Softwaretechnik / Software-Engineering

Lecture 3: Software Project Management

2019-05-02

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

Topic Area Project Management: Content

VL 2

- **Software Metrics**
  - Metrics, Properties of Metrics
  - Software Metrics
  - Software Metrics Issues

⋮

- **Cost Estimation**
  - (Software) Economics in a Nutshell
  - Software Cost Estimation
  - Expert's / Algorithmic Estimation

VL 3

⋮

- **Project Management**
  - Project
  - Process and Process Modelling
  - Procedure Models
  - Process Models

VL 4

⋮

- **Process Metrics**
  - CMMI, Spice

Content

- **Cost Estimation**
  - Software Cost Estimation
    - Expert's Estimation (Delphi Method)
    - Algorithmic Estimation (COCOMO, Function Points)
- **(Software) Project**
- **Project Management**
  - Goals, Common Activities
  - Excursion: Risk
- **Software Development Processes**
  - Roles, Artefacts, Activities
  - Costs and Deadlines
    - phase, milestone, deadline
    - cycle, life cycle, software life cycle
- **Development Process Modelling**
  - process vs. process model
- **Procedure and Process Models**
  - "Code and Fix"
  - The (infamous) Waterfall Model

Software Cost Estimation Cont'd

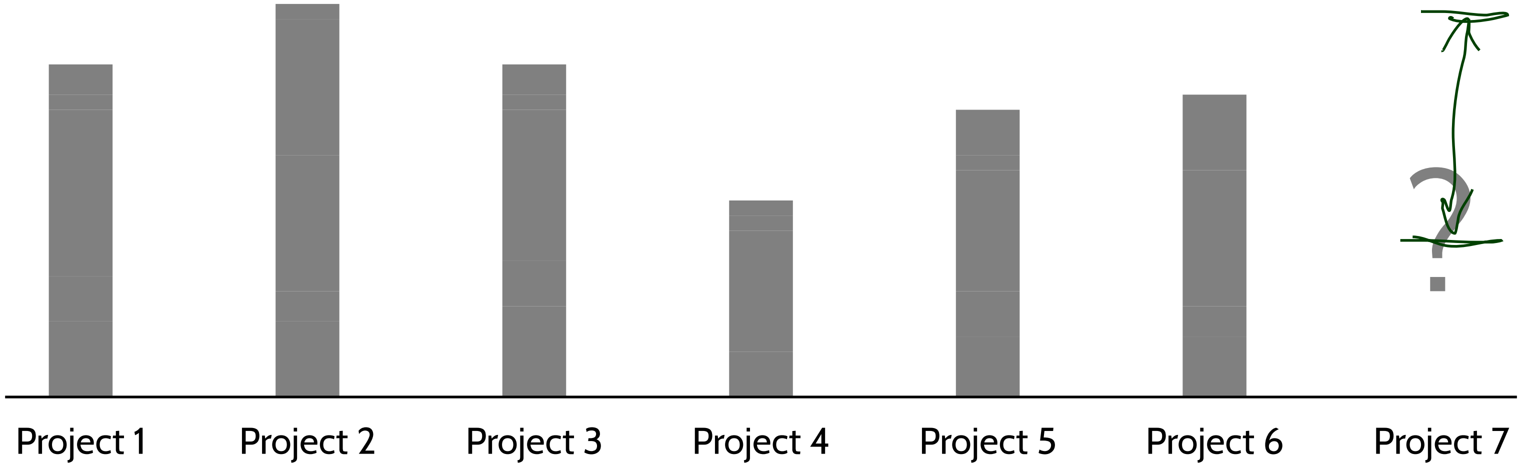
Principles of Software Cost Estimation

In the end, it's **experience**, **experience**, **experience**:

> "Estimate, document, estimate better." (Ludewig and Lichter, 2013)

Example:

- Assume these were the overall costs of previous, all similar projects:



- What could be an estimate of the new (also similar) Project 7?

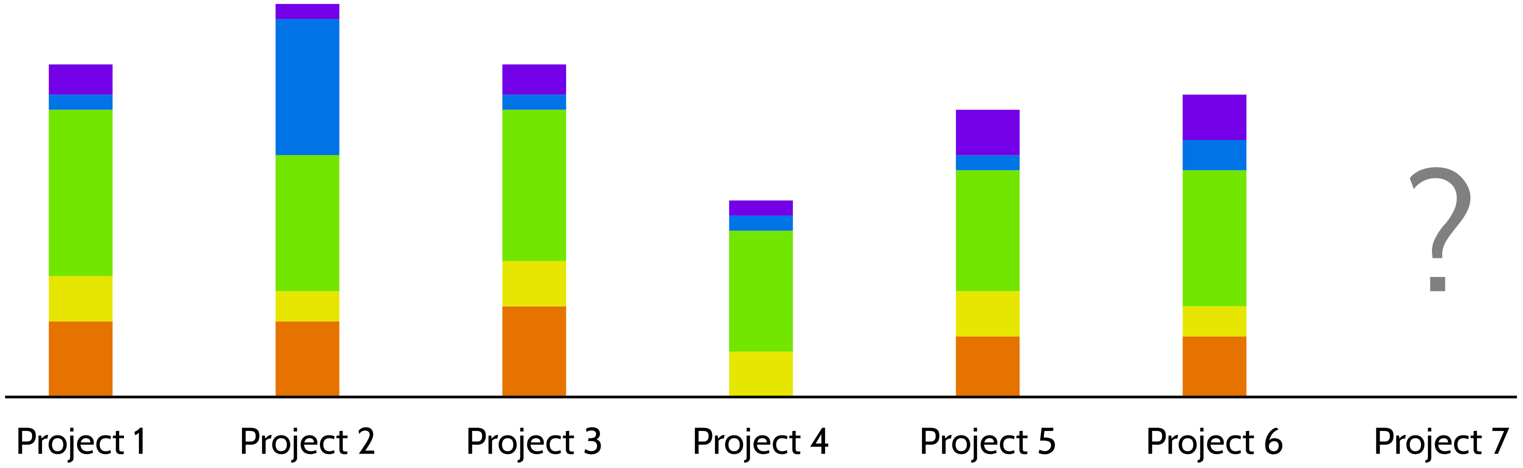
Principles of Software Cost Estimation

In the end, it's **experience**, **experience**, **experience**:

> "Estimate, document, estimate better." (Ludewig and Lichter, 2013)

Example:

- Assume these were the overall costs of previous, all similar projects:



- What could be an estimate of the new (also similar) Project 7?
- For a better estimate: analyse what costs are composed of.

