

Topic Area Project Management: Content

- VL2
 - Software Metrics
 - ↳ Properties of Metrics
 - ↳ Scales
 - ↳ Examples
- VL3
 - Cost Estimation
 - ↳ Software Economics in a Nutshell
 - ↳ Experts Estimation
 - ↳ Algorithmic Estimation
 - Project Management
 - ↳ Project
 - ↳ Process and Process Modelling
 - ↳ Process Models
 - ↳ Process Models
- VL4
 - Project Management
 - ↳ Process Models
 - ↳ Process Models
- VL5
 - Process Metrics
 - ↳ OWL Spike

- (Software) Project
 - Project Management
 - ↳ Goals and Activities
 - ↳ Common Activities
 - ↳ Escalation Risk
 - Software Project Planning
 - ↳ Costs and Deadlines
 - ↳ phase milestones, deadline
 - ↳ Tasks and Activities
 - ↳ gantt, PERT, cycle
 - ↳ software life cycle
 - ↳ responsibilities and rights
 - Software Development Process
 - ↳ Procedure and Process Models

Vocabulary: Project

Project

project - A temporary activity that is characterized by having

- a start date,
- specific objectives and constraints,
- established responsibilities,
- a budget and schedule, and
- a completion date.

If the objective of the project is to develop a software system, the project is called a **software development project**. R. H. Thayer (1977) or software engineering project.

- We could refine our earlier definition as follows: a project is **successful** if and only if
- started at start date,
 - achieved objectives, respected constraints,
 - adheres to budget and schedule,
 - stops at completion date.
- Whether, e.g., objectives have been achieved can still be **subjective** (← customer/user happy)

Vocabulary: Software Project

(software) project - characteristics:

- Duration is limited
- Has an **originator** (person or institution which initiated the project)
- The project owner is the originator or its representative
- The project leader reports to the project owner
- Has a **purpose**, i.e. to prepare, build or modify software
- The project is called **successful** if the goals are reached to a high degree
- Has a **recipient** (or will have one)
- The recipient is the customer
- Later users (conceptually) belong to the customer
- The project **links people**, results in **intermediate/final products**, and **resources**
- The organization determines their **roles and relations**, and the **external interface** of the project.

Ludewig & Lehner (2011)



Project Management

7/20

Goals and Activities of Project Management

- **Main and general goal: a successful project**
 - i.e. the project **delivers**
 - in demanded quality
 - in planned time
 - using the assigned resources
 - There may be **secondary goals**, e.g.
 - build or strengthen good reputation on market
 - acquire **knowledge** which is useful for later projects
 - develop or **create** components (to save resources later)
 - be attractive to employees
 - ...
- **Main project management activities (and responsibilities of project manager):**
 - Planning
 - Assessment and Control
 - Description and fixing of possible problems as early as possible
 - Communication
 - Leading and Motivation of Employees
 - Creation and Preservation of Beneficial Conditions

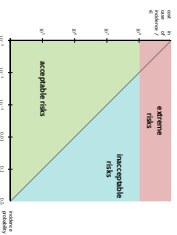


 Customer-Centered Software Delivery

8/20

Quick Excursion: Risk and Riskvalue

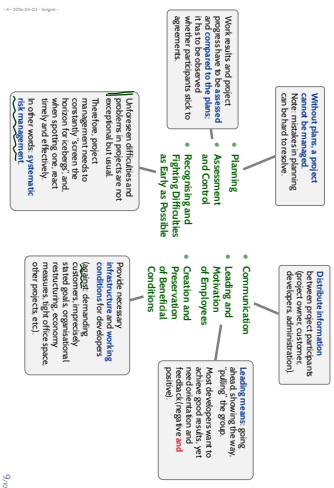
risk - a problem, which did not occur yet, but on occurrence threatens important project goals or results. Whether it will occur, cannot be surely predicted
 Ludwig & Lehner (2013)



- **Average** requires "Average Probability per Flight hour for Catastrophic Failure Conditions of 10⁻⁷" or "Extremely improbable" (FAK 25.3597-3)
- Problems with $p = 0.5$ are not risks, but environment conditions to be dealt with!


