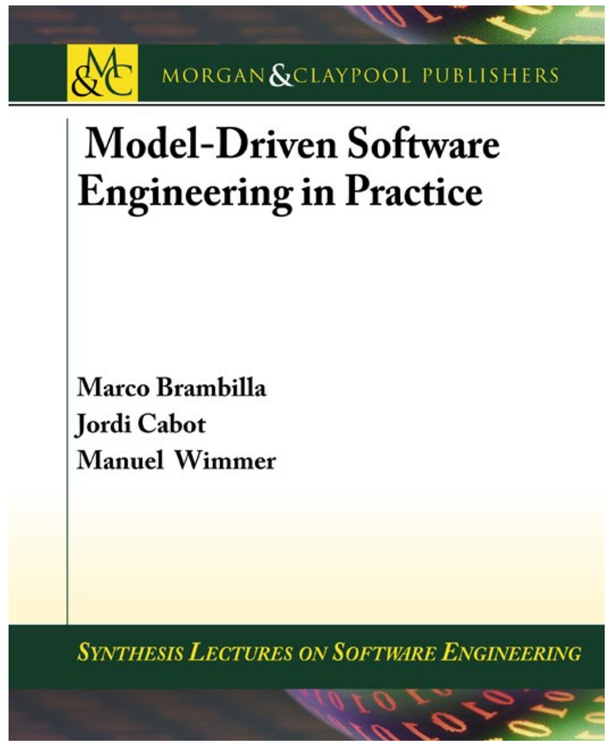
Modellierung und Analyse von Software Systemen

Bernhard Bauer

Software Methodologies
for Distributed Systems

Slides based on Model-Driven Software Engineering in Practice



Software Methodologies
for Distributed Systems

OVERVIEW - preliminary

Overview

1 Introduction

1.1 Human Cognitive Processes
1.2 Models
1.3 Model Engineering

2 MDSE Principles

2.1 MDSE Basics
2.2 The MD* Jungle of Acronyms
2.3 Modeling Languages and Metamodeling
2.4 Overview Considered Approaches
2.5 Tool Support
2.6 Criticisms of MDSE

3 MDSE Use Cases

3.1 MDSE applications
3.2 USE CASE1 – Model driven development
3.3 USE CASE2 – Systems interoperability
3.4 USE CASE3 – Model driven reverse engineering

Overview

4 Modeling Languages at a Glance

5.1 Anatomy of Modeling Languages

5.2 General Purpose vs. Domain-Specific Modeling Languages

5.3 Overview of UML Diagrams

5.4 UML Behavioural or Dynamic Diagrams

5.5 UML Extensibility: The Middle Way Between GPL and DSL

5.6 Domain Specific Languages

5.7 Defining Modeling Constraints (OCL)

Overview

5 Developing your Own Modeling Language

6.1 Metamodel-Centric Language Design
6.2 Programming Languages
6.3 Metamodel Development Process
6.4 MOF - Meta Object Facility
6.5 Example DSML: sWML
6.6 EMF and Ecore
6.7 Abstract Syntax Development
6.8 Graphical Concrete Syntax Development
6.9 Textual Concrete Syntax Development

6 Model-to-Model Transformations

7.1 Model Transformations and their Classification
7.2 Exogenous, Out-Place Transformations
7.3 Endogenous, In-Place Transformations
7.4 ATL Overview
7.5 Mastering Model Transformations and their Rules

Overview

7 Model-to-Text Transformations

- 8.1 Basics of Model-Driven Code Generation
- 8.2 Code Generation Through Programming Languages
- 8.3 Code Generation Through M2T Transformation Languages
- 8.4 Mastering Code Generation
- 8.5 Excursus: Code Generation Through M2M Transformations and TCS

8 Analyzing Models (not included in the book)

- 9.1 Model Analysis
- 9.2 Tooling
- 9.3 Use Cases

Overview

9 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
- 10.8 Collaborative Modeling

10 Integration of MDSE in your Development Process

- 11.1 Introducing MDSE in your Software Development Process
- 11.2 Classical Development Processes and MDSE
- 11.3 Agile and MDSE
- 11.4 Domain-Driven Design and MDSE
- 11.5 Test-Driven Development and MDSE