  Maybe, Project 4 could re-use parts of Project 3, maybe Project 2 is the only one with a new customer. For Project 7 check: can we re-use parts? Is it a new customer?

A Classification of Software Costs

Distinguish **current cost** ('laufende Kosten'), e.g.

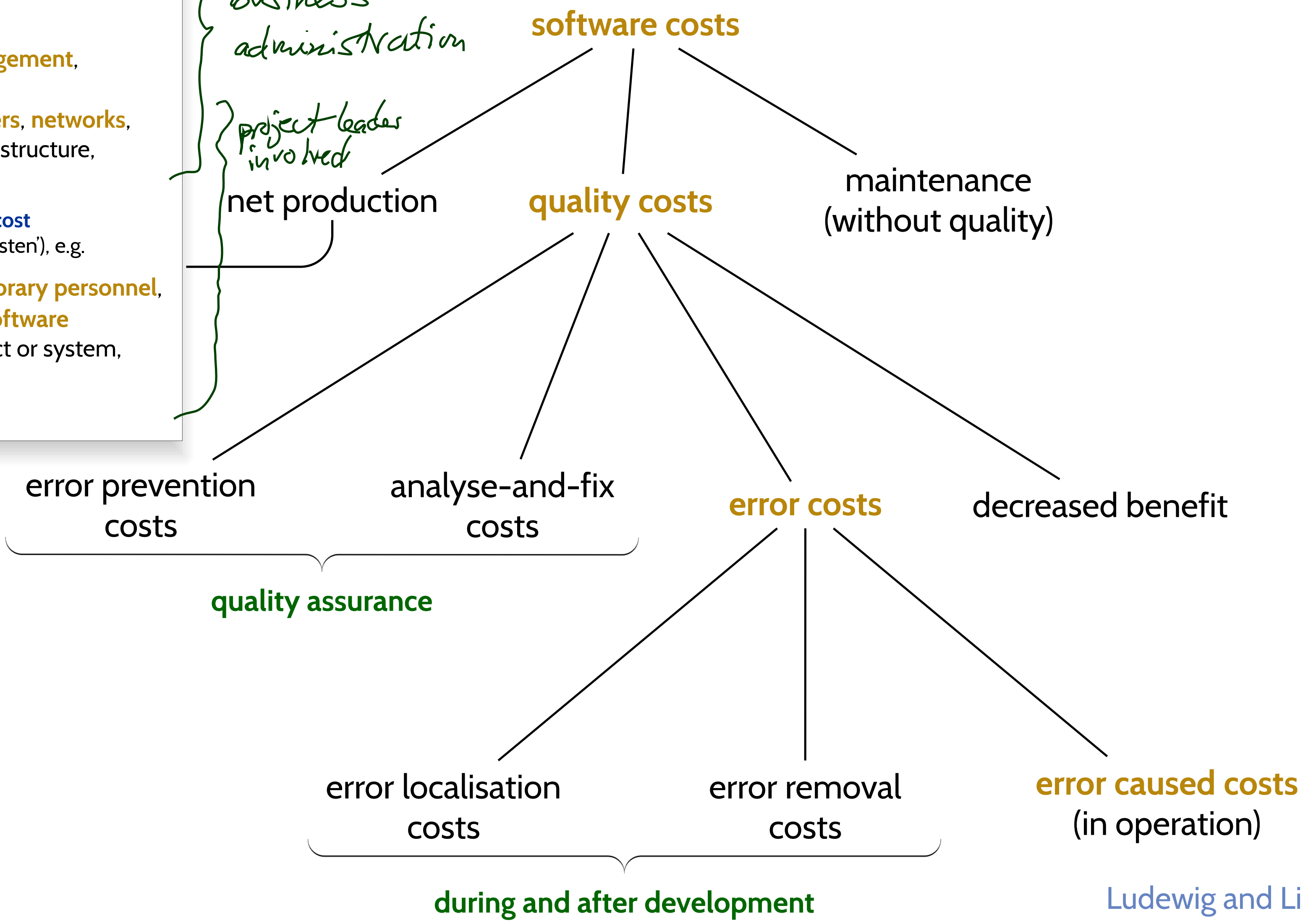
- fixed **personnel**,
- (business) **management**, **marketing**,
- **rooms**, **computers**, **networks**, **software** as infrastructure,
- …

business administration

and **project-related cost** ('projektbezogene Kosten'), e.g.

