Software Engineering and Scientific Computing

Barbara Paech, Hanna Valtokari

Institute of Computer Science

Im Neuenheimer Feld 326

69120 Heidelberg, Germany

http://se.ifi.uni-heidelberg.de

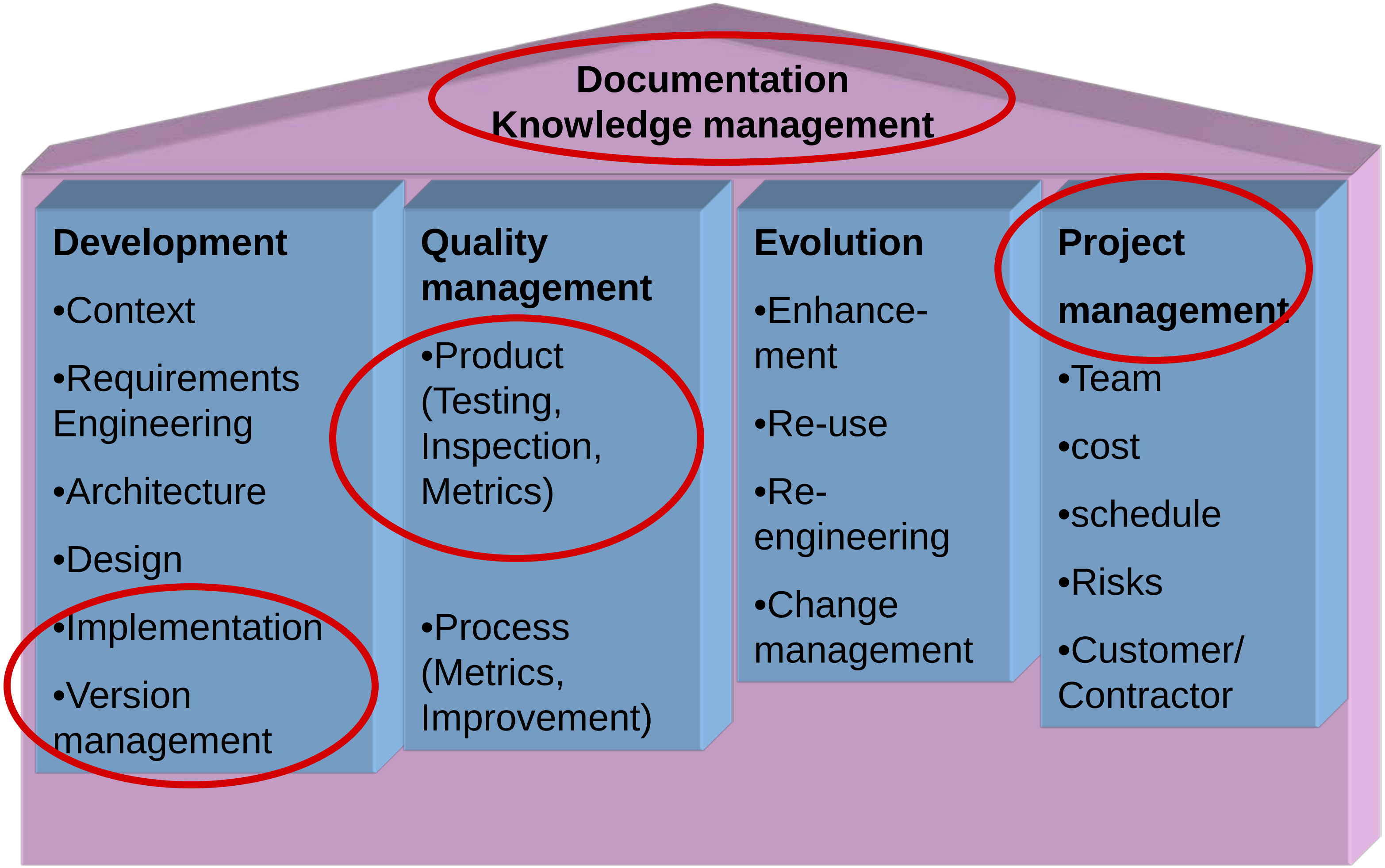
paech@informatik.uni-heidelberg.de

RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Schedule Third Day (Thursday)

9:00	Modeling Knowledge Management
10:30	Break
11:00	Best practices
12:00	Lunch
13:00 Incl. a short break	Tools, Exercises Branches and Tagging in Subversion IDE Wrap-Up, Feedback
16.00	End

In this course: Programming in a small team



Documentation
Knowledge management

Development

- Context
- Requirements Engineering
- Architecture
- Design
- Implementation
- Version management

Quality management

- Product (Testing, Inspection, Metrics)
- Process (Metrics, Improvement)

Evolution

- Enhance-ment
- Re-use
- Re-engineering
- Change management

Project management

- Team
- cost
- schedule
- Risks
- Customer/ Contractor

Programming in a small team

What is
Ron doing?

