Model-Based Software Engineering

Lecture 01 – Introduction

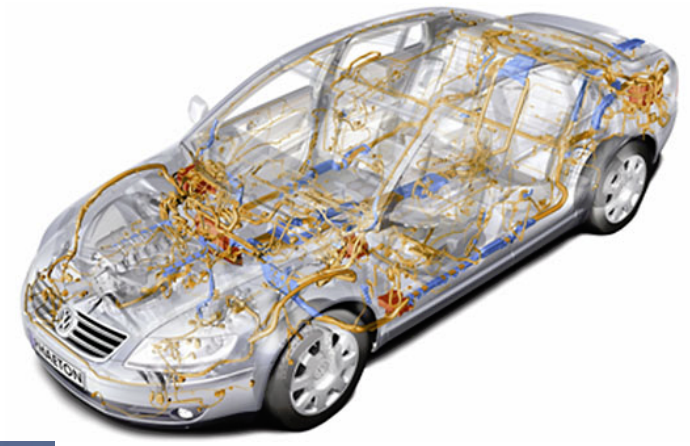
Prof. Dr. Joel Greenyer

April 5, 2015

Discussion

- What do **you** associate with the term "Model-Based Software Engineering?"

Critical Software-Intensive Systems



SOFTWARE ENGINEERING

Complexity

- Software systems become **more and more complex**
  - Systems and applications become increasingly interconnected
  - Processes and data become more complex

Complexity

- Software systems become **more and more complex**
  - Systems and applications become increasingly interconnected
  - Processes and data become more complex
- The ongoing “Software Crisis”
  - “The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.”

Complexity

- Software systems become **more and more complex**
  - Systems and applications become increasingly interconnected
  - Processes and data become more complex
- The ongoing “Software Crisis”
  - “The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.”
    - Edsger Dijkstra, The Humble Programmer

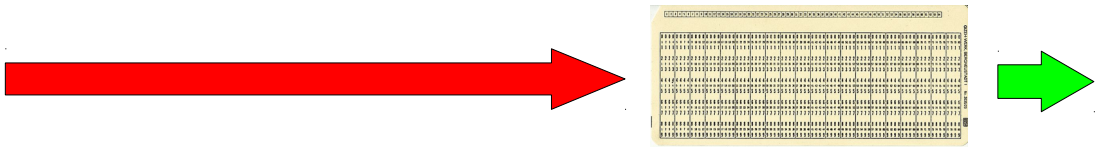
SOFTWARE ENGINEERING

Complexity

- Software systems become **more and more complex**
  - Systems and applications become increasingly interconnected
  - Processes and data become more complex
- The ongoing “Software Crisis”
  - “The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.”
    - Edsger Dijkstra, The Humble Programmer
      Communications of the ACM, **1972**
      (http://www.cs.utexas.edu/users/EWD/ewd03xx/EWD340.PDF)

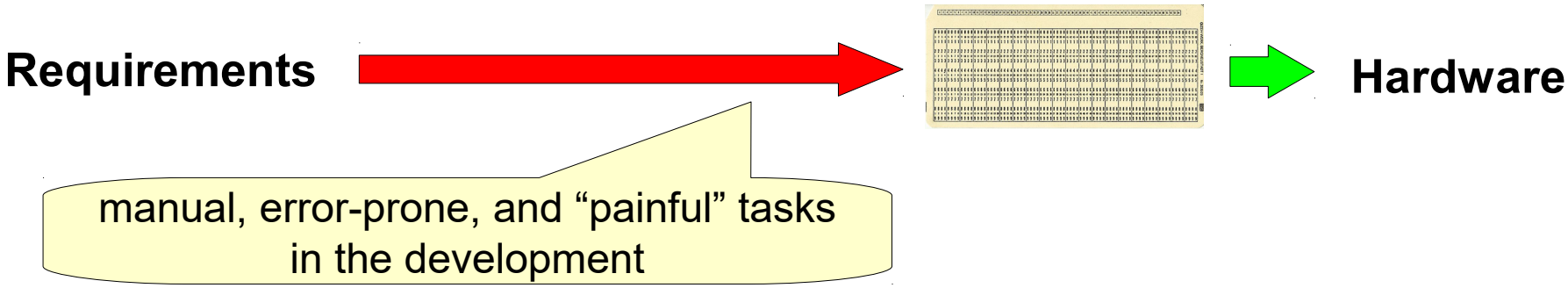
Historical Perspective

- in the early days of computers: punch cards

Requirements  **Hardware**

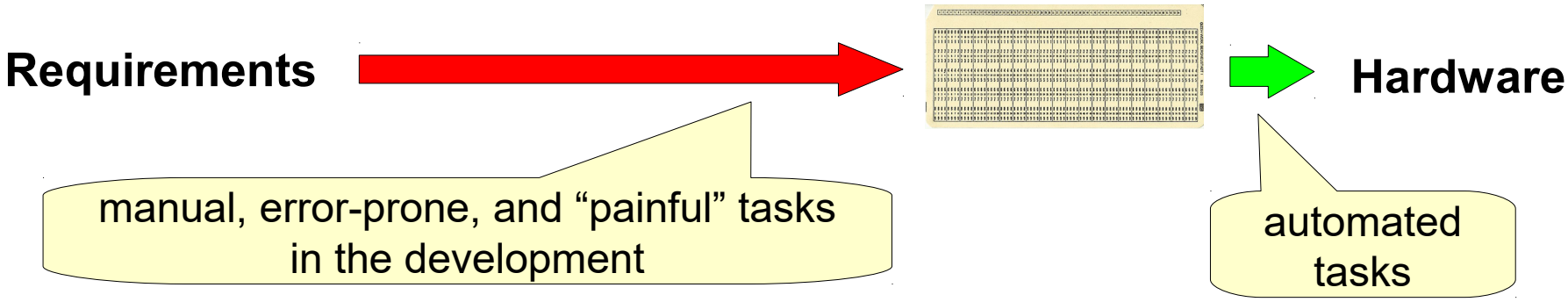
Historical Perspective

- in the early days of computers: punch cards



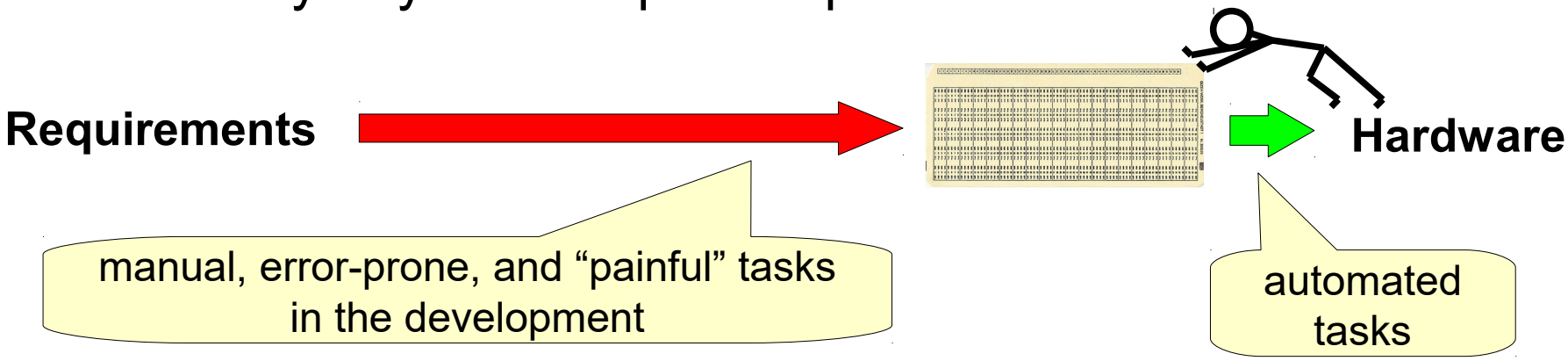
Historical Perspective

- in the early days of computers: punch cards



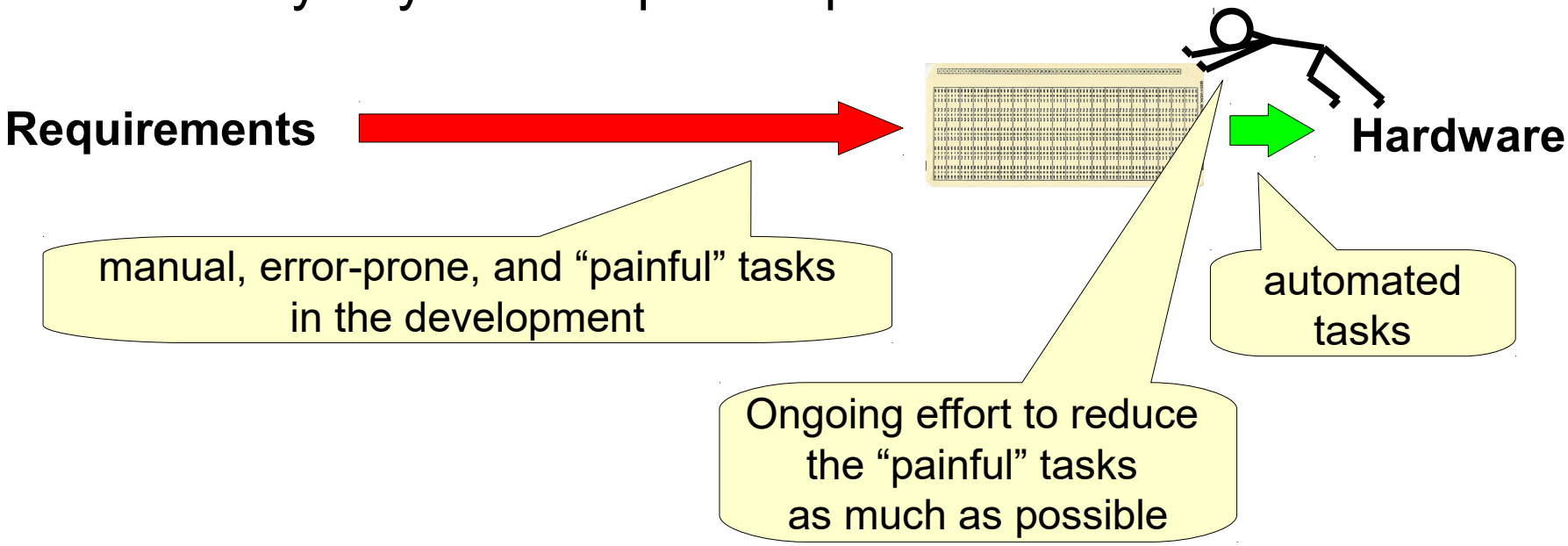
Historical Perspective

- in the early days of computers: punch cards



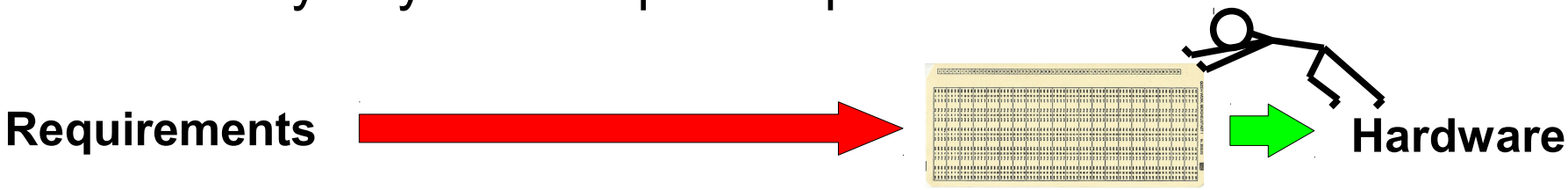
Historical Perspective

- in the early days of computers: punch cards

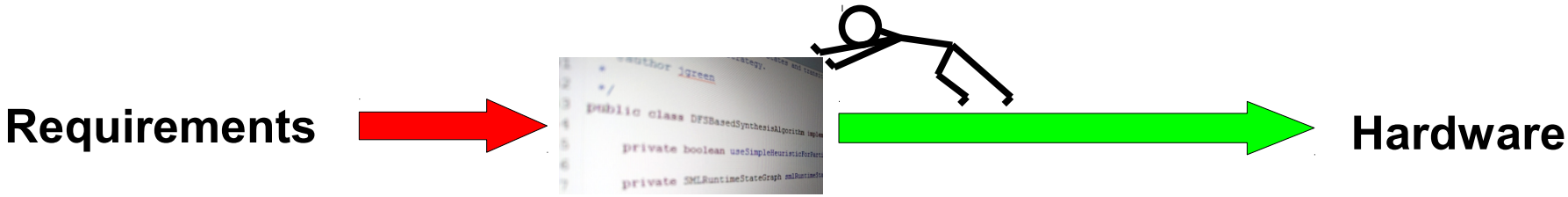


Historical Perspective

- in the early days of computers: punch cards

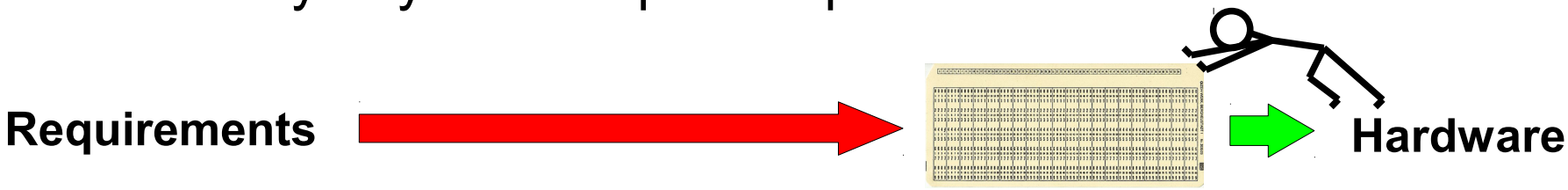


- Today:

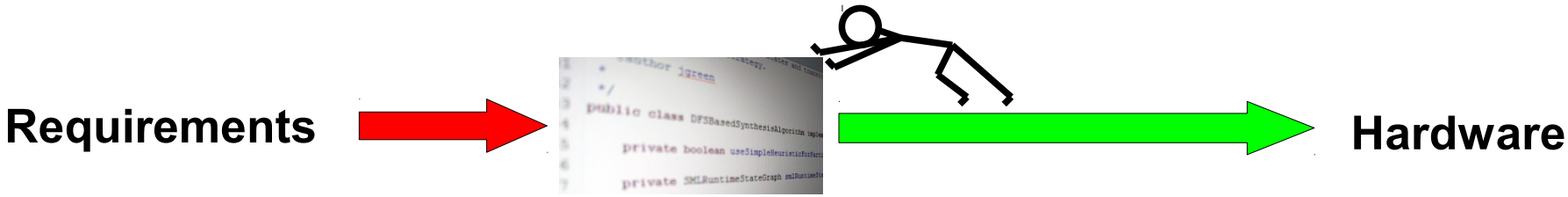


Historical Perspective

- in the early days of computers: punch cards



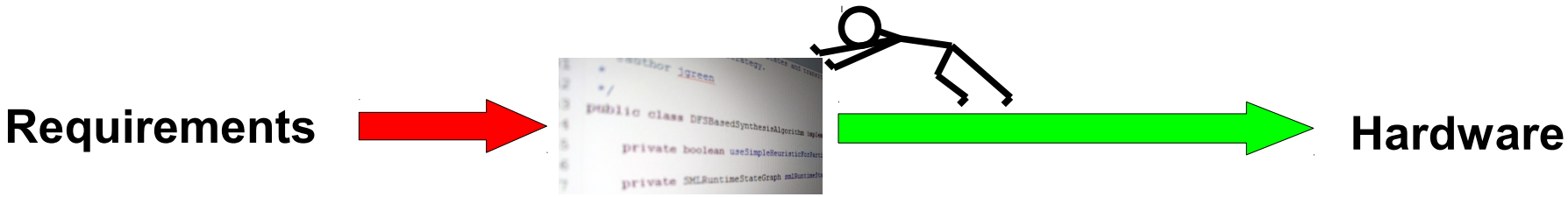
- Today:



Goal:
1. Describe software on **adequate abstraction** level
2. **Automate** (code) generation as much as possible
3. **Automate analysis** as much as possible

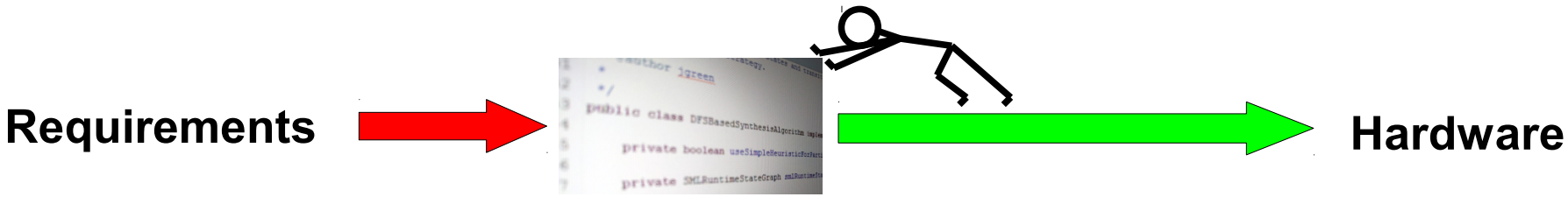
Model-Based Software Engineering

- Model-based software engineering is about continuing the trend towards more adequate ways of describing software, and automating as much of the development as possible



Model-Based Software Engineering

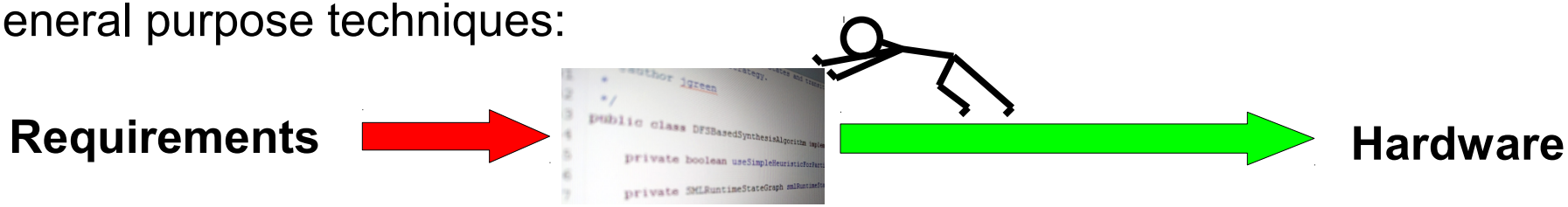
- Model-based software engineering is about continuing the trend towards more adequate ways of describing software, and automating as much of the development as possible
  - adequate: in many cases tailored for a **specific domain**



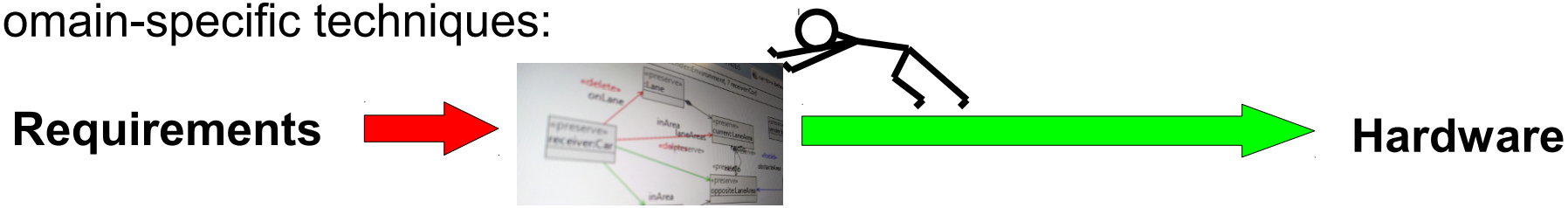
Model-Based Software Engineering

- Model-based software engineering is about continuing the trend towards more adequate ways of describing software, and automating as much of the development as possible
  - adequate: in many cases tailored for a **specific domain**

General purpose techniques:



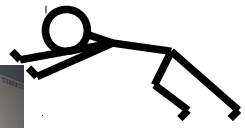
Domain-specific techniques:



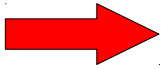
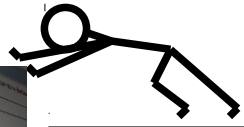
Model-Based Software Engineering

- Model-based software engineering is about continuing the trend towards more adequate ways of describing software, and automating as much of the development as possible
  - adequate: in many cases tailored for a **specific domain**

General purpose techniques:

Requirements     **Hardware**

Domain-specific techniques:

Requirements     **Hardware**

"The last push" is only possible when languages and tools are tailored for a specific domain.

Agenda

1. Motivation ✓
2. What you will learn in this lecture
   - Learning Objectives
   - Topics
   - Literature
3. Organizational Issues
4. Introduction:
   - What is a model?
   - What is model-based software engineering (MBSE)?
   - What are reasons for doing MBSE?

SOFTWARE ENGINEERING

1.2 What you will learn in this lecture

Leibniz Universität Hannover

SOFTWARE ENGINEERING

Learning Objective

- The students know **concepts, methods, and tools** for the **development of modeling languages and tools**.

Learning Objective

- The students know **concepts, methods, and tools** for the **development of modeling languages and tools**.
- They have learned about meta-modeling and the definition of a graphical or textual language syntax and are able to design a DSL (Domain Specific Language) themselves for a specific domain.

Learning Objective

- The students know **concepts, methods, and tools** for the **development of modeling languages and tools**.
- They have learned about meta-modeling and the definition of a graphical or textual language syntax and are able to design a DSL (Domain Specific Language) themselves for a specific domain.
- Furthermore they can develop interpreters, model transformations, and code generators for modeling languages to execute, simulate or otherwise analyze the models.

Learning Objective

- The students know **concepts, methods, and tools** for the **development of modeling languages and tools**.
- They have learned about meta-modeling and the definition of a graphical or textual language syntax and are able to design a DSL (Domain Specific Language) themselves for a specific domain.
- Furthermore they can develop interpreters, model transformations, and code generators for modeling languages to execute, simulate or otherwise analyze the models.
- Students have deepened their understanding of modeling and abstraction and of modern software engineering technologies.

SOFTWARE ENGINEERING

Topics

- Introduction and Overview

SOFTWARE ENGINEERING

Topics

- Introduction and Overview
- Meta-Modeling, MOF, Eclipse Modeling Framework and other technologies

SOFTWARE ENGINEERING

Topics

- Introduction and Overview
- Meta-Modeling, MOF, Eclipse Modeling Framework and other technologies
- Object Constraint Language (OCL)

SOFTWARE ENGINEERING

Topics

- Introduction and Overview
- Meta-Modeling, MOF, Eclipse Modeling Framework and other technologies
- Object Constraint Language (OCL)
- UML: Profiles, Understanding UML as an instance of MOF

SOFTWARE ENGINEERING

Topics

- Introduction and Overview
- Meta-Modeling, MOF, Eclipse Modeling Framework and other technologies
- Object Constraint Language (OCL)
- UML: Profiles, Understanding UML as an instance of MOF
- Domain Specific Languages, defining a textual or graphical syntax

SOFTWARE ENGINEERING

Topics

- Introduction and Overview
- Meta-Modeling, MOF, Eclipse Modeling Framework and other technologies
- Object Constraint Language (OCL)
- UML: Profiles, Understanding UML as an instance of MOF
- Domain Specific Languages, defining a textual or graphical syntax
- Model-to-text transformations and code generation

SOFTWARE ENGINEERING

Topics

- Introduction and Overview
- Meta-Modeling, MOF, Eclipse Modeling Framework and other technologies
- Object Constraint Language (OCL)
- UML: Profiles, Understanding UML as an instance of MOF
- Domain Specific Languages, defining a textual or graphical syntax
- Model-to-text transformations and code generation
- Interpreters

SOFTWARE ENGINEERING

Topics

- Introduction and Overview
- Meta-Modeling, MOF, Eclipse Modeling Framework and other technologies
- Object Constraint Language (OCL)
- UML: Profiles, Understanding UML as an instance of MOF
- Domain Specific Languages, defining a textual or graphical syntax
- Model-to-text transformations and code generation
- Interpreters
- Graph transformations

SOFTWARE ENGINEERING

Topics

- Introduction and Overview
- Meta-Modeling, MOF, Eclipse Modeling Framework and other technologies
- Object Constraint Language (OCL)
- UML: Profiles, Understanding UML as an instance of MOF
- Domain Specific Languages, defining a textual or graphical syntax
- Model-to-text transformations and code generation
- Interpreters
- Graph transformations
- Model-to-model transformations: Operational (ATL, QVT-O, others) and declarative (Triple Graph Grammars, QVT-R)

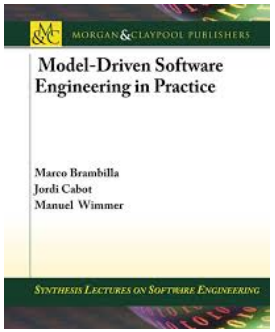
SOFTWARE ENGINEERING

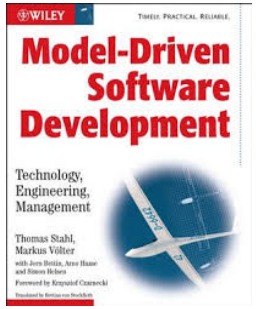
Topics

- Introduction and Overview
- Meta-Modeling, MOF, Eclipse Modeling Framework and other technologies
- Object Constraint Language (OCL)
- UML: Profiles, Understanding UML as an instance of MOF
- Domain Specific Languages, defining a textual or graphical syntax
- Model-to-text transformations and code generation
- Interpreters
- Graph transformations
- Model-to-model transformations: Operational (ATL, QVT-O, others) and declarative (Triple Graph Grammars, QVT-R)
- Further Topics (TBD)

Literature

- Brambilla, Cabot, Wimmer: Model-Driven Software Engineering in Practice, Morgan & Claypool Publishers, 2012.
- Stahl, Völter, Efftinge, Haase: Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management, dpunkt, 2. Auflage, 2007.
- Völter: DSL Engineering: Designing, Implementing and Using Domain-Specific Languages, CreateSpace Independent Publishing Platform, 2013
  (http://voelter.de/data/books/markusvoelter-dslengineering-1.0.pdf)
- Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006.
  (http://www.voelter.de/data/books/mdsd-en.pdf)





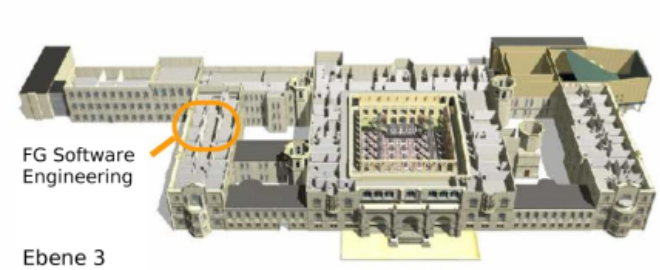
SOFTWARE ENGINEERING

1.3 Organizational Issues

Leibniz Universität Hannover

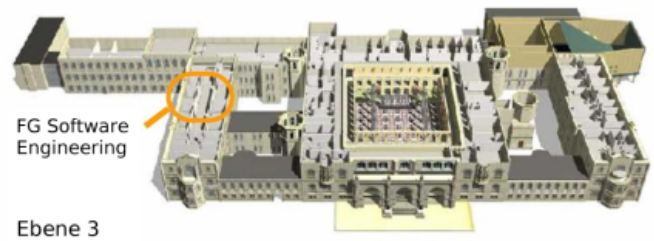
Who am I and why am I here?

- Prof. Dr. Joel Greenyer
  - Juniorprofessor at the Leibniz Universität Hannover



Who am I and why am I here?

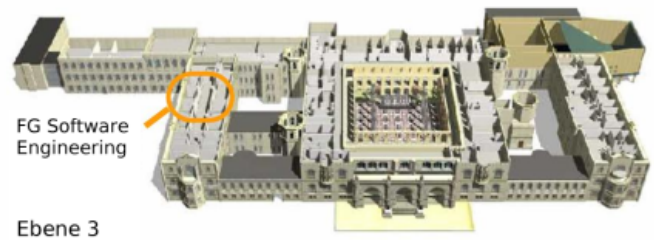
- Prof. Dr. Joel Greenyer
  - Juniorprofessor at the Leibniz Universität Hannover
  - 10 years of research in the field of model-based software engineering
    - Graph- and model transformations
    - Behavior modeling
    - Formal analysis and synthesis



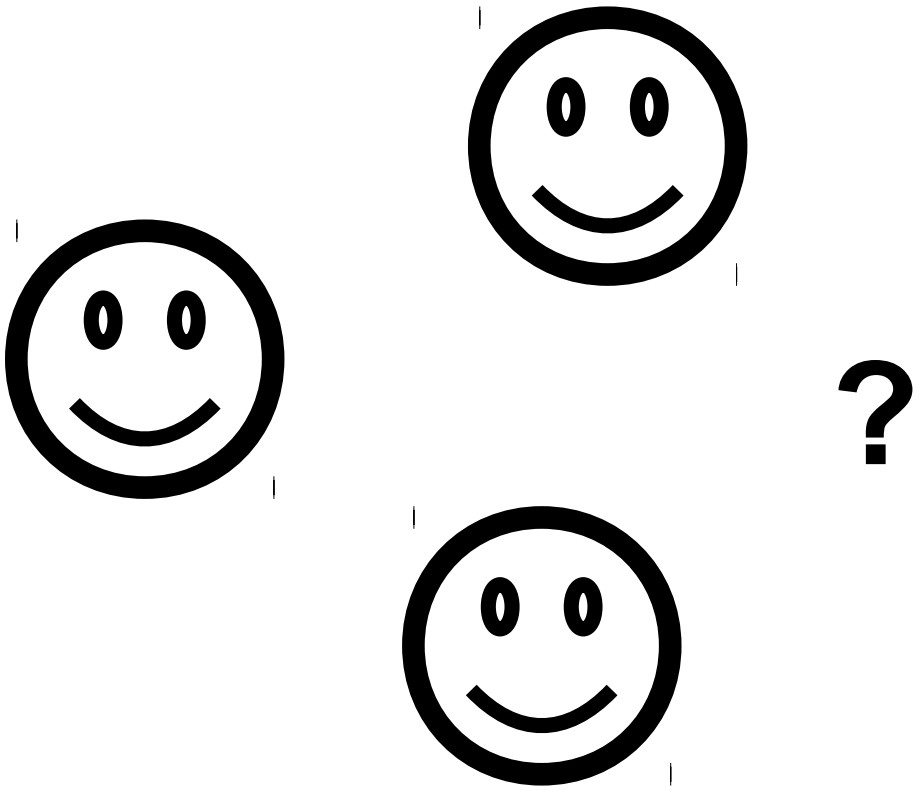
Who am I and why am I here?