10/20

Activities of Project Management



9/20

Project Management



 Customer-Centered Software Delivery

Software Engineering as defensive discipline

Analogy: safety belt or hygiene in hospital

"Dear patient, we're working hard to prevent you from getting hurt. Well, doctor, I thought you were working to get me well again!"

Software Engineering is *proactive* and *defensive* for people who do not value the defence of failures as a positive achievement!"

04.01.2008 17:49

10/20

Software Project Planning

12/20

What to (Plan and) Manage?

Planning and managing software projects involves

- costs and deadlines,
- tasks and activities,
- people and roles.

13/01

Phases, Milestones

A phase is a continuous, i.e. not interrupted range of time in which certain works are carried out and completed. At the end of each phase, there is a milestone.

A phase is successfully completed if the criteria defined by the milestone are satisfied. Learning & Master (1999)

- Phases (in this sense) do not overlap, yet they may be different. The order of development (running in parallel) structured by different milestones.

• Splitting a project into phases makes controlling easier:

milestones may involve the customer (accept intermediate results) and trigger payments.

- The granularity of the phase structuring is critical:
 - very short phases may not be justified by the expense
 - very long phases may mask significant delays (longer than necessary)

It is necessary to define internal (customer not involved) and external (customer involved) milestones.

14/01

Milestones, Deadlines

A phase is a continuous, i.e. not interrupted range of time in which certain works are carried out and completed. At the end of each phase, there is a milestone.

A phase is successfully completed if the criteria defined by the milestone are satisfied. Learning & Master (1999)

15/01

Milestones, Deadlines

A phase is a continuous, i.e. not interrupted range of time in which certain works are carried out and completed. At the end of each phase, there is a milestone.

A phase is successfully completed if the criteria defined by the milestone are satisfied. Learning & Master (1999)

- Whether a milestone is reached (or successfully completed) must be assessable by
 - clear,
 - objective, and
 - unambiguous
- criteria.
- The definition of a milestone often comprises:
 - a definition of the results which need to be achieved,
 - the separation of the properties of these results,
 - the assignment of responsibilities to the persons who are to be held responsible, and
 - the resource person or committee which decides whether the milestone is reached.

• Milestones can be part of the **development contract**:
not reaching a defined milestone as planned can lead to **legal claims**.

15/01

What to (Plan and) Manage?

Planning and managing software projects involves

- costs and deadlines,
- tasks and activities,
- people and roles.

16/01

Cycle and Life Cycle

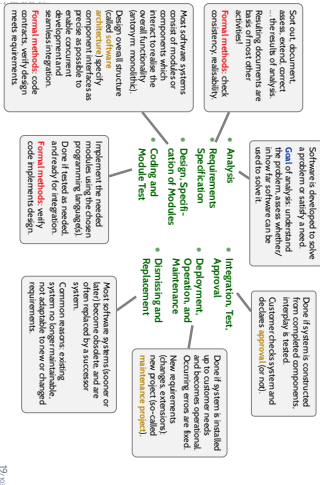
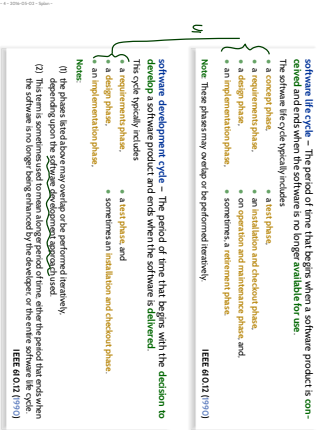
cycle – (1) A period of time during which a set of events is completed. See also: – IEEE 60012 (1990)

system life cycle – The period of time that begins when a system is acquired and ends when it is no longer available for use. IEEE 60012 (1990)

software life cycle – The period of time that begins when a software product is conceived and ends when the software is no longer available for use. IEEE 60012 (1990)

software development cycle – The period of time that begins with the decision to develop a software product and ends when the software is delivered. IEEE 60012 (1990)

17/01



- Planning and managing software projects involves
- costs and deadlines,
- tasks and activities,
- people and roles.

The Concept of Roles

In a software project, at each point in time, there is a set R of (active) roles, e.g. $R = \{PM, PR, SE, QA\}$.