Project management
Issue Tracking

I want to explain
my ideas to Hermione

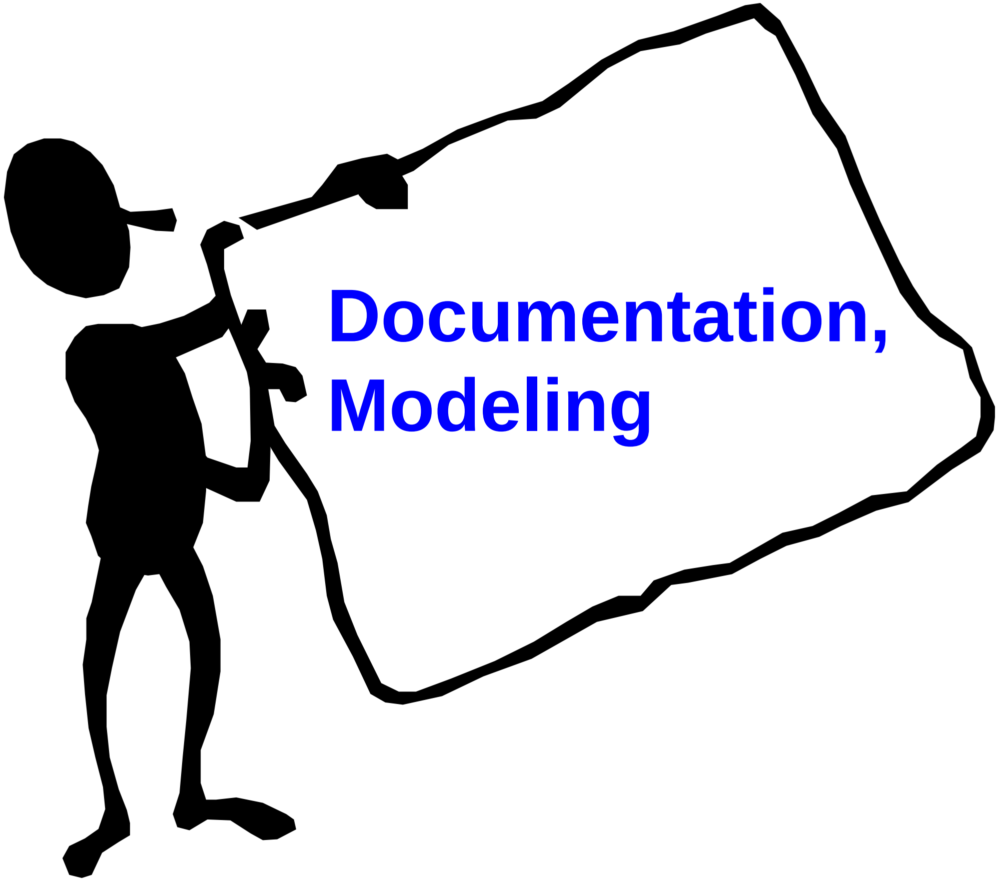
Modeling
Knowledge Management

I want to change
Ginnys code

Version management,
Build management

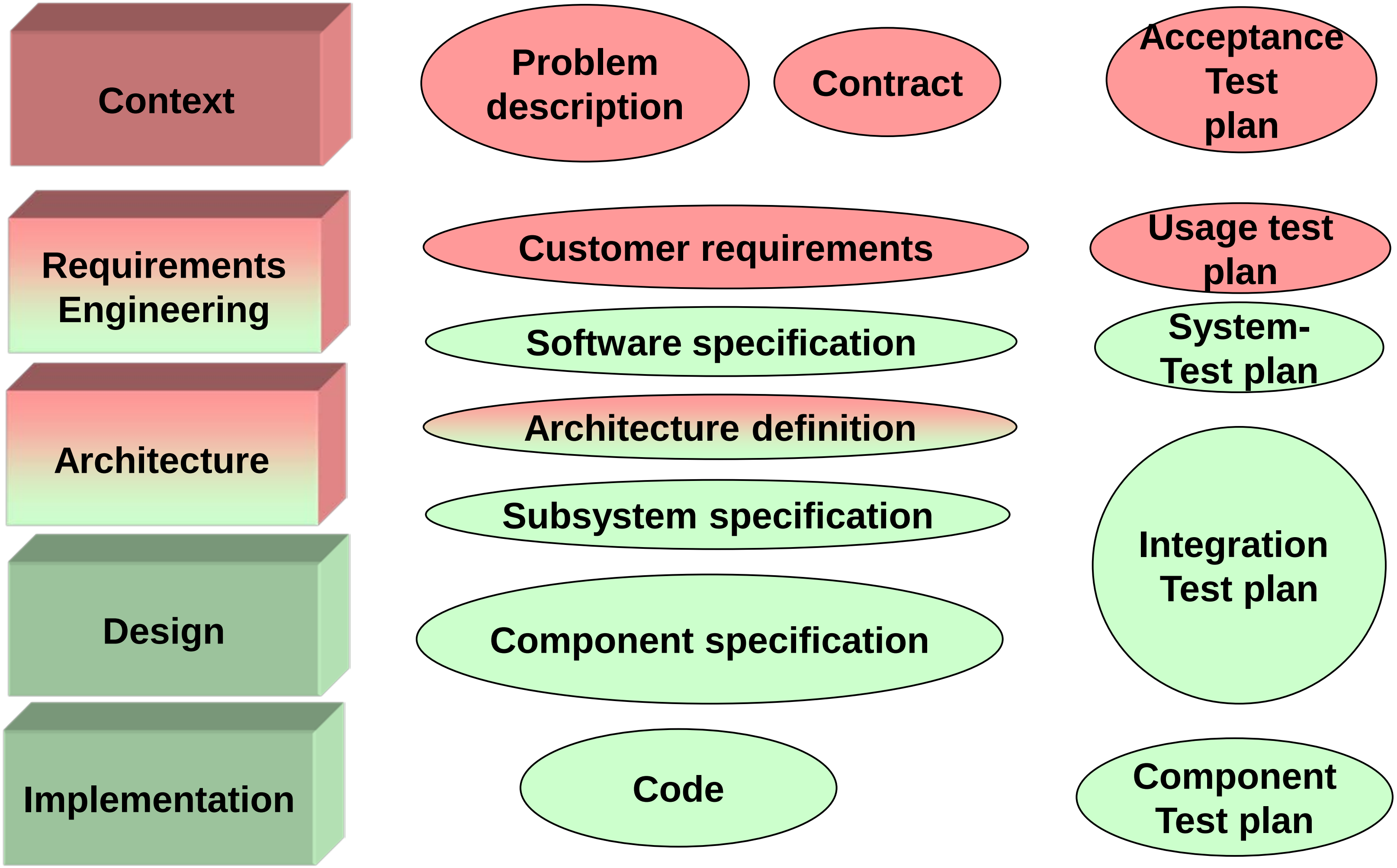
I want to check
Harrys changes

Quality assurance
Testing



Documentation, Modeling

Documentation



Context

Problem description

Contract

Acceptance Test plan

Requirements Engineering

Customer requirements

Usage test plan

Software specification

System-Test plan

Architecture

Architecture definition

Subsystem specification

Integration Test plan

Design

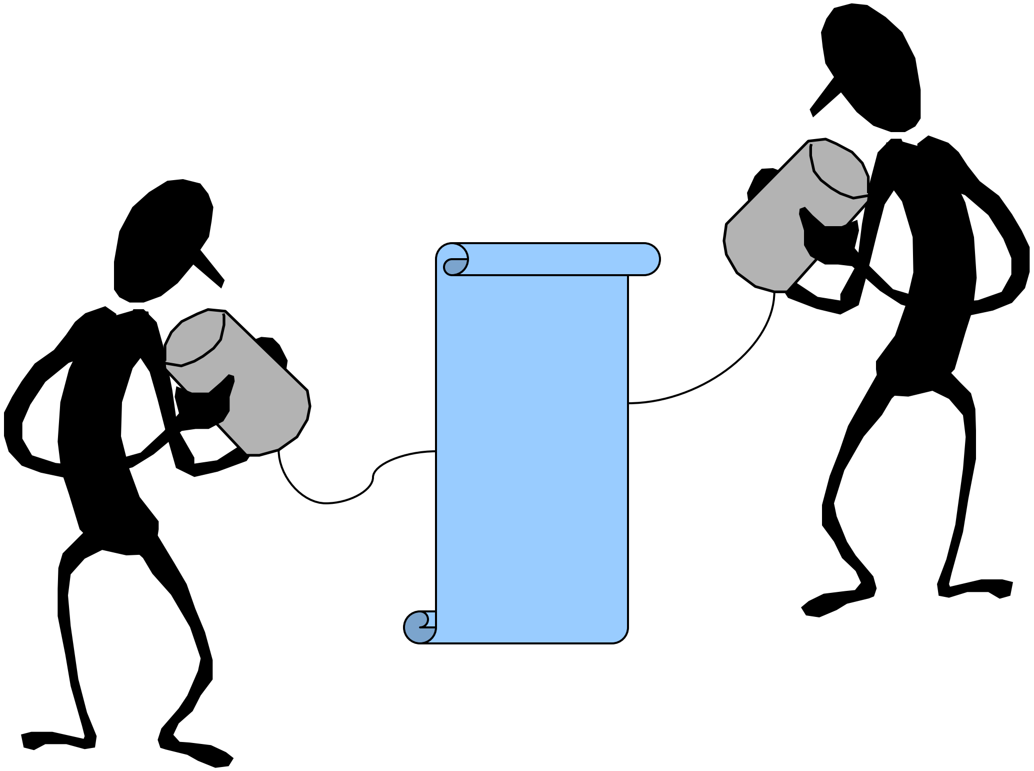
Component specification

Implementation

Code

Component Test plan

Documents needed for Communication



- Important if
  - **Long-time use of the SW**
  - **System complex**
  - **Many project participants**
  - **Decisions have legal relevance**
- Software documents contain typically text and models

