Software Engineering

Lesson 01
Software Development Processes
v1.1

Uwe Gühl

Fall 2007/ 2008

Contents

- Software Development Process
- Classical Process Models for Software Engineering (SE)
  - Structured Analysis / Structured Design (SA/SD)
  - Data oriented Software Development
- Object Oriented Process Models for Object Oriented Software Engineering (OOSE)
  - Rational Unified Process (RUP)
  - Object Engineering Process (OEP)
- Agile Approaches
  - Extreme Programming (XP)
- More Process Models / Outlook
- Software Development Process Improvements (CMMI)
- Summary

Software Development Process
History of Abstraction

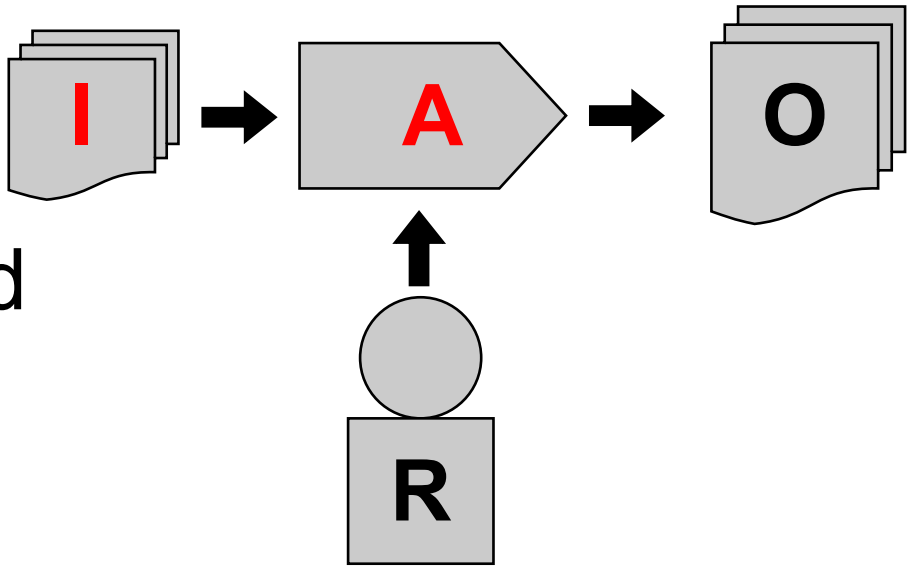
Abstraction from …	**... to …**
Voltage and electricity	Programming language
Command and Date	von-Neumann-Machine
Several bits:	Octal- and hexadecimal code
Command codes	Operator symbols
Memory addresses	Identifier
Machine operations	Expressions
Calculation rules and data	Functions
Command sequences	Procedures
special operations	Function / procedure parameter
Different characteristics	Generic units
Data implementation	Encapsulation
Type implementation	Abstract data types (ADT)
Similar operations	Overloading
Similar types	Inheritance

Software Development Process Thoughts

- Methods: How to do or make something? A series of steps taken to build software
- Processes: How to proceed? Requirements-Management, Architecture-driven, Test-driven
- Rules, Constraints: Programming guidelines, document guidelines, modelling language, ...
- IT-Project Management
  - Organization, Planning, Control, Reviews, Communication, Culture

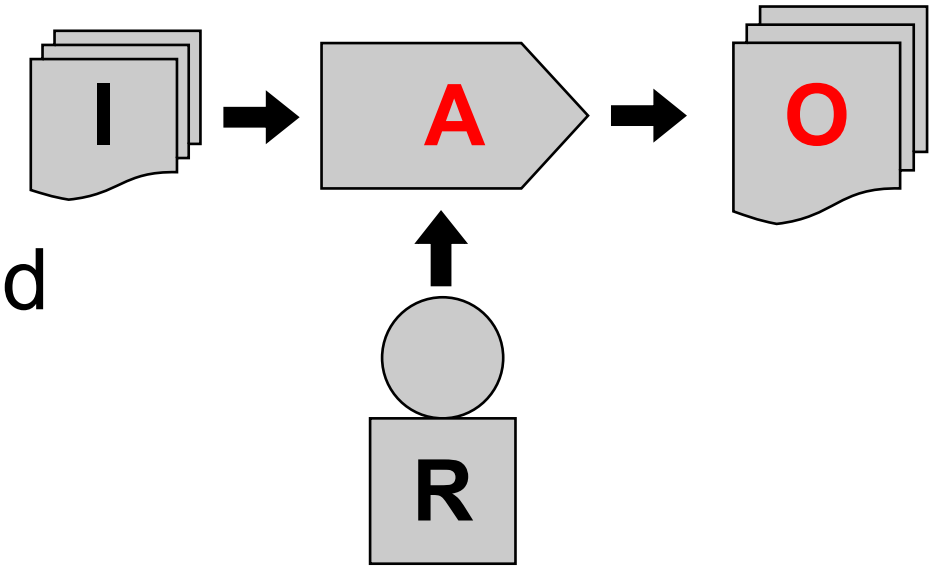
Software Development Process What is it?

- Structuring of Development into processes and activities



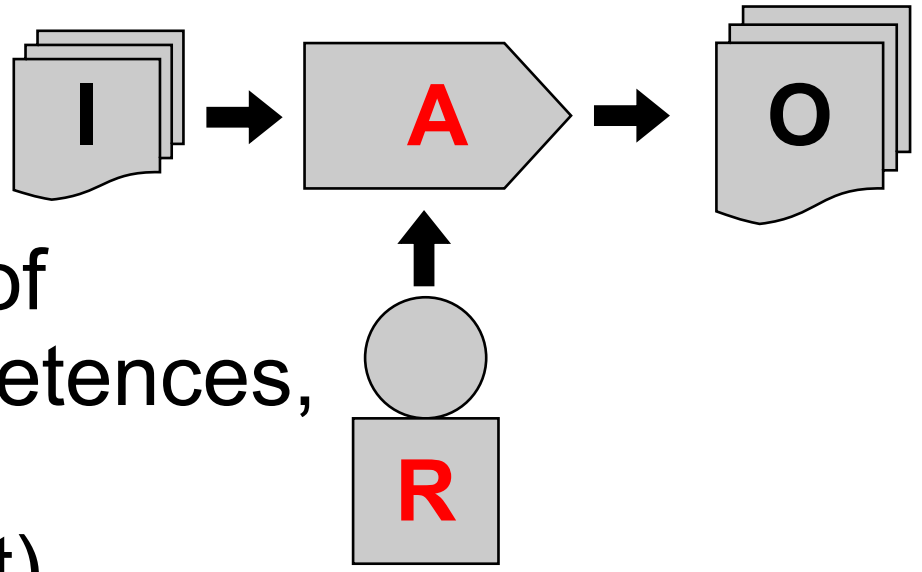
- **A**ctivities have defined
  - **I**nputs:
    - Events
    - Documents
    - Time

Software Development Process What is it?



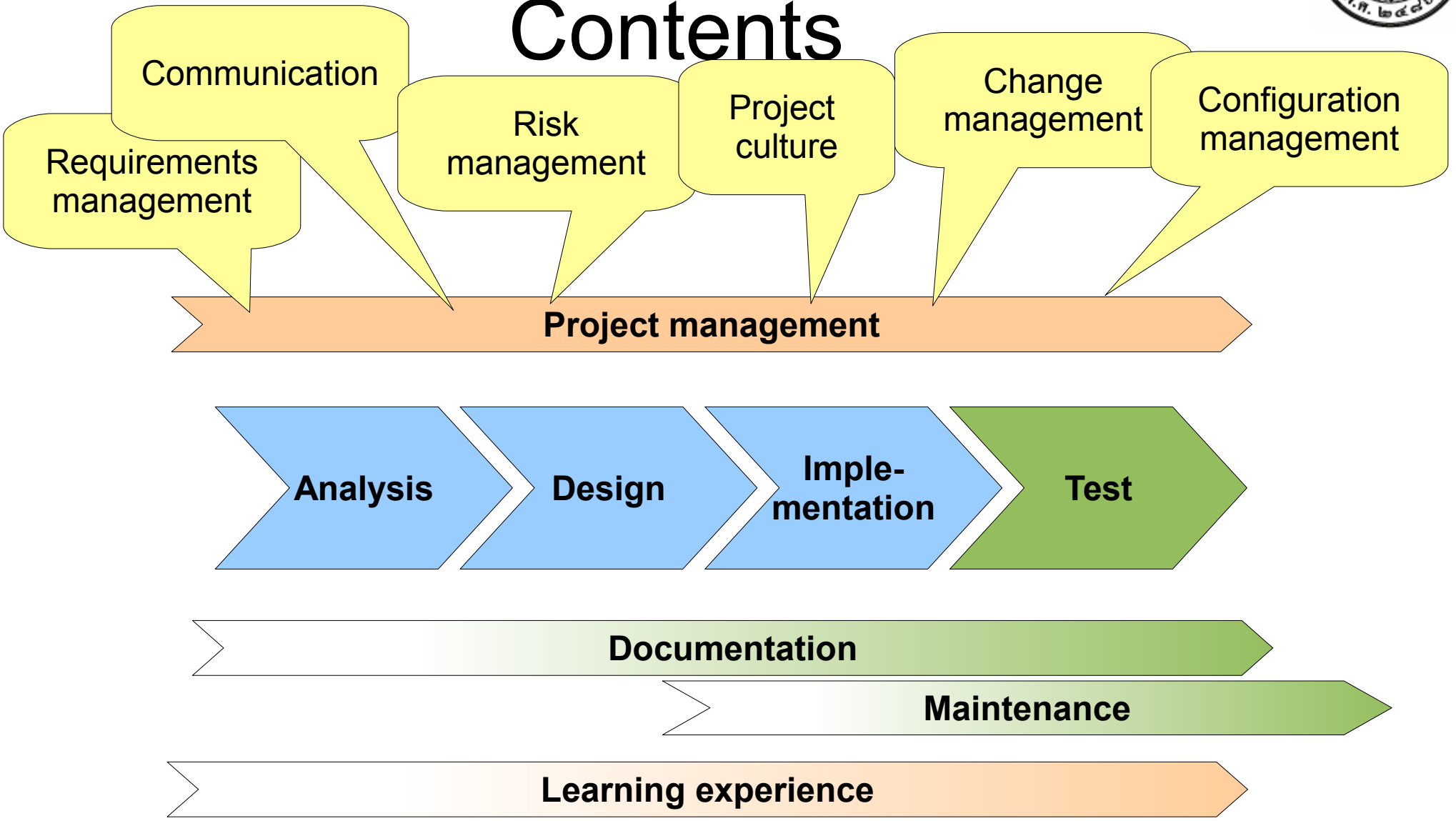
- **A**ctivities have defined
  - **O**utputs (Results):
    - Documents
    - Milestones
    - Trigger for other activities

Software Development Process What is it?