A role has **responsibilities** and **rights**, and necessary skills and capabilities.

For example,

- **PM** project manager
 - has the **right** to allocate team reports
 - is **responsible** for allocating team reports
- **PR** programmer
 - has the **right** to change the code
 - is **responsible** for reporting unexplained problems to the project manager
 - is **responsible** for respecting coding conventions
 - is **responsible** for addressing warnings
- **SE** test engineer
 - has the **right** to allocate team reports
 - is **responsible** for equality control

The Concept of Roles Cont'd

Given a set R of roles, e.g. $R = \{PM, PR, SE, QA\}$, and a set P of people, e.g. $P = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z\}$, each with skills or capabilities.

An aspect of project management is to assign (a set of) people to each role.

such that each person $p \in assign(r)$ assigned to role r has at least the skills and capabilities required by role r .

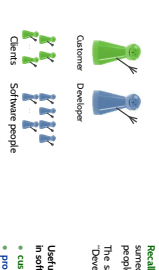
Note: $assign$ may change over time, there may be different assignments for different phases.

Sanity check: ensure that $assign(r) \neq \emptyset$ for each role r .

Example:

$assign = \{ \{PM\} \rightarrow A, \{PR\} \rightarrow B, \{SE\} \rightarrow C, \{QA\} \rightarrow D, \{QA\} \rightarrow E, \{QA\} \rightarrow F, \{QA\} \rightarrow G, \{QA\} \rightarrow H, \{QA\} \rightarrow I, \{QA\} \rightarrow J, \{QA\} \rightarrow K, \{QA\} \rightarrow L, \{QA\} \rightarrow M, \{QA\} \rightarrow N, \{QA\} \rightarrow O, \{QA\} \rightarrow P, \{QA\} \rightarrow Q, \{QA\} \rightarrow R, \{QA\} \rightarrow S, \{QA\} \rightarrow T, \{QA\} \rightarrow U, \{QA\} \rightarrow V, \{QA\} \rightarrow W, \{QA\} \rightarrow X, \{QA\} \rightarrow Y, \{QA\} \rightarrow Z \}$

Useful and Common Roles

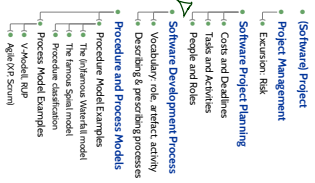


Real-life roles "Customer" and "Developer" are assumed by **multiple** persons, which often represent many people.

The same legal person may act as "Customer" and "Developer" in the same project.

Useful and common roles in software projects:

- customer, user
- project manager
- systems analyst
- software architect, designer
- (lead) developer
- programmer, QA, test engineer
- systems administrator
- interface clients, legislator
- norm/standard supervisory committee



Software Development Process

- Process –
- (1) A sequence of steps performed for a given purpose; for example, the software development process.
 - (2) See also: task, job
 - (3) To perform operations on data

IEEE 6002.1 (1992)

Software Development Process –

The process by which user needs are translated into a software product. The process involves formulating user needs into software requirements, transforming the software requirements into design and code, testing and validating the code, and sometimes, installing and checking out the software for operational use.

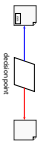
IEEE 6002.1 (1992)

- The process of a software development project may be
 - implicit,
 - informally agreed on, or
 - explicitly prescribed by a procedure or process model.
- **No**: each software development project has a process!

Describing Software Development Processes

Over time, the following notions proved useful to describe an artifact (→ in a model) software development processes:

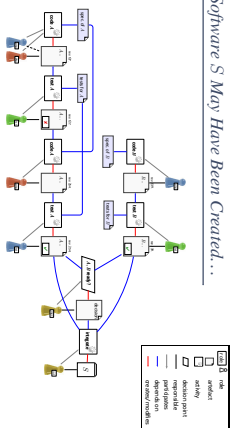
- **role** – has responsibilities and rights, needs skills and capabilities. In particular, responsibility for artifacts, participates in activities.
- **artifact** – all documents, evaluation methods, software modules, etc., all products emerging during a development process. If processed by activities, may have state.
- **activity** – any processing of artifacts, manually or automatic.
- Depends on artifacts, creates/modifies artifacts.