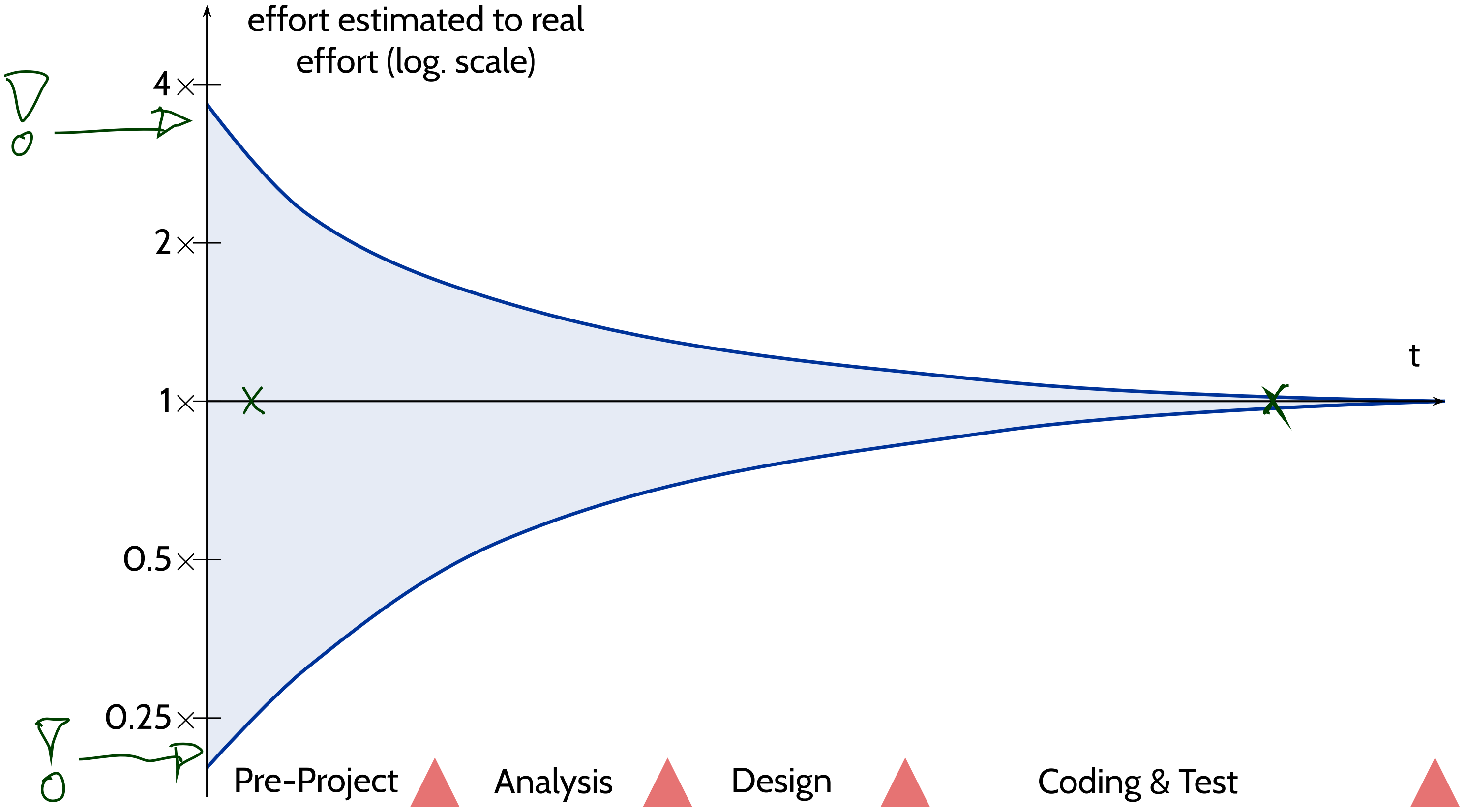
- additional **temporary personnel**,
- **hardware and software** as part of product or system,
- **contract costs**,
- …

project leader involved



Ludewig and Lichter (2013)

The "Estimation Funnel"



Uncertainty with estimations (following (Boehm et al., 2000), p. 10).

Visualisation: Ludewig and Lichter (2013)

Approaches to Software Cost Estimation

- **Expert's Estimation**

  For example,

  - **Delphi Method**

- **Algorithmic Estimation**

  For example,

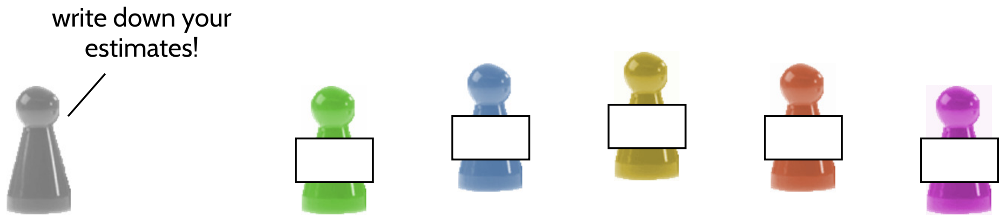
  - **COCOMO**
  - **Function Points**

Expert's Estimation

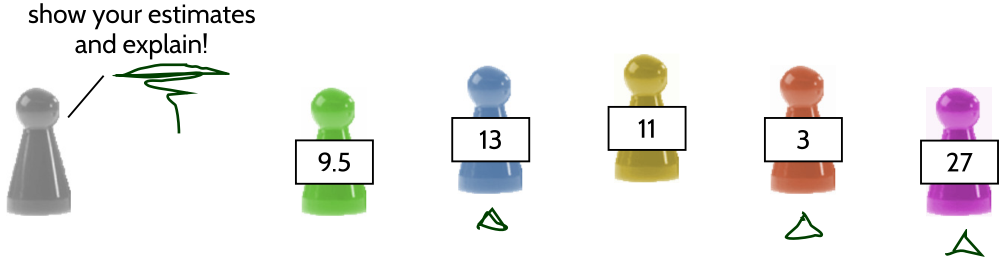
Expert's Estimation

One approach: the **Delphi** method.

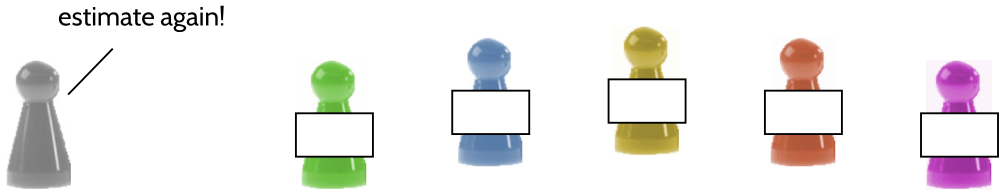
- Step 1:



- Step 2:



- Step 3:



- Then take the median, for example.

Algorithmic Estimation: COCOMO

Algorithmic Estimation: COCOMO

- **Co**nstructive **Co**st **Mo**del:

  Formulae which fit a huge set of archived project data (from the late 70's).
- Flavours:
  - COCOMO 81 (Boehm, 1981): variants **basic**, **intermediate**, **detailed**
  - COCOMO II (Boehm et al., 2000)
- All flavours are based on estimated program size S measured in DSI (Delivered Source Instructions) or kDSI (1000 DSI).
- Factors like security requirements or experience of the project team are mapped to values for parameters of the formulae.