Typical Document Structure

1. Introduction

1.1 Purpose

- Who has created the document how?
- Who should read the document why??
- Whoy has to adhere to the content?

1.3 Definitions and Abbreviations

- Including glossary

1.4 References, Standards and Directives

1.5 Overview

- Content and Structure of the document

2. – X. Main Content

X+1. Conclusion

X+2. Appendices

Example Requirements Document

2. Product Context

2.1 Purpose of the Product

Business goals

2.2 Stakeholder

Everyone who is participating in, interested in, affected by the product

2.3 Context Processes

Business processes which involve the product

3. System Requirements

3.1 Main Features

3.2 Architectural constraints

3.3 Functional requirements

Use cases, system functions, GUI

3.4 Non-functional requirements

4. Project requirements

4.1. Assumptions and Dependencies (incl. Risks)

4.2. Product Acceptance

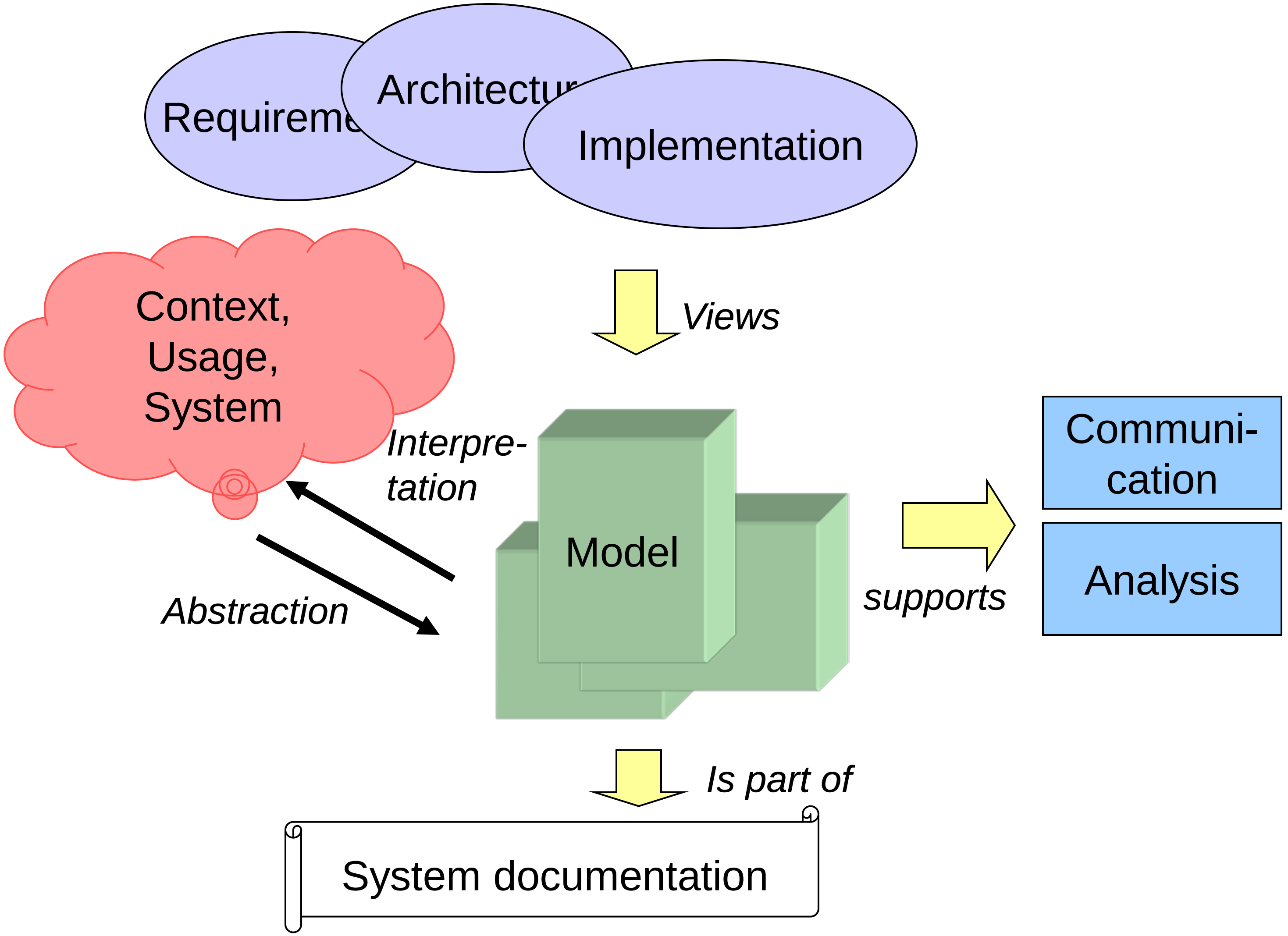
How much and how much detail?