- **decision point** – specific case of activity. A decision is made based on artifacts in a certain state!

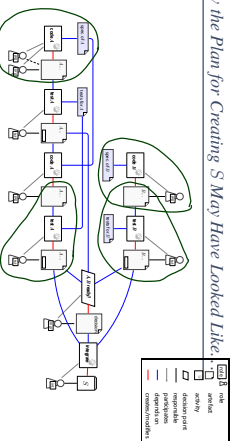
Denote phases, corresponds to milestones

How Software S May Have Been Created...

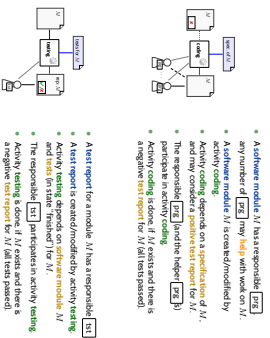


- S consists of modules A and B.
- Assume specifications and test cases for A and B were available.
- Person coded B (according to spec). Then person tested B with test cases, no errors found.
- Person coded A, with the help of person B. Then person tested A, some errors found.
- Person fixed A, person tested again, no errors found.
- A and B ready caused a positive decision, then person integrated A and B and obtained S.

How the Plan for Creating S May Have Looked Like

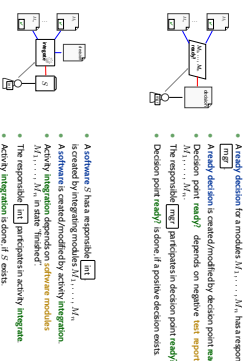


- S consists of modules A and B; specifications and test cases for A and B are available.
- Some milestones (circles) occur: Person B tests B (with test cases) and creates test report.
- Some milestones (circles) occur: Person A tests A, and creates test report.
- There is a fixed some single milestones A, same milestones, and creates test report.
- If A and B ready causes a positive decision, then some milestones integrated A and B and obtains S.



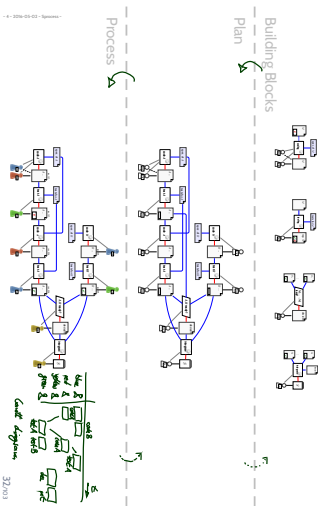
Building Blocks Can Be Arbitrarily Complicated

- **Example:** Distinguish coding and fixing software
- If there is a negative test result for M_i
- A **Lead(ing) architect** is responsible for fixing M_i
- The **Programmer** who was responsible for the initial version acts as **test case**; in addition to the **responsibility of M_i**
- a report/analysis of the error/documentation of this fix is created
- Using such building blocks, the project management can **precise** particular procedures, analyze, which roles need to be filled in a project, avoid to "forget" things



Content

- **(Software) Project**
- **Project Management**
 - Excursion Risk
- **Software Project Planning**
 - Costs and Deadlines
 - Resource and Activities
 - People and Roles
- **Software Development Process**
 - Vocabulary role, artefact, activity
 - Describing & prescribing processes
- **Procedure and Process Models**
 - Procedure Model Examples
 - The Johnson-Vincent model
 - The Irons-Siegel model
 - Procedure classification
 - Process Model Examples
 - V-Model (RUP)
 - Agile (XP, Scrum)



Process vs. Procedure Models

process description – documented expression of a set of activities performed to achieve a given purpose.

NOTE A process description provides an operational definition of the major components of a The description specifies, in a complete, precise, and verifiable manner, the requirements, design, behavior, or other characteristics of a process. It also may include procedures for determining whether those provisions have been satisfied. Process descriptions can be found at the activity, project, or organizational level. **IEEE 24765 (2010)**

process reference model – a model comprising definitions of processes in a life cycle described in terms of process purpose and outcomes, together with an architecture describing the relationships between the processes. **IEEE 24765 (2010)**

Procedure Models

(Ludewig and Lehner, 2013) propose to distinguish **process model** and **procedure model**.