- COCOMO examples:
  - textbooks like Ludewig and Lichter (2013) (most probably made up)
  - an exceptionally large example:
    COCOMO 81 for the Linux kernel (Wheeler, 2006) (and follow-ups)

COCOMO 81

Characteristics of the Type				a	b	Software Project Type
Size	Innovation	Deadlines/ Constraints	Dev. Environment			
Small (<50 KLOC)	Little	Not tight	Stable	3.2	1.05	Organic
Medium (<300 KLOC)	Medium	Medium	Medium	3.0	1.12	Semi-detached
Large	Greater	Tight	Complex HW/ Interfaces	2.8	1.20	Embedded

estimation 10^3 delivered source instruction

Basic COCOMO:

- **effort required**: $E = a \cdot (S/\underline{kDSI})^b$ [PM (person-months)]
- **time to develop**: $T = c \cdot E^d$ [months]
- **headcount**: $H = E/T$ [FTE (full time employee)]
- **productivity**: $P = S/E$ [DSI per PM] (← use to check for plausibility)

Intermediate COCOMO:

$$E = M \cdot a \cdot (S/kDSI)^b \quad \text{[person-months]}$$

$$M = RELY \cdot CPLX \cdot TIME \cdot ACAP \cdot PCAP \cdot LEXP \cdot TOOL \cdot SCED$$

COCOMO 81: Some Cost Drivers

$$M = RELY \cdot CPLX \cdot TIME \cdot ACAP \cdot PCAP \cdot LEXP \cdot TOOL \cdot SCED$$

factor		very low	low	normal	high	very high	extra high
RELY	required software reliability	0.75	0.88	1	1.15	1.40	
CPLX	product complexity	0.70	0.85	1	1.15	1.30	1.65
TIME	execution time constraint			1	1.11	1.30	1.66
ACAP	analyst capability	1.46	1.19	1	0.86	0.71	
PCAP	programmer capability	1.42	1.17	1	0.86	0.7	
LEXP	programming language experience	1.14	1.07	1	0.95		
TOOL	use of software tools	1.24	1.10	1	0.91	0.83	
SCED	required development schedule	1.23	1.08	1	1.04	1.10	

- **Note**: what, e.g., "extra high" TIME means, may depend on project context. (Consider data from previous projects.)

COCOMO II (Boehm et al., 2000)

Consists of

- **Application Composition Model** – project work is configuring components, rather than programming
- **Early Design Model** – adaption of **Function Point** approach (in a minute); does not need completed architecture design
- **Post-Architecture Model** – improvement of **COCOMO 81**; needs completed architecture design, and size of components estimatable

COCOMO II: Post-Architecture

$$E = 2.94 \cdot S^X \cdot M$$

- **Program size**: $S = (1 + REVL) \cdot (S_{new} + S_{equiv})$
  - **requirements volatility** $REVL$:
    e.g., if new requirements make 10% of code unusable, then $REVL = 0.1$
  - S_{new}: estimated size minus size w of **re-used code**,
  - $S_{equiv} = w/q$, if writing new code takes q-times the effort of re-use.
- **Scaling factors**:
  $X = \delta + \omega, \quad \omega = 0.91, \quad \delta = \frac{1}{100} \cdot (PREC + FLEX + RESL + TEAM + PMAT)$

factor		very low	low	normal	high	very high	extra high
PREC	**precedentness** (experience with similar projects)	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	**development flexibility** (development process fixed by customer)	5.07	4.05	3.04	2.03	1.01	0.00
RESL	**Architecture/risk resolution** (risk management, architecture size)	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	**Team cohesion** (communication effort in team)	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	**Process maturity** (see CMMI)	7.80	6.24	4.69	3.12	1.56	0.00

COCOMO II: Post-Architecture Cont'd

$$M = RELY \cdot DATA \cdot \dots \cdot SCED$$

group	factor	description
Product factors	RELY	required software reliability
	DATA	size of database
	CPLX	complexity of system
	RUSE	degree of development of reusable components
	DOCU	amount of required documentation
Platform factors	TIME	execution time constraint
	STOR	memory consumption constraint
	PVOL	stability of development environment
Team factors	ACAP	analyst capability
	PCAP	programmer capability
	PCON	continuity of involved personnel
	APEX	experience with application domain
	PLEX	experience with development environment
	LTEX	experience with programming language(s) and tools
Project factors	TOOL	use of software tools
	SITE	degree of distributedness
	SCED	required development schedule

(also in COCOMO 81, new in COCOMO II)

Algorithmic Estimation: Function Points

Algorithmic Estimation: Function Points

	Complexity			Sum
Type	low	medium	high	
input	____ $\cdot 3 =$	____ $\cdot 4 =$	____ $\cdot 6 =$	
output	____ $\cdot 4 =$	____ $\cdot 5 =$	____ $\cdot 7 =$	
query	____ $\cdot 3 =$	____ $\cdot 4 =$	____ $\cdot 6 =$	
user data	____ $\cdot 7 =$	____ $\cdot 10 =$	____ $\cdot 15 =$	
reference data	____ $\cdot 5 =$	____ $\cdot 7 =$	____ $\cdot 10 =$	
Unadjusted function points		UFP		
Value adjustment factor		VAF		
Adjusted function points		AFP = UFP · VAF		