CHAPTER 1 - Introduction

Overview

1 Introduction

- » Human Cognitive Processes
- » Models
- » Model Engineering

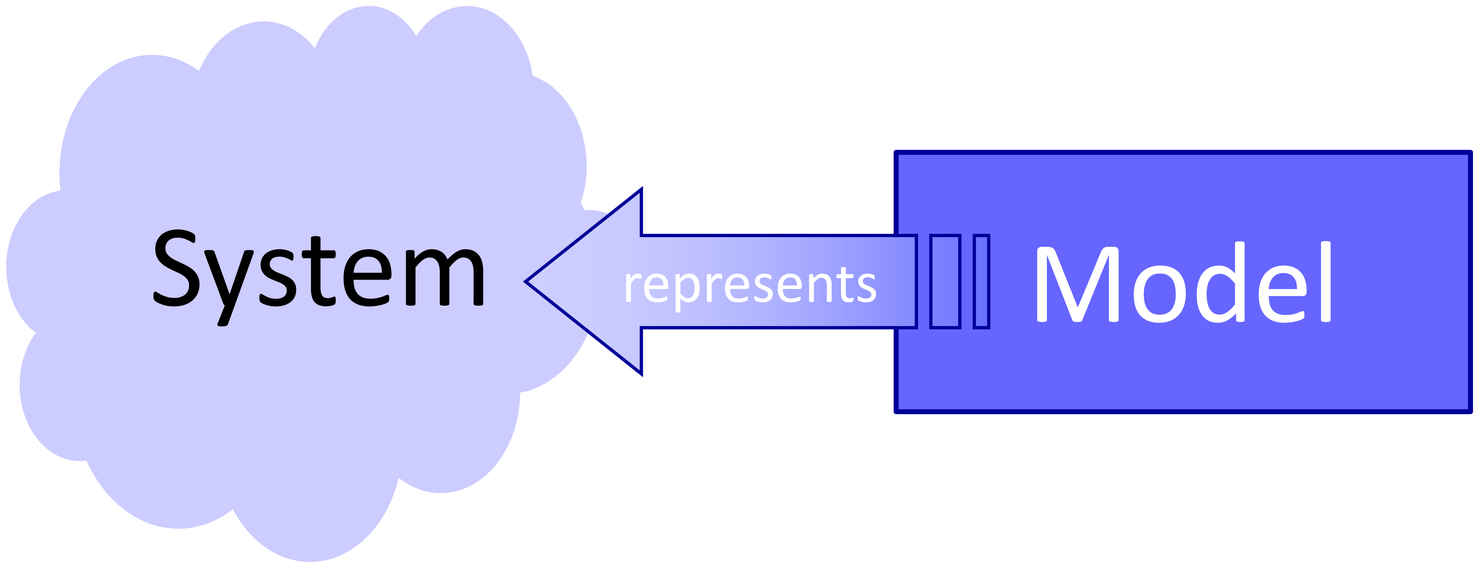
Abstraction and human mind

- 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

Models
What is a model?

"Modeling, in the broadest sense, is the cost-effective ***use of something in place of something else for some cognitive purpose****. It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose. A* ***model represents reality for the given purpose****;* ***the model is an abstraction of reality*** *in the sense that it cannot represent all aspects of reality. This* ***allows us to deal with the world in a simplified manner****, avoiding the complexity, danger and irreversibility of reality."*

"The Nature of Modeling."
Jeff Rothenberg
in Artificial Intelligence, Simulation, and Modeling,
L.E. William, K.A. Loparo, N.R. Nelson, eds.
New York, John Wiley and Sons, Inc., 1989, pp. 75-92

Challenges of today's software engineering

- Like businesses, IT systems are expected to be more adaptable and flexible
- Strong focus on IT departments to deliver business value
  - » Software must be business relevant
  - » Miscommunication between business and IT people can lead to projects that, successful from an IT-delivery viewpoint, are deemed business failures (e.g. wrong functionality)
- Cost and time control
  - » IT departments operate under strong budget constraints and are expected to demonstrate value for money
- Increasing complexity
  - » Software systems continue to increase in scale and complexity
  - » Techniques that work well for small-scale development do not necessarily work enterprise-wide
- Skills availability
  - » Specialists' knowledge is required to deliver software (complex IT platforms, e.g.)
  - » Projects often depend on key individuals and suffer significantly if those individuals leave a project or organization
- Changing middleware environment
  - » Applications are deployed to a huge variety of middleware platforms
  - » The rate of change in platform technology is showing no sign of slowing up
  - » Businesses want to take advantage of advances in middleware but do not want to repeatedly rewrite their applications