- Depends on the risk, if the readers of the documents do not find the necessary information
- … if it is difficult for the readers to ask the authors personally
  - E.g. distributed development
- … if there is a high risk involved when the software has bugs
  - E.g. life-critical software
- … if there is a high probability of later changes
  - E.g. context changes (new business processes)

Good Documents should be …

- …Understandable for all readers
  - Non-ambiguous, complete, consistent
- …Understandable for all readers relying on the information, e.g. for requirements documents
  - Customer:
    - Correct: are these really the customer requirements?
  - Requirements engineering: assess progress, ensure consistent change
    - Prioritized, traceable, changeable?
  - Developer:
    - Realizable?
  - Tester:
    - Can test cases be derived?

Models in software development

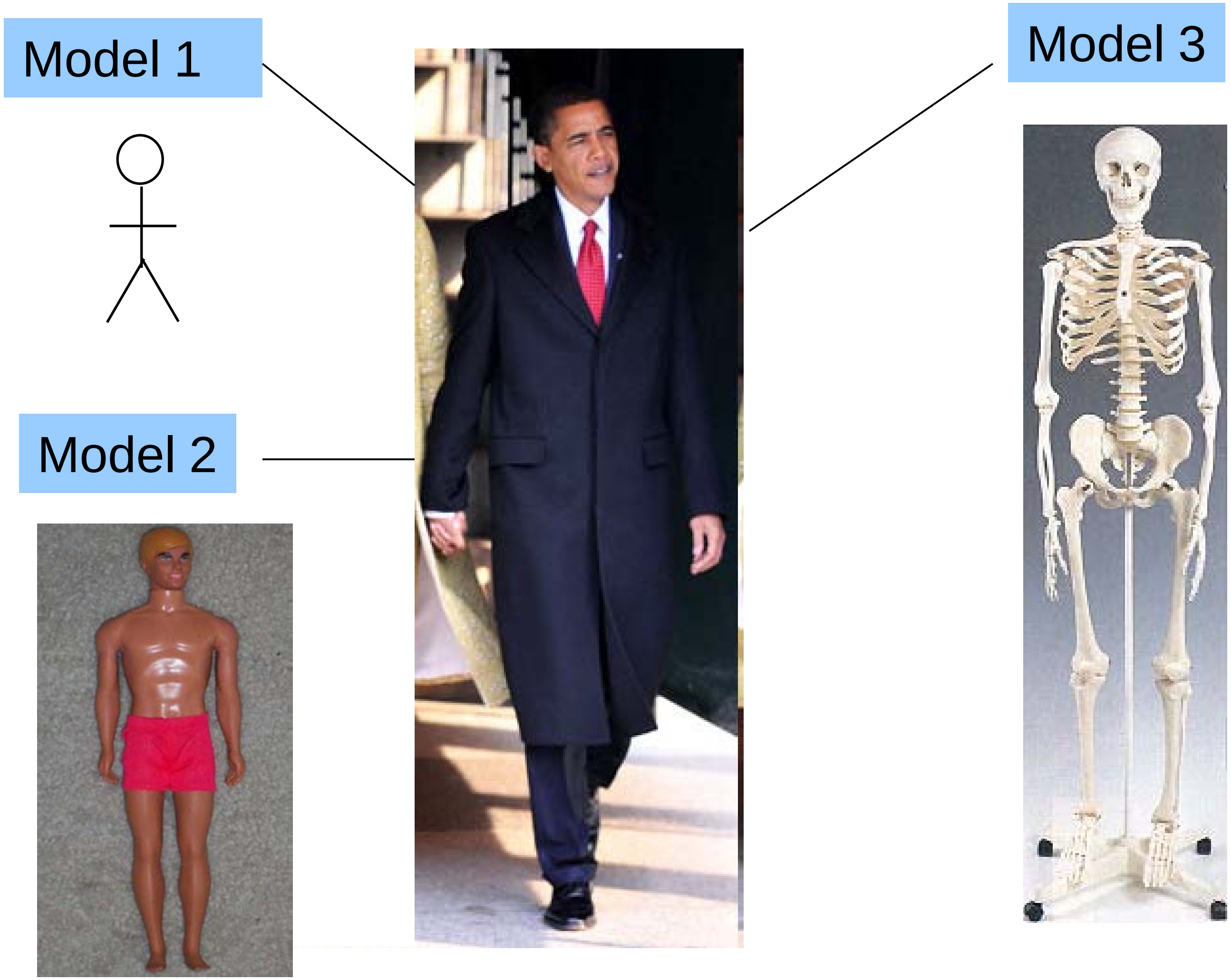


Characteristics of a model

- A model is essentially a representation of an original, which is reduced in size or abstracts from details [Stachowiak 73]
- A model is a system abstraction with the purpose to support thinking about the system (leaving out details) [Brügge 00].
- 3 important model characteristics
  - mapping (there is an original)
  - reduction (original is not represented completely)
  - pragmatic (model should be used instead of the original within in a specific context, purpose)