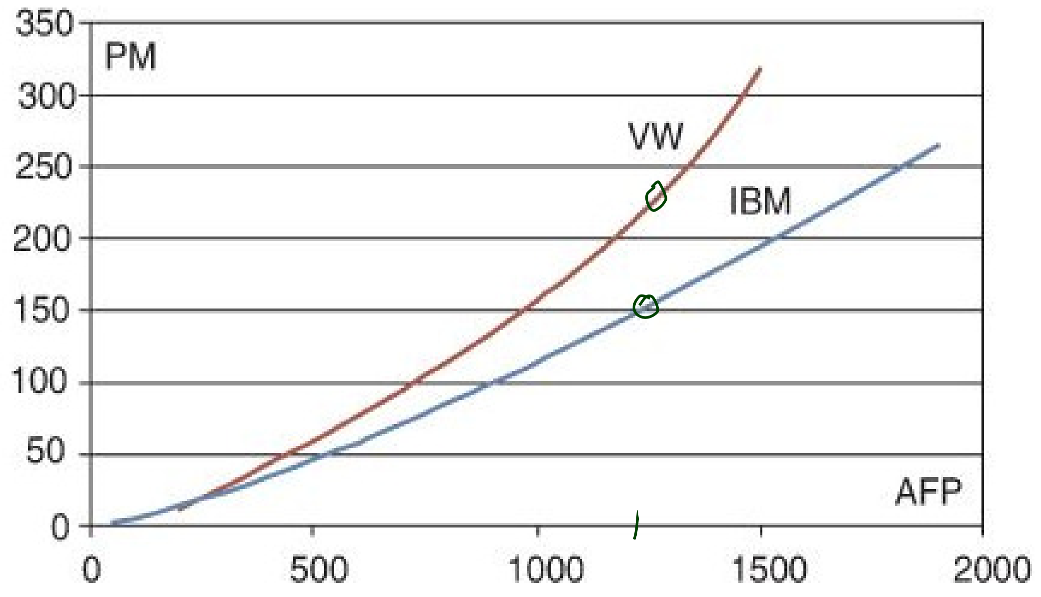
UFP

$$VAF = 0.65 + \frac{1}{100} \cdot \sum_{i=1}^{14} GSC_i,$$

$$0 \leq GSC_i \leq 5.$$

Algorithmic Estimation: Function Points

	Co			
Type	low	me		
input	____$\cdot 3 =$	____		
output	____$\cdot 4 =$	____		
query	____$\cdot 3 =$	____		
user data	____$\cdot 7 =$	____		
reference data	____$\cdot 5 =$	____$\cdot 7 =$	____$\cdot 10 =$	
Unadjusted function points		UFP		
Value adjustment factor		VAF		
Adjusted function points		AFP = UFP · VAF		



IBM and VW curve for the conversion from AFPs to PM according to (Noth and Kretzschmar, 1984) and (Knöll and Busse, 1991).

$$VAF = 0.65 + \frac{1}{100} \cdot \sum_{i=1}^{14} GSC_i,$$

$$0 \leq GSC_i \leq 5.$$

Discussion

Cost Estimation is Everywhere

- For example: **Bachelor's Thesis**

  Estimation Task: Which results can I promise to deliver in 3 months time?

- **Suggestion**: start to **quantify** your experience **now**.
  - **Take notes on your projects**:

    (e.g., Softwarepraktikum, Bachelor Projekt, Bachelor's Thesis, Master Projekt, Master's Thesis, …)
    - timestamps,
    - size of program created,
    - number of errors found,
    - number of pages written,
    - etc. …
  - **Try to identify factors**: what hindered productivity, what boosted productivity, …
  - Which **detours and mistakes** were **avoidable** in hindsight? How?

Content

- **Cost Estimation**
  - Software Cost Estimation
    - Expert’s Estimation (Delphi Method)
    - Algorithmic Estimation (COCOMO, Function Points)
- **(Software) Project**
- **Project Management**
  - Goals, Common Activities
  - Excursion: Risk
- **Software Development Processes**
  - Roles, Artefacts, Activities
  - Costs and Deadlines
    - phase, milestone, deadline
    - cycle, life cycle, software life cycle
- **Development Process Modelling**
  - process vs. process model
- **Procedure and Process Models**
  - “Code and Fix”
  - The (infamous) Waterfall Model

Project

Vocabulary: 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, then it is sometimes called a **software development project** or **software engineering project**. R. H. Thayer (1997)

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).

Project Management

Goals of Project Management

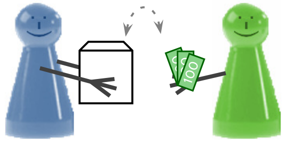
- **Main and general goal**:

  Have a **successful** project,
  i.e. the project delivers

  - defined **results**
  - in demanded **quality**
  - within scheduled **time**
  - using the assigned **resources**.

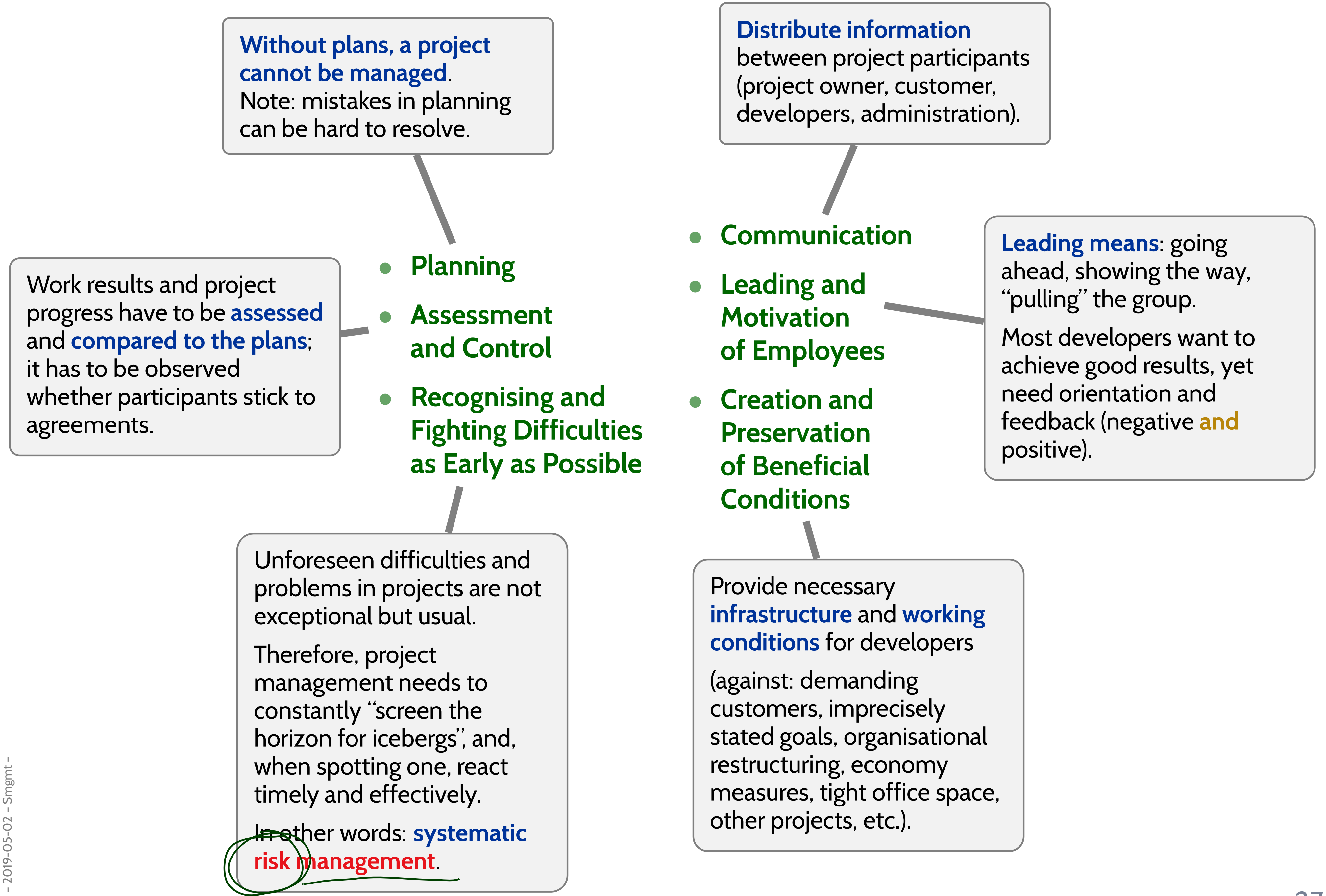
  There may be **secondary goals**, e.g.,