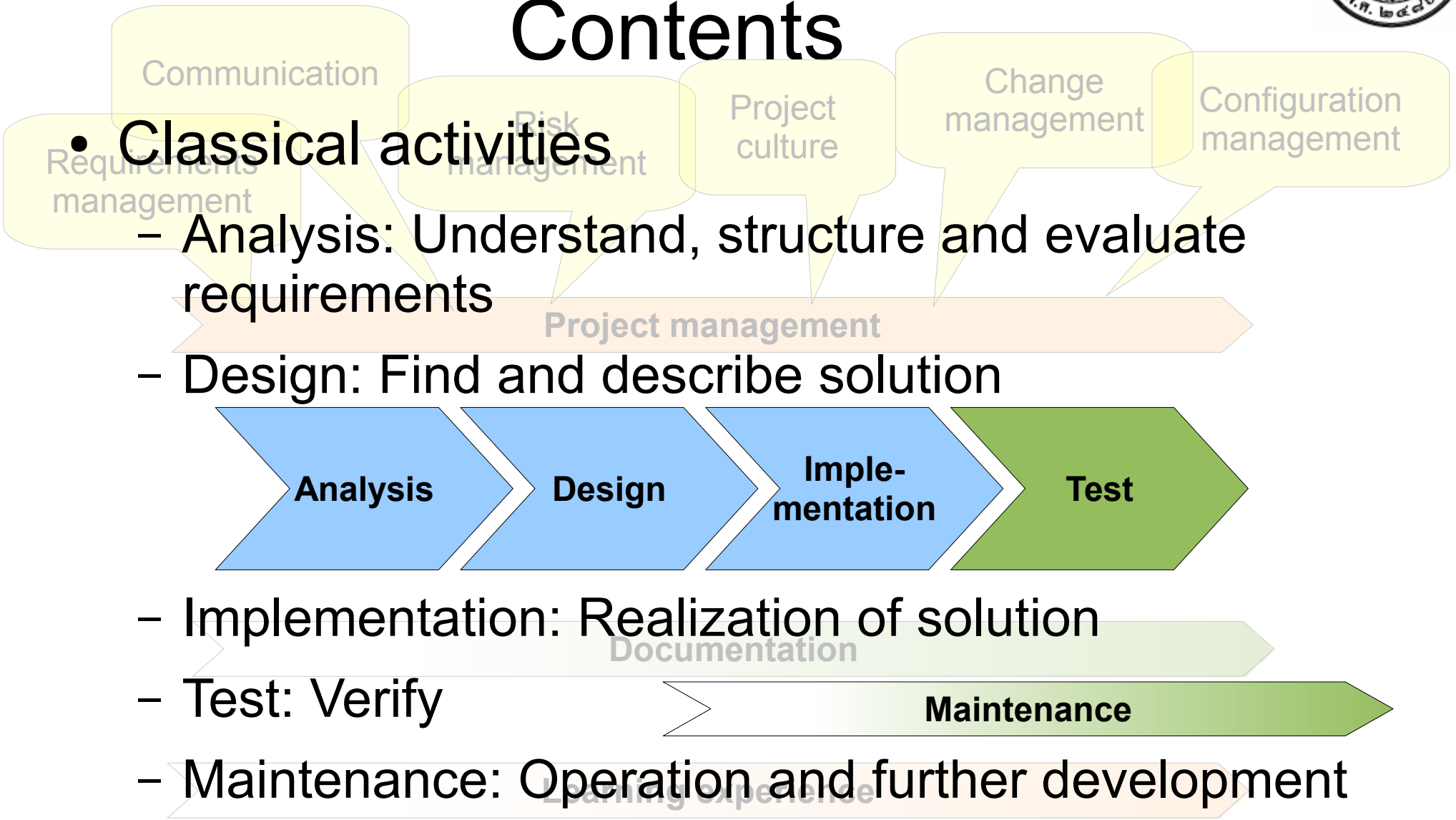


- **R**oles are definitions of responsibilities, competences, and capabilities
  (e. g. system architect)
  - Assigning activities to roles
  - Assigning persons to roles

Software Development Process Contents

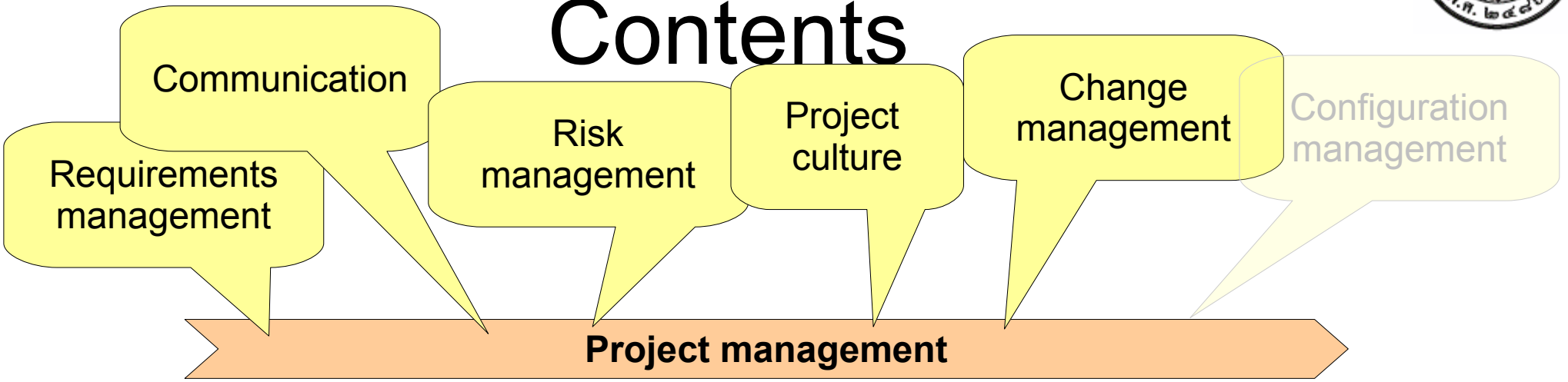


Software Development Process Contents



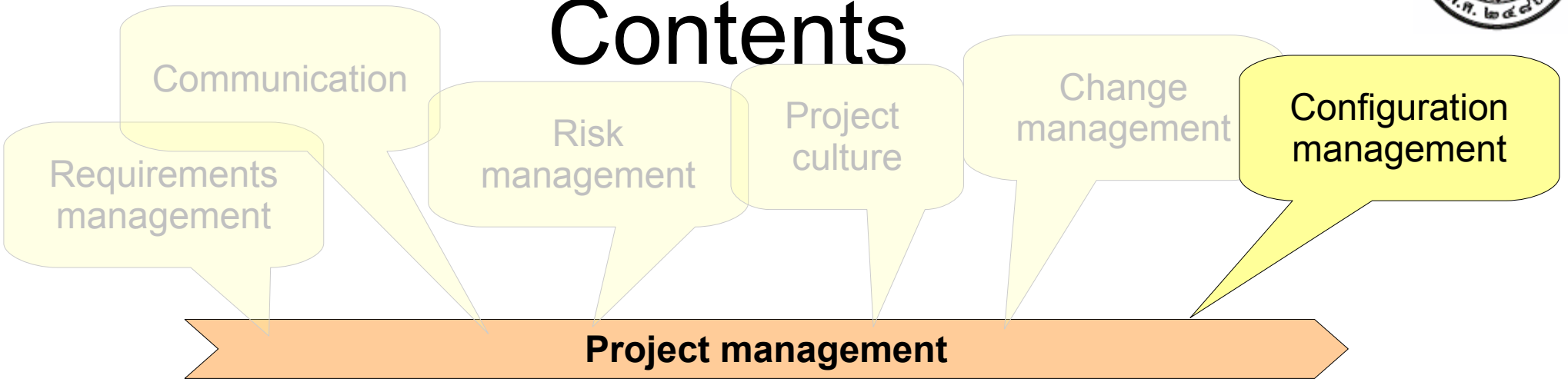
- Classical activities
  - Analysis: Understand, structure and evaluate requirements
  - Design: Find and describe solution
  - Implementation: Realization of solution
  - Test: Verify
  - Maintenance: Operation and further development

Software Development Process Contents



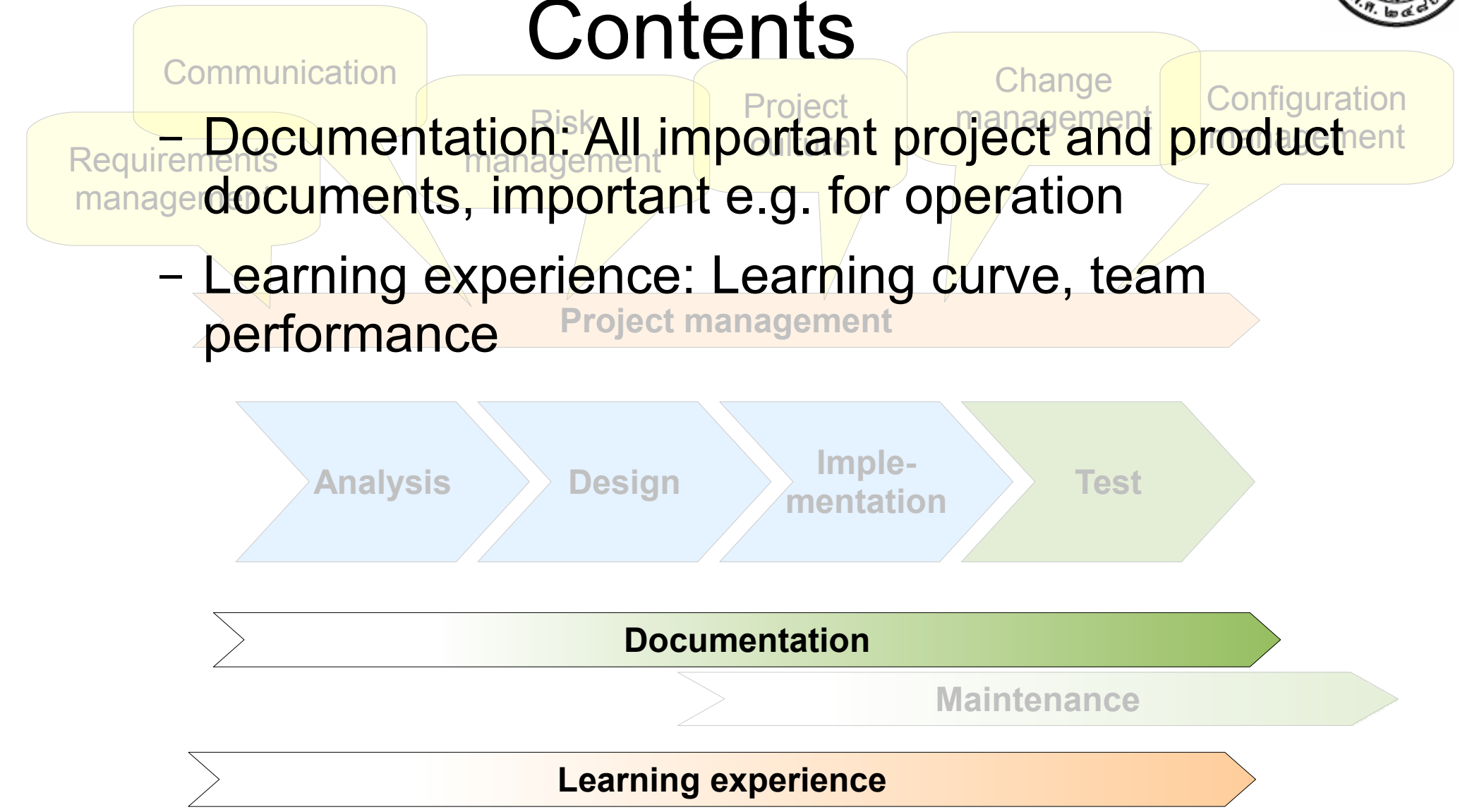
- Project management: Plan and control
- Communication: Communication model
- Risk management: Collect and evaluate risks
- Project culture: Project rules
- Requirements management: Collect, evaluate, and trigger requirements
- Change management: Deal with changes

Software Development Process Contents



- Configuration management
  - Tracking and managing of changes of all artefacts in the project (Sources, Requirements, Documentation, ...)
  - To be able to answer questions like
    - Which requirement was valid when?
    - Which requirement was realized in which release?
    - When was a specific requirement new, changed, realized?

Software Development Process Contents



- Documentation: All important project and product documents, important e.g. for operation
- Learning experience: Learning curve, team performance

Software Development Process Changes

- Paradigm shift
  - Stages not separated strictly, but flowing
  - Iterative and incremental approach
  - Concept of Round trip Engineering
- Focus is moving from applications and application organization to data modelling and the interacting with data
- Reason: Different life cycles [Hel94, p. 180]
  - Hardware: 2 to 4 years
  - Applications: 3 to 6 years
  - Models: 1 to 30 years

Software Development Process Why not try without?

- Why should we use a Software Development Process?
- Even today there are projects without a software developments process
- After clarification of requirements (more or less) one programs, tests, corrects until the software has an acceptable status.

Software Development Process Why not try without?

- **Pros**
  - No „Overhead“ for design, documentation, no following to standards, ...
  - No previous knowledge necessary
- **Cons**
  - No possibility to get the current status
  - Difficult to control
  - Problems with maintainability and enhancement highly probable
- **So**
  - Only a (possible) idea for small short projects, e. g. Throwaway-prototypes, „Proof-of-Concept“-Demonstrations, etc.

Software Development Process Why?

- Why models?
  A good model should answer questions to a system without using it
- But: Models are always wrong - some are useful
- Idea: Good process → good product

Software Development Process Why?