- Note: model can be pre-scriptive or de-scriptive

Example model



Model 1

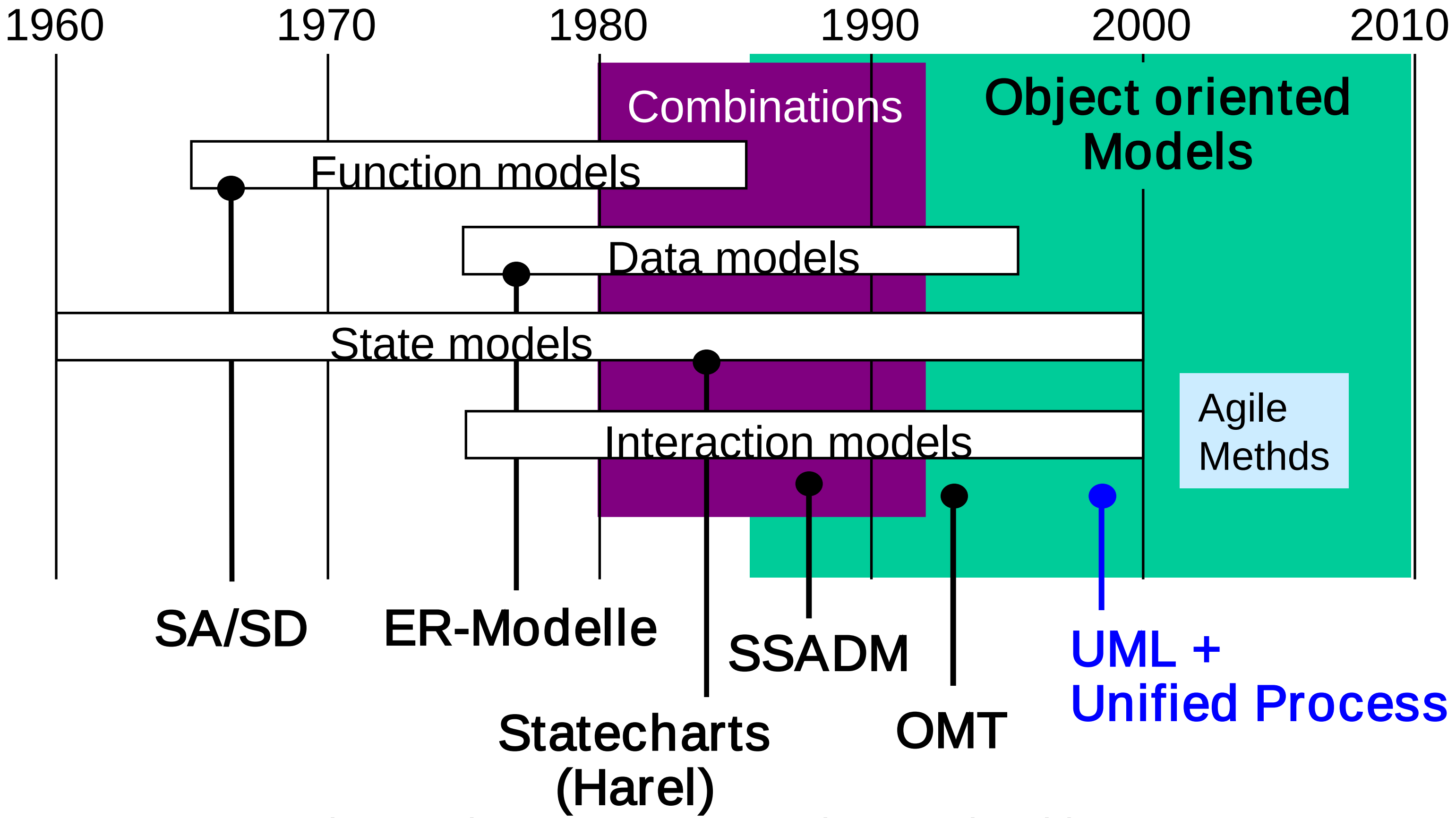
Model 2

Model 3

Notation (Language) for a model

- Everything you need to know to create and use models
- Syntax
  - Symbols allowed
  - Abstract Syntax (main concepts)
  - Well-formedness conditions
- Semantic
  - How to interpret the model
- Pragmatics (usage methodology)
  - Techniques for analysis (e.g. type checking, consistency checks)
  - Techniques for simulation
  - Techniques for transformation (e.g. Refactoring)
  - Techniques for generation

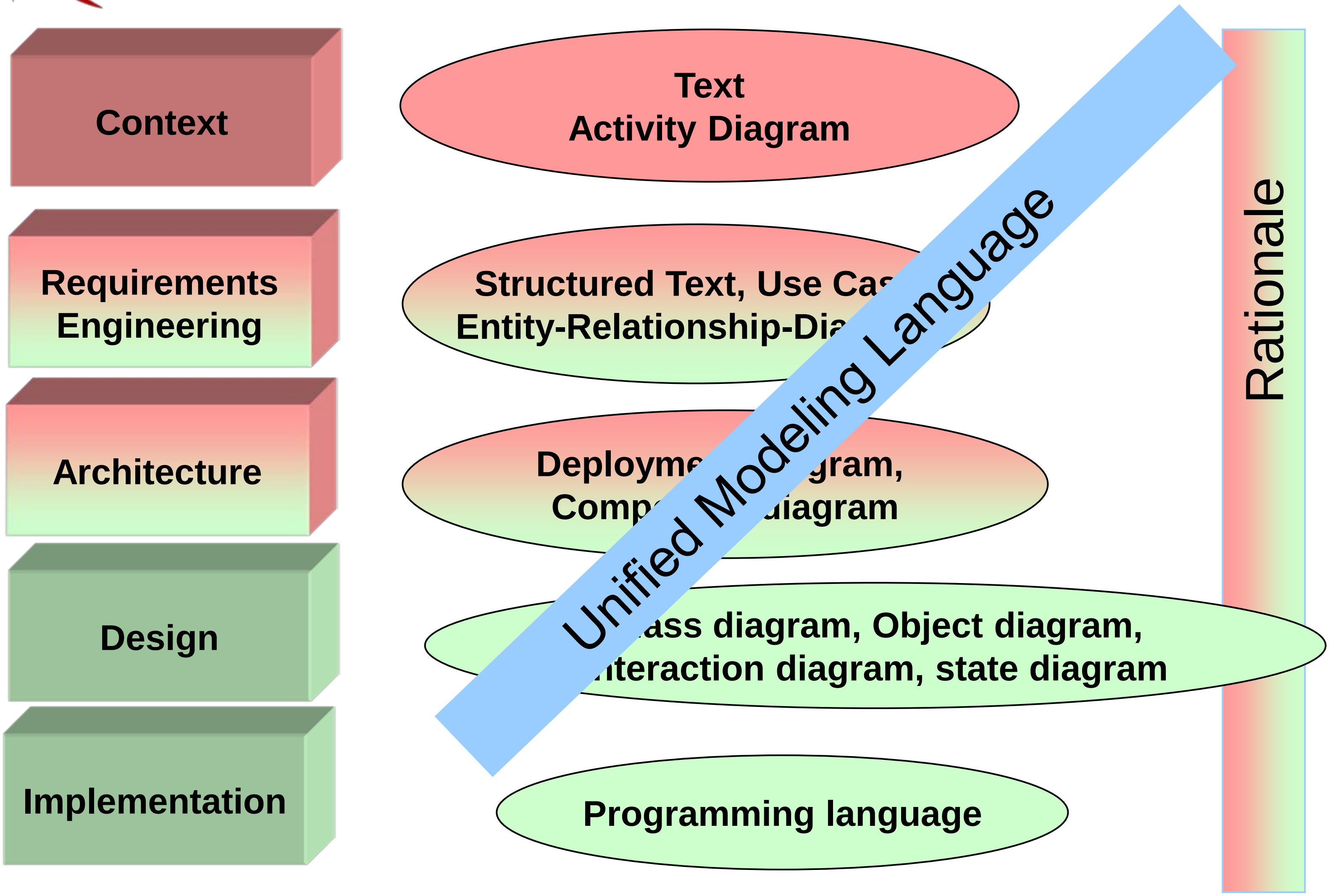
Modeling techniques



SA = Structured Analysis
SD = Structured Design
SSADM = Structured Systems Analysis and Design Method