  - build or strengthen good **reputation** on market,
  - acquire **knowledge** which is useful for later projects,
  - develop **re-usable components** (to save resources later),
  - be attractive to **employees**.
  - …



software delivery

Common Activities of Project Management



- Planning
- Assessment and Control
- Recognising and Fighting Difficulties as Early as Possible
- Communication
- Leading and Motivation of Employees
- Creation and Preservation of Beneficial Conditions

Without plans, a project cannot be managed.
Note: mistakes in planning can be hard to resolve.

Work results and project progress have to be **assessed** and **compared to the plans**; it has to be observed whether participants stick to agreements.

Unforeseen difficulties and problems in projects are not exceptional but usual.

Therefore, project management needs to constantly "screen the horizon for icebergs", and, when spotting one, react timely and effectively.

In other words: **systematic risk management**.

Distribute information between project participants (project owner, customer, developers, administration).

Leading means: going ahead, showing the way, "pulling" the group.

Most developers want to achieve good results, yet need orientation and feedback (negative **and** positive).

Provide necessary **infrastructure** and **working conditions** for developers

(against: demanding customers, imprecisely stated goals, organisational restructuring, economy measures, tight office space, other projects, etc.).

Quick Excursion: Risk and Riskvalue

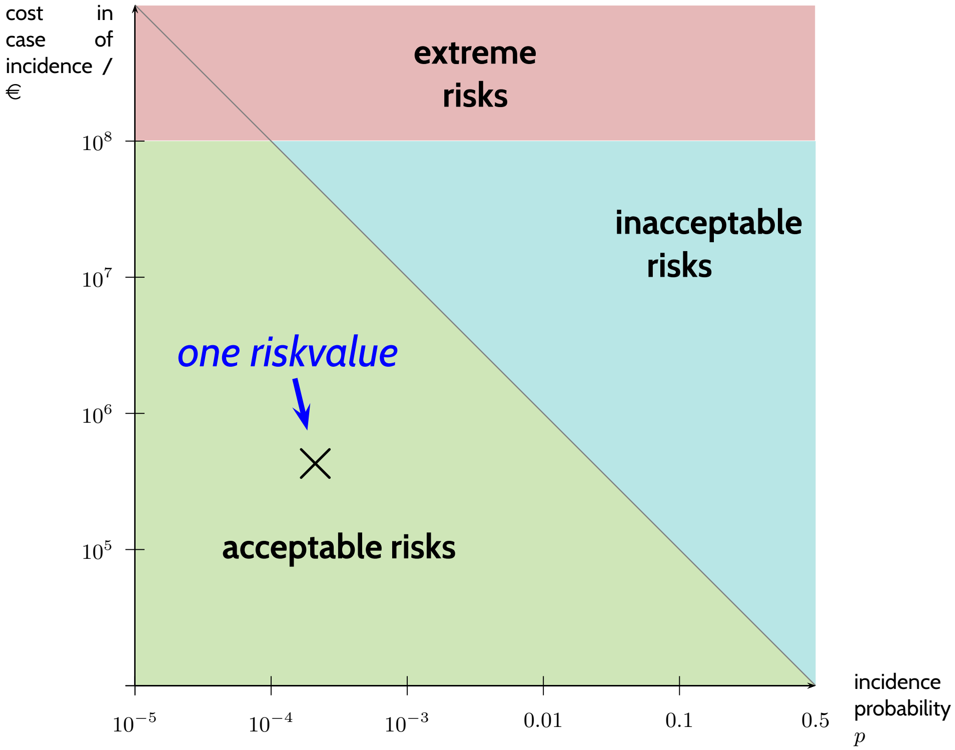
risk – a problem, which did not occur yet, but on occurrence threatens important project goals or results. Whether it will occur, cannot be surely predicted.

Ludewig & Lichter (2013)

$$\text{riskvalue} = p \cdot K$$

p: **probability** of problem occurrence,

K: **cost** in case of problem occurrence.



- **Avionics** requires: “Catastrophic Failure Conditions have Average Probability per Flight Hour of 10^{-9} (or ‘Extremely Improbable’)” (AC 25.1309-1).
- “problems with $p = 0.5$ are not risks, but environment conditions to be dealt with”

Content

- **Cost Estimation**
  - Software Cost Estimation
    - Expert's Estimation (Delphi Method)
    - Algorithmic Estimation (COCOMO, Function Points)
- **(Software) Project**
- **Project Management**
  - Goals, Common Activities
  - Excursion: Risk
- **Software Development Processes**
  - Roles, Artefacts, Activities
  - Costs and Deadlines
    - phase, milestone, deadline
    - cycle, life cycle, software life cycle
- **Development Process Modelling**
  - process vs. process model
- **Procedure and Process Models**
  - "Code and Fix"
  - The (infamous) Waterfall Model

Software Development Process

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. pursues a bunch of goals.
  - The most important goal is usually to create or modify software; this software is thus the result of the project, the **product**.