- Software Development Processes and you
  - Let's hear from your experience
  - Group discussion (3 to 4 people in one group)
    - Have you ever done software development?
    - What was your part / your role?
    - Was there a process, did you like it?
    - What were you experiences?
    - What was the craziest, funniest, most stupid thing what happened during the software development ?
  - Results to the class

Classical Process Models History - On the way

- Reason: Software Crisis End of 1960s / Begin of 1970s
  - Software could not keep up with increasing performance characteristics of the hardware
  - Software got more and more complex, fault-prone and unmanageable with more and more requirements and enlargements
  - Software got more expansive because of high manual programming effort (low automation, no software reusability)
- Goal: From non coordinated non structured program development to Software Engineering!

Classical Process Models SA / SD

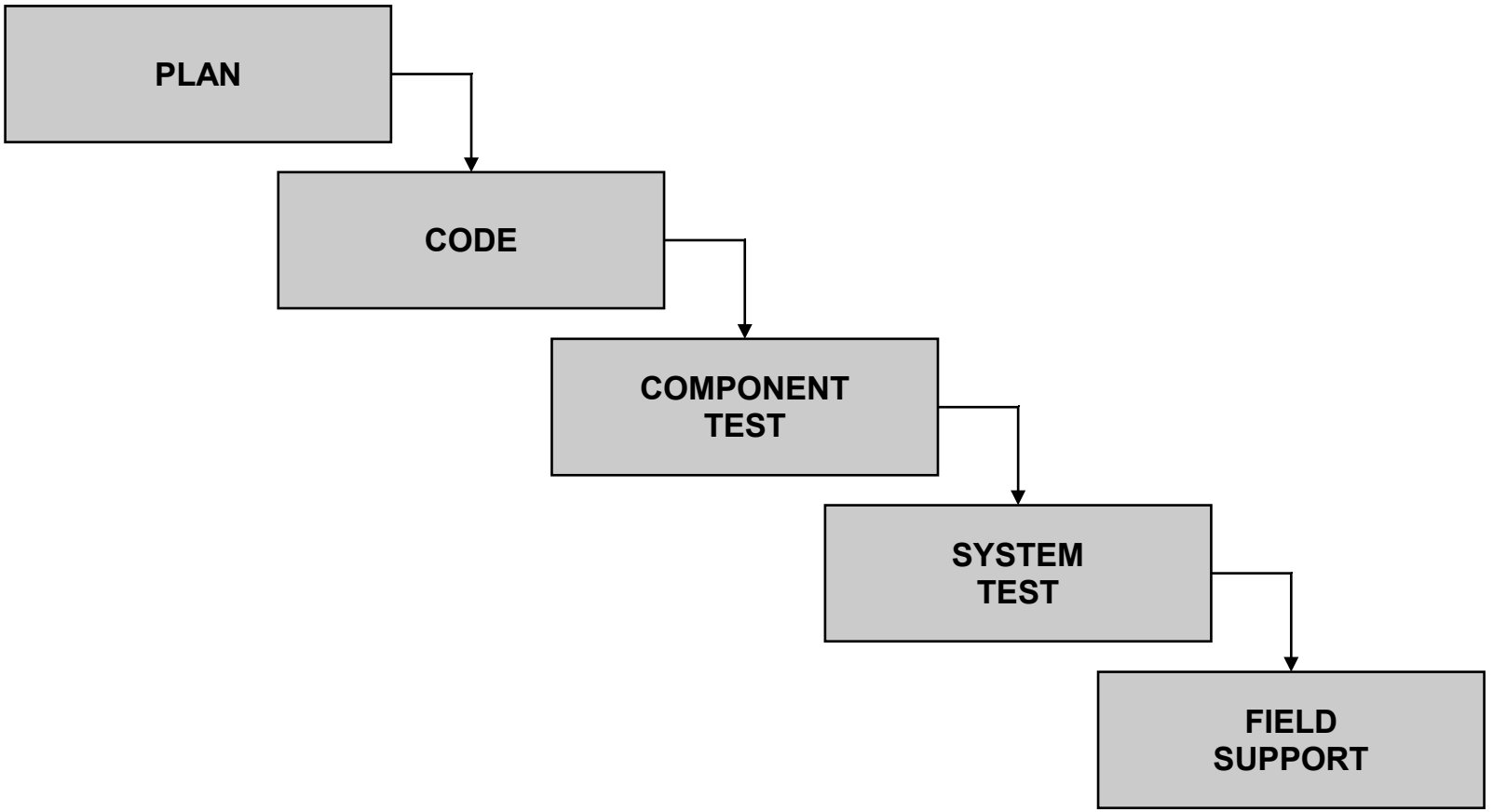
- Structured Analysis (SA)
  - From DeMarco, McMenamin und Palmer [DeM78]
  - Collects requirements for implementation
  - Modelling: Structured Analysis Design Technique (SADT)
  - Contents: Flow charts, data dictionary, process specification
  - Dynamic modeling with sequence diagrams and Petri Nets – enhancement to SA/RT (SA for Realtime Systems)

Classical Process Models SA / SD

- Structured Design (SD)
  - Planning of implementation
  - Stage between analysis and implementation
  - Analysis of underlying data
  - Development of conceptional data models in structure diagrams

Classical Process Models SA / SD

- Process Model: Waterfall model [Roy70]



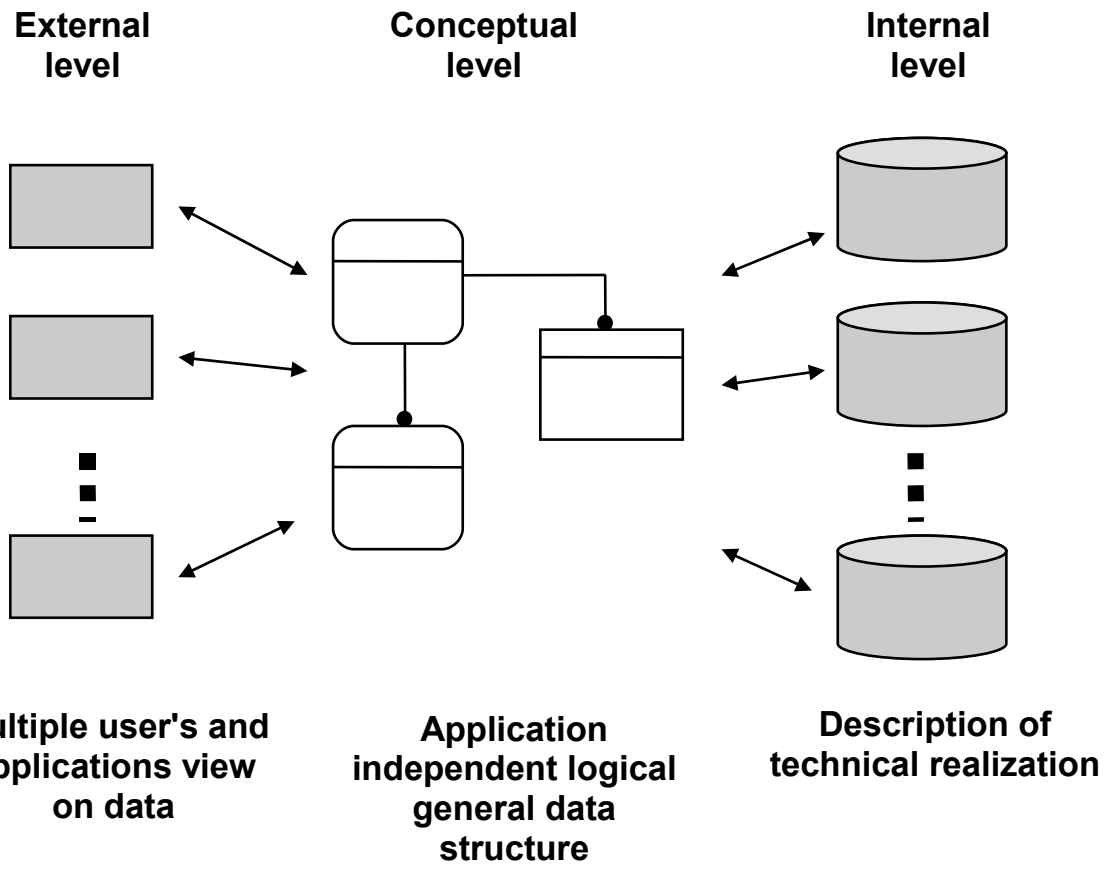
Classical Process Models SA / SD

- Critics
  - Brooks: Waterfall model is wrong [FPB95, pp. 264]
  - One way from analysis to design too ideal and inflexible
  - Problems detected later lead to local fixes without considerable common changes
  - Difficult to handle
  - Not clear
  - Gap between analysis and design model
  - Good for small well arranged projects

Classical Process Models
Data oriented approach

- The data oriented approach is standard practice in database applications
  - Used data are in the centre
  - Less important: Procedural techniques how to work with the data
- More areas of the data oriented approach in software development
  - Conversion processors
  - EDM-Systems (EDM = Engineering Data Management)
  - Integration architecture (discuss SOA)

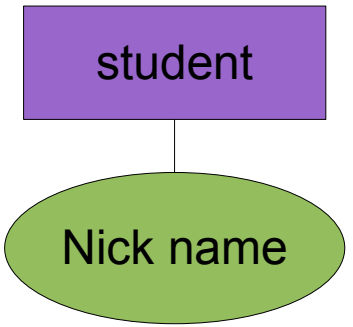
Classical Process Models
Data oriented approach



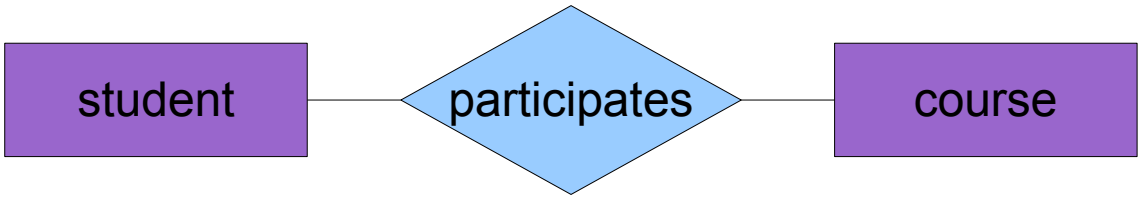
Proposal of a 3-Level-Architecture for Databases of ANSI-SPARC (American National Standards Institute - Standard Planning and Requirement Committee)

Classical Process Models
Data oriented approach

- Modelling with Entity-Relationship-Model (ERM) from Chen [Che76]
- Basic elements
  - Entities as objects of the real world
  - Relationships between entities



Entity with attribute



Two related entities

Primary key

Classical Process Models Data oriented approach

- Comments
  - Semantic of Entity-Relationship-Models is not sufficient for complex application
    - Dynamic behaviour could not be shown
  - Further development to EER-Model (Extended Entity-Relationship-Model) with
    - Parent classes and child classes
    - Inheritance hierarchies
  - Development of object oriented and object relational data bases

Object Oriented Process Models UML

- In the 1980s:
  - From process orientation to object orientation
  - From waterfall model to iterative incremental approach
- From the beginning of the 1990s
  - Ideas to formalize object oriented analysis (OOA) and object oriented design (OOD)