Model engineering

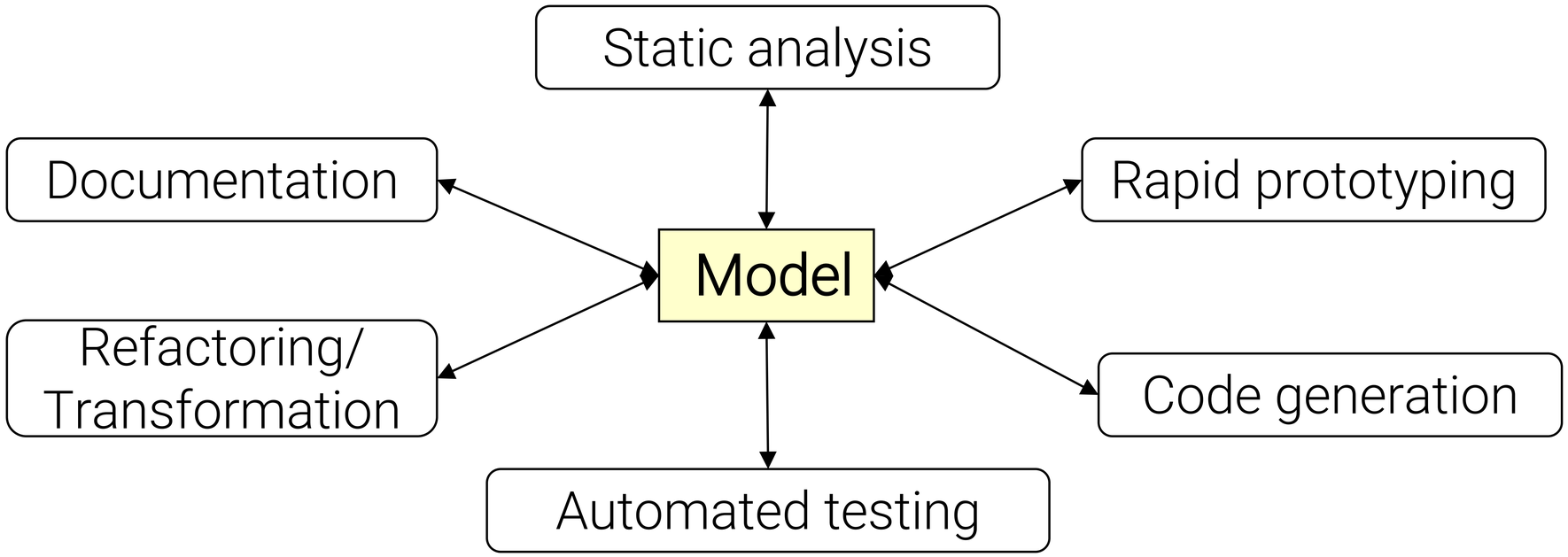
- Traditional role of models in software development
  - » Used for communication purposes with the customer and within the development team (requirements specification, prototypes, etc.)
  - » Support for software design, capturing of the intention
  - » Specification for the programmer
  - » Code visualization, for example in TogetherJ

- What is the difference to model engineering?

  ➔ ***Models are more than code or documentation***

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)

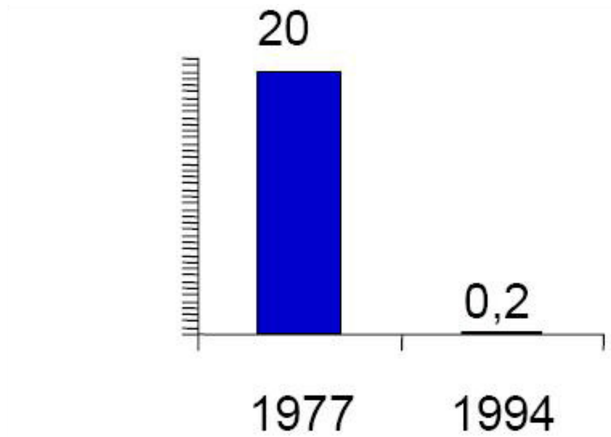
[Illustration by Bernhard Rumpe]

Motivation
Why Model Engineering?

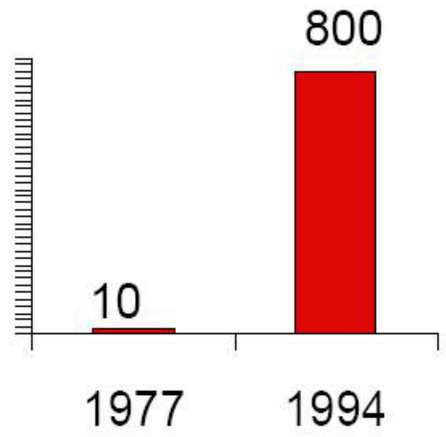
- Increasing complexity of software
  - » Increasing basic requirements, e.g., adaptable GUIs, security, network capabilities, …
  - » Complex infrastructures, e.g., operating system APIs, language libraries, application frameworks
- Software for specific devices
  - » Web browser, mobile phone, navigation 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
Quality problems in software development

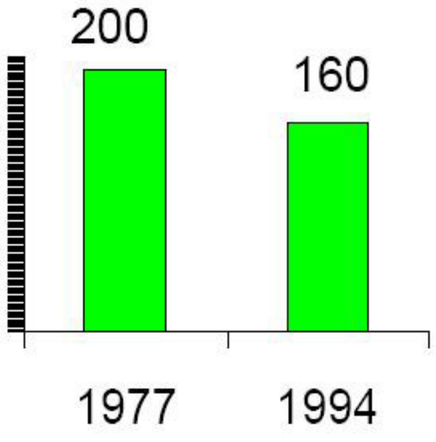
- Why Model Engineering?



