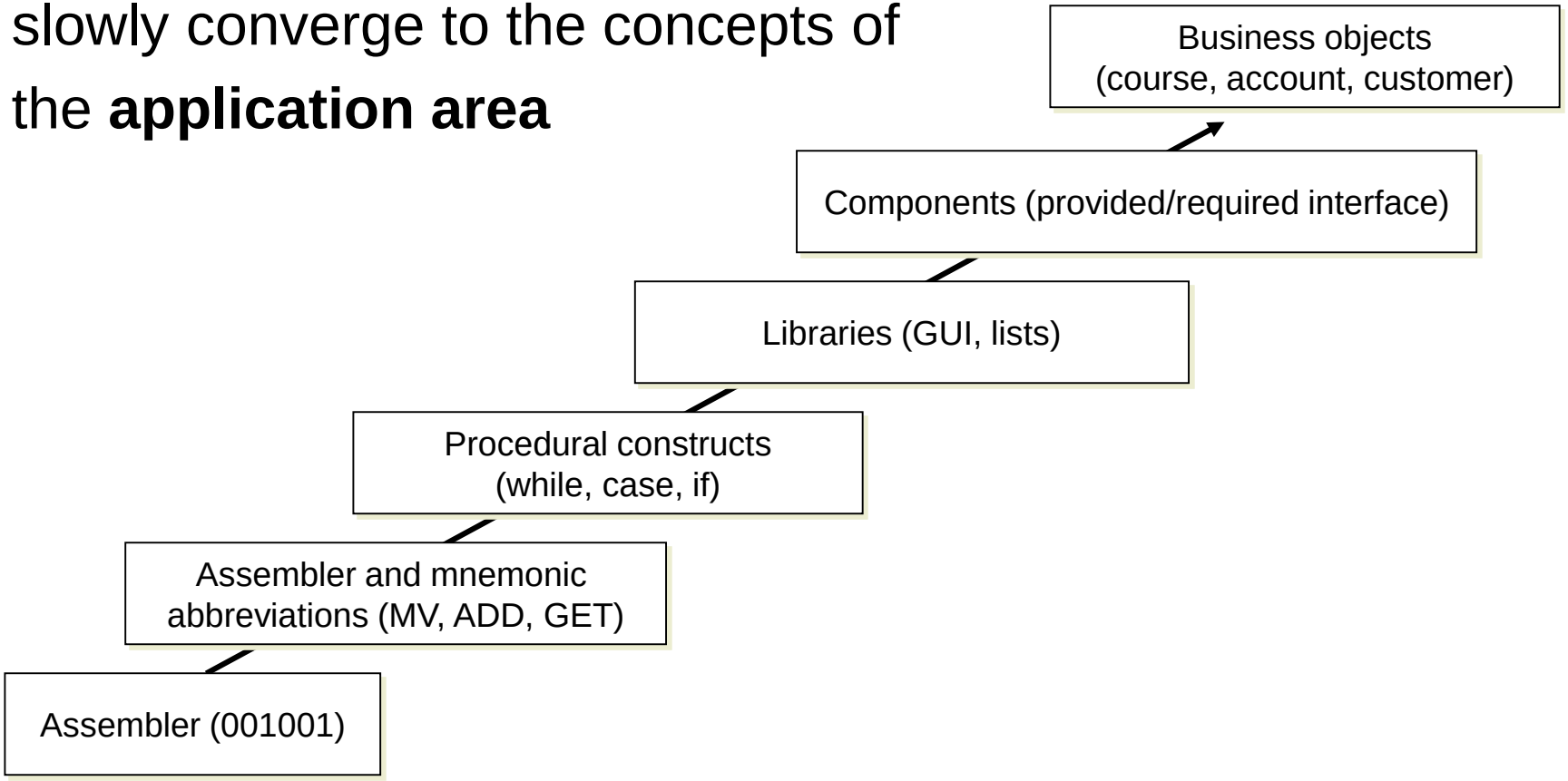


- ***Models as drafts***
  - Communication of ideas and alternatives
  - Objective: modeling per se
- ***Models as guidelines***
  - Design decisions are documented
  - Objective: instructions for implementation
- ***Models as programs***
  - Applications are generated automatically
  - Objective: models are source code and vice versa

Motivation

Increasing abstraction in software development

- The **used artifacts of software development** slowly converge to the concepts of the **application area**



[Illustration by Volker Gruhn]

Structure of the book

PART 1: MDSE Foundations

- **1 Introduction**
- 1.1 Purpose and Use of Models
- 1.2 Modeling for Software Development
- 1.3 How to Read this Book