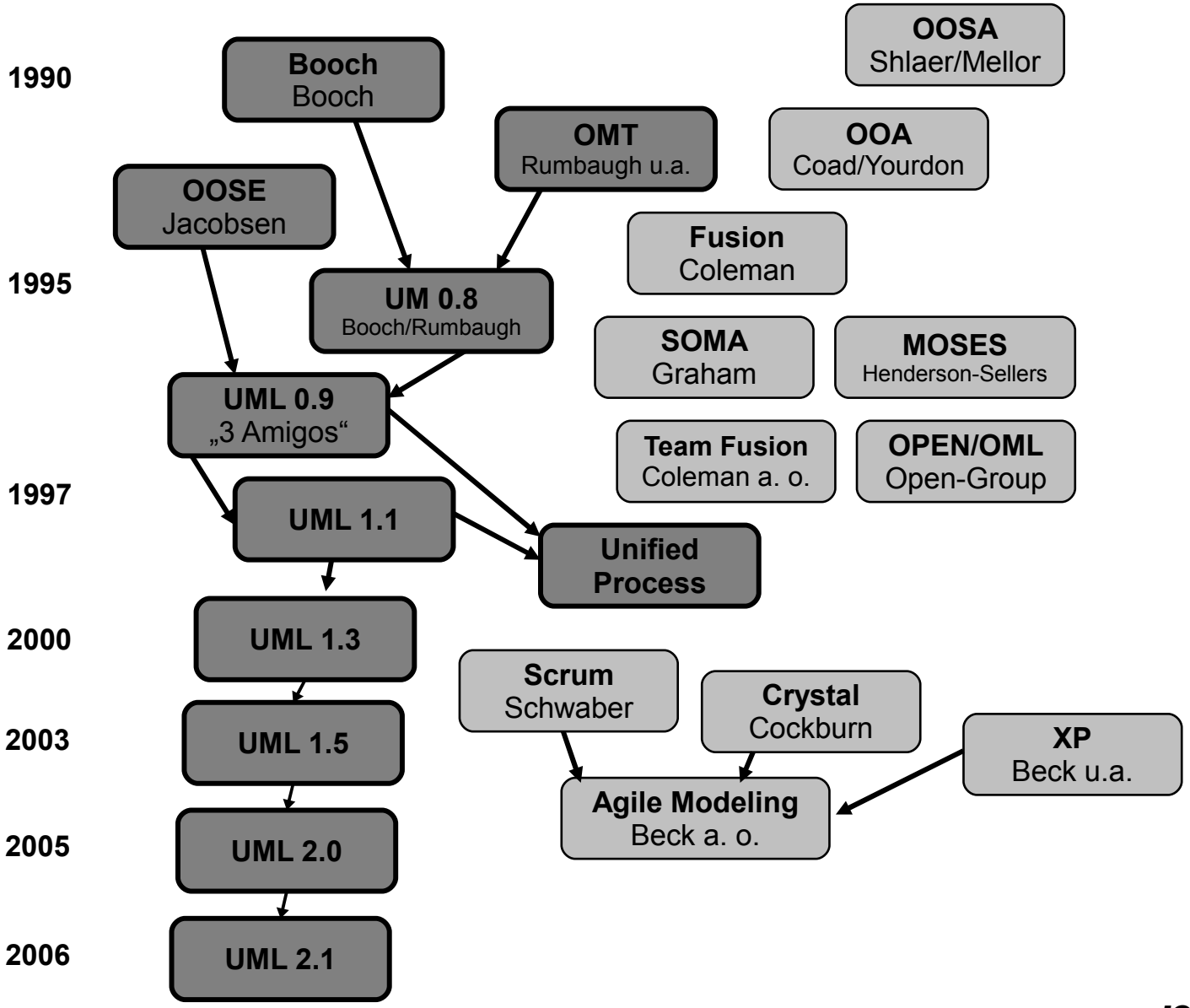
Object Oriented Process Models UML

- Object oriented Design (OOD) by Booch
  - Focus: Commercial applications
- Object Modeling Technique (OMT) by Rumbaugh
  - Referring to structured methods
- Object-Oriented Software-Engineering (OOSE) by Jacobson
- Object Oriented Analysis (OOA) by Coad, Yourdon
- Rumbaugh and Booch started to work together, supported by Jacobson later ➔ „3 Amigos“

Object Oriented Process Models UML

- Lead managed by the “Rational” company the „3 Amigos“ developed together the Unified Modeling Language (UML):
  - Graphical language to describe object oriented models
  - Unified notation - UML could be used as one language for the analysis as well for the design of object oriented systems
  - Standardization by OMG (www.omg.org)
  - No Method, but could be fundament for methods

Object Oriented Process Models



compare [Oes06]

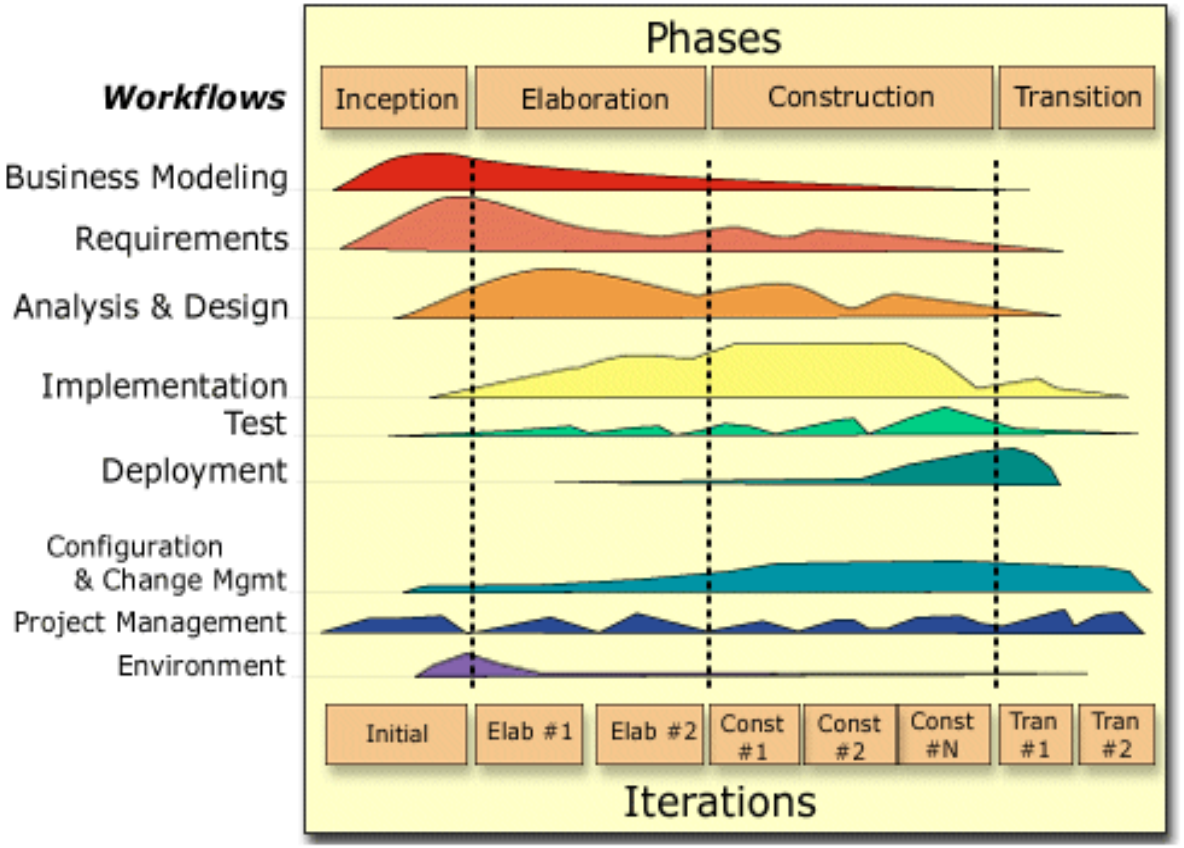
Object Oriented Process Models

- Rational Unified Process (RUP)
- Object Engineering Process (OEP) [Oes06]
- The German Government Process Model (German: “V-Modell”)
  - Standard for the software development at the German Federal Armed Forces
- In-house process models

Object Oriented Process Models RUP

- The Rational Unified Process (RUP)
  - was created by Jacobson, Booch and Rumbaugh in 1999
  - is based on the UML
  - is an incremental and iterative software development process
  - is based on
    - Phases – where one or more iterations take place
    - Iterations – during the iterations the different quantified activities in the subprocesses take place
    - Processes - Subprocesses are assigned to workflows

Object Oriented Process Models
RUP

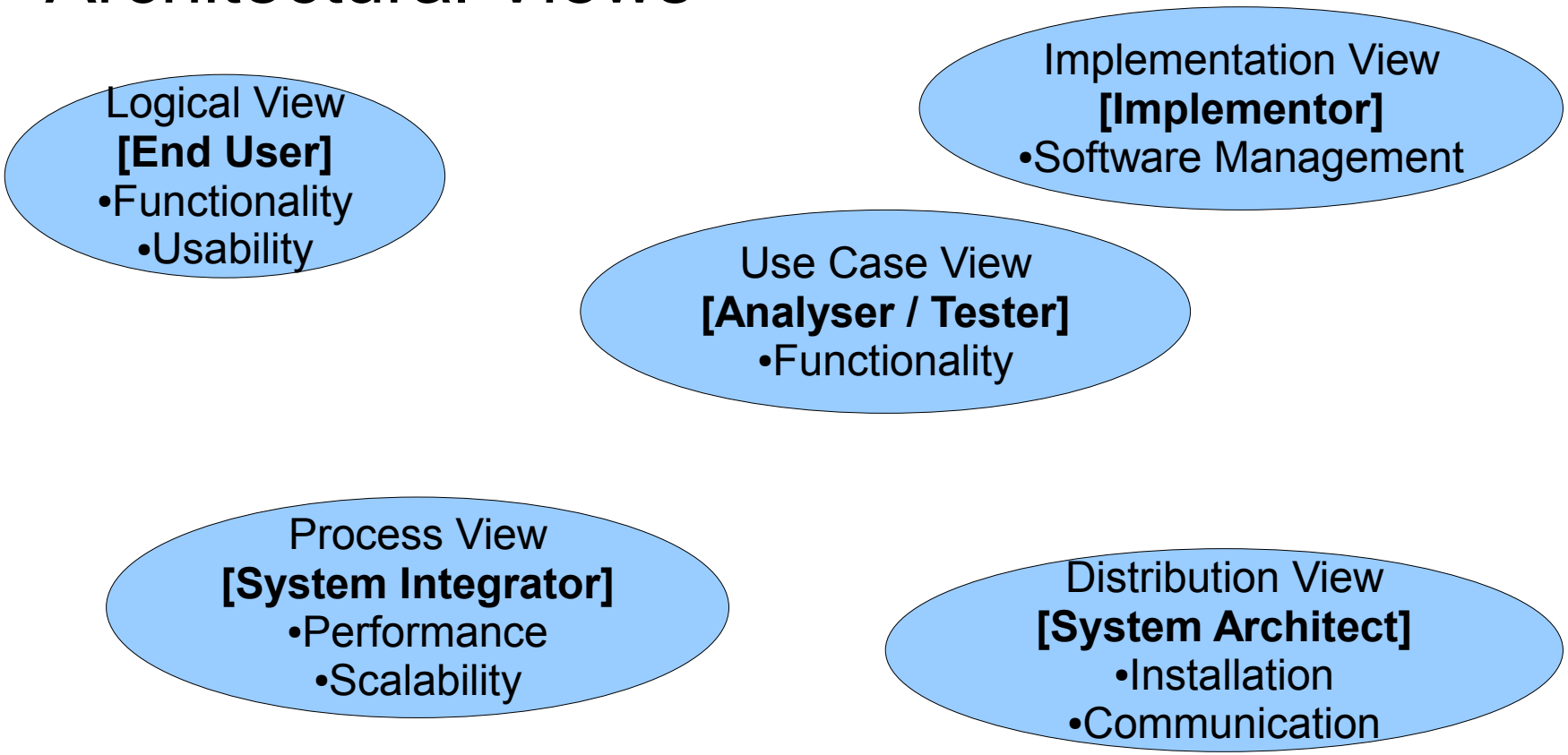


Object Oriented Process Models RUP

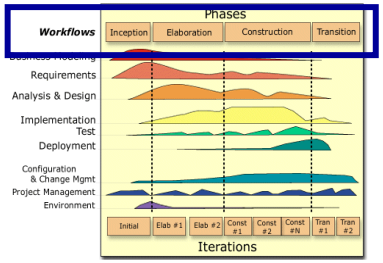
- Six core workflows
  - Business Modelling Workflow
  - Requirement Workflow
  - Analysis & Design Workflow
  - Implementation Workflow
  - Test Workflow
  - Deployment Workflow
- Three supporting workflows
  - Project Management Workflow
  - Configuration and Change Management Workflow
  - Environment Workflow