- Prof. Dr. Joel Greenyer
  - Juniorprofessor at the Leibniz Universität Hannover
  - 10 years of research in the field of model-based software engineering
    - Graph- and model transformations
    - Behavior modeling
    - Formal analysis and synthesis

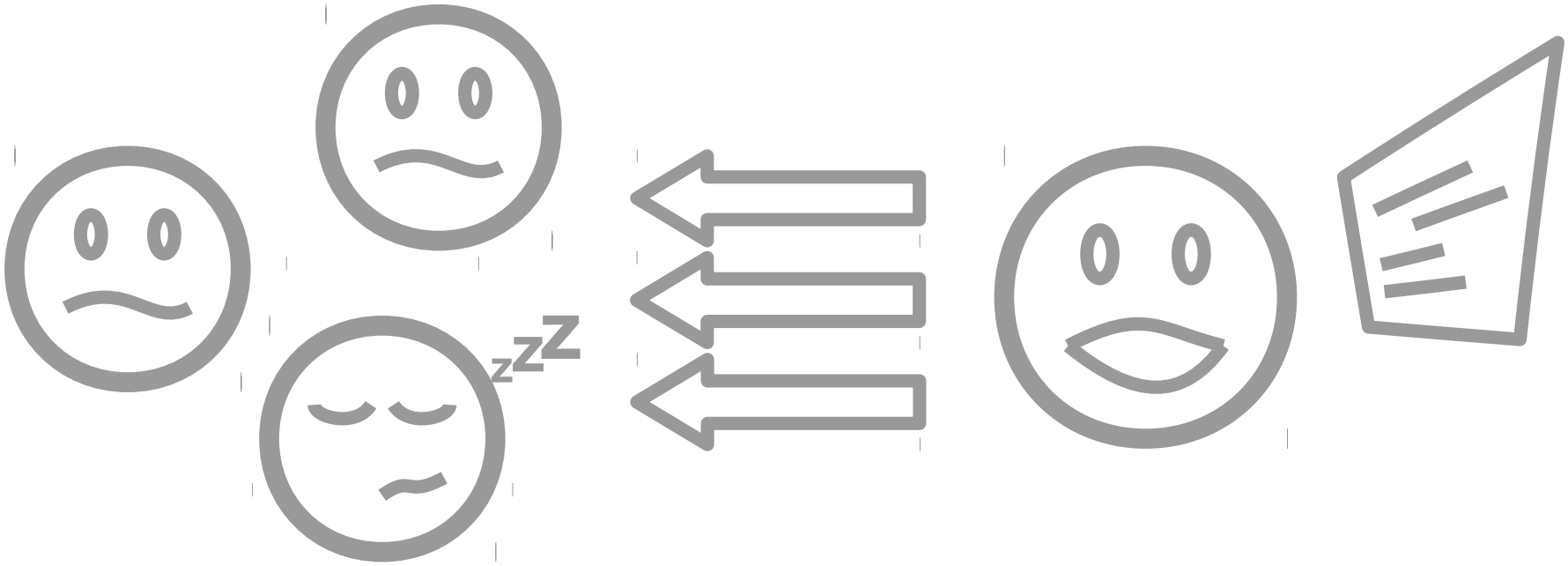
- Contact
  - Email: greenyer@uni-hannover.de
  - Office G322
  - You can make appointments via email



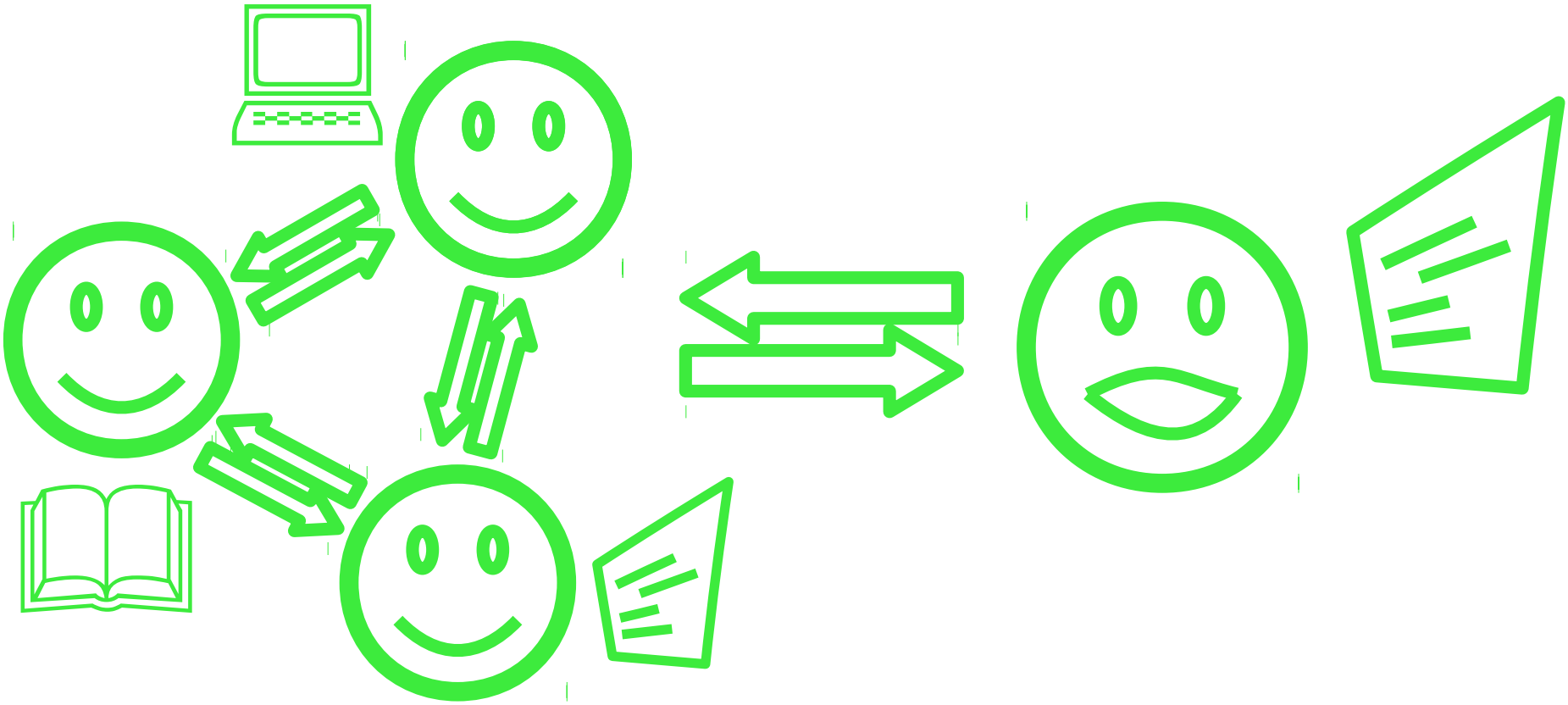
Who are you and why are you here?



Interaction in the lecture



Interaction in the lecture



Mini Projects

- During the lecture period, you can work on **mini projects** in **groups of three or four** students
  - you will build modeling languages and tools yourself!

Mini Projects

- During the lecture period, you can work on **mini projects** in **groups of three or four** students
  - you will build modeling languages and tools yourself!

- This way, you can obtain **bonus points**:

Mini Projects

- During the lecture period, you can work on **mini projects** in **groups of three or four** students
  - you will build modeling languages and tools yourself!

- This way, you can obtain **bonus points**:
  - For each successfully **submitted and presented** mini project, you can earn **3 points**

Mini Projects

- During the lecture period, you can work on **mini projects** in **groups of three or four** students
  - you will build modeling languages and tools yourself!

- This way, you can obtain **bonus points**:
  - For each successfully **submitted and presented** mini project, you can earn **3 points**
  - There will be three mini-projects, so you can earn **9 points**

Mini Projects

- During the lecture period, you can work on **mini projects** in **groups of three or four** students
  - you will build modeling languages and tools yourself!

- This way, you can obtain **bonus points**:
  - For each successfully **submitted and presented** mini project, you can earn **3 points**
  - There will be three mini-projects, so you can earn **9 points**
  - They will be added to your exam score **if you pass the exam**

Mini Projects

- During the lecture period, you can work on **mini projects** in **groups of three or four** students
  - you will build modeling languages and tools yourself!

- This way, you can obtain **bonus points**:
  - For each successfully **submitted and presented** mini project, you can earn **3 points**
  - There will be three mini-projects, so you can earn **9 points**
  - They will be added to your exam score **if you pass the exam**
  - With 9 additional points you can improve by a **full grade**

Mini Projects

- During the lecture period, you can work on **mini projects** in **groups of three or four** students
  - you will build modeling languages and tools yourself!

- This way, you can obtain **bonus points**:
  - For each successfully **submitted and presented** mini project, you can earn **3 points**
  - There will be three mini-projects, so you can earn **9 points**
  - They will be added to your exam score **if you pass the exam**
  - With 9 additional points you can improve by a **full grade**
  - However, it is required that you **present at least once** the results of your group (otherwise you can only get 1 point per mini project)

Mini Projects

- During the lecture period, you can work on **mini projects** in **groups of three or four** students
  - you will build modeling languages and tools yourself!

- This way, you can obtain **bonus points**:
  - For each successfully **submitted and presented** mini project, you can earn **3 points**
  - There will be three mini-projects, so you can earn **9 points**
  - They will be added to your exam score **if you pass the exam**
  - With 9 additional points you can improve by a **full grade**
  - However, it is required that you **present at least once** the results of your group (otherwise you can only get 1 point per mini project)
  - Only one participant can present the result of one mini project

Mini Projects

- During the lecture period, you can work on **mini projects** in **groups of three or four** students
  - you will build modeling languages and tools yourself!

- This way, you can obtain **bonus points**:
  - For each successfully **submitted and presented** mini project, you can earn **3 points**
  - There will be three mini-projects, so you can earn **9 points**
  - They will be added to your exam score **if you pass the exam**
  - With 9 additional points you can improve by a **full grade**
  - However, it is required that you **present at least once** the results of your group (otherwise you can only get 1 point per mini project)
  - Only one participant can present the result of one mini project
  - Use the Stud.IP forum if you are still looking to join a group

Exercises and Tutorials

- Exercise Sheets:
  - There will be occasional exercise sheets
  - But we will not correct or grade them

Exercises and Tutorials

- Exercise Sheets:
  - There will be occasional exercise sheets
  - But we will not correct or grade them

- Weekly tutorials:

Exercises and Tutorials

- Exercise Sheets:
  - There will be occasional exercise sheets
  - But we will not correct or grade them

- Weekly tutorials:
  - We will use them to discuss exercise tasks

Exercises and Tutorials

- Exercise Sheets:
  - There will be occasional exercise sheets
  - But we will not correct or grade them

- Weekly tutorials:
  - We will use them to discuss exercise tasks
  - Some exercise tasks will be marked with *“presentation bonus”*

Exercises and Tutorials

- Exercise Sheets:
  - There will be occasional exercise sheets
  - But we will not correct or grade them

- Weekly tutorials:
  - We will use them to discuss exercise tasks
  - Some exercise tasks will be marked with *“presentation bonus”*
    - you can earn your presentation bonus also by presenting you solution to this exercise task

Exercises and Tutorials

- Exercise Sheets:
  - There will be occasional exercise sheets
  - But we will not correct or grade them

- Weekly tutorials:
  - We will use them to discuss exercise tasks
  - Some exercise tasks will be marked with *“presentation bonus”*
    - you can earn your presentation bonus also by presenting you solution to this exercise task
  - Otherwise, you can use the tutorials to work on your mini projects

Organizational Issues

- Date and time of the **lecture**:
  - Tuesdays 14.15-15.45 PM, A310
- Date and time of the **tutorial**:
  - Tuesdays, 16:00-16:45, B302 (per request: 16:05 – 16:50)

Organizational Issues

- Date and time of the **lecture**:
  - Tuesdays 14.15-15.45 PM, A310
- Date and time of the **tutorial**:
  - Tuesdays, 16:00-16:45, B302 (per request: 16:05 – 16:50)

- **Tutorials start next week**

Organizational Issues

- Date and time of the **lecture**:
  - Tuesdays 14.15-15.45 PM, A310
- Date and time of the **tutorial**:
  - Tuesdays, 16:00-16:45, B302 (per request: 16:05 – 16:50)
- **Tutorials start next week**
- **Exam:**

Organizational Issues

- Date and time of the **lecture**:
  - Tuesdays 14.15-15.45 PM, A310
- Date and time of the **tutorial**:
  - Tuesdays, 16:00-16:45, B302 (per request: 16:05 – 16:50)
- **Tutorials start next week**
- **Exam:**
  - there will be an exam of 60 minute

Organizational Issues

- Date and time of the **lecture**:
  - Tuesdays 14.15-15.45 PM, A310
- Date and time of the **tutorial**:
  - Tuesdays, 16:00-16:45, B302 (per request: 16:05 – 16:50)
- **Tutorials start next week**
- **Exam:**
  - there will be an exam of 60 minute
  - date (preliminary): 30 August, 15.15-16.45

ITIS Students

- This course is also offered to ITIS students under the name **Model-Driven Software Development**

- Because the course is worth **6 ECTS points** for ITIS students (instead of 4 ECTS points for non-ITIS students), the ITIS students will have to work on and present an additional mini project towards the end of the lecture period
  - details will be announced later

Lecture Dates

- 05.04.2016 – today
- 12.04.2016
- 19.04.2016
- 26.04.2016
- 03.05.2016
- 10.05.2016 – **no lecture**
- 17.05.2016 – no lecture (lecture-free week)
- 24.05.2016
- 31.05.2016
- 07.06.2016
- 14.06.2016 – **no lecture**
- 21.06.2016
- 28.06.2016
- 05.07.2016
- 12.07.2016

SOFTWARE ENGINEERING

1.4 Introduction

Leibniz Universität Hannover

SOFTWARE ENGINEERING

What is a Model?

What is a Model?

- popular models:

Quelle: http://de.wikipedia.org/wiki/Heidi_Klum

What is a Model?

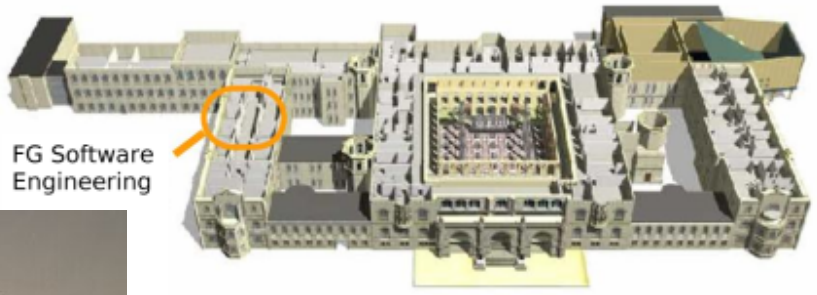
- Models of boats:

http://de.wikipedia.org/wiki/Modellboot

Portrevel shiphandling school
http://www.portrevel.com/

What is a Model?