    Other important goals are extension of know-how, preparation of building blocks for later projects, or utilisation of employees.

  The project is called **successful** if the goals are reached to a high degree.
- Has a **recipient** (or will have one).
  - This recipient is the **customer**.
  - Later **users** (conceptionally) belong to the customer.
- **Connects people**, **results** (intermediate/final products), and **resources**.

  The **organisation** determines roles of and relations between peoples/results/resources, and the **external interfaces** of the project.

Ludewig & Lichter (2013)

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 610.12 (1990)

Software Development Process –
The process by which user needs are translated into a software product. The process involves **translating** user needs into **software requirements**, **transforming** the software requirements into **design**, **implementing** the design in **code**, **testing** the code, and sometimes, **installing and checking out** the software for **operational use**.

IEEE 610.12 (1990)

- The **process** of a software development **project** may be
  - implicit,
  - informally agreed on, or
  - explicitly prescribed (by a **procedure** or **process model**).
- **Note**: each software development project **has** a process!

Describing Software Development Processes

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

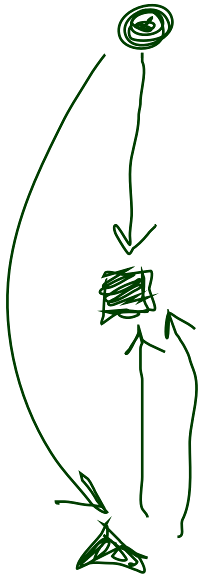
- **role** – has responsibilities and rights, needs skills and capabilities.

  In particular: has responsibility for **artefacts**, participates in **activities**.

- **artefact** (or **product**) – all documents, evaluation protocols, software modules, etc.; all products emerging during a development process.

  Is processed by **activities**, may have **state**.

- **activity** – any processing of artefacts, manually or automatic; solves tasks. Depends on **artefacts**, creates/modifies **artefacts**.



Describing Software Development Processes

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

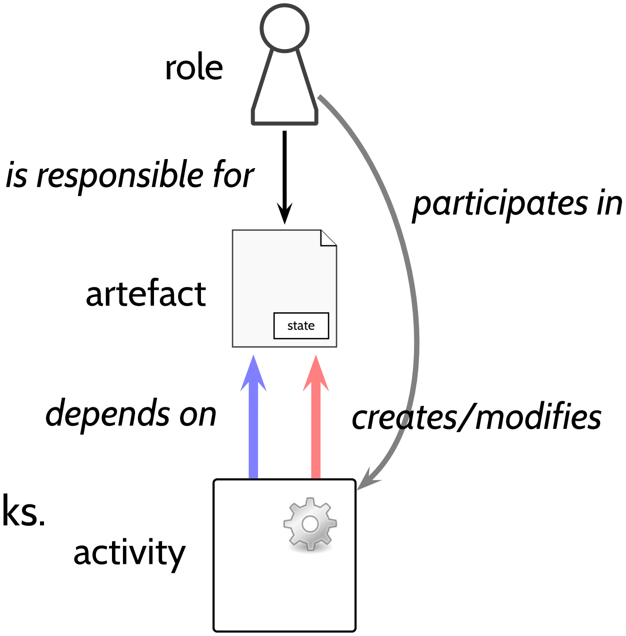
- **role** – has responsibilities and rights, needs skills and capabilities.

  In particular: has responsibility for **artefacts**, participates in **activities**.

- **artefact** (or **product**) – all documents, evaluation protocols, software modules, etc.; all products emerging during a development process.

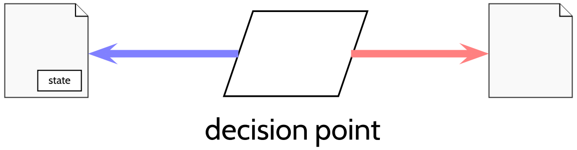
  Is processed by **activities**, may have **state**.

- **activity** – any processing of artefacts, manually or automatic; solves tasks. Depends on **artefacts**, creates/modifies **artefacts**.



- **decision point** – special case of activity: a decision is made based on **artefacts** (in a certain state), creates a **decision artefacts**.

  Delimits phases, may **correspond to milestone**.



The Concept of Roles

In a software project, at each point in time,
there is a set R of (active) **roles**, e.g. $R = \left\{ \boxed{\text{mgr}}, \boxed{\text{prg}}, \boxed{\text{tst}}, \boxed{\text{ana}} \right\}$.

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

For example,

- $\boxed{\text{mgr}}$: project manager
  - has the **right** to raise issue reports
  - is **responsible** for closing issue reports
- $\boxed{\text{prg}}$: programmer
  - has the **right** to change the code
  - is **responsible** for reporting unforeseen problems to the project manager
  - is **responsible** for respecting coding conventions
  - is **responsible** for addressing issue reports
- $\boxed{\text{tst}}$: test engineer
  - has the **right** to raise issue reports
  - is **responsible** for quality control

The Concept of Roles Cont'd

Given a set R of roles, e.g. $R = \left\{ \boxed{\text{mgr}}, \boxed{\text{prg}}, \boxed{\text{tst}}, \boxed{\text{ana}} \right\}$,

and a set P of people, e.g. $P = \{$ $\}$, each with **skills** or **capabilities**.

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

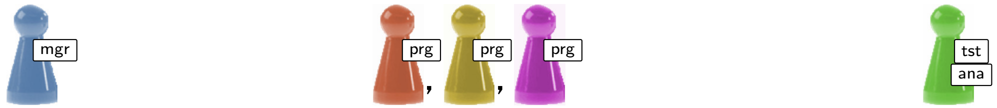
$$assign : R \to 2^P$$

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.