Object Oriented Process Models RUP

- Architectural Views



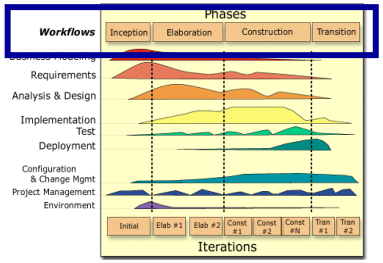
Object Oriented Process Models RUP



- Phases
  - Inception (Concept phase)
    - Define scope of project
  - Elaboration (Design phase)
    - Planning of project
    - Define functionality and architecture
  - Construction
    - Implementation of product
    - Enhancement of functionality
  - Transition
    - Product delivery

Object Oriented Process Models RUP

- Phases



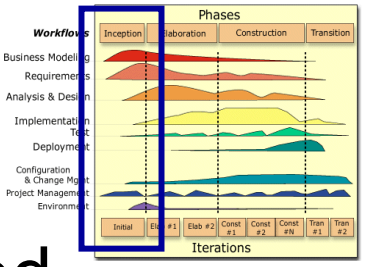
	Inception	Elaboration	Construction	Transition
Effort	**~ 5%**	**20%**	**65%**	**10%**
Time	**10%**	**20%-40%**	**40%-50%**	**10%**

Table 1: Characteristic proportions of the phases

Short project	**3 Iterations (0 I, 1 E, 1 C, 1 T)**
Typical project	**6 Iterations (1 I, 2 E, 2 C, 1 T)**
Complex project	**9 Iterations (1 I, 3 E, 3 C, 2 T)**
Risky project	**10 Iterations (2 I, 3 E, 3 C, 2 T)**

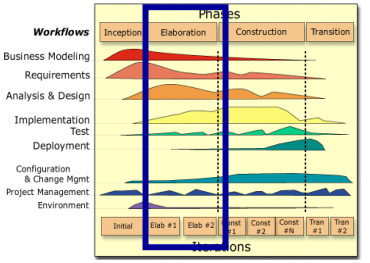
Table 2: Characteristic values for iterations per phase depending on kind and size of projects

Object Oriented Process Models RUP



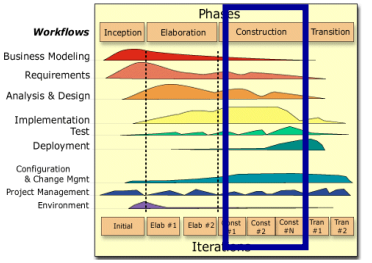
- Inception Phase
  - Small team (up to 6 people) – high qualified
  - Milestones
    - Planning of Responsibilities, rough planes for each phase
    - Baseline vision and requirements
    - Architecture models
  - Evaluation
    - Agreement of all project members
    - **Go** / **No go** Decision

Object Oriented Process Models RUP



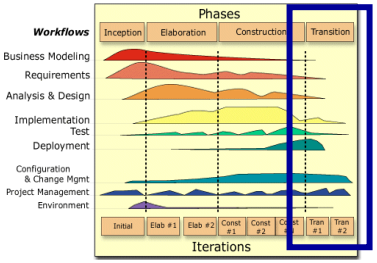
- Elaboration Phase
  - Small team (up to 10 people)
  - Milestones
    - Planning of Construction Phase
    - Stable requirements
    - Stable design model
  - Evaluation
    - Product vision and architecture are stable
    - **Don'ts:** Analysis Paralysis

Object Oriented Process Models RUP



- Construction Phase
  - Maximum number of people
  - Release every 2 to 3 months up to 6 to 9 months depending on project size
  - Milestones
    - Stable implementation with critical features in transparent quality
  - Evaluation
    - Quality of releases

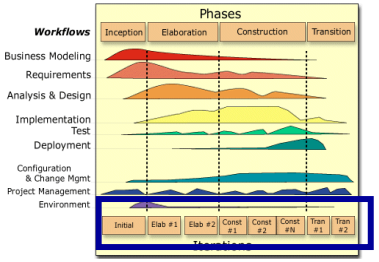
Object Oriented Process Models RUP

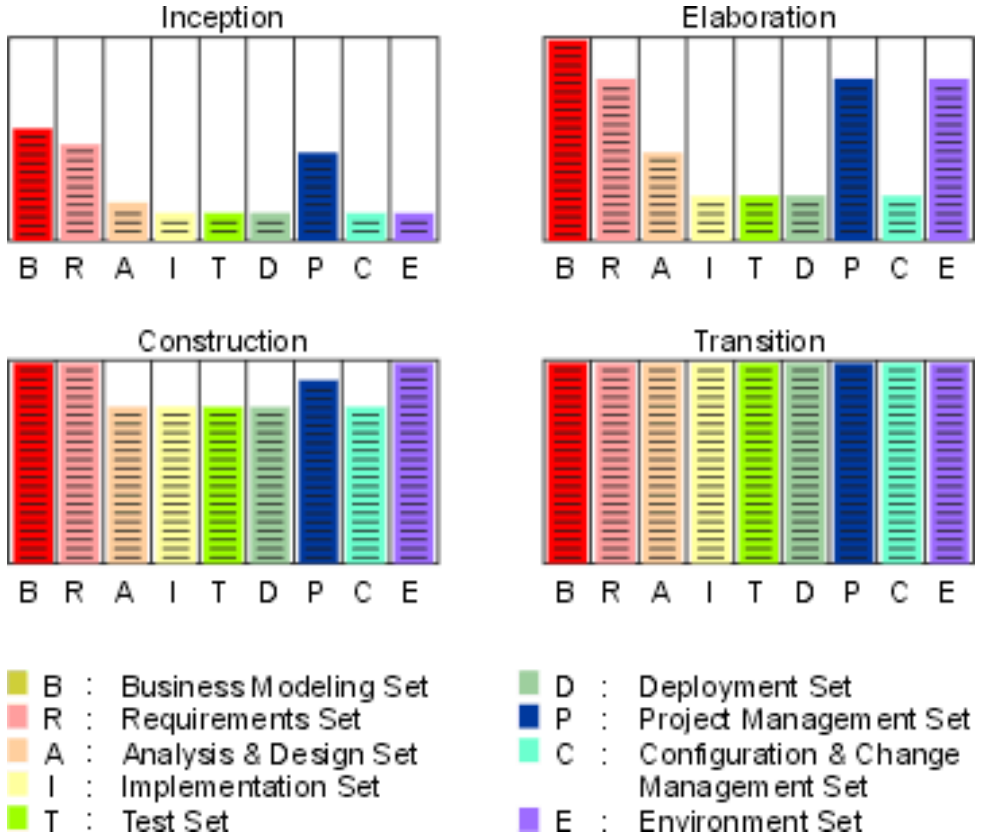


- Transition Phase
  - Small team, support, trainer
  - Milestones
    - User documentation
    - Stable delivery with all features in highest quality
  - Evaluation
    - Acceptance of customer
    - Real project costs to predicted project costs

Object Oriented Process Models RUP

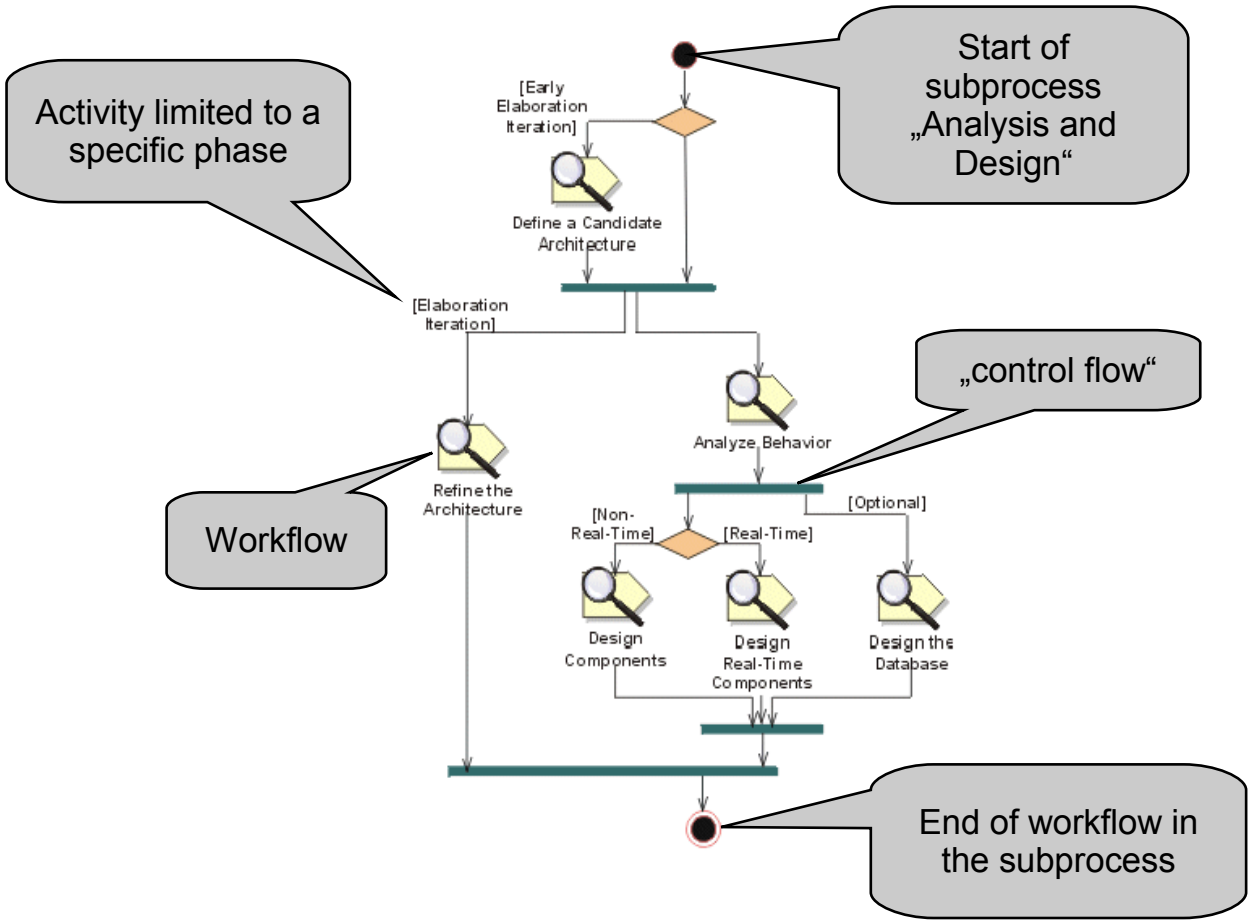
- Artifacts get finished until the end of the project