ER = Entity-Relationship
OMT = Object Modeling Technique

Modeling



Context

Requirements Engineering

Architecture

Design

Implementation

Text
Activity Diagram

Structured Text, Use Cas
Entity-Relationship-Dia

Deployme gram,
Comp iagram

ass diagram, Object diagram,
nteraction diagram, state diagram

Programming language

Unified Modeling Language

Rationale

Unified Modeling Language (UML)

- Since 1997 De-Facto-Standard in industry (www.uml.org)
- Mainly for object-oriented development
- defines
  - Structure diagrams (System statics)
  - Behaviour diagrams (System dynamics)

Structure diagrams UML

- Design
  - Class diagram
  - Object diagram (complex situations)
  - Package diagram (sets of classes)

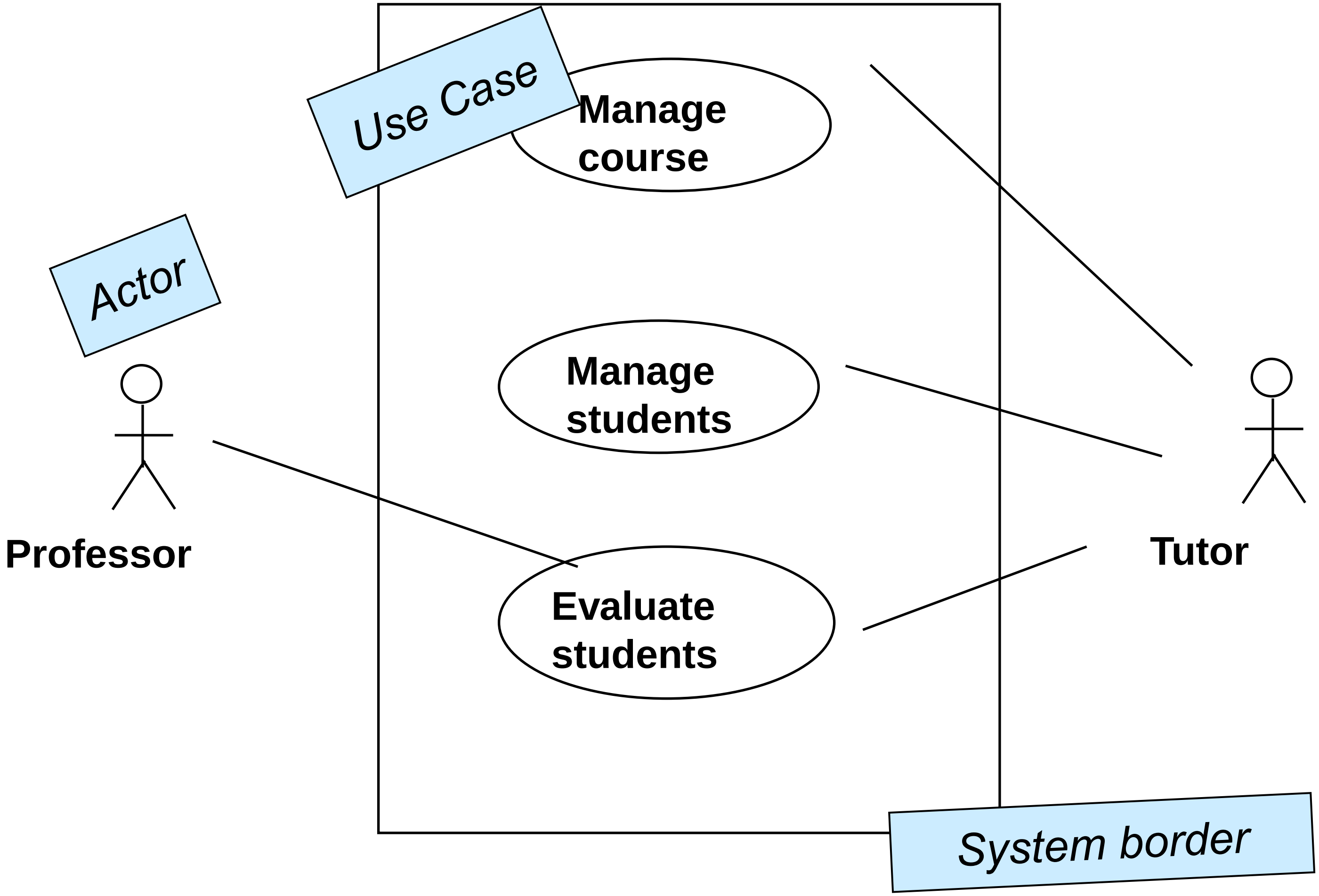
- Architecture
  - Composition diagram (internal and external interface)
  - (logical) Component diagrams
  - Deployment diagram (physical components distributed)