Number of bugs per 1000 LOC



Program size (1000 LOC)



Resulting absolute bug count

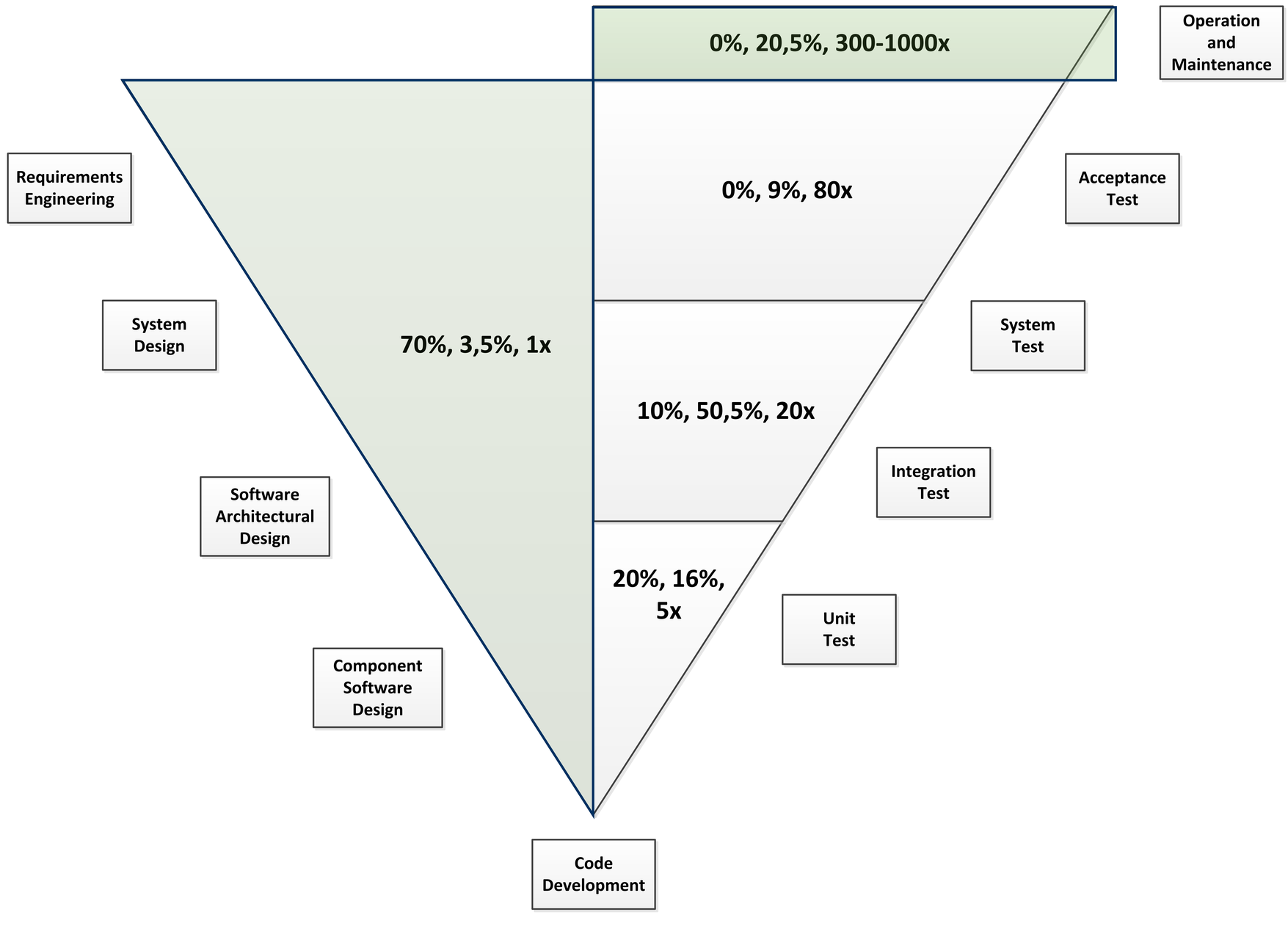
Real quality improvements are only possible if the increase in program complexity is **overcompensated** !

(Average values, from Balzert 96)