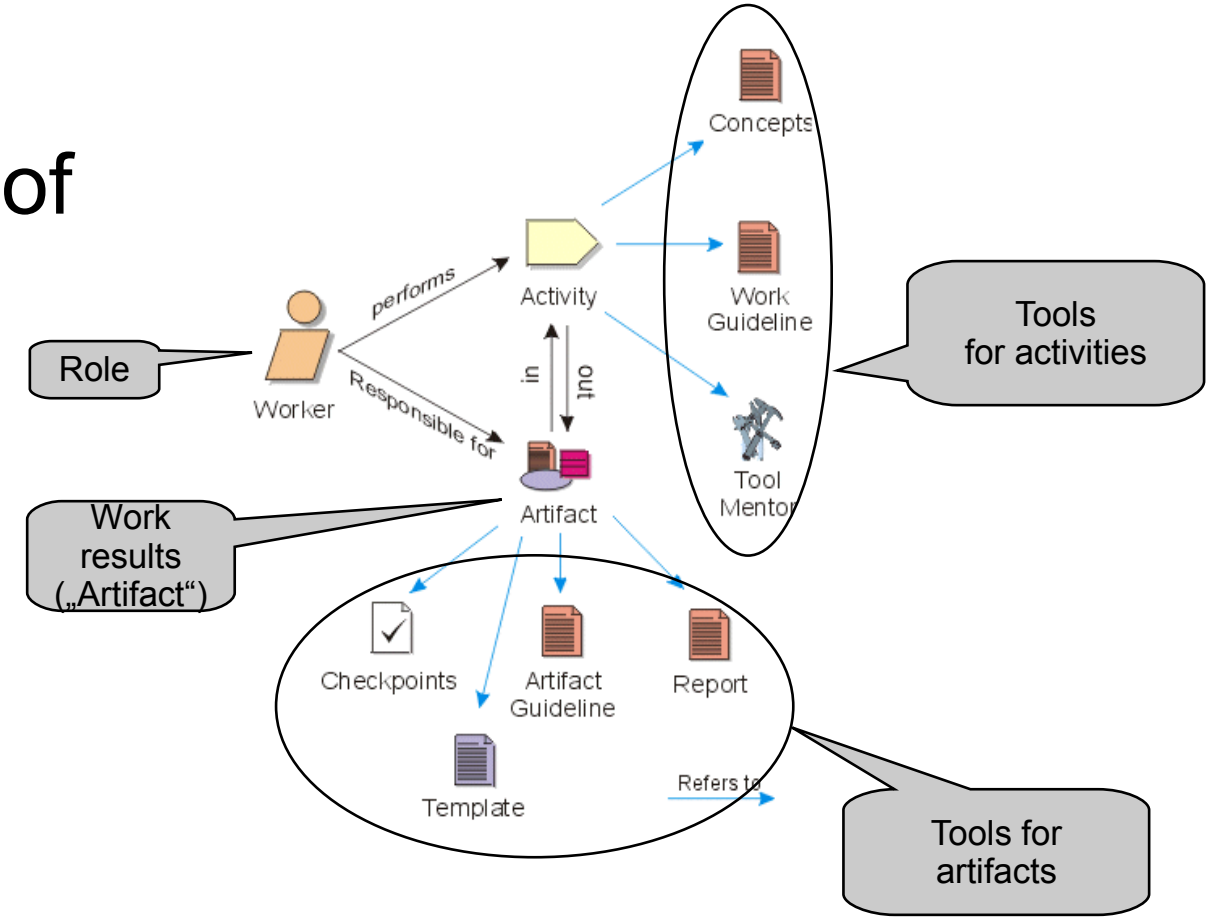
Object Oriented Process Models RUP

- Subprocess „Analysis and Design“



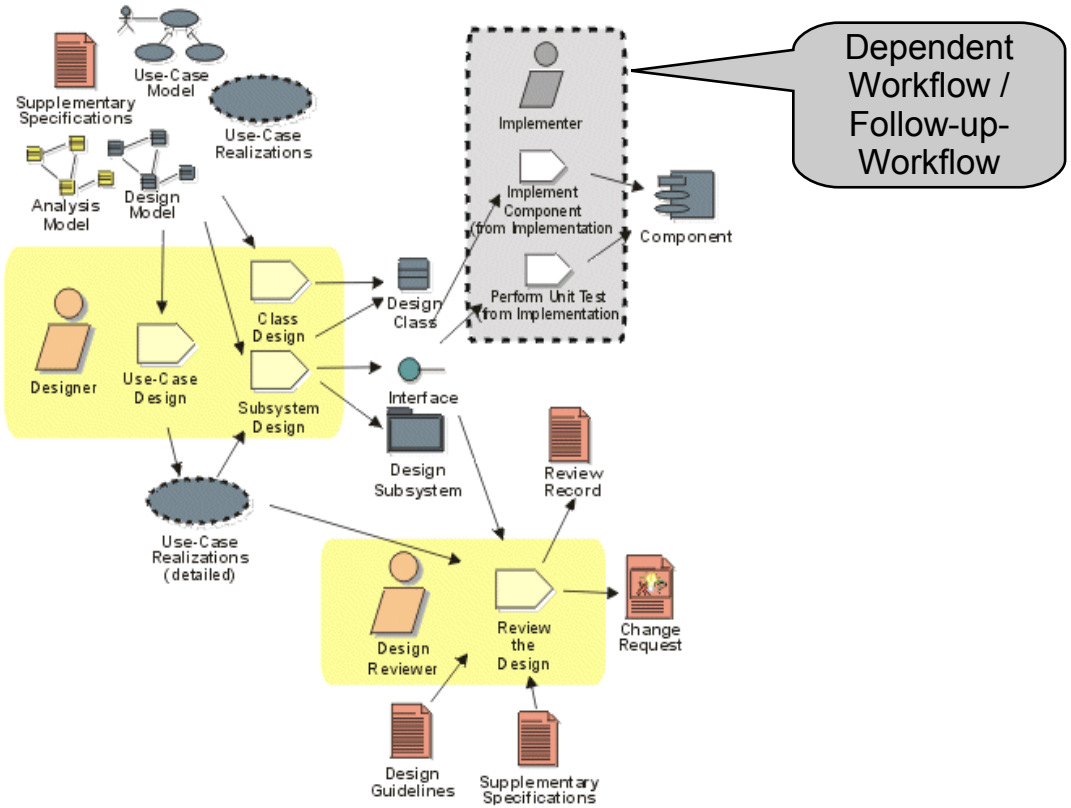
Object Oriented Process Models RUP

- Basic elements of Workflow



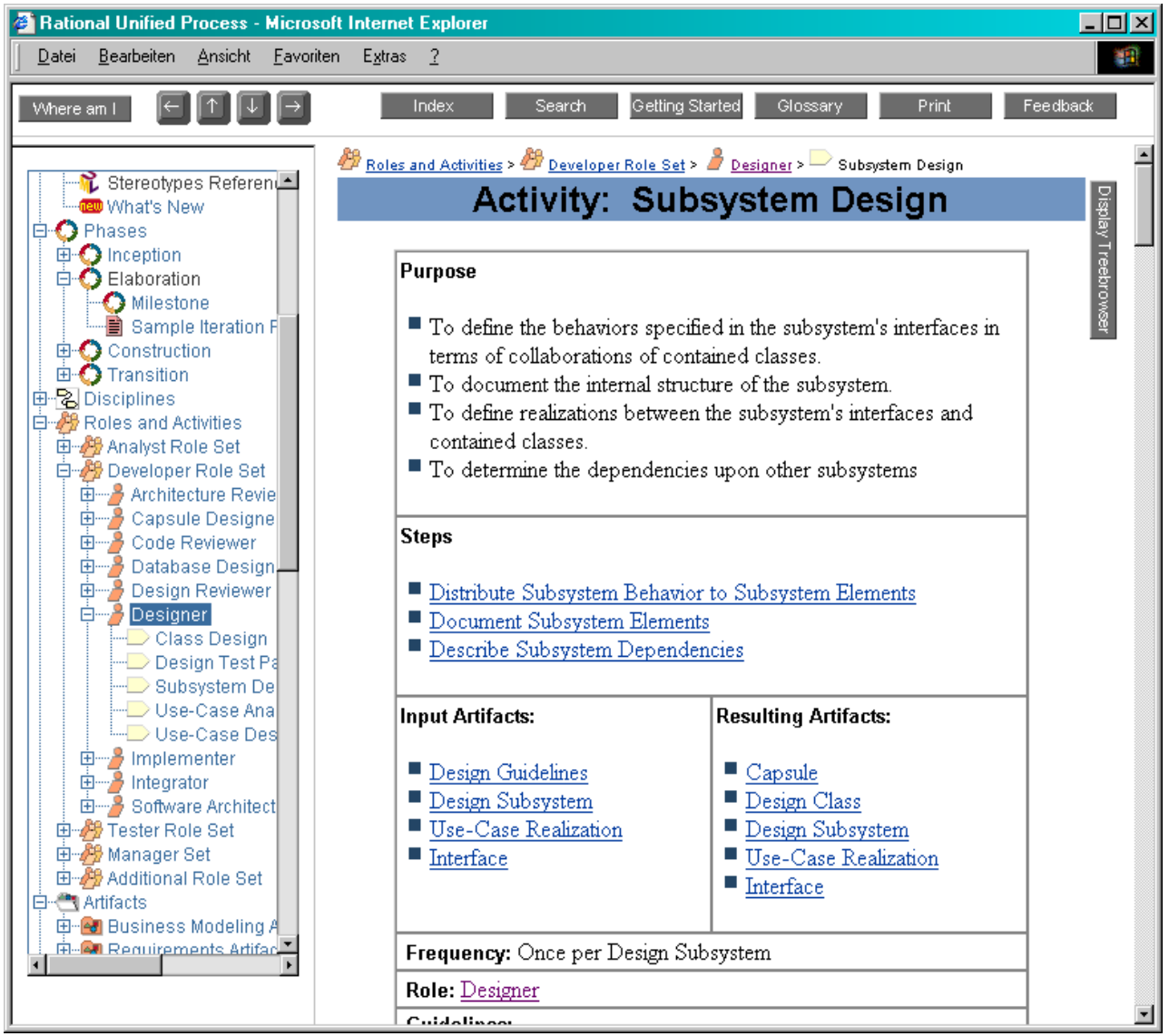
Object Oriented Process Models RUP

- Workflow „Design Components“



Object Oriented Process Models RUP

- Activity „Subsystem Design“



Object Oriented Process Models RUP

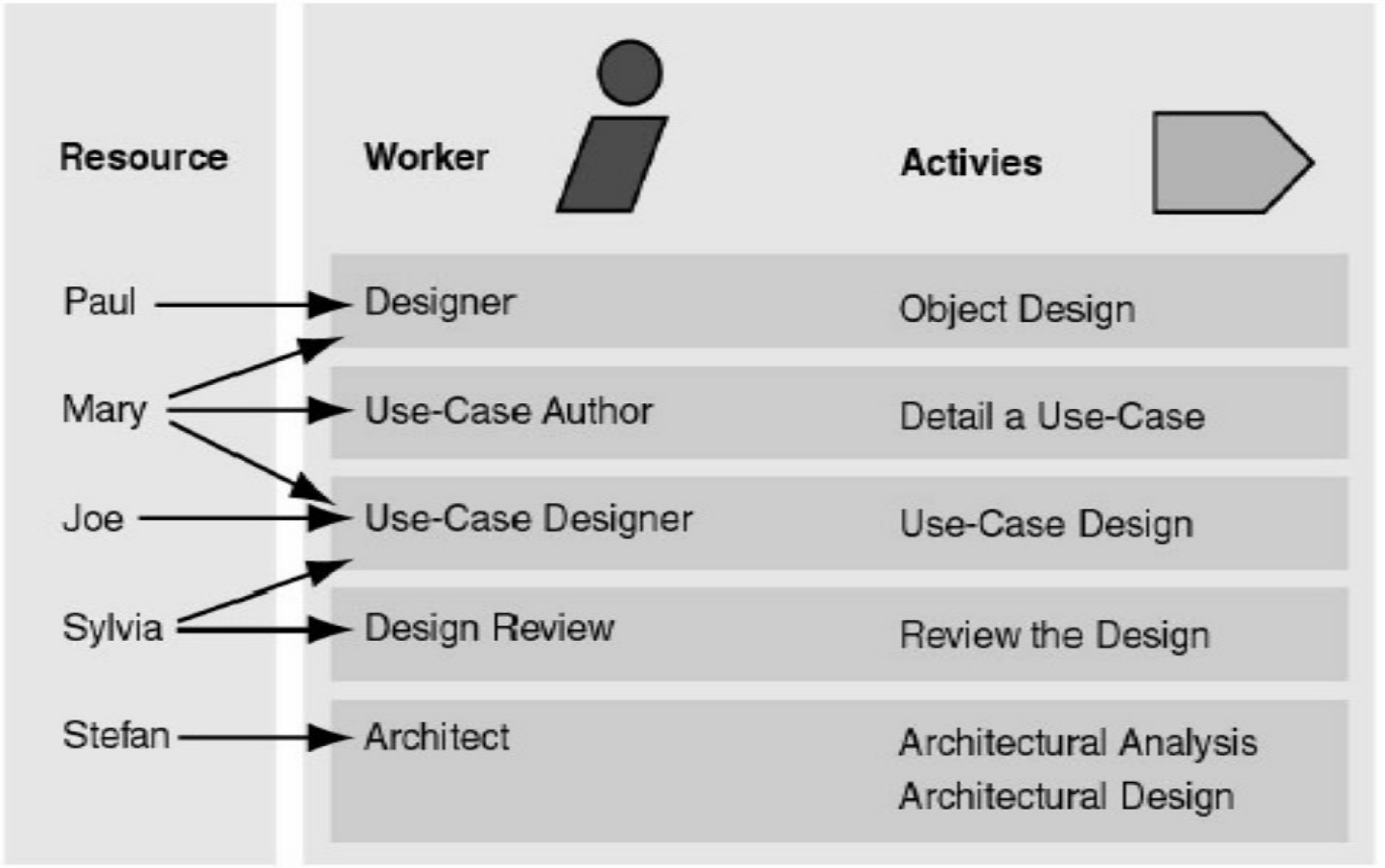
- People and Roles
  - Examples for roles in the Rational Unified Process:
  - Analysts (Business-Process Analyst, Business Designer, Requirement Specifier, System Analyst,etc.)
  - Developers (Software Architect, Capsule Designer, Database Designer, Design Reviewer, Designer, Implementor, etc.)
  - Testers (Test Designer, Tester)
  - Managers (Change Control Manager, Configuration Manager, Deployment Manager, Project Manager, etc.)