Behaviour diagrams UML

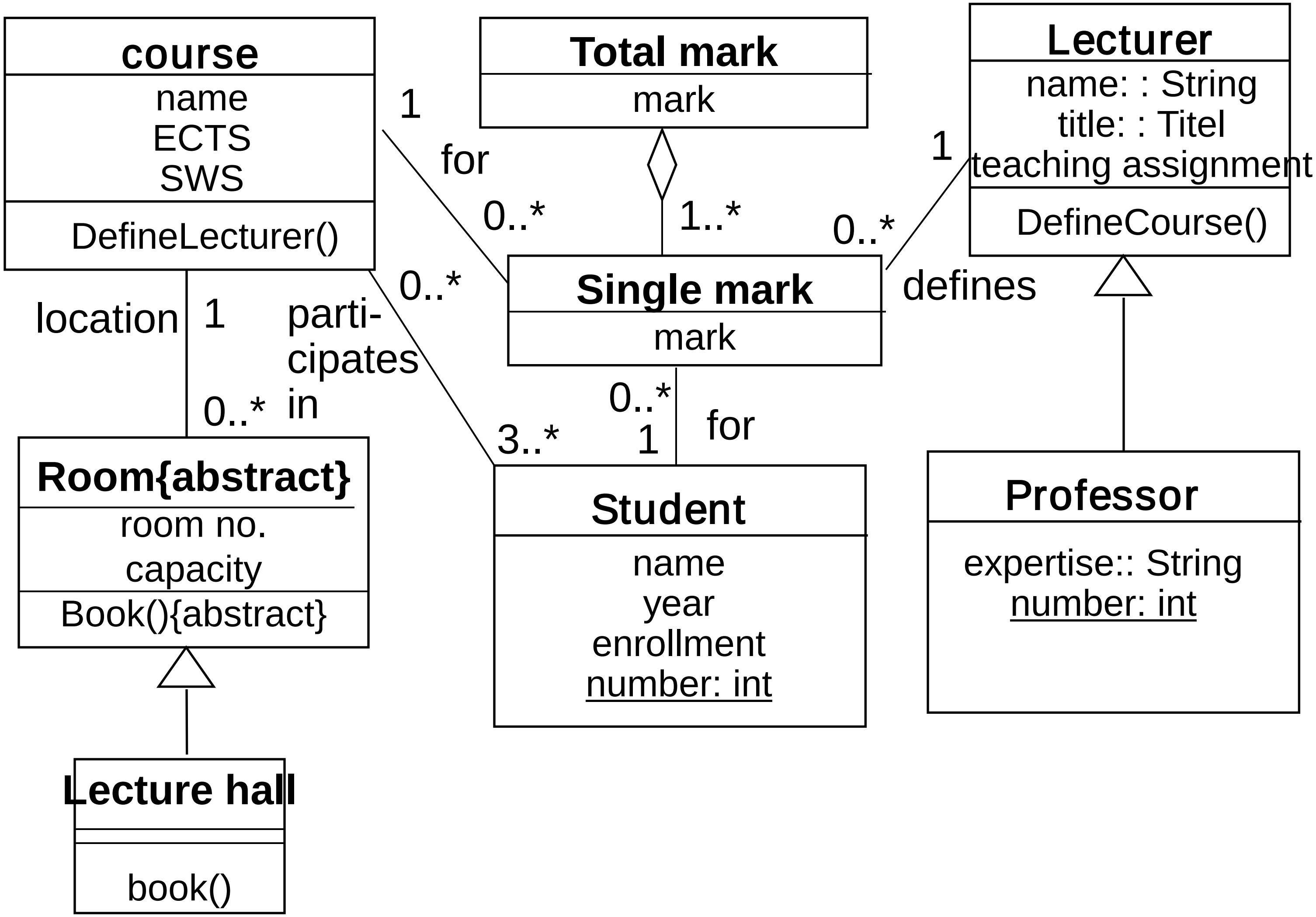
- Processes
  - Use Case Diagrams (overview of system functionsality)
  - Activity diagrams (sequences of activities)
  - State diagrams (sequences of states)

- Interaction
  - Sequence diagram (sequences of messages)
  - Communication diagrams (focus on one component)
  - Timing diagrams (Communication between automata)
  - Interaction overview diagram