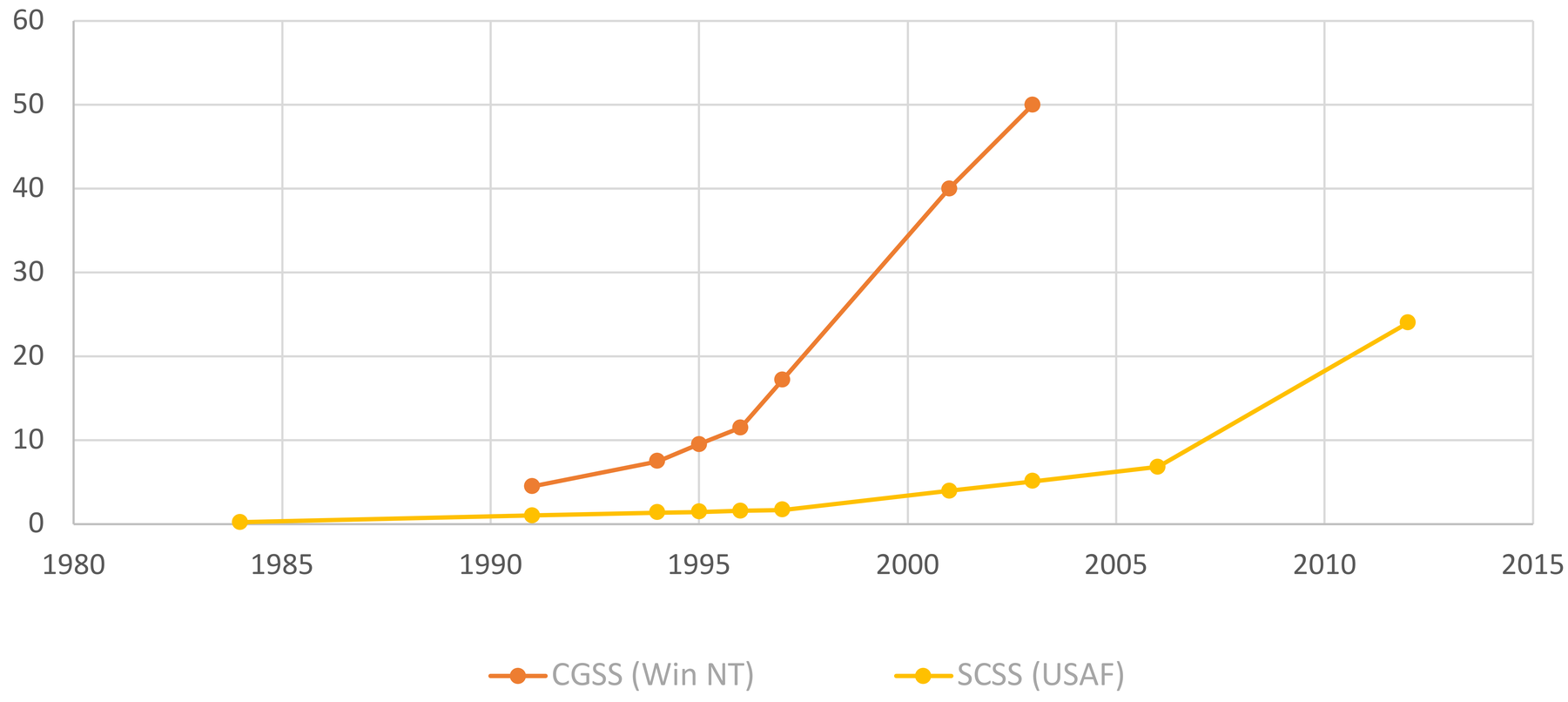
[Slide by Bernhard Rumpe]