- Architecture models:



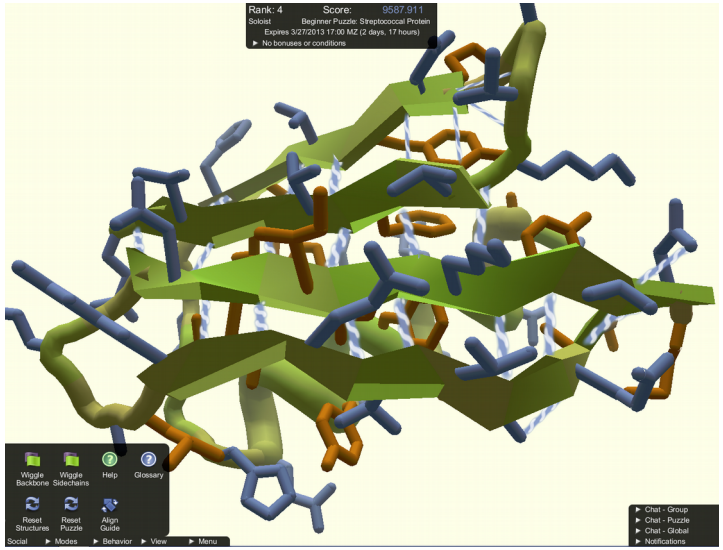
(this was already the first model in the lecture)

Model of the Opéra Garnier, Paris

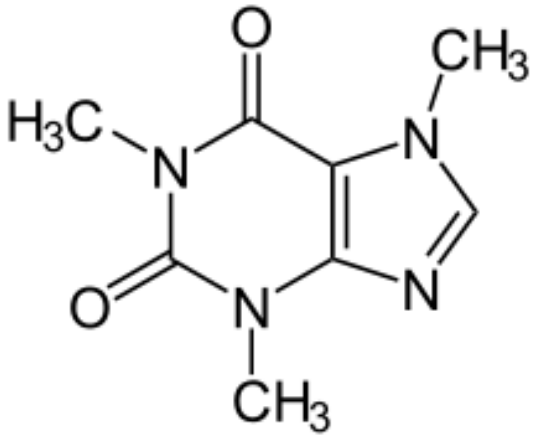
http://de.wikipedia.org/wiki/Op%C3%A9ra_Garnier

Neues Rathaus in Hannover

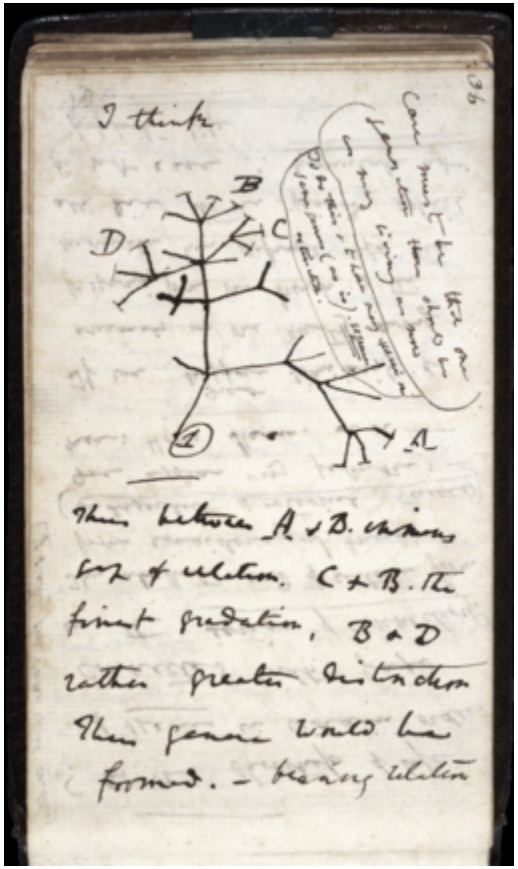
What is a Model?



model of a protein molecule



model of a caffein molecule

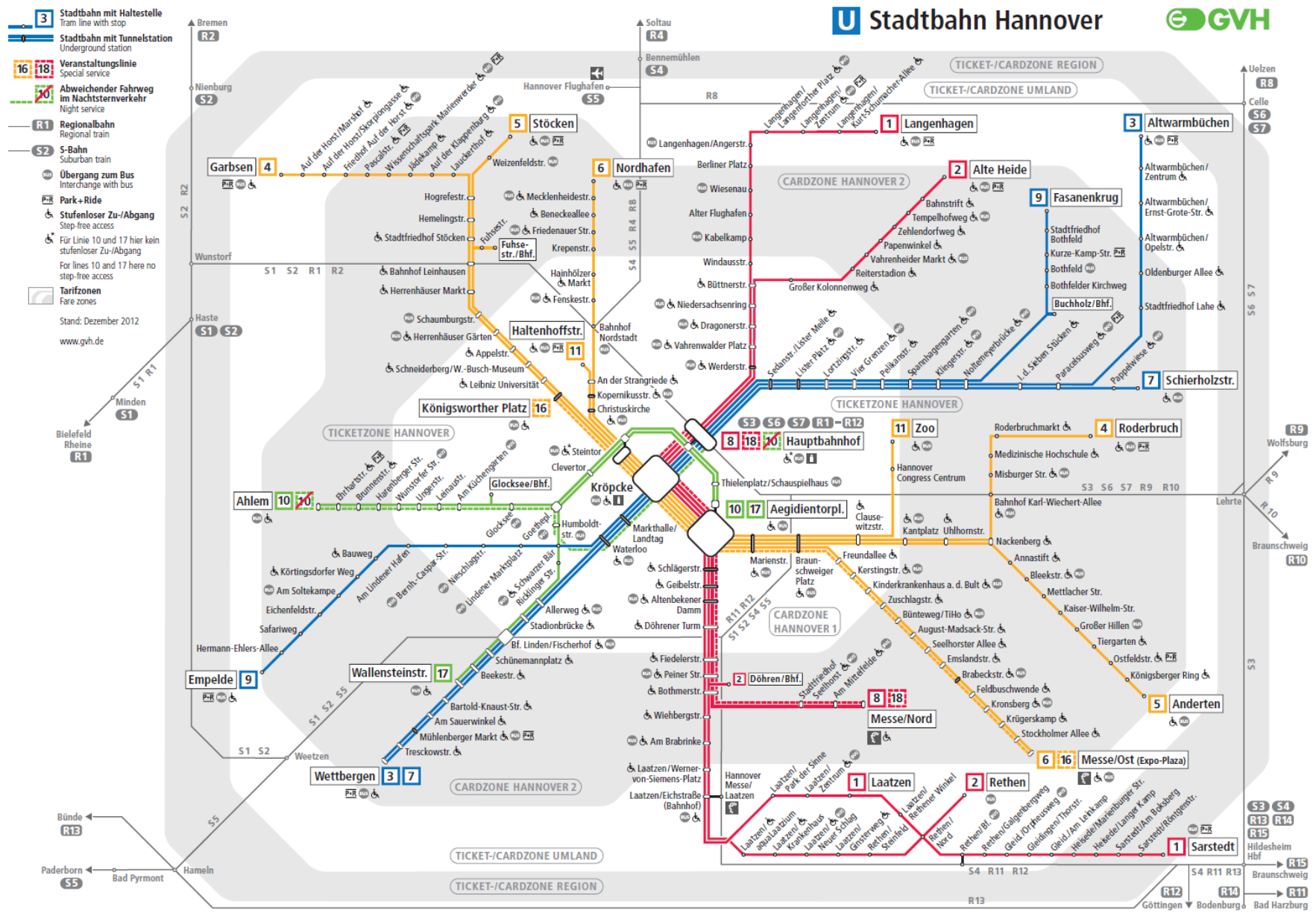


page from Darwin's notebook around July 1837 showing his first sketch of an evolutionary tree

https://en.wikipedia.org/wiki/Tree_of_life_%28biology%29

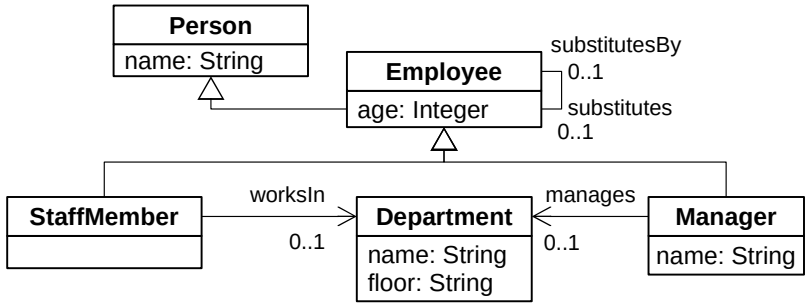
SOFTWARE ENGINEERING

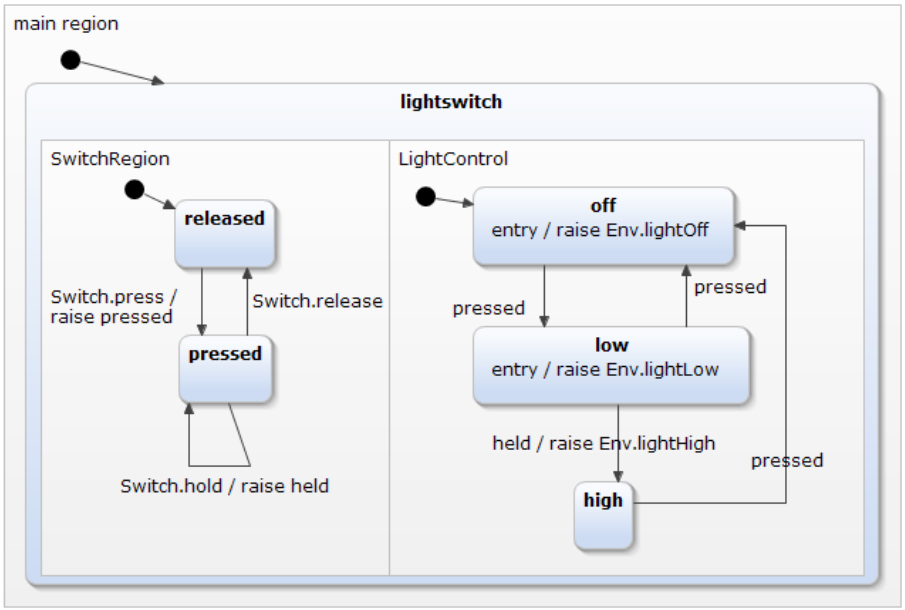
What is a Model?

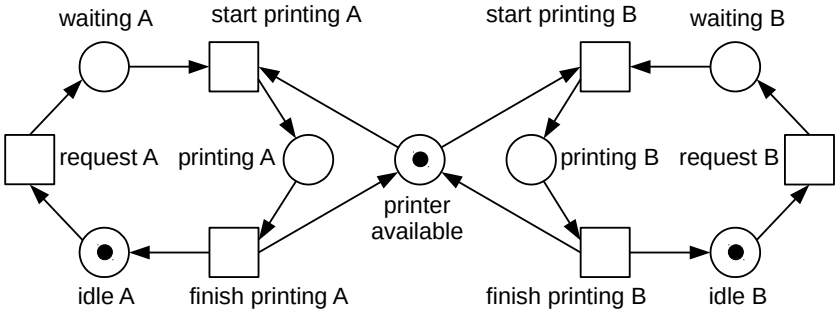


What is a Model?

- Models in computer science:








```
1 procedure DFS(G,v):
2   label v as discovered
3   for all edges from v to w in
          G.adjacentEdges(v) do
4     if vertex w is not labeled as
          discovered then
5       call DFS(G,w)
```

What is a Model?

- Definition: **Model** (based on H. Stachowiak and extended)

What is a Model?

- Definition: **Model** (based on H. Stachowiak and extended)
  - A **representation** of entities and relationships of an original system (which can itself be a model)

What is a Model?

- Definition: **Model** (based on H. Stachowiak and extended)
  - A **representation** of entities and relationships of an original system (which can itself be a model)
  - with a certain **correspondence** between the entities and relationships of the model to that of the original

What is a Model?

- Definition: **Model** (based on H. Stachowiak and extended)
  - A **representation** of entities and relationships of an original system (which can itself be a model)
  - with a certain **correspondence** between the entities and relationships of the model to that of the original
  - A model is usually an **abstraction (reduction)** of the original: it does not capture all of the attributes of an original

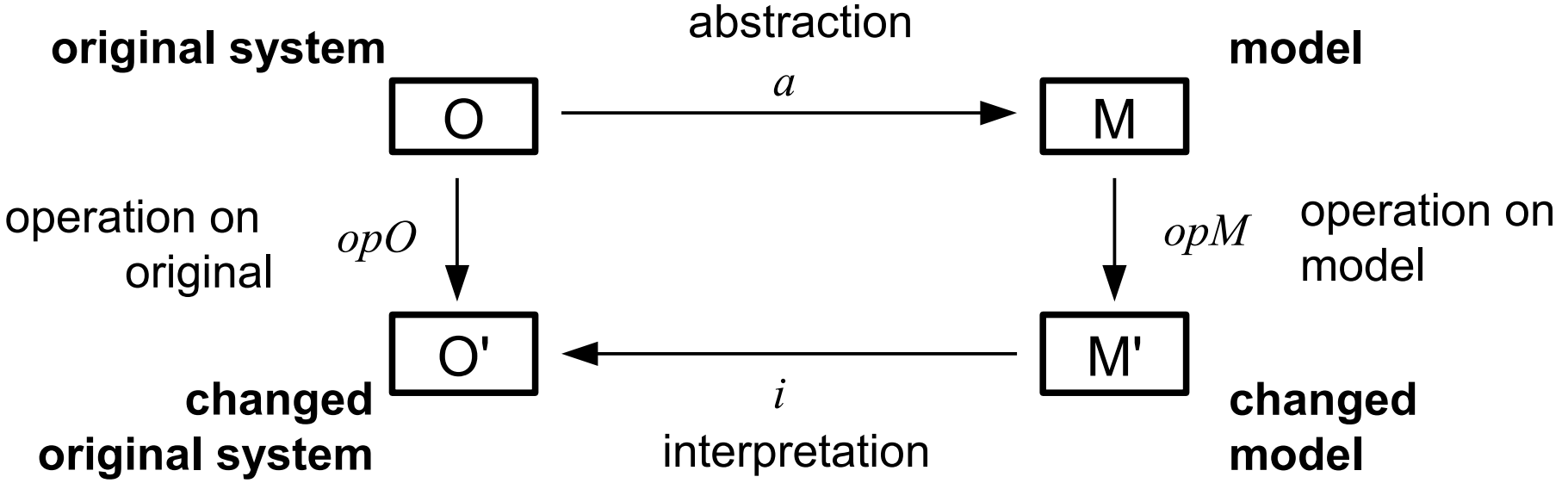
What is a Model?

- Definition: **Model** (based on H. Stachowiak and extended)
  - A **representation** of entities and relationships of an original system (which can itself be a model)
  - with a certain **correspondence** between the entities and relationships of the model to that of the original
  - A model is usually an **abstraction (reduction)** of the original: it does not capture all of the attributes of an original
  - but it captures those attributes which are relevant to the modeler for a particular use of the model (**pragmatics**)

What is a Model?