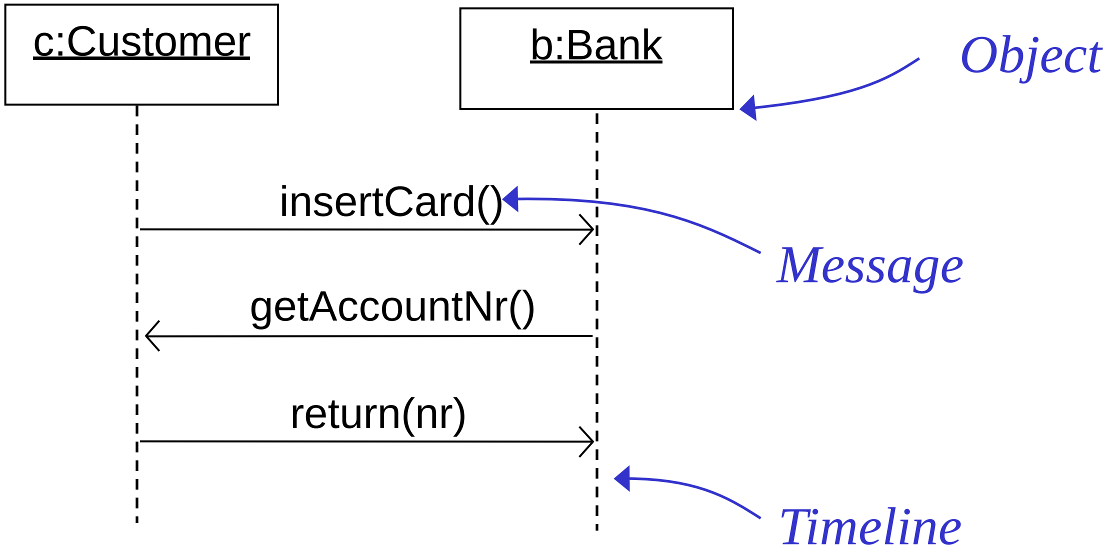
Example Use Case Diagram



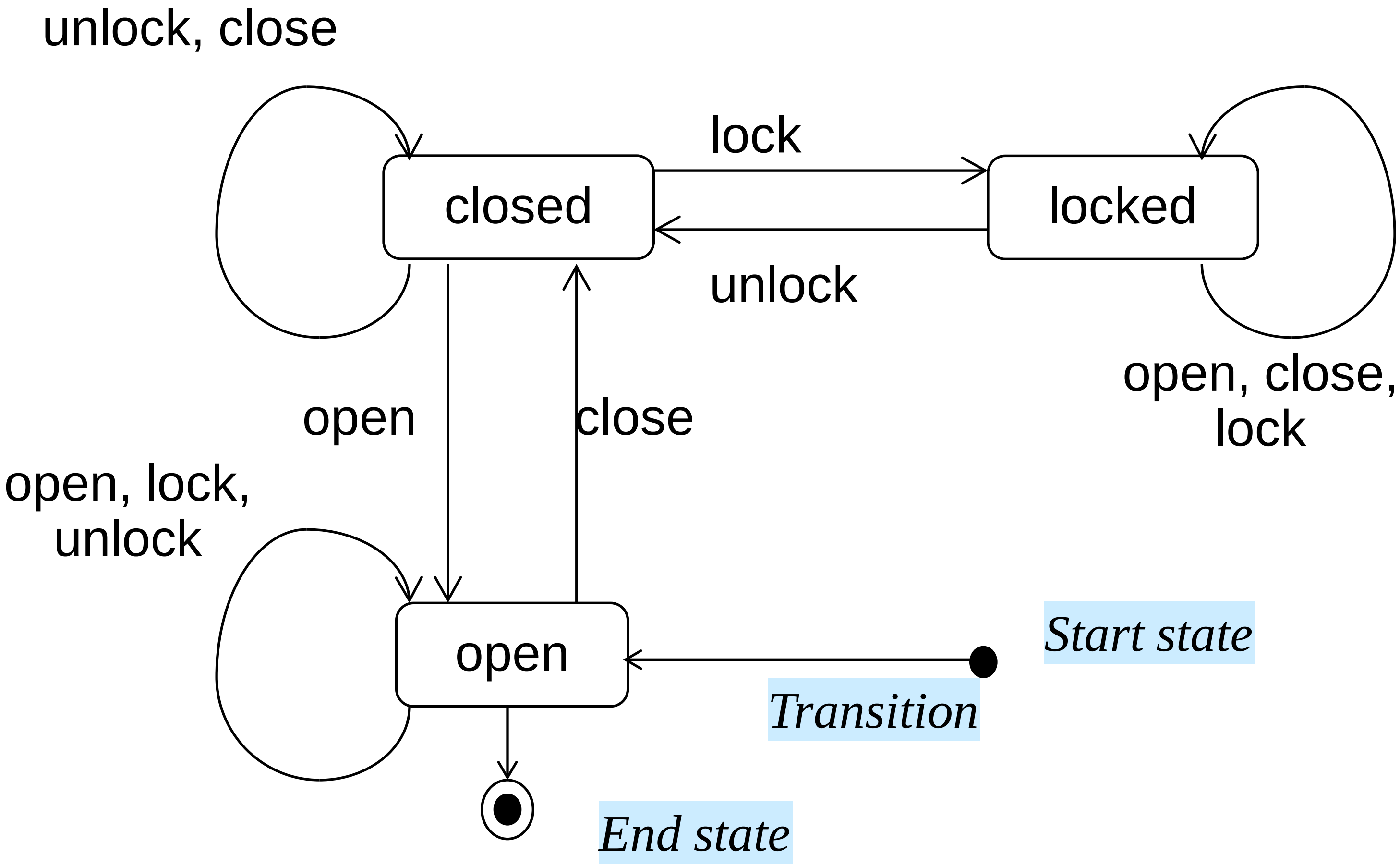
Example class diagram



Example Sequence diagram



Example State diagram

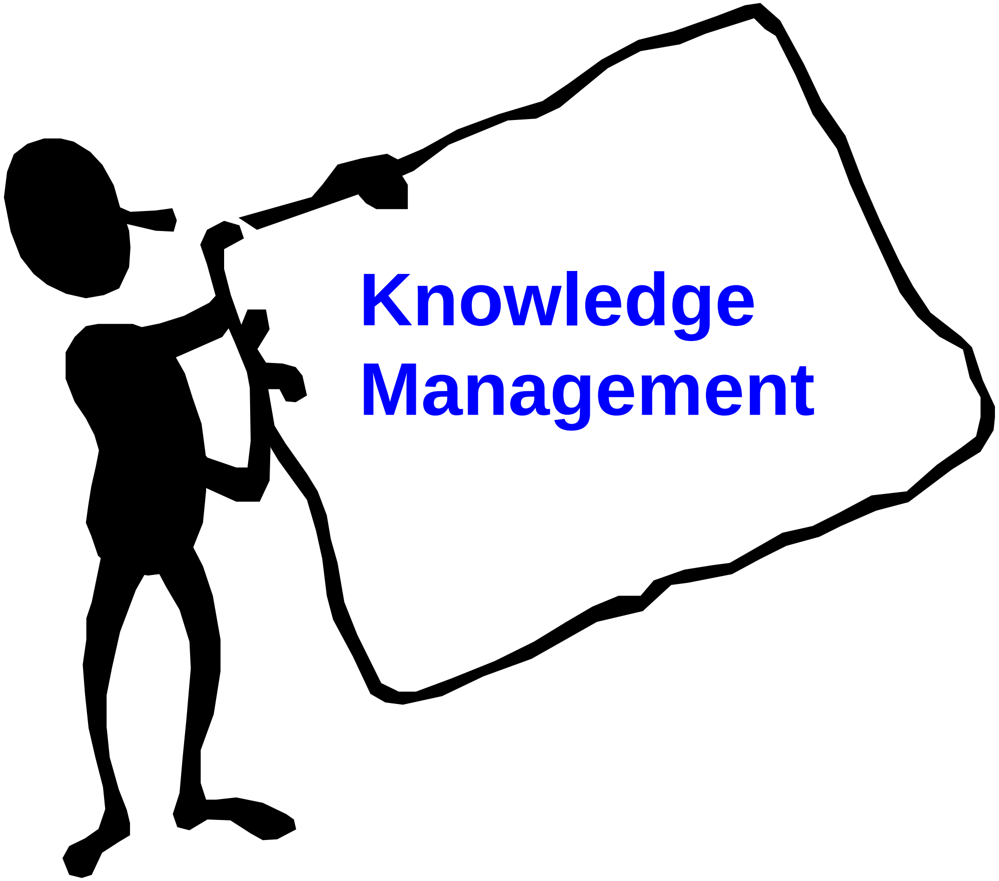


- What documents do you create or use?
- What documents could you create or use?

- What modeling techniques do you use
- What modeling techniques could you use?

Literature

- Ch. Rupp, S. Queins, B. Zengler, „UML 2 glasklar“, Hanser Verlag 2007



Knowledge Management

Motivation Knowledge Management

Why this one ?

- Documents mostly contain the final decision
- Discarded options and criteria are not documented
- => Decisions
  - Do not reflect all important criteria
  - Are not convincing for others
  - Get overthrown (and dead ends are entered again)

Software Engineering Knowledge

	Knowledge about the system	Knowledge about the process (Roles, Activities, Dokuments)
Product know-ledge	**Content:** Specification, Design, Code, Test plan, etc **Rationale:** Product Goals, Criteria, Options, Assessments	**Content:** Project plan, Cost plan, Tasks, Guidelines **Rationale:** Project goals, Risk assessment, Criteria, Options
Organi-sational Know-ledge	**Content:** Domain model, System architecture, Design pattern **Rationale:** on the level of generalized models (e.g. Forces for patterns)	**Content:** Process model, Best Practices, Experiences **Rationale:** on the level of generalized models (e.g. sucess factors for best practices)

Example rationale as a table

Criteria

Options

	flexibility	Low cost	security
PIN / account number	--	++	--
Card / PIN	+	+	-
Body characteristics	++	--	++

Evaluations

- Which knowledge do you capture?
- Which knowledge could you capture?