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

Motivation
Quality problems in software development

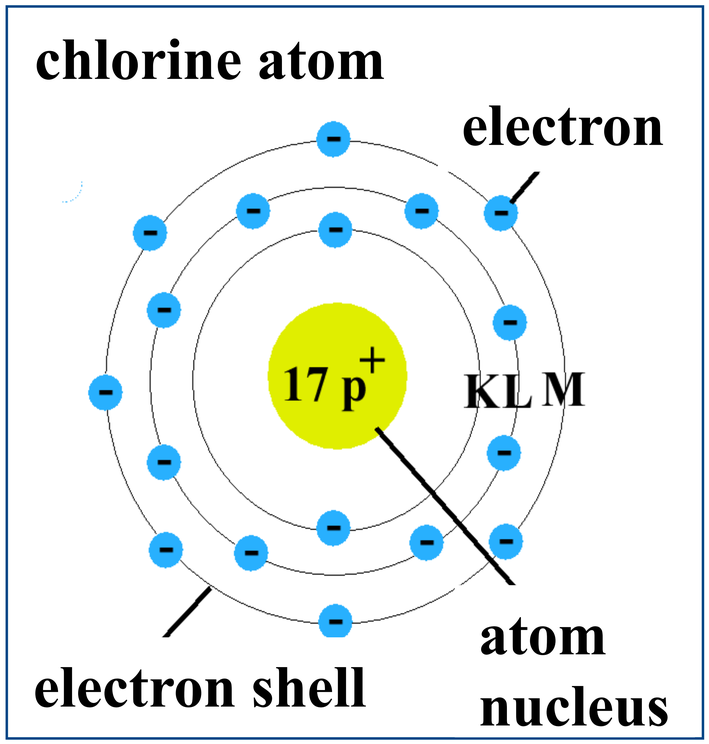


Motivation
Quality problems in software development

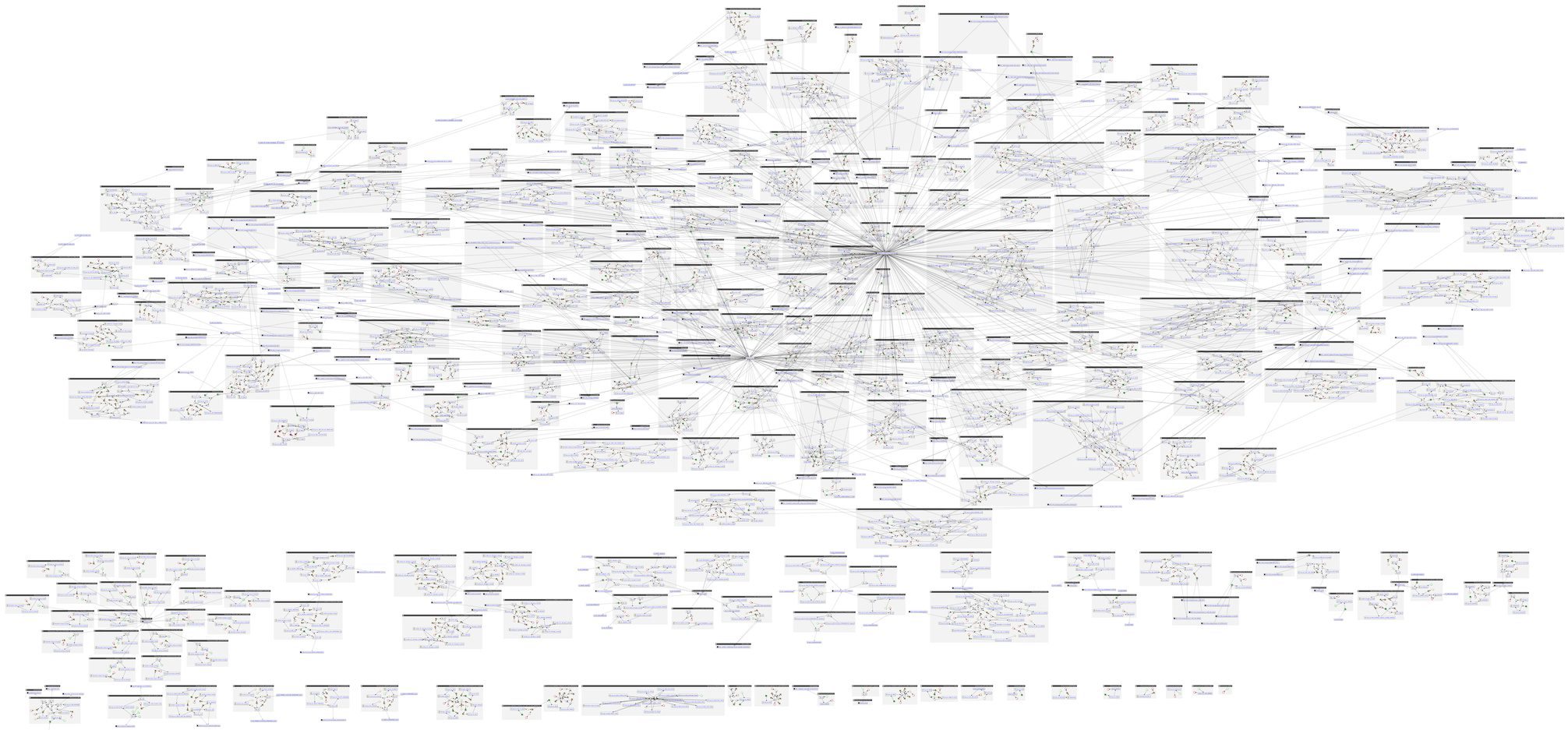


Motivation
Usage of models

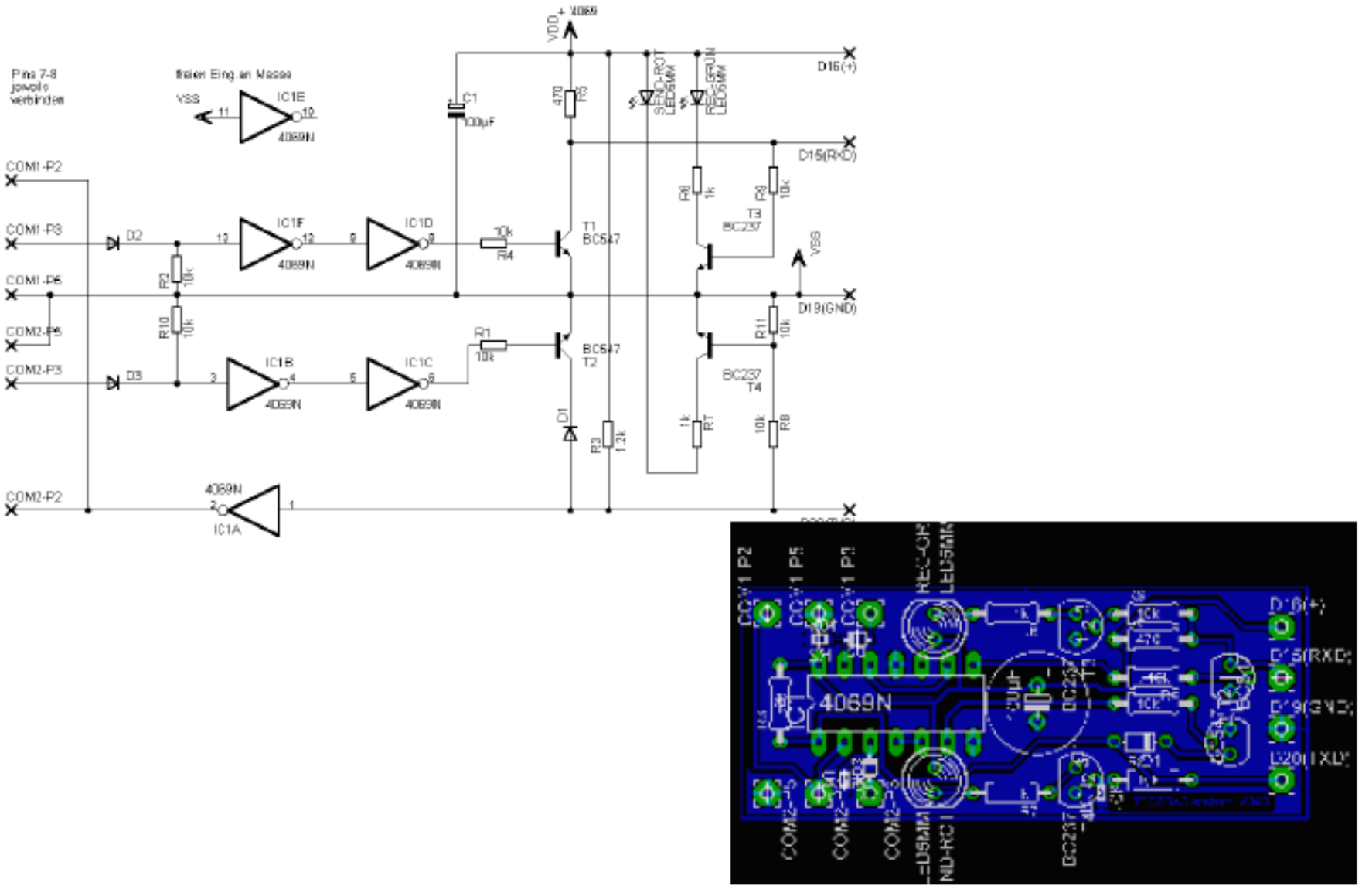
- Do not apply models as long as you have not checked the underlying simplifications and evaluated its practicability.
- Never mistake the model for the reality.
  - » Attention: abstraction, abbreviation, approximation, visualization, …