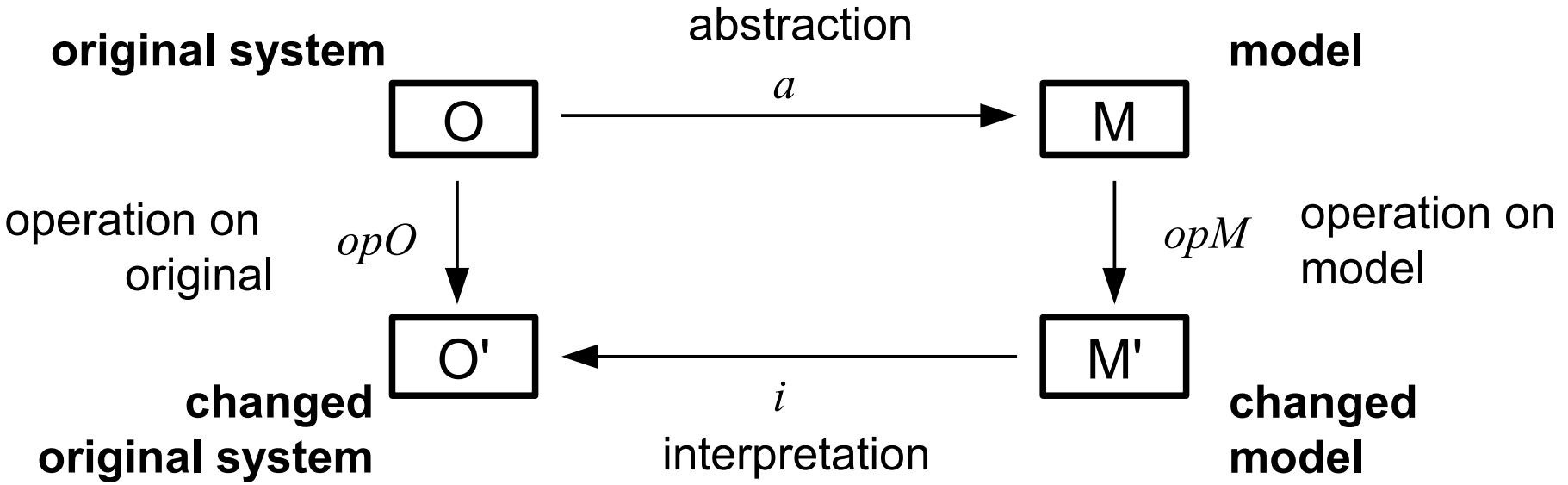
- Definition: **Model** (based on H. Stachowiak and extended)
  - A **representation** of entities and relationships of an original system (which can itself be a model)
  - with a certain **correspondence** between the entities and relationships of the model to that of the original
  - A model is usually an **abstraction (reduction)** of the original: it does not capture all of the attributes of an original
  - but it captures those attributes which are relevant to the modeler for a particular use of the model (**pragmatics**)
  - Operations on the model allow the modeler to deduce information about the original (**interpretation**)

What is a Model?



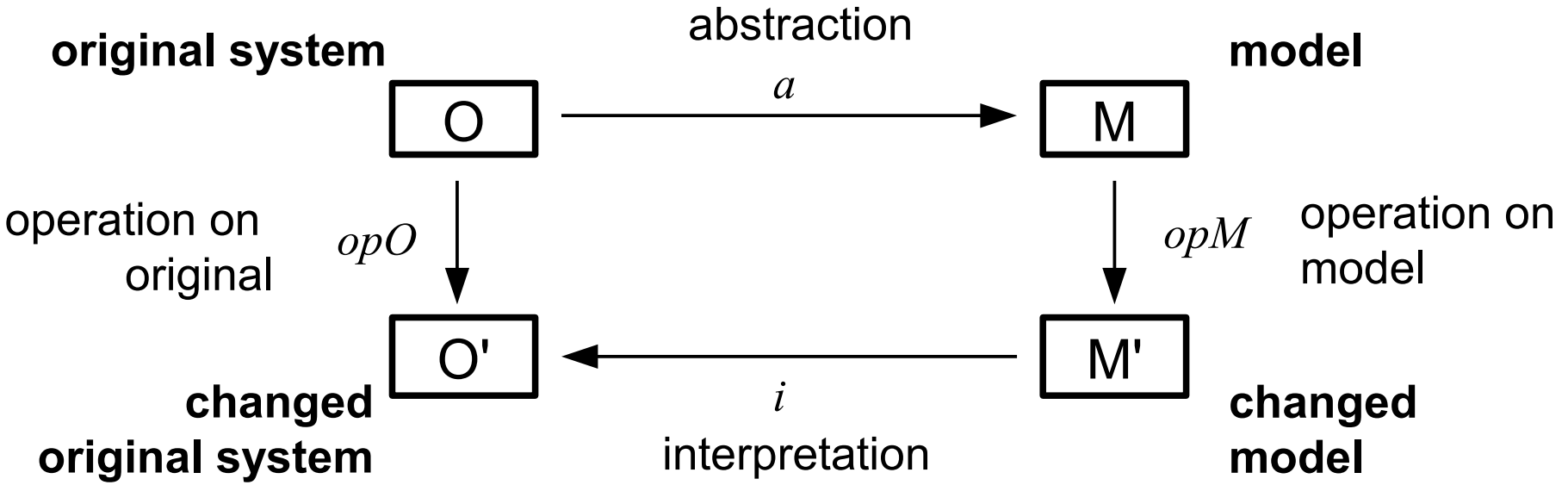
What is a Model?

- *a*: abstraction, projection, reduction, formalization



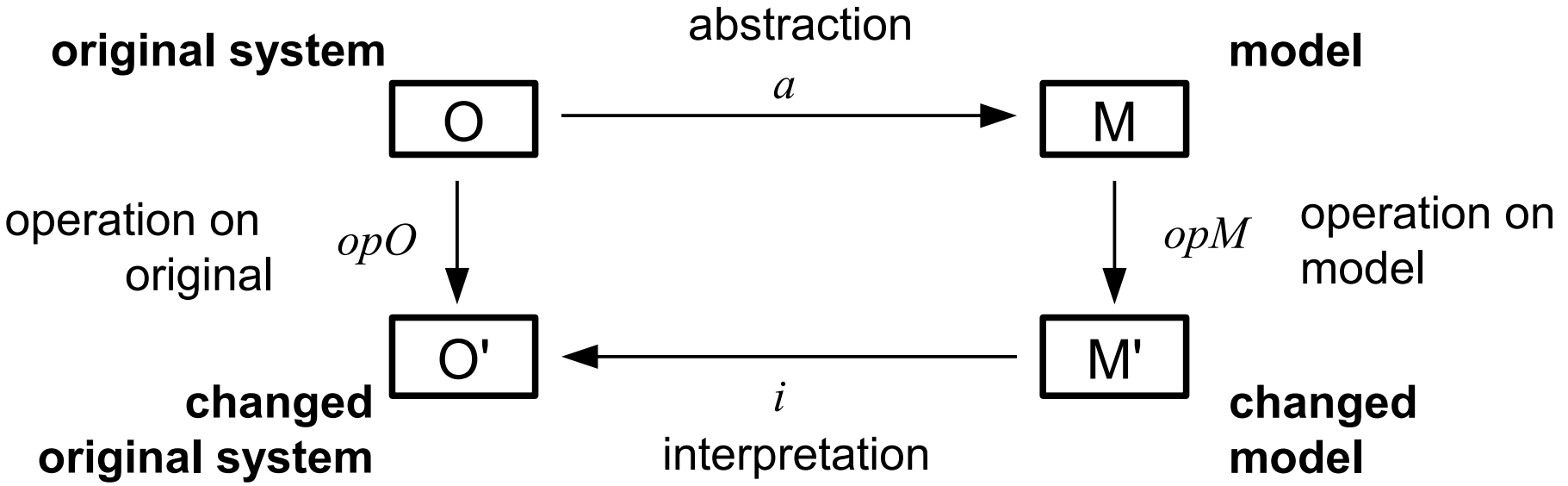
What is a Model?

- a: abstraction, projection, reduction, formalization
- opM: operations on the model



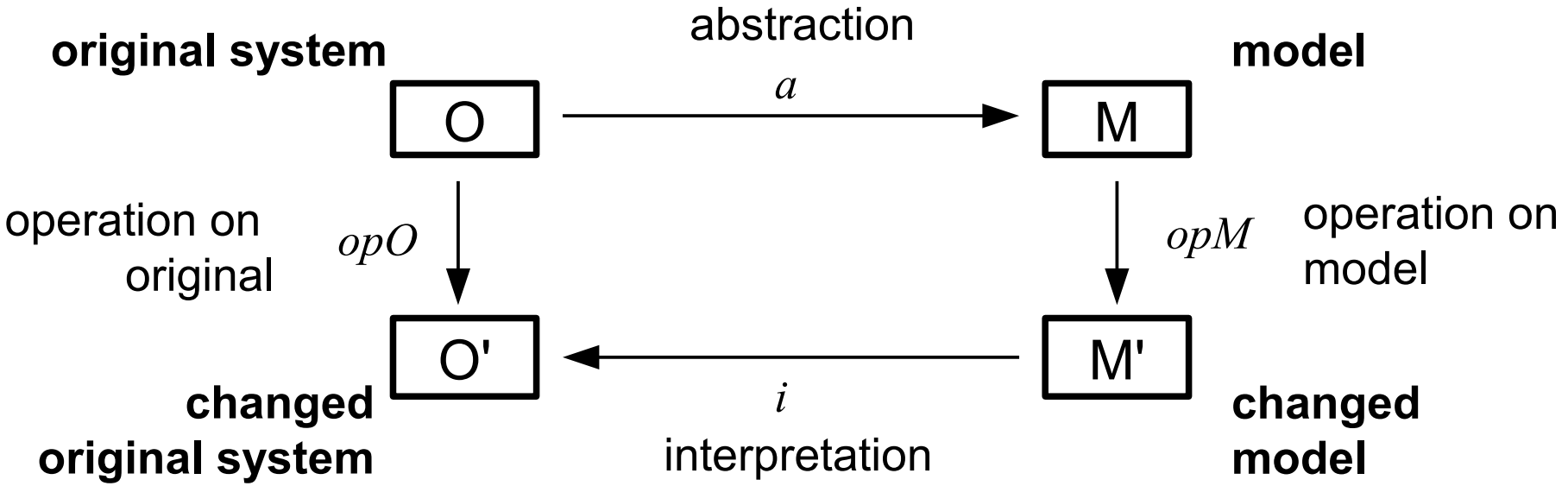
What is a Model?

- a: abstraction, projection, reduction, formalization
- opM: operations on the model
- i: interpretation of model leads to information about original

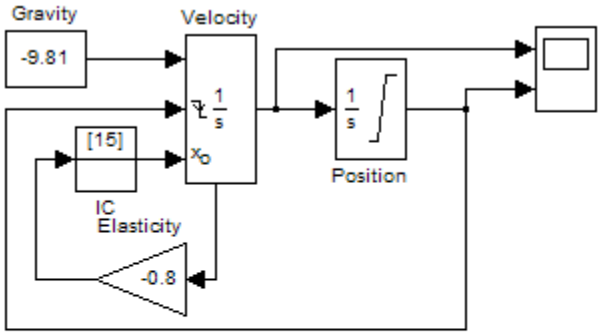


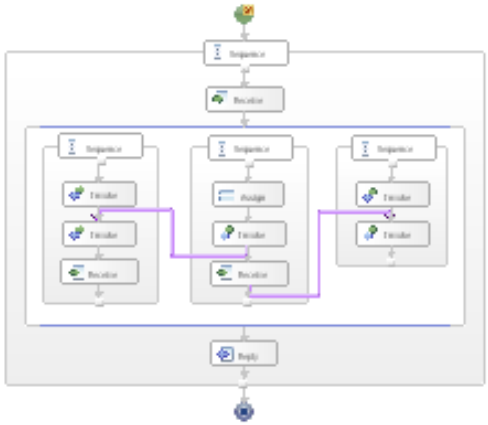
What is a Model?

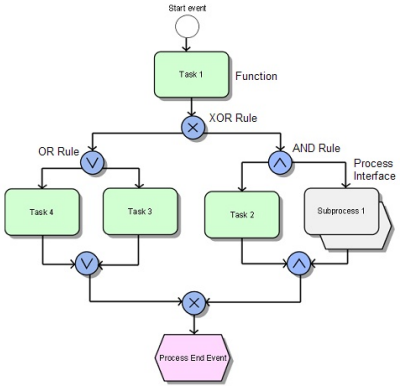
- a: abstraction, projection, reduction, formalization
- opM: operations on the model
- i: interpretation of model leads to information about original
- Operations on the original must must correspond to those on the model, i.e., $opO(O) = i\,(opM(a(O)))$



What is Model-Based Software Engineering?







UNIFIED MODELING LANGUAGE™

OMG SYSTEMS MODELING LANGUAGE™

AADL

What is Model-Based Software Engineering?

- **Model-Based Software Engineering** describes software engineering techniques that rely on the use of specific models for certain tasks in the software development

What is Model-Based Software Engineering?

- **Model-Based Software Engineering** describes software engineering techniques that rely on the use of specific models for certain tasks in the software development

  also:

- **Model-Driven Software Development (MDSD)**:

  Techniques that rely on the use of specific formal models for automatically generating executable code

What is Model-Based Software Engineering?

- **Model-Based Software Engineering** describes software engineering techniques that rely on the use of specific models for certain tasks in the software development

  also:

- **Model-Driven Software Development (MDSD)**:

  Techniques that rely on the use of specific formal models for automatically generating executable code

- or **Model-Driven Engineering (MDE)**:

  Techniques that create and use (domain) specific models for solving specific (software) engineering problems

What is Model-Based Software Engineering?

- **Model-Driven Architecture (MDA):**

  A standardized software development methodology by the Object Management Group (OMG) for developing software by using the modeling languages defined by the OMG (like UML)

What is Model-Based Software Engineering?

- **Model-Driven Architecture (MDA):**

  A standardized software development methodology by the Object Management Group (OMG) for developing software by using the modeling languages defined by the OMG (like UML)

- **Computer-Aided Software Engineering (CASE):**

  Software development tools that use (often graphical) models, inspired by Computer-Aided Design (CAD)

What is Model-Based Software Engineering?

- **Round-Trip Engineering:**

  A software development methodology that aims to support changes in the model as well as in the generated code. Not only can code be generated from models, but changes in the code can also be translated back into the model (reverse engineering)

What is Model-Based Software Engineering?

- **Round-Trip Engineering:**

  A software development methodology that aims to support changes in the model as well as in the generated code. Not only can code be generated from models, but changes in the code can also be translated back into the model (reverse engineering)

- **Domain-Driven Design (DDD):**

  Principles and patterns for creating models for a particular domain

What is Model-Based Software Engineering?

- The lecture is called **Model-Based Software Engineering** because it is the general term that also includes MDSD, MDE, MDA, etc.

What is Model-Based Software Engineering?

- The lecture is called **Model-Based Software Engineering** because it is the general term that also includes MDSD, MDE, MDA, etc.

- Some people say that they are doing *model-driven* instead of *"only" model-based* to emphasize that they are doing "more" with their models
  - code generation, execution instead of only design and documentation

What is Model-Based Software Engineering?

- The lecture is called **Model-Based Software Engineering** because it is the general term that also includes MDSD, MDE, MDA, etc.

- Some people say that they are doing *model-driven* instead of *"only" model-based* to emphasize that they are doing "more" with their models
  - code generation, execution instead of only design and documentation

- In that sense, we will do mainly *model-driven* in this lecture

What is Model-Based Software Engineering?

- The lecture is called **Model-Based Software Engineering** because it is the general term that also includes MDSD, MDE, MDA, etc.