Object Oriented Process Models RUP

- People and Roles
- Responsibilities of roles have to be clarified
- Tool to help: RACI-Matrix
- RACI = Responsible, Accountable, Consulted, Informed
- Roles must be mapped to people (`m:n`-mapping)
- Roles can change at each start of a project phase

Object Oriented Process Models RUP

- People and Roles



Source: www.jeckle.de

Object Oriented Process Models RUP

- Comments
  - RUP is established, but there are points of critique [Hes01]:
    - RUP still follows the waterfall model
    - RUP is not sufficient architecture concentrated
    - Iterations are coupled to phases, not to packages
    - Needless complexity of RUP workflows (named now: disciplines)
    - Hierarchy, Recursion, and orthogonality are not used
    - Not sufficient management support, milestones concept is too weak

Agile Approaches Overview

- Agile Process Models are an alternative to traditional Software Development Processes
- Examples (1/2)
  - eXtreme Programming (XP), see www.extremeprogramming.org
  - Scrum, see www.controlchaos.com
    - Incremental process, the end of an iteration results in a set of features
    - Authors: Takeuchi / Nonaka, 1986; Schwaber / Beedle, 2002
    - Terms out of Rugby: Pregame Phase, Development Phase (Sprint), Postgame Phase

Agile Approaches Overview

- Examples (2/2)
  - Crystal-Process family
    - Author: Cockburn, 1998
    - Crystal Clear for small projects (up to 6 developer)
    - Crystal Orange for midsize projects (up to 40 developer)
  - Feature Driven Development (FDD) (Coad et. al. 2000)

Agile Approaches Overview

- Developers of agile methods met, compared their ideas, and clarified differences

  www.agilemanifesto.org

- Basic values:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

Agile Approaches
XP

- Developed 1996 by Kent Beck
- Basic values
  - **Communication**: Regularly between customer and developer and between developer themselves
  - **Simplicity**: Regular rework of source code and avoiding needlessness
  - **Feedback**: Based on „User Stories“ many small releases, continuous integration, continuous testing
  - **Courage**: What is possible and what is not possible

Agile Approaches
XP

„XP is the most important movement in our field today. I predict that it will be as essential to the present generation as the S.E.I. and its Capability Maturity Model were to the last.“

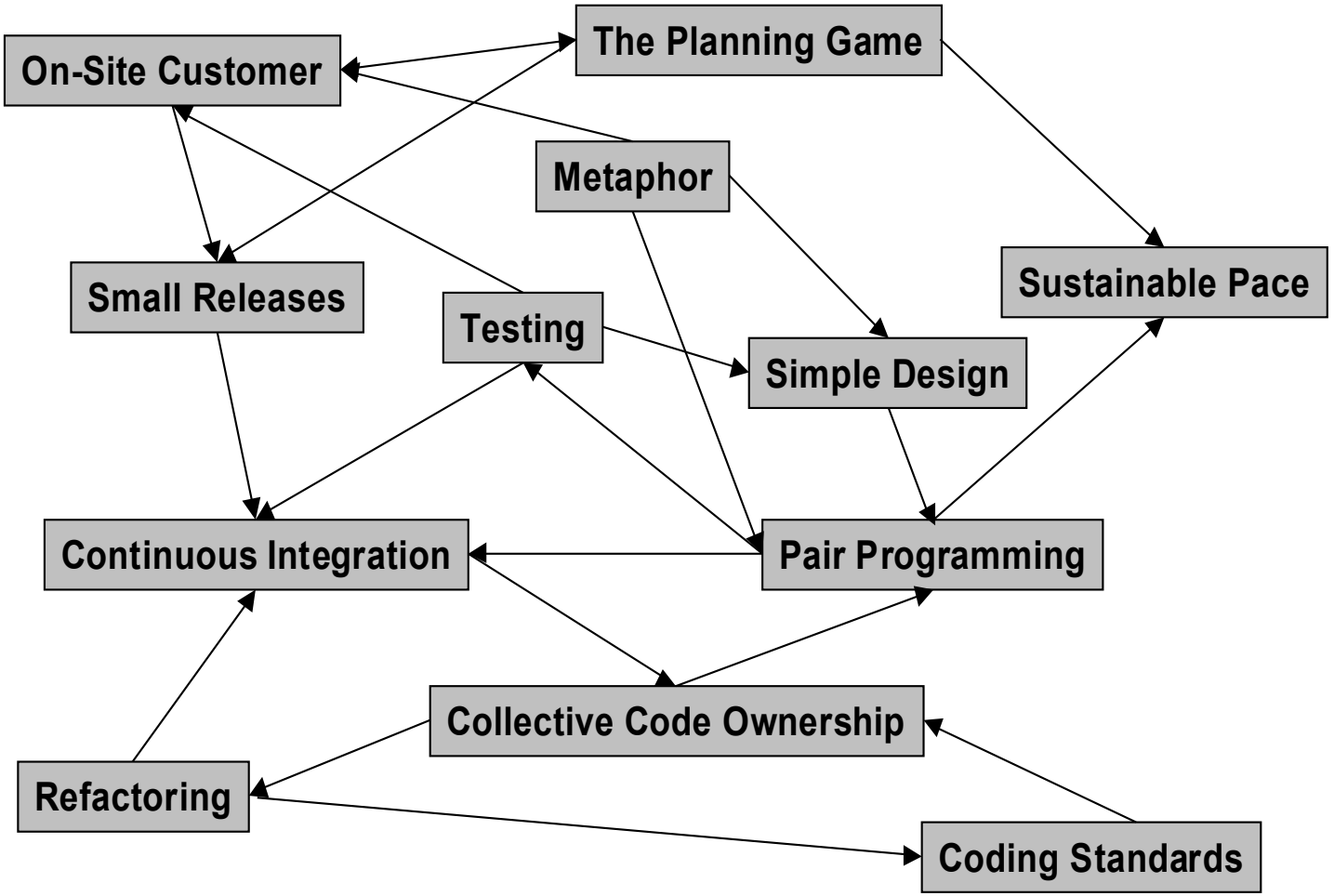
-- Tom DeMarco, *Preface to „Planning Extreme Programming“*, 2001

Agile Approaches
XP

- User stories / Storycards
  - User Stories to handle requirements in XP
  - During the project the customer writes new user stories, deletes old, changes, but has to prioritize
  - Small piece of paper
  - Basic for reports

Agile Approaches
XP

- 12 Practices



Agile Approaches
XP – 12 Practices

- Fine scale feedback
  - Pair programming
    - 2 developers work together,
    - Always at least 2 know the code
    - Change of roles as necessary (other user stories)
  - Testing – Test Driven Development
    - Developers write (to be automated) Unit-Tests before coding ("Test-First"-approach)
    - Customer defines parallel functionality tests

Agile Approaches
XP – 12 Practices

- Fine scale feedback
  - The Planning Game
    - Common release planning based on user stories
    - Prioritization by customer – Effort guess by developers
  - On-Site Customer
    - A representative of the customer is always available to discuss / answer questions and to get decisions concerning user stories and test

Agile Approaches
XP – 12 Practices

- Continuous process
  - Continuous Integration
    - If a user story is done, it gets integrated in the whole system
    - Testing before and after integration to ensure functionality
  - Refactoring
    - Every time when it is detected that the design could be improved, it has to be done
    - Unit-Tests assure, that the functionality still works
  - Small Releases
    - About every +/- 4 weeks to get early customer feedback

Agile Approaches
XP – 12 Practices

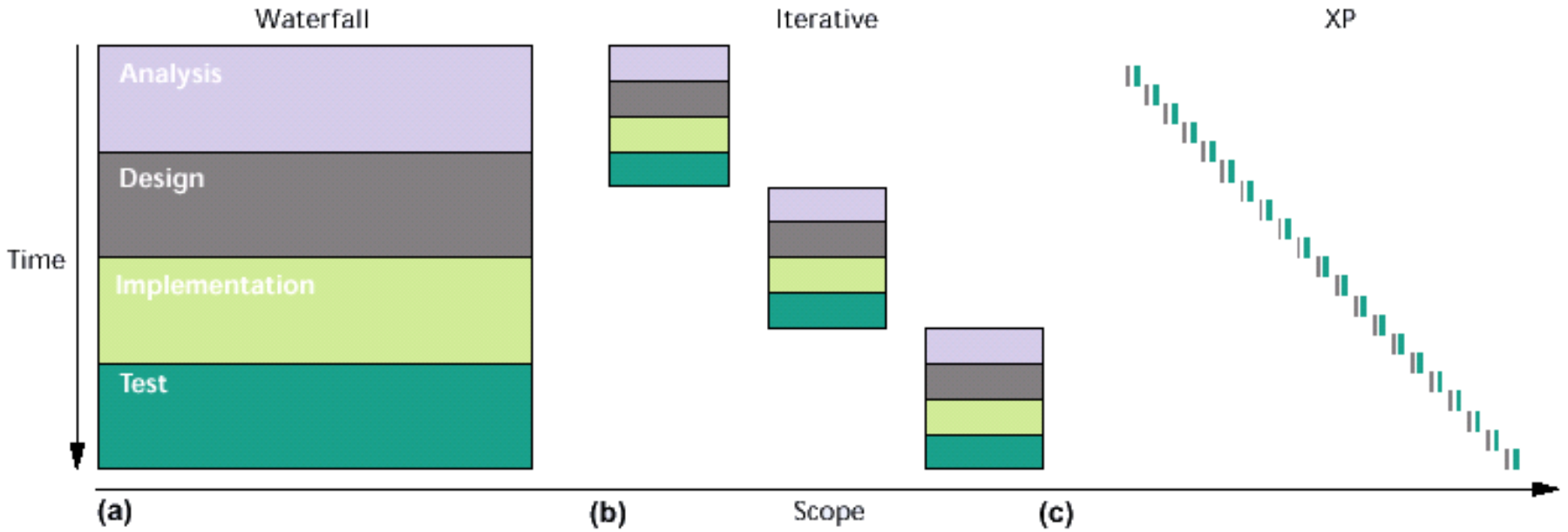
- Shared understanding
  - Coding Standards
  - Collective Code Ownership
    Everybody could change everywhere
  - Simple Design (→ Refactoring)
    - Design and Code as simple as possible
    - Not needed code gets deleted immediately
    - implement only what is needed to fulfill an user story
  - Metaphor
    Simple story how the system should work instead of a complex architecture description

Agile Approaches
XP – 12 Practices

- Programmer welfare
  - Sustainable Pace
    The big needs in XP lead to intensive work, so that overtime should not be done

Agile Approaches
XP

- Summary:



K. Beck, Embracing Change with Extreme Programming, IEEE Computer, Oct 1999.

Agile Approaches
XP

- Summary: Learning from XP
  XP-Practices could be used in other Software Development Processes as well
  - "Test-First"-approach
  - Small releases and continuous / frequent integrations
  - Pair Programming
  - Refactoring to keep “projects well”
- A process model supporting XP is Scrum
  ... see www.methodsandtools.com