Live SW-Architectures



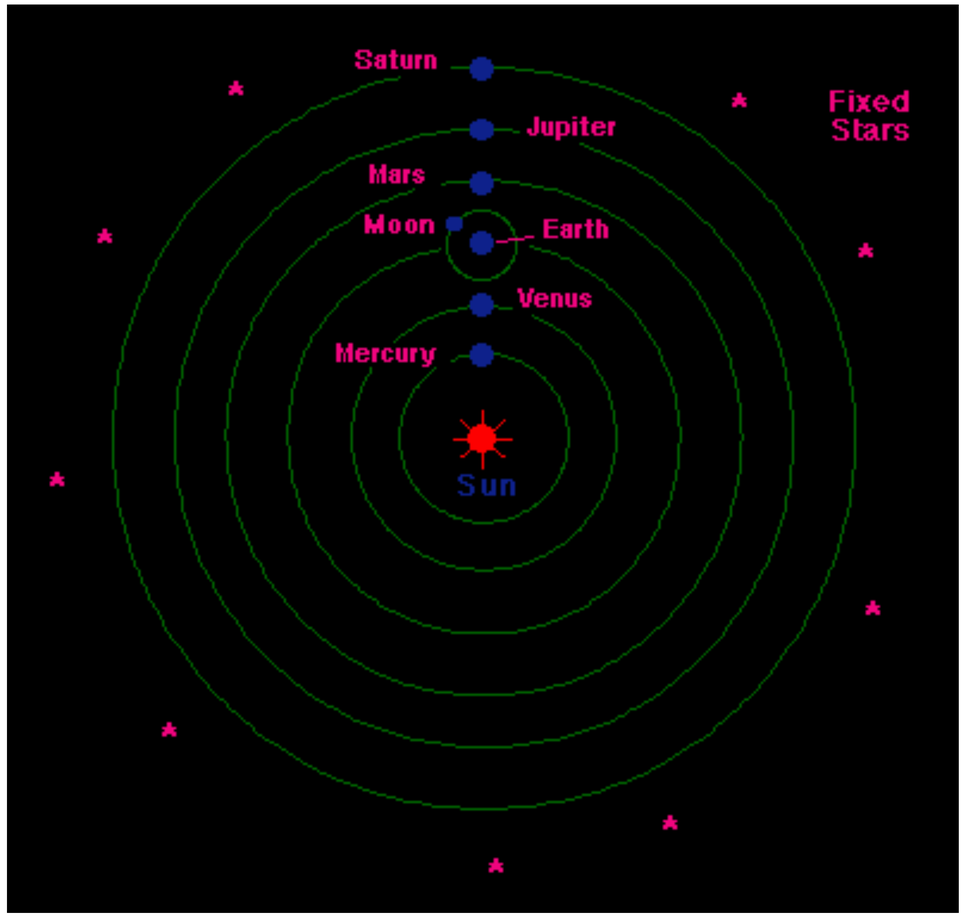
Motivation - Constructive models (Example: Electrical Engineering)



[Slide by Bernhard Rumpe]