- Some people say that they are doing *model-driven* instead of *"only" model-based* to emphasize that they are doing "more" with their models
  - code generation, execution instead of only design and documentation

- In that sense, we will do mainly *model-driven* in this lecture
  - in case anybody asks ;-))

What is Model-Based Software Engineering?

- The lecture is called **Model-Based Software Engineering** because it is the general term that also includes MDSD, MDE, MDA, etc.

- Some people say that they are doing *model-driven* instead of *"only" model-based* to emphasize that they are doing "more" with their models
  - code generation, execution instead of only design and documentation

- In that sense, we will do mainly *model-driven* in this lecture
  - in case anybody asks ;-))

Goals and Reasons for MBSE

- Abstraction via models reduces complexity of the development

Goals and Reasons for MBSE

- Abstraction via models reduces complexity of the development
  - Adequate models are easier to understand and to communicate

Goals and Reasons for MBSE

- Abstraction via models reduces complexity of the development
  - Adequate models are easier to understand and to communicate
  - Some modeling languages support separation of concerns using different views

Goals and Reasons for MBSE

- Abstraction via models reduces complexity of the development
  - Adequate models are easier to understand and to communicate
  - Some modeling languages support separation of concerns using different views
- This increase development speed and software quality

Goals and Reasons for MBSE

- Abstraction via models reduces complexity of the development
  - Adequate models are easier to understand and to communicate
  - Some modeling languages support separation of concerns using different views
- This increase development speed and software quality
  - specific modeling languages and tools help engineers to make less errors

Goals and Reasons for MBSE

- Abstraction via models reduces complexity of the development
  - Adequate models are easier to understand and to communicate
  - Some modeling languages support separation of concerns using different views
- This increase development speed and software quality
  - specific modeling languages and tools help engineers to make less errors
  - automated transformations and code generation reduce errors

Goals and Reasons for MBSE

- Abstraction via models reduces complexity of the development
  - Adequate models are easier to understand and to communicate
  - Some modeling languages support separation of concerns using different views
- This increase development speed and software quality
  - specific modeling languages and tools help engineers to make less errors
  - automated transformations and code generation reduce errors
  - platform independence can be reached via different transformations (especially a goal of MDA)

Goals and Reasons for MBSE

- Abstraction via models reduces complexity of the development
  - Adequate models are easier to understand and to communicate
  - Some modeling languages support separation of concerns using different views
- This increase development speed and software quality
  - specific modeling languages and tools help engineers to make less errors
  - automated transformations and code generation reduce errors
  - platform independence can be reached via different transformations (especially a goal of MDA)
- Models can be used for other analysis tasks (for example formal verification)

Goals and Reasons for MBSE

- Reduction of development costs

Goals and Reasons for MBSE

- Reduction of development costs
- Earlier time-to-market

Goals and Reasons for MBSE

- Reduction of development costs
- Earlier time-to-market
- Better ways to handle new challenges

Goals and Reasons for MBSE

- Reduction of development costs
- Earlier time-to-market
- Better ways to handle new challenges
  - for example multi-platform software, (software-) production lines, etc.

Use of MBSE

- Model-Based (software) engineering tools are used heavily for example in the automotive, avionics, and robotics domain

Use of MBSE

- Model-Based (software) engineering tools are used heavily for example in the automotive, avionics, and robotics domain
  - Often SCADE or MATLAB Simulink for control engineering (models of continuous dynamics)

Use of MBSE

- Model-Based (software) engineering tools are used heavily for example in the automotive, avionics, and robotics domain
  - Often SCADE or MATLAB Simulink for control engineering (models of continuous dynamics)
  - software for discrete dynamics, for example for communication of subsystems, becomes increasingly relevant

Use of MBSE

- Model-Based (software) engineering tools are used heavily for example in the automotive, avionics, and robotics domain
  - Often SCADE or MATLAB Simulink for control engineering (models of continuous dynamics)
  - software for discrete dynamics, for example for communication of subsystems, becomes increasingly relevant

- Information systems and business process systems are often based or driven by models

Use of MBSE

- Model-Based (software) engineering tools are used heavily for example in the automotive, avionics, and robotics domain
  - Often SCADE or MATLAB Simulink for control engineering (models of continuous dynamics)
  - software for discrete dynamics, for example for communication of subsystems, becomes increasingly relevant

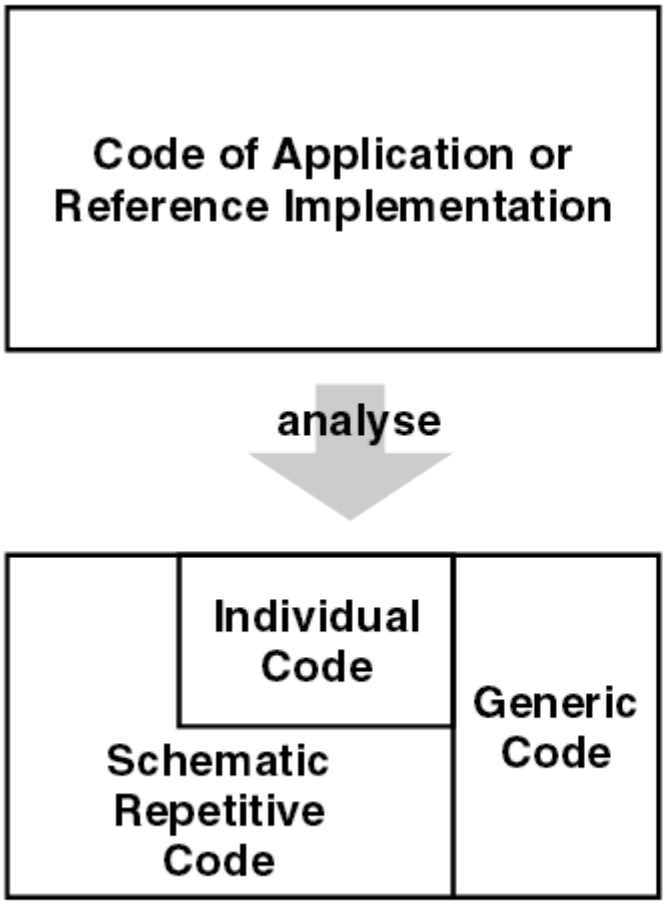
- Information systems and business process systems are often based or driven by models
  - Event-Driven Process Chain (EPCs) used to describe business processes in SAP systems

Use of MBSE

- Model-Based (software) engineering tools are used heavily for example in the automotive, avionics, and robotics domain
  - Often SCADE or MATLAB Simulink for control engineering (models of continuous dynamics)
  - software for discrete dynamics, for example for communication of subsystems, becomes increasingly relevant

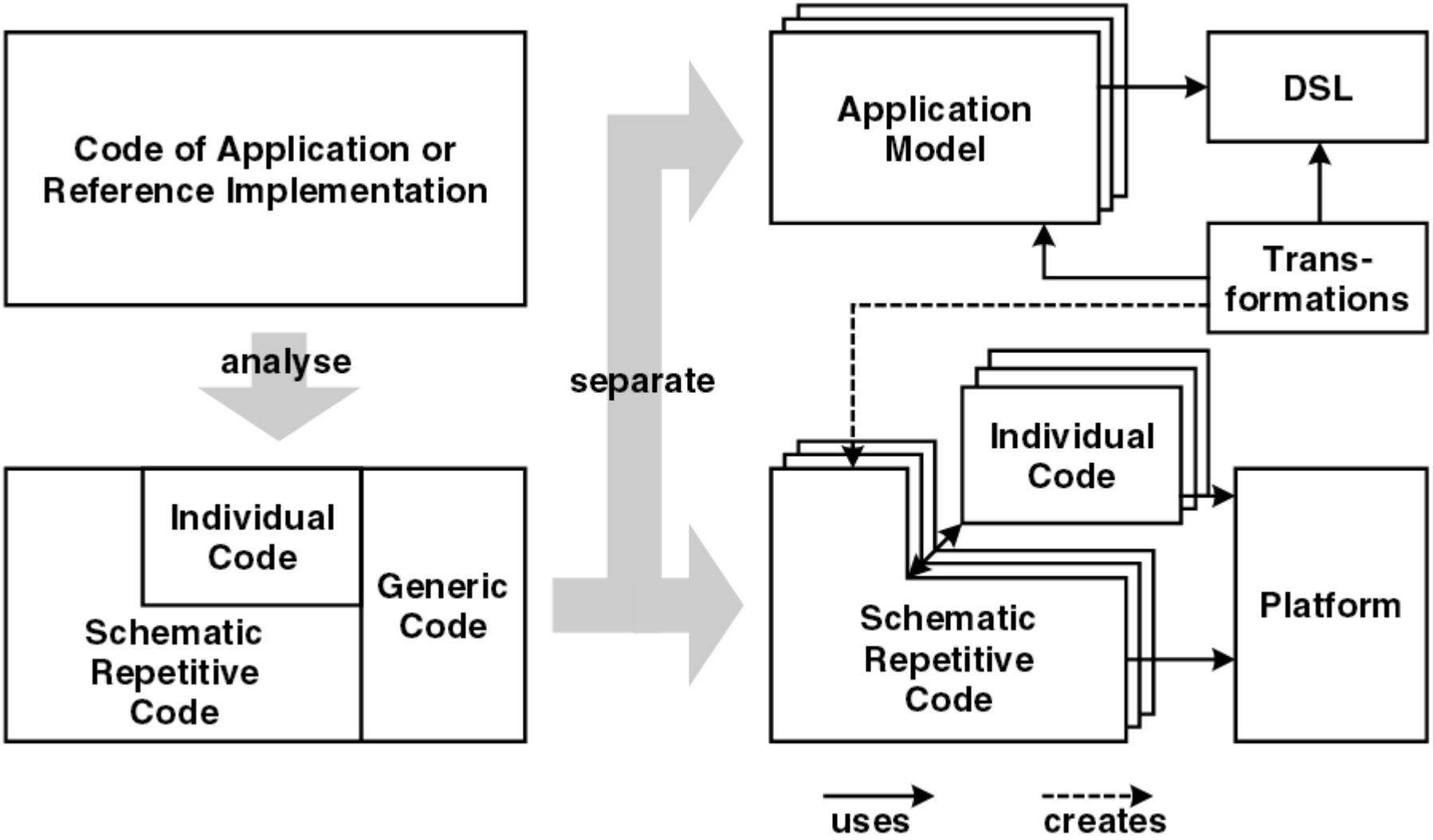
- Information systems and business process systems are often based or driven by models
  - Event-Driven Process Chain (EPCs) used to describe business processes in SAP systems
  - BPEL and other languages for describing processes in Service-Oriented Architectures

The idea of MBSE/MDSD in a bit more detail



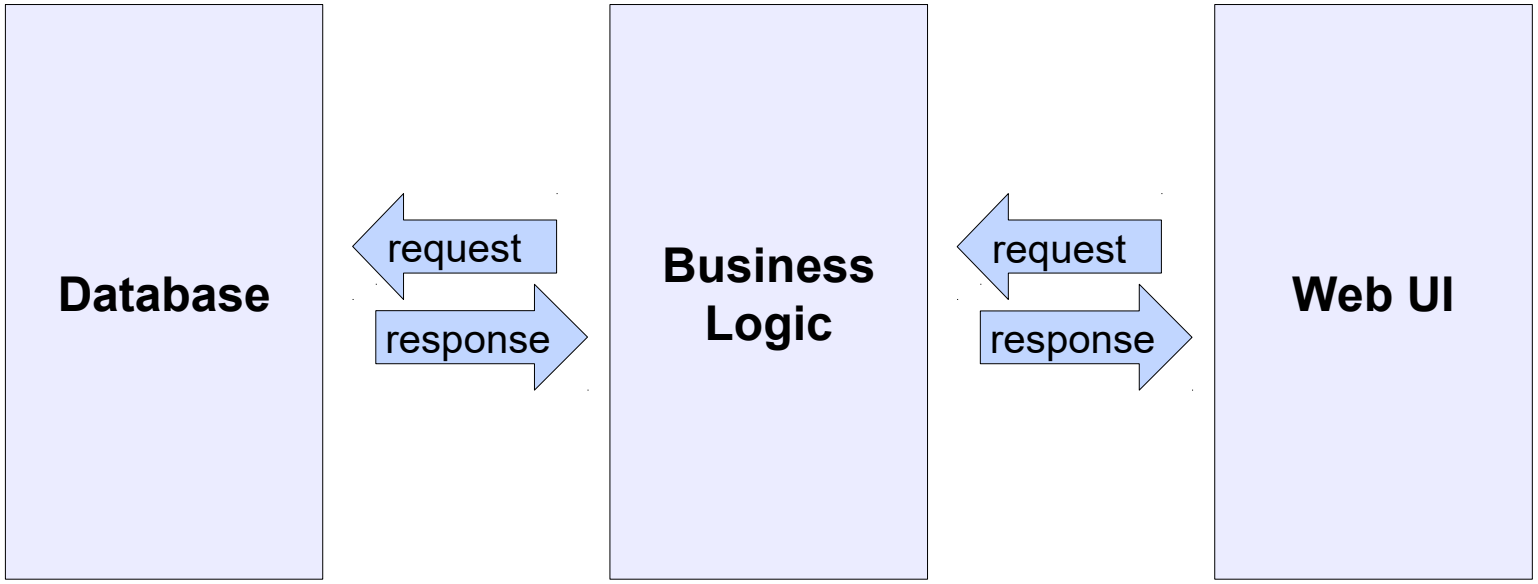
from Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006.

The idea of MBSE/MDSD in a bit more detail



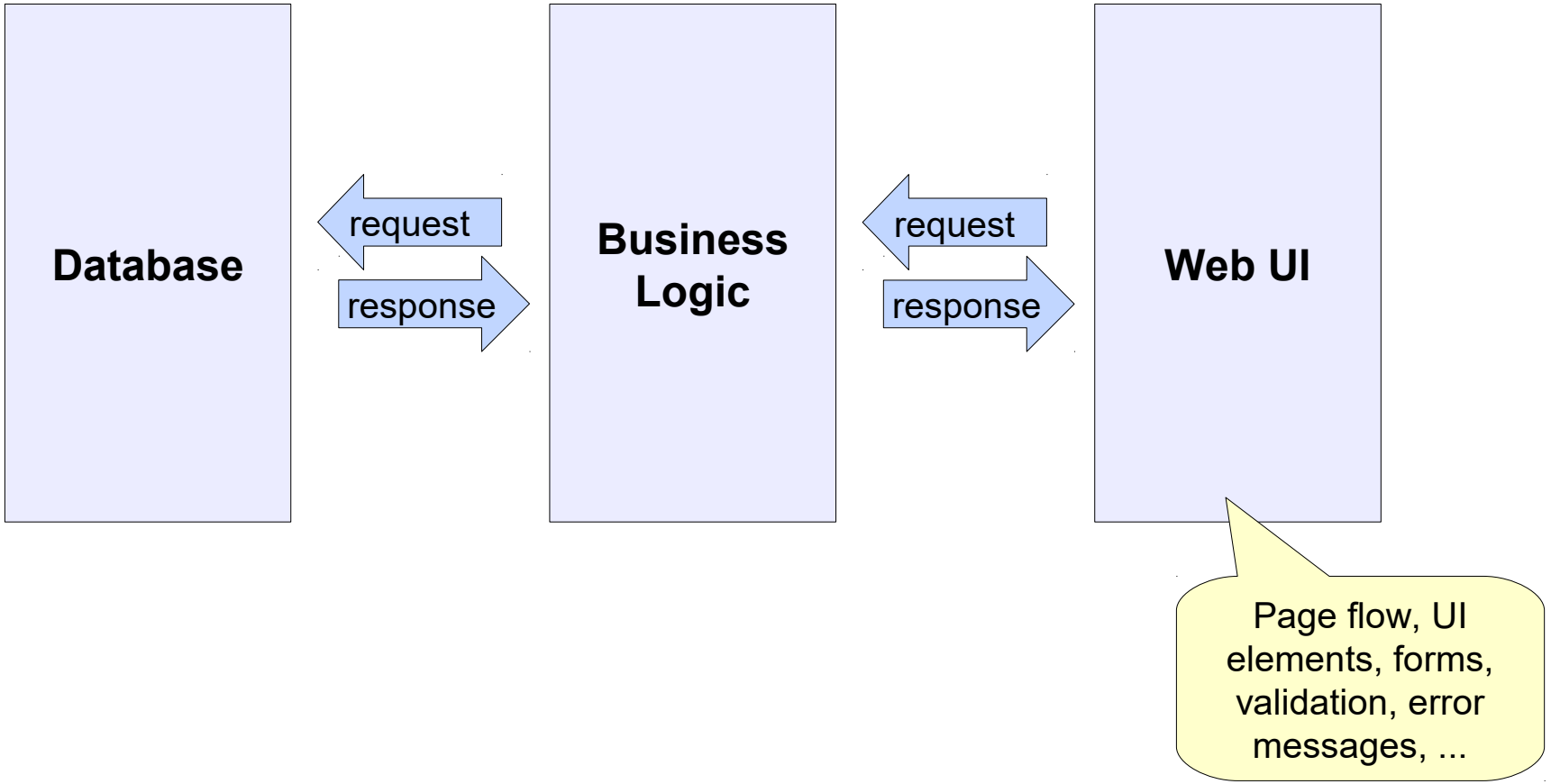
from Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006.