- **2 MDSE Principles**
- 2.1 MDSE Basics
- 2.2 Lost in Acronyms: The MD* Jungle
- 2.3 Overview of the MDSE Methodology
- 2.3.1 Overall Vision
- 2.3.2 Target of MDSE: Domains, Platforms,Technical Spaces, and Scenarios
- 2.3.3 Modeling Languages
- 2.3.4 Metamodeling
- 2.3.5 Transformations
- 2.3.6 Model Classification
- 2.4 MDSE Adoption in Industry
- 2.5 Tool Support
- 2.5.1 Drawing Tools vs Modeling Tools
- 2.5.2 Model-Based vs Programming-Based MDSE Tools
- 2.5.3 Eclipse and EMF
- 2.6 Criticisms of MDSE

Structure of the book

PART 1: MDSE Foundations (continued)

- **3 MDSE Use Cases**
- 3.1 Automating Software Development
- 3.1.1 Code Generation
- 3.1.2 Model Interpretation
- 3.1.3 Combining Code Generation and Model Interpretation
- 3.2 System Interoperability
- 3.3 Reverse Engineering

- **4 Model-Driven Architecture (MDA)**
- 4.1 MDA Definitions and Assumptions
- 4.2 The Modeling Levels: CIM, PIM, PSM
- 4.3 Mappings
- 4.4 General Purpose and Domain-Specific Languages in MDA
- 4.5 Architecture-Driven Modernization

- **5 Integration of MDSE in your Development Process**
- 5.1 Introducing MDSE in your Software Development Process
- 5.1.1 Pains and Gains of Software Modeling
- 5.1.2 Socio-Technical Congruence of the Development Process
- 5.2 Traditional Development Processes and MDSE
- 5.3 Agile and MDSE
- 5.4 Domain-Driven Design and MDSE
- 5.5 Test-Driven Development and MDSE
- 5.5.1 Model-Driven Testing
- 5.5.2 Test-Driven Modeling

Structure of the book

PART 1: MDSE Foundations (continued)

- **6 Modeling Languages at a Glance**
- 6.1 Anatomy of Modeling Languages
- 6.2 General Purpose vs Domain-Specific Modeling Languages
- 6.3 General-Purpose Modeling: The Case of UML
- 6.4 UML Extensibility: The MiddleWay Between GPL and DSL
- 6.5 Overview on DSLs (Domain Specific Languages)
- 6.5.1 Principles of DSLs
- 6.5.2 Some Examples of DSLs
- 6.6 Defining Modeling Constraints (OCL)

PART 2: MDSE Technologies

- **7 Developing yourOwn Modeling Language**
- 7.1 Metamodel-Centric Language Design
- 7.1.1 Abstract Syntax
- 7.1.2 Concrete Syntax
- 7.1.3 Language Ingredients at a Glance
- 7.2 Example DSML: sWML
- 7.3 Abstract Syntax Development
- 7.3.1 Metamodel Development Process
- 7.3.2 Metamodeling in Eclipse
- 7.4 Concrete Syntax Development
- 7.4.1 Graphical Concrete Syntax (GCS)
- 7.4.2 Textual Concrete Syntax (TCS)

Structure of the book

PART 2: MDSE Technologies (continued)

- **8 Model-to-ModelTransformations**
- 8.1 Model Transformations and their Classification
- 8.2 Exogenous, Out-Place Transformations
- 8.3 Endogenous, In-Place Transformations
- 8.4 Mastering Model Transformations
- 8.4.1 Divide and Conquer: Model Transformation Chains
- 8.4.2 HOT: Everything is a Model, Even Transformations!
- 8.4.3 Beyond Batch: Incremental and Lazy Transformations
- 8.4.4 Bi-Directional Model Transformations

- **9 Model-to-TextTransformations**
- 9.1 Basics of Model-Driven Code Generation
- 9.2 Code Generation Through Programming Languages
- 9.3 Code Generation Through M2T Transformation Languages
- 9.3.1 Benefits of M2T Transformation Languages
- 9.3.2 Template-Based Transformation Languages: an Overview
- 9.3.3 Acceleo: An Implementation of the M2T Transformation Standard
- 9.4 Mastering Code Generation
- 9.5 Excursus: Code Generation Through M2M Transformations and TCS

Structure of the book

PART 2: MDSE Technologies (continued)

- **10 Managing Models**
- 10.1 Model Interchange
- 10.2 Model Persistence
- 10.3 Model Comparison
- 10.4 Model Versioning
- 10.5 Model Co-Evolution
- 10.6 Global Model Management
- 10.7 Model Quality
- 10.7.1 Verifying Models
- 10.7.2 Testing and Validating Models
- 10.8 Collaborative Modeling

- **11 Summary**

- Bibliography

- Authors' Biographies

MORGAN & CLAYPOOL PUBLISHERS

MODEL-DRIVEN SOFTWARE ENGINEERING IN PRACTICE

Marco Brambilla,
Jordi Cabot,
Manuel Wimmer.
Morgan & Claypool, USA, 2012.

www.mdse-book.com
www.morganclaypool.com
or buy it at: www.amazon.com