Motivation - Declarative models (Example: Astronomy)

- Heliocentric model by Kopernikus

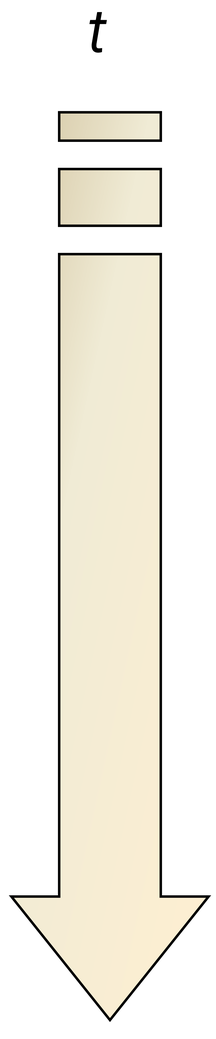


Motivation
Application area of modeling

- Models as drafts
  - » Communication of ideas and alternatives
  - » Objective: modeling per se

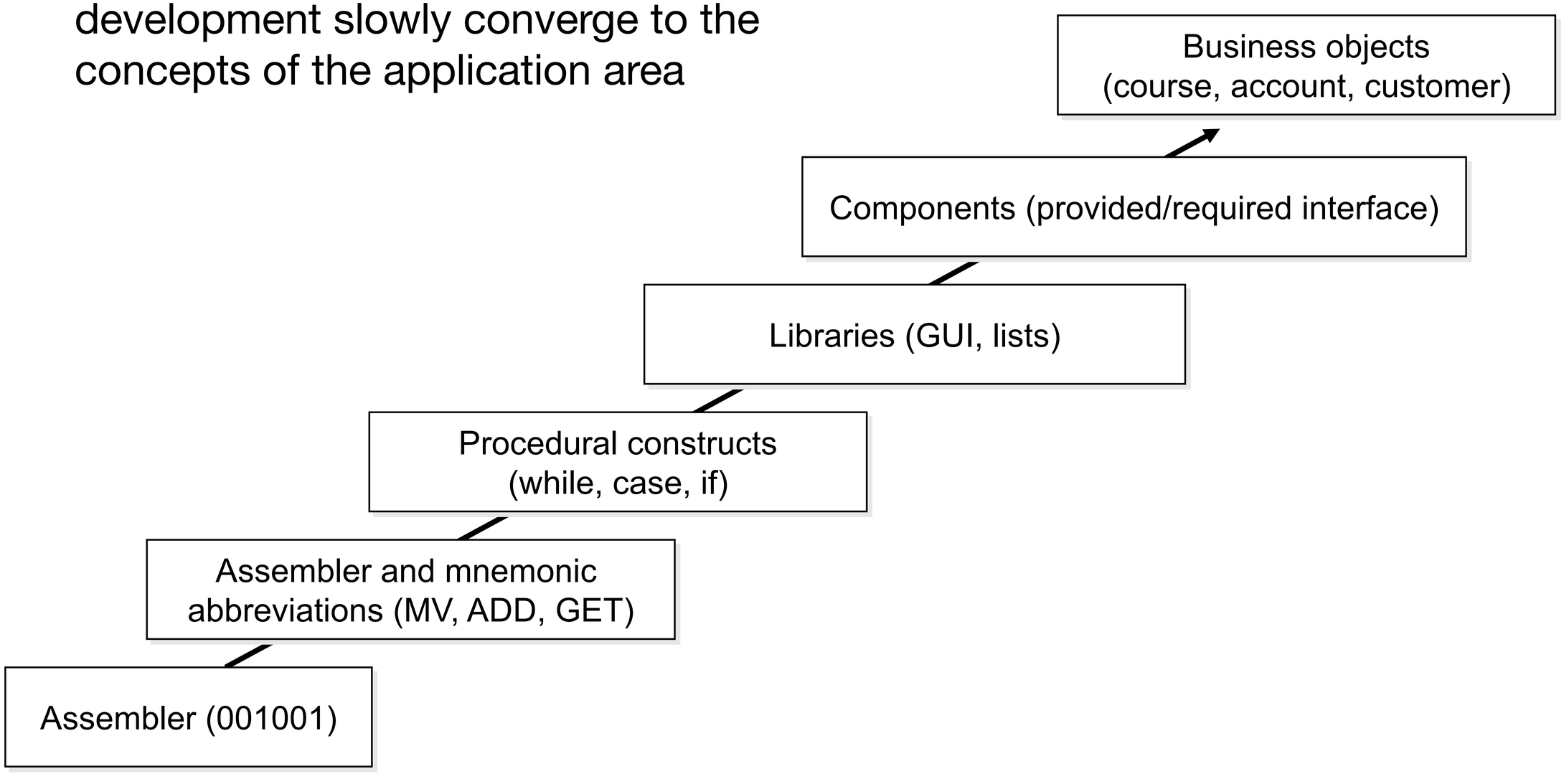
- Models as guidelines
  - » Design decisions are documented
  - » Objective: instructions for implementation

- 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]