Example: MDSD for a Three-Tier Web Application



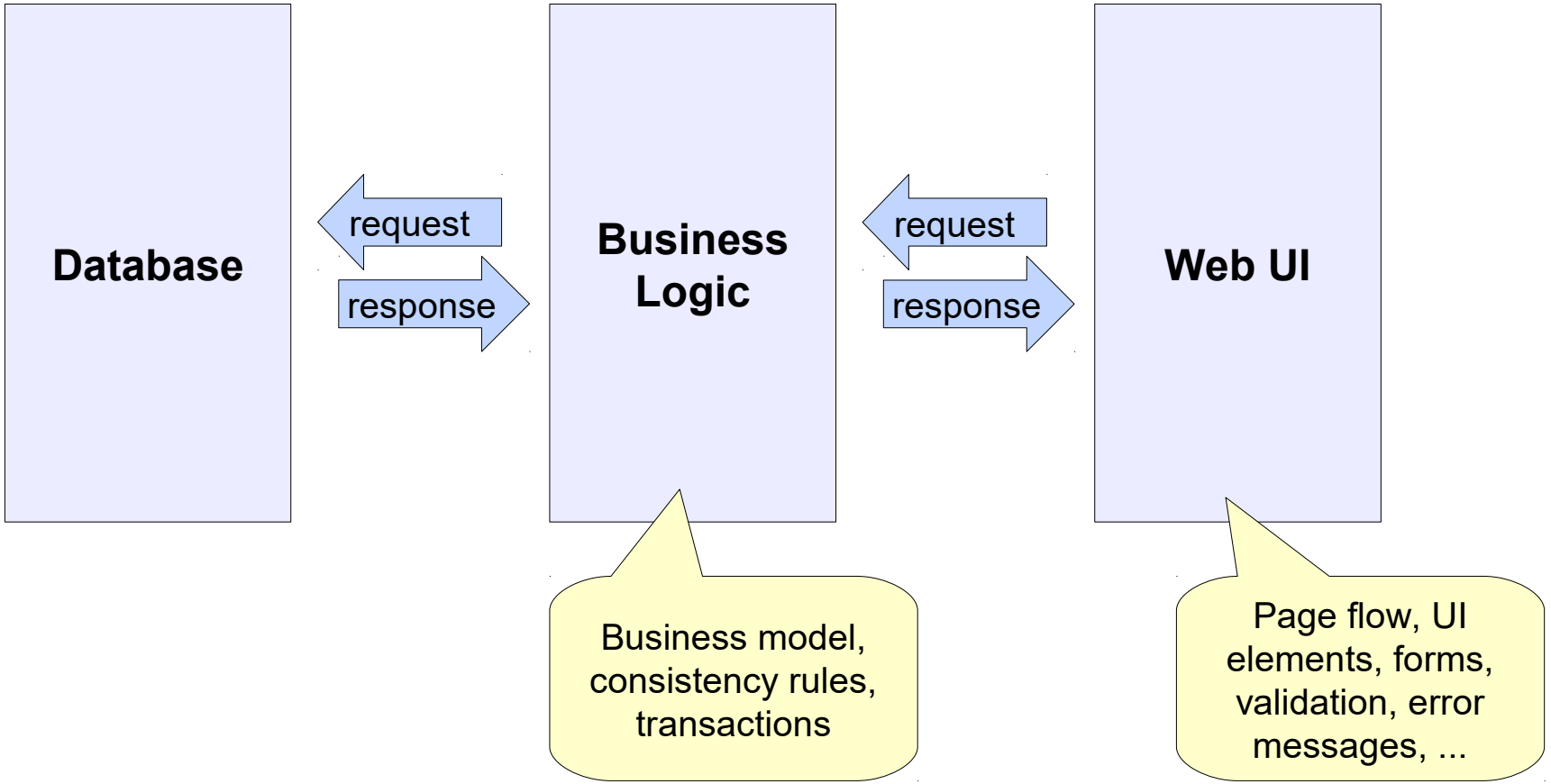
inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006.
and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application



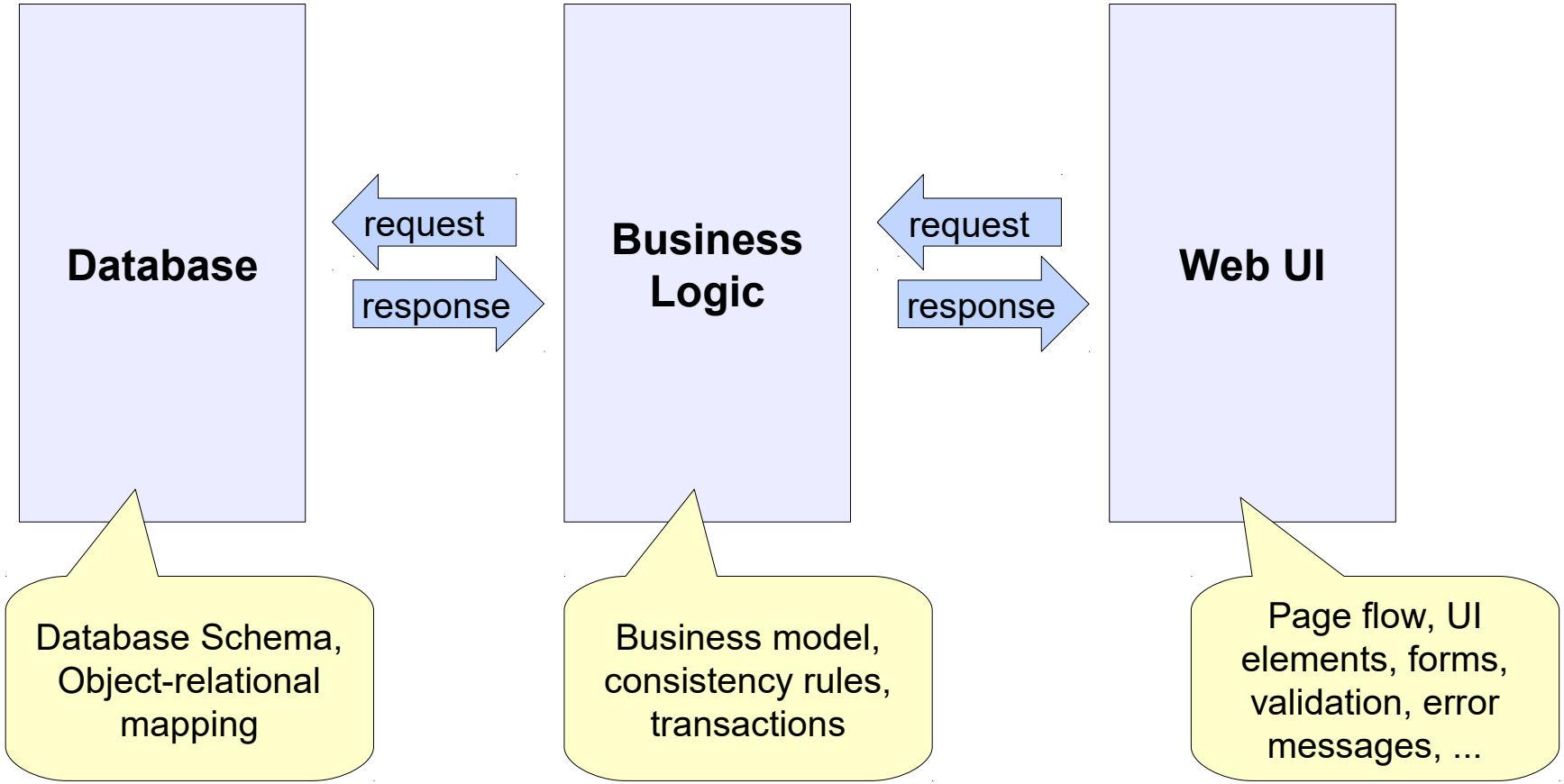
inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application



inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006.
and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

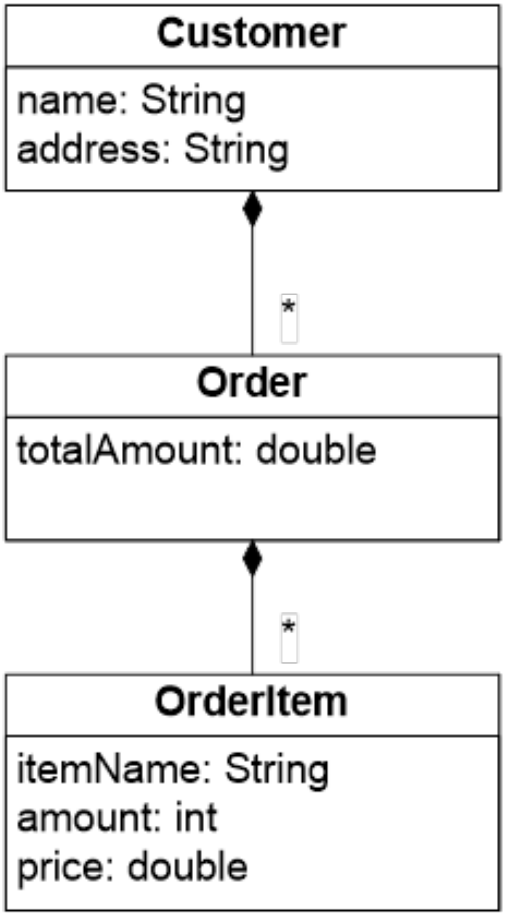
Example: MDSD for a Three-Tier Web Application



inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- The business model:

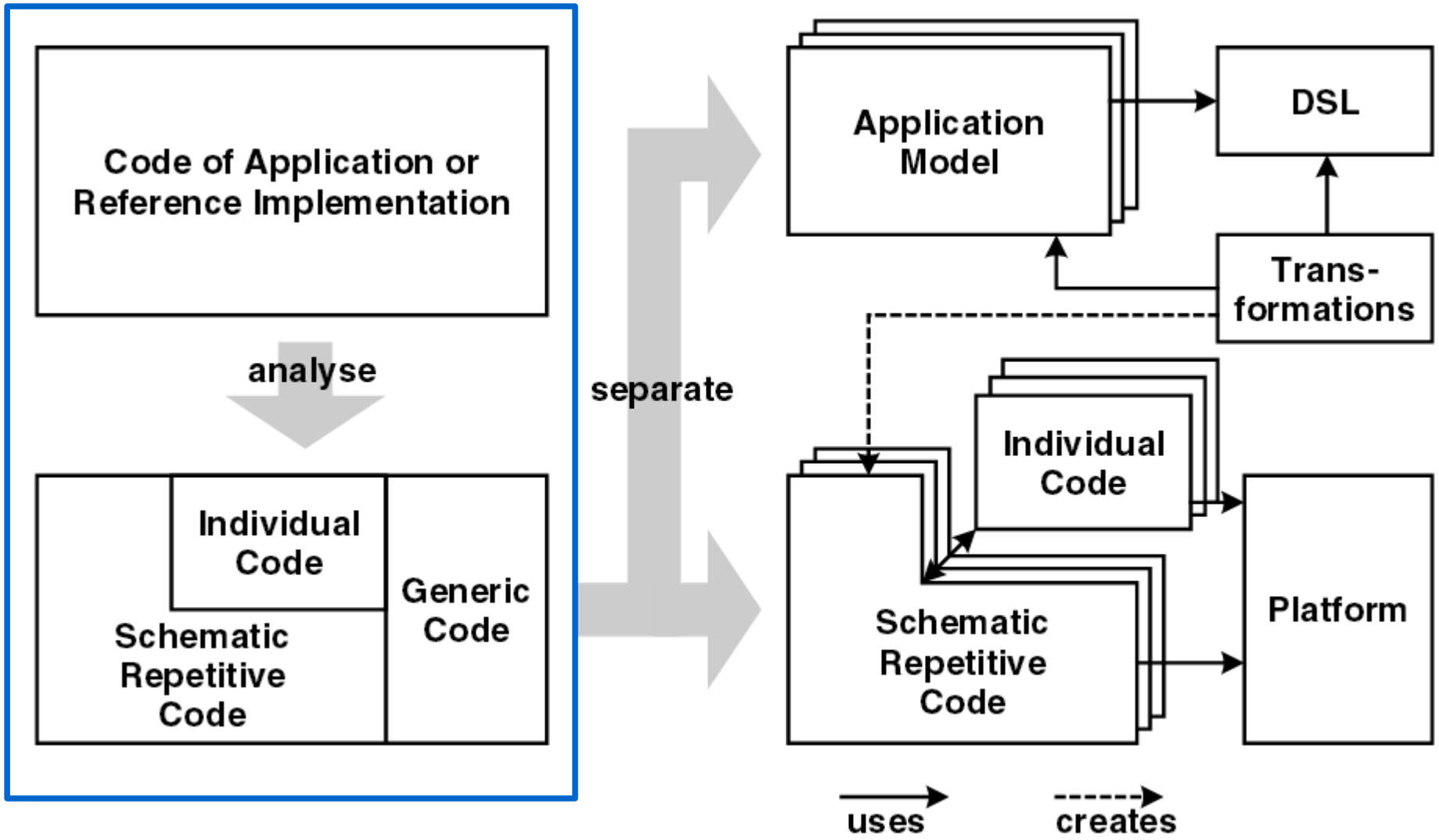


Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application (WAR)
  - Java Script UI & Validation
  - Struts UI Framework
- Business Logic
  - Java EE Session Beans
  - Entity Beans
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