Comparison

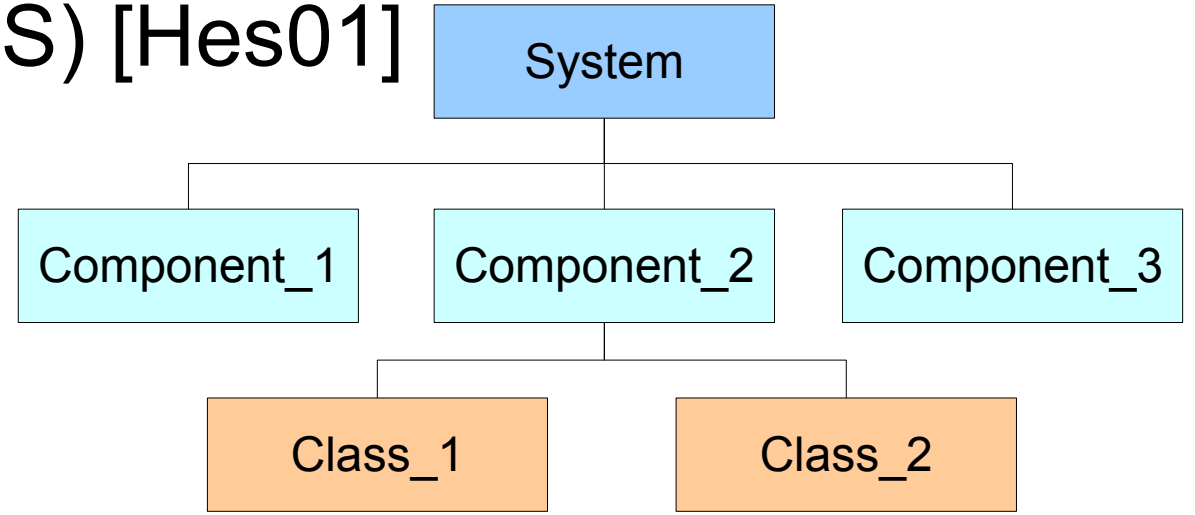
RUP versus XP

- Heavy process
- Document / Artifact concentrated
- Organization important
- Based on UML
- Roles
- Phases with Artifacts
- Specification

- Lean process
- User Stories / On-site Customer
- Organization minimized
- Based on User Stories / On-site Customer
- Collective Code Ownership
- Timeboxes
- User stories / storycards

More Process Models

- Evolutionary object oriented Software-Development (EOS) [Hes01]



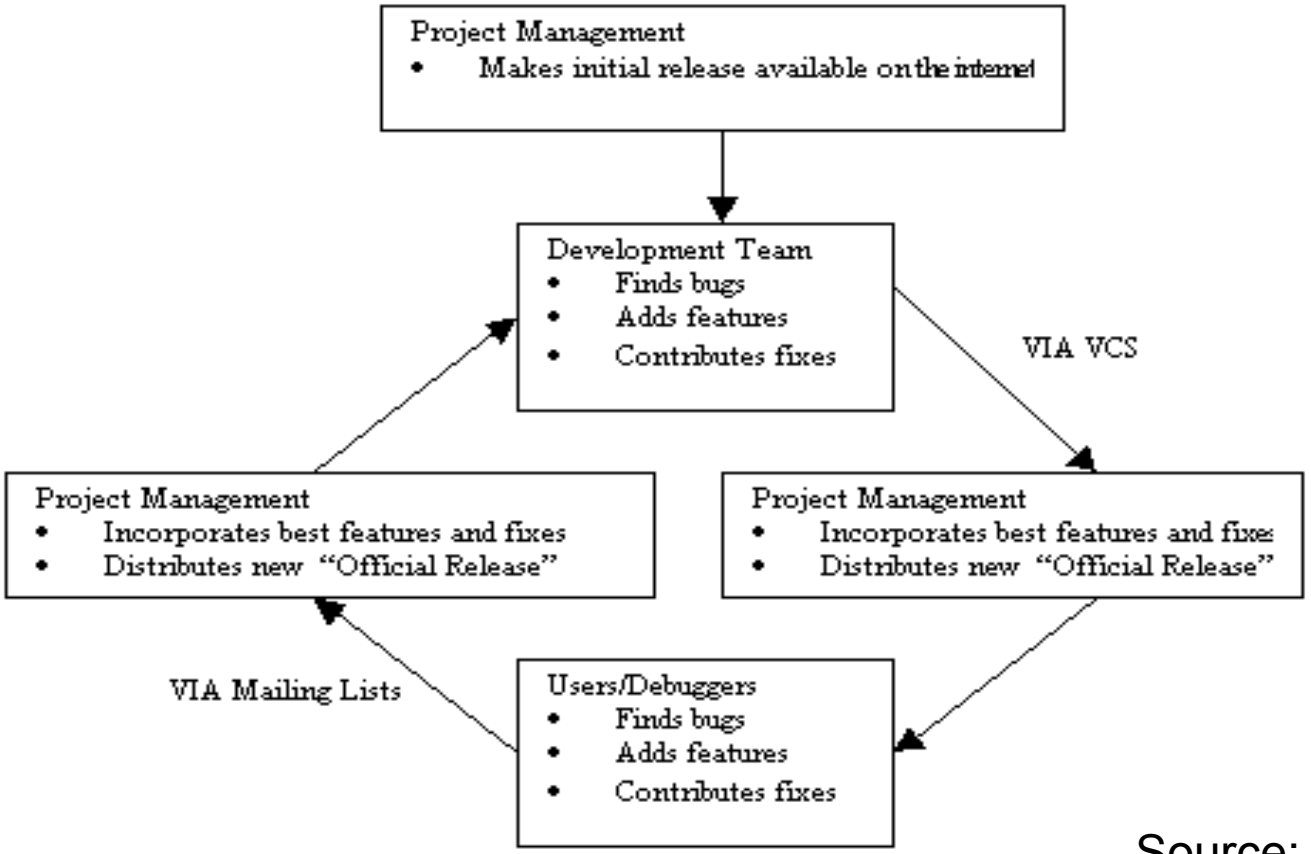
- Fountain Model [Pil96]
  - Activities could be done parallel
  - Highly iterative
  - Specific phases could overlap

More Process Models

- Chaos model from Raccoon [Rac95]
  - extends the spiral model and waterfall model.
  - Idea [wikipedia.org]
    - An important change in perspective is if projects can be thought of as whole units, or must be thought of in pieces.
    - Nobody writes thousands of lines of code in one sitting. They write small pieces, one line at a time, verifying that the small pieces work. Then they build up from there.
    - The behaviour of a complex system emerges from the combined behaviour of the smaller building blocks.
  - Flexibility, to iterate in specific steps in the Software Engineering Process, if requested

More Process Models

- Open Source Development Process



Source: LinuxCare, 2000, p4

More Process Models Outlook

- New developments like the orientation on components will lead to new process models, that are based on basic concepts of the object oriented process models
- Model based Software Development with the idea to map business processes directly to code (Model Driven Architecture – MDA)

Software Development Process Improvements

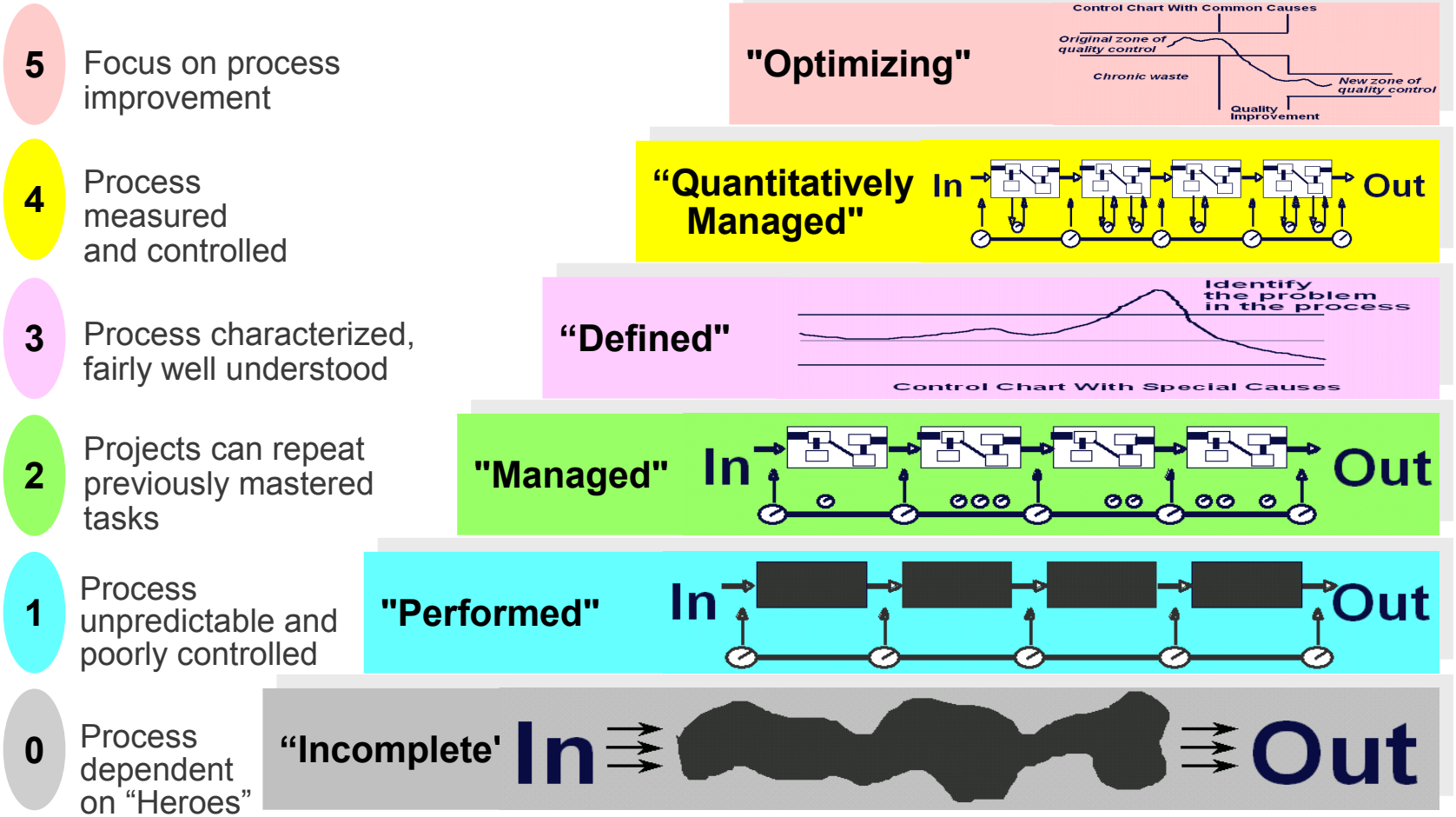
- So: Great processes! But which process to use for my project?
  - There is no ideal process for each problem with every possible constraint
  - Every process has to be adapted to the specific project
- Good: At least ***there should be any*** defined process to make a project successful
- Better: Continuous process improvement

Software Development Process Improvements – with CMMI

- CMMI = Capability Maturity Model Integrated
- Processes with Capability Level
  - 0 (Incomplete)
  - .... , up to
  - 5 (Optimizing)
- Organization units get Maturity Level
  - 1 (Initial)
  - .... , up to
  - 5 (Optimizing)
- The Maturity Level predicts the performance of an Organization

Software Development Process Improvements – with CMMI

- Maturity Levels



Software Development Process Improvements – with CMMI

- Success stories

Category	Result	Company
Costs	33% decrease in the average cost to fix a defect	Boeing, Australia
Costs	20% reduction in unit software costs	Lockheed Martin M&DS
Costs	60% reduction in cost of customer acceptance	Thales Research & Technology
Costs	Saved $2 million in first 6 months after reaching CMM ML3	Sanchez Computer Associates, Inc
Schedule	Increased the percentage of milestones met from approximately 50% to approximately 95%	General Motors
Schedule	Met every milestone (25 in a row) on time, with high quality and customer satisfaction	Northrop Grumman IT2
Schedule	15% improvement in internal on-time delivery	Bosch Gasoline Systems
Quality	Reduction in defects found from 6.6 per KLOC to 2.1 over 5 causal analysis cycles	Northrop Grumman IT2
Return on Investment	5:1 ROI for quality activities	Accenture
Return on Investment	13:1 ROI calculated as defects avoided per hour spent in training and defect prevention	Northrop Grumman IT2

Source: D.R. Goldenson, D.L. Gibson, „Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results“, SEI Special Report CMU/SEI-2003-SR-009, October 2003 http://www.sei.cmu.edu/pub/documents/03.reports/pdf/03sr009-revised.pdf.

Summary

- Substantial character of all process models: Abstraction of problems to be solved
- There are alternatives to the waterfall model
- All up-to-date processes have in common: Iterative Approach
- Trend: Agile Processes (only as much formalism as necessary)
- Hint: Learn and improve your processes!