- A **Process model** (Prozessmodell) comprises
 - Procedure model** (Verfahrenmodell) e.g., "waterfall model" (70%/80%)
 - Organisational structure** – comprising requirements on project management and responsibilities, quality assurance, documentation, document structure, revision control.
- e.g. V-Modell, RUP, XP, 90/10/0/4
- In the literature, **process model** and **procedure model** are often used as synonyms; there is not universally agreed distinction.

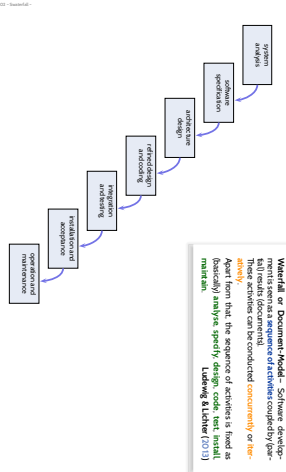
Procedure Model (?): Code and Fix

Code and Fix - denotes an approach, where coding and correction alternating with ad-hoc tests are the **only conditionally** conducted activities of software development. **Ludewig & Lehner (2013)**

- **Advantages:**
 - Corresponds to our desire to "get ahead" to solve the stated problem quickly.
 - The conducted activities (coding and ad-hoc testing) are easy.
- **Disadvantages:**
 - It is hard to plan the proper means to address specific decisions.
 - It is hard to distribute work over multiple persons or groups. (**see waterfall**)
 - If requirements are not stated, there is no notion of **error-free** (= meeting requirements).
 - Tests are lacking expected outcomes (otherwise, e.g. derived from requirements).
 - Resulting programs often hard to maintain.
 - Effort for maintenance high; most errors are only detected in operation.
 - Important concepts and decisions are not documented, but only in the heads of the developers (harder to transfer).

- **"economy of thought"**
 - don't reinvent principles
- **quantification, reproducibility**
 - one can assess the quality of **new** products are created (= QMII).
 - Identify weaknesses, learn from (bad) experience, improve the process.
- **fewer errors**
 - fewer errors in models cannot be forgotten because the "ready" decision point dependent on model with "test passed" flagged.
 - **clear responsibilities**
 - fewer "I thought so id. fr. the model!"
- **Process modeling is really *overdone*** – the best process model is **not used** (your software people don't use it).
- **Before introducing a process model**
 - assess what the new/changed process model makes matters better or worse (= metrics)
 - **Note:** customer may require a certain process model.

The (In)Famous Waterfall Model (Roscoe, 1967)



References

- Abrahamson, P., Salo, O., Ronkainen, J., and Warsta, J. (2002). Agile software development methods: review and analysis. Technical Report 478.
- Beck, K. (1999). *Extreme Programming Explained - Embrace Change*. Addison-Wesley.
- Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21(5), 61-72.
- Burnham, K., Dittmann, L., Henkel, B., and Müller, M. (2006). *SPICE in der Praxis: Interpretationshilfe für Anwender und Assessorn*. dpunktverlag.
- IEEE (1990). *IEEE Standard Glossary of Software Engineering Terminology*. Std 61012-1990.
- ISO/IEC/IEEE (2010). *Systems and software engineering - Vocabulary*. 2, 42452-01(01).
- Ludewig, J. and Lichter, H. (2013). *Software Engineering*. dpunktverlag, 3. edition.
- Roosch, E. (1987). *Developing Computer-based Information Systems*. John Wiley and Sons.
- Schubert, K. (1995). *SCRM - Softwareentwicklung*. In Schäfers, J. et al., editors. *Business Object Design and Implementation*. ODP&L/95 Workshop Proceedings. Springer-Verlag.
- Team, C. P. (2010). *Grant for development, version 1.3*. Technical Report ESCTR-2010-033. OML/SEI.
- Thayer, R. H. (1997). *Iterativ - Software Engineering Project Management*. IEEE Society Press, revised edition.
- V-MoDeLL XT (2008). *V-MoDeLL XT, Version 1.4*.
- Zülligbroven, H. (2005). *Object-Oriented Construction Handbook - Developing Application-Oriented Software with the UML and Rational Approach*. dpunktverlag/Vieweg+Tutmann.