The idea of MBSE/MDSD in a bit more detail



from Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application
  - Java Script UI & Validation
  - Struts UI Framework
- Business Logic
  - Java EE Session Beans
  - Entity Beans
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application
  - Java Script UI & Validation
  - Struts UI Framework
- Business Logic
  - Java EE Session Beans
  - Entity Beans
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database

When developing this application based on the business model, many development tasks are

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application
  - Java Script UI & Validation
  - Struts UI Framework
- Business Logic
  - Java EE Session Beans
  - Entity Beans
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database

When developing this application based on the business model, many development tasks are

generic or

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application
  - Java Script UI & Validation
  - Struts UI Framework
- Business Logic
  - Java EE Session Beans
  - Entity Beans
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database

When developing this application based on the business model, many development tasks are

generic or

repetitive

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application
  - Java Script UI & Validation
  - Struts UI Framework
- Business Logic
  - Java EE Session Beans
  - Entity Beans
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database

When developing this application based on the business model, many development tasks are

generic or

repetitive

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application (Setup&Config)
  - Java Script UI & Validation
  - Struts UI Framework
- Business Logic
  - Java EE Session Beans
  - Entity Beans
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database

When developing this application based on the business model, many development tasks are

generic or

repetitive

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application (Setup&Config)
  - Java Script UI & Validation
  - Struts UI Framework (Setup&Config)
- Business Logic
  - Java EE Session Beans
  - Entity Beans
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database

When developing this application based on the business model, many development tasks are

generic or

repetitive

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application (Setup&Config)
  - Java Script UI & Validation
  - Struts UI Framework (Setup&Config)
- Business Logic
  - Java EE Session Beans
  - Entity Beans
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database (Setup&Config)

When developing this application based on the business model, many development tasks are

generic or

repetitive

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application (Setup&Config)
  - Java Script UI & Validation
  - Struts UI Framework (Setup&Config)
- Business Logic
  - Java EE Session Beans
  - Entity Beans
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database (Setup&Config)

When developing this application based on the business model, many development tasks are

generic or

repetitive

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application — Setup&Config
  - Java Script UI & Validation — Derive from business model
  - Struts UI Framework — Setup&Config
- Business Logic
  - Java EE Session Beans
  - Entity Beans
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database — Setup&Config

When developing this application based on the business model, many development tasks are

generic or

repetitive

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application — Setup&Config
  - Java Script UI & Validation — Derive from business model
  - Struts UI Framework — Setup&Config
- Business Logic
  - Java EE Session Beans — Derive from business model
  - Entity Beans
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database — Setup&Config

When developing this application based on the business model, many development tasks are

generic or

repetitive

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application — Setup&Config
  - Java Script UI & Validation — Derive from business model
  - Struts UI Framework — Setup&Config
- Business Logic
  - Java EE Session Beans — Derive from business model
  - Entity Beans — Derive from business model
  - Pricing rules
- Database
  - Hibernate OR Mapping
  - MySQL Database — Setup&Config

When developing this application based on the business model, many development tasks are

generic or

repetitive

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application — Setup&Config
  - Java Script UI & Validation — Derive from business model
  - Struts UI Framework — Setup&Config
- Business Logic
  - Java EE Session Beans — Derive from business model
  - Entity Beans — Derive from business model
  - Pricing rules
- Database
  - Hibernate OR Mapping — Derive from business model
  - MySQL Database — Setup&Config

When developing this application based on the business model, many development tasks are

generic or

repetitive

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application — Setup&Config
  - Java Script UI & Validation — Derive from business model
  - Struts UI Framework — Setup&Config
- Business Logic
  - Java EE Session Beans — Derive from business model
  - Entity Beans — Derive from business model
  - Pricing rules
- Database
  - Hibernate OR Mapping — Derive from business model
  - MySQL Database — Derive DB-Schema from — Setup&Config

When developing this application based on the business model, many development tasks are

generic or

repetitive

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application — Setup&Config
  - Java Script UI & Validation — Derive from business model
  - Struts UI Framework — Setup&Config
- Business Logic
  - Java EE Session Beans — Derive from business model
  - Entity Beans — Derive from business model
  - Pricing rules
- Database
  - Hibernate OR Mapping — Derive from business model
  - MySQL Database — Setup&Config — Derive DB-Schema from business model

When developing this application based on the business model, many development tasks are

generic or

repetitive

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Frontend
  - Apache Tomcat Web Application — Setup&Config
  - Java Script UI & Validation — Derive from business model
  - Struts UI Framework — Setup&Config
- Business Logic
  - Java EE Session Beans — Derive from business model
  - Entity Beans — Derive from business model
  - Pricing rules
- Database
  - Hibernate OR Mapping — Derive from business model
  - MySQL Database — Setup&Config — Derive DB-Schema from business model

When developing this application based on the business model, many development tasks are

generic or

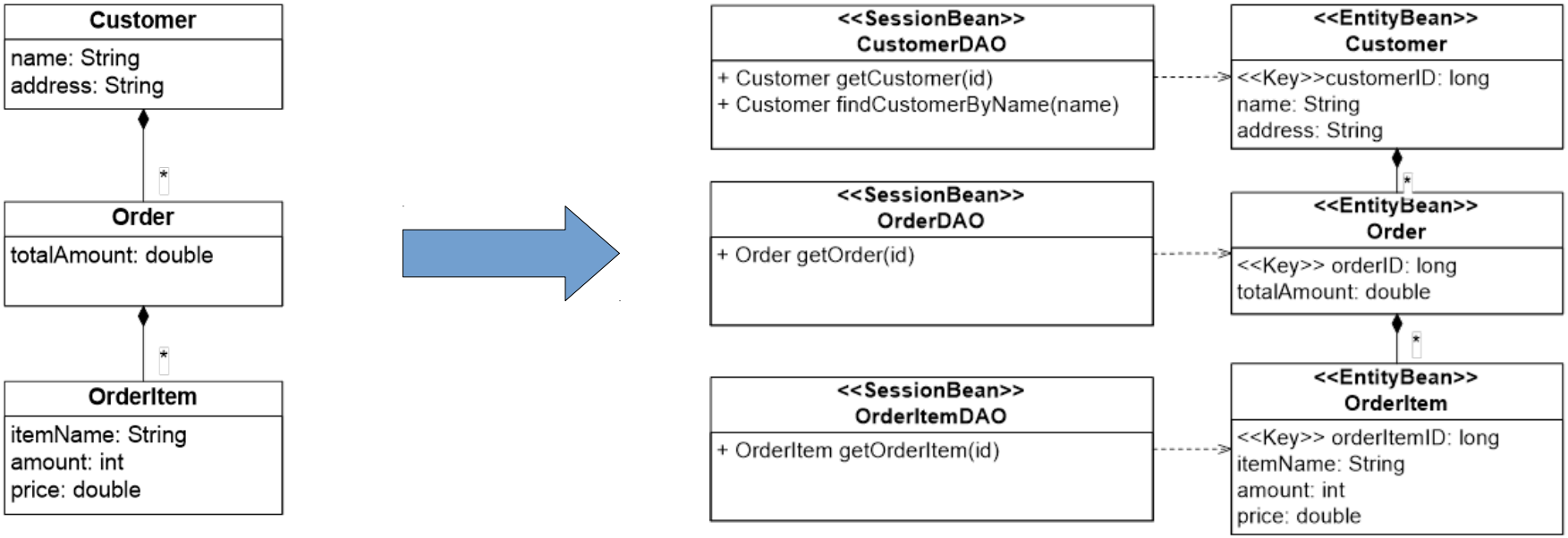
repetitive

Some individual tasks remain

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

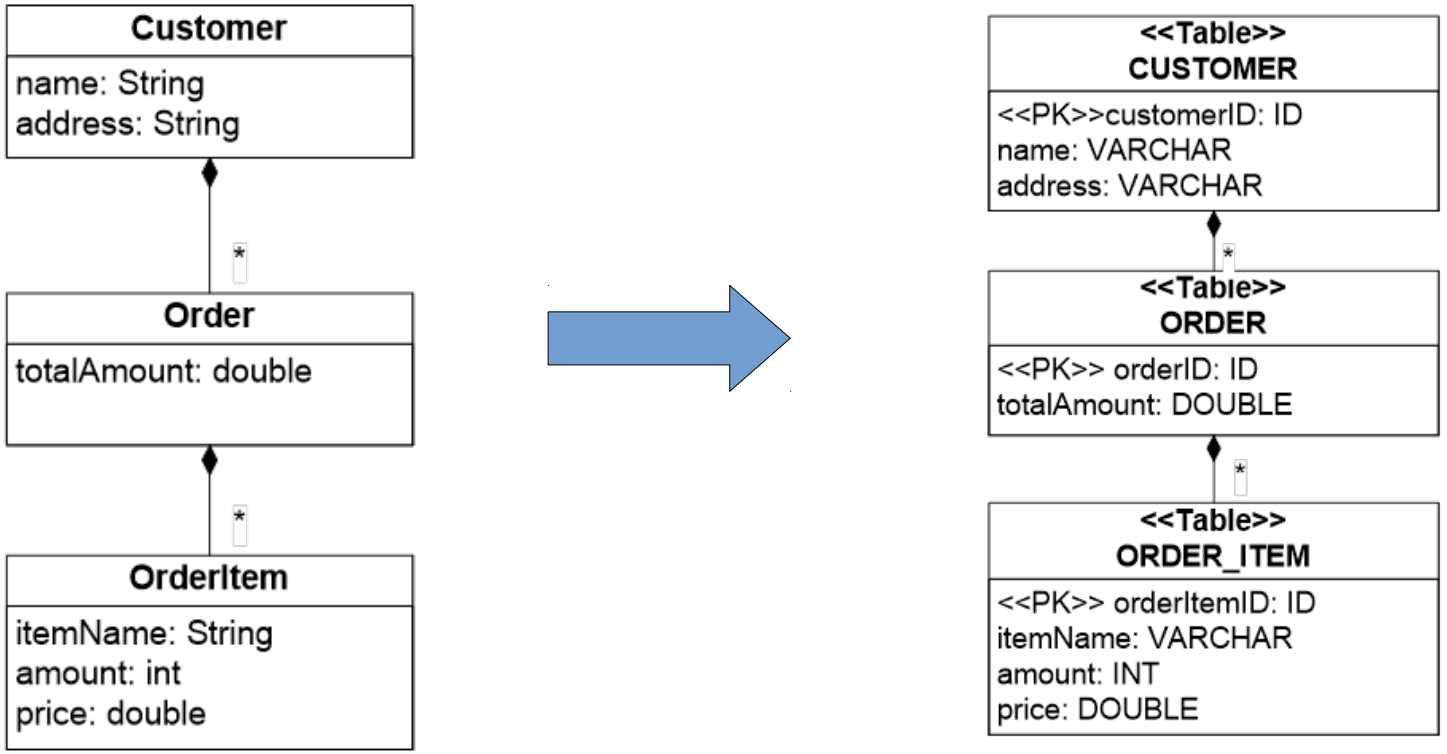
- Approach: Transform the platform-independent business model into different platform-specific models,
- For example: Generating a Bean model



inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Also generate the database schema:



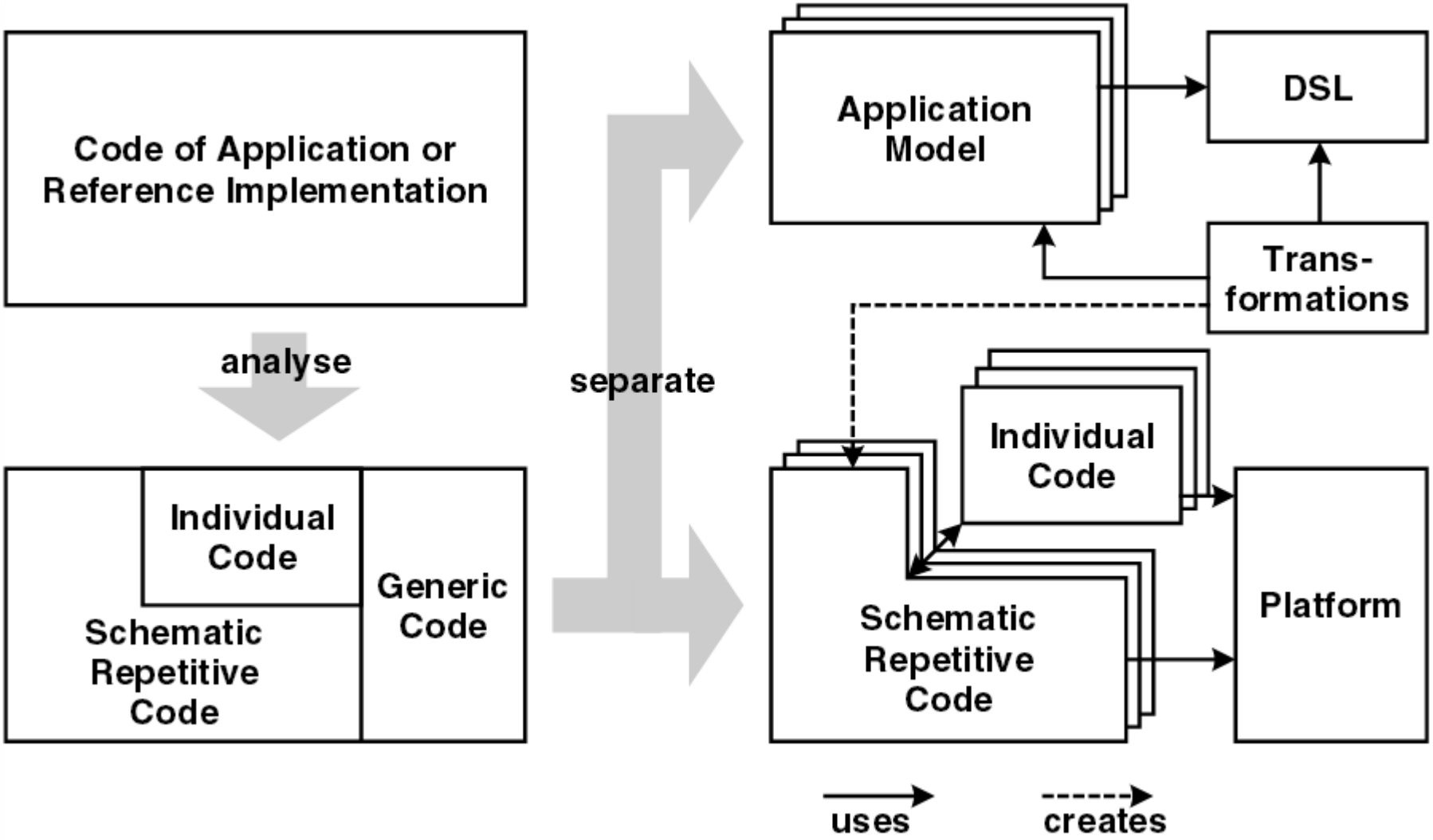
inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

Example: MDSD for a Three-Tier Web Application

- Furthermore, generate forms, validation rules, error messages, object-relational mapping rules, configuration files, ...

inspired by Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006. and a lecture on Model-Driven Software Development by Prof. Steffen Becker, 2011.

The idea of MBSE/MDSD in a bit more detail



from Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006.

Agenda / Summary and Outlook

1. Motivation ✓
2. What you will learn in this lecture ✓
   - Learning Objectives
   - Topics
   - Literature
3. Organizational Issues ✓
4. Introduction: ✓
   - What is a model?
   - What is model-based software engineering (MBSE)?
   - What are reasons for doing MBSE?

What's next: Meta-modeling – defining modeling languages