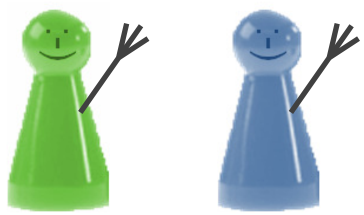
- **Example**:



one person, one role   multiple persons, one role   one person, multiple roles

$assign = \Big\{ \boxed{\text{mgr}} \mapsto \{$ $\}, \boxed{\text{prg}} \mapsto \{$ $\}, \boxed{\text{tst}} \mapsto \{$ $\}, \boxed{\text{ana}} \mapsto \{$ $\} \Big\}$

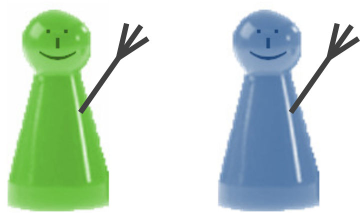
Useful and Common Roles

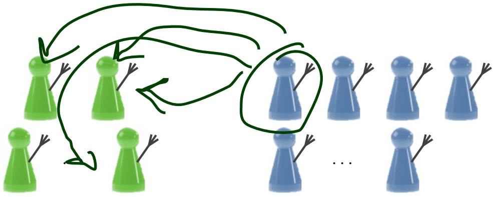


Recall: **roles** “Customer” and “Developer” are assumed by **legal persons**, which often represent many people.

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

Useful and Common Roles





Recall: **roles** “Customer” and “Developer” are assumed by **legal 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**
- **(sytems) analyst**
- **software architect**, **designer**
- (lead) **developer**
  programmer, **tester**, …
- **maintenance engineer**
- **systems administrator**
- **invisible clients**: legislator, norm/standard supervisory committee

Describing Software Development Processes

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

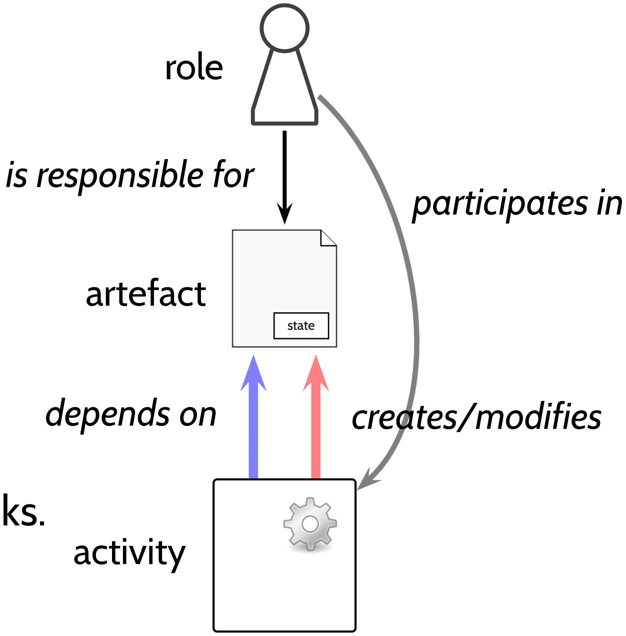
- **role** – has responsibilities and rights, needs skills and capabilities.

  In particular: has responsibility for **artefacts**, participates in **activities**.

- **artefact** (or **product**) – all documents, evaluation protocols, software modules, etc.; all products emerging during a development process.

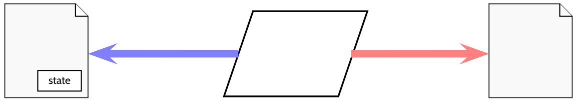
  Is processed by **activities**, may have **state**.

- **activity** – any processing of artefacts, manually or automatic; solves tasks. Depends on **artefacts**, creates/modifies **artefacts**.



- **decision point** – special case of activity: a decision is made based on **artefacts** (in a certain state), creates a **decision artefacts**.

  Delimits phases, may **correspond to milestone**.

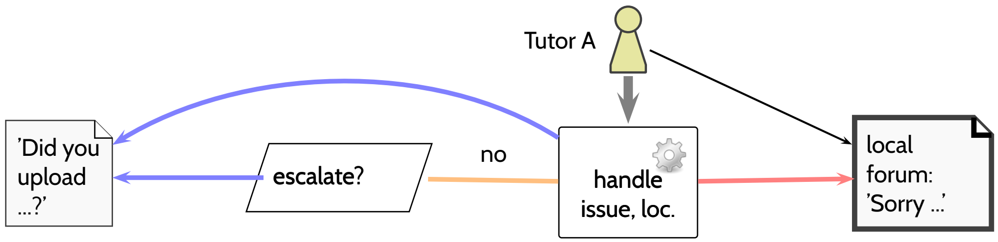


decision point

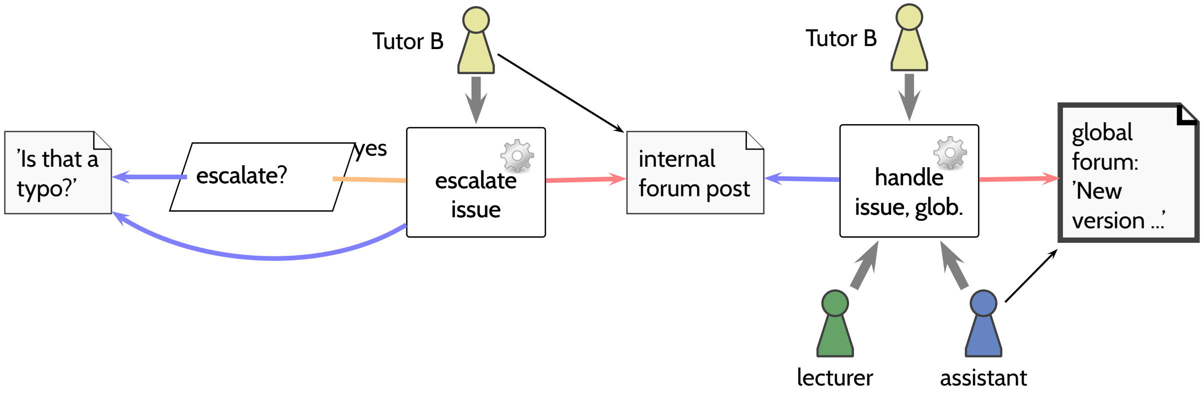
Describe Processes

Example: Forum Work of the Course

- A particular post is handled locally by Tutor A:
  - Friday, 2019-05-10, 19:37: a new post appears in the group forum: ‘Did you upload the notes?’
  - 20:03: Tutor A decides that the issue can be handled locally (by uploading the forgotten notes);
  - 20:21: Tutor A writes a local forum post ‘Sorry, forgot! Thanks for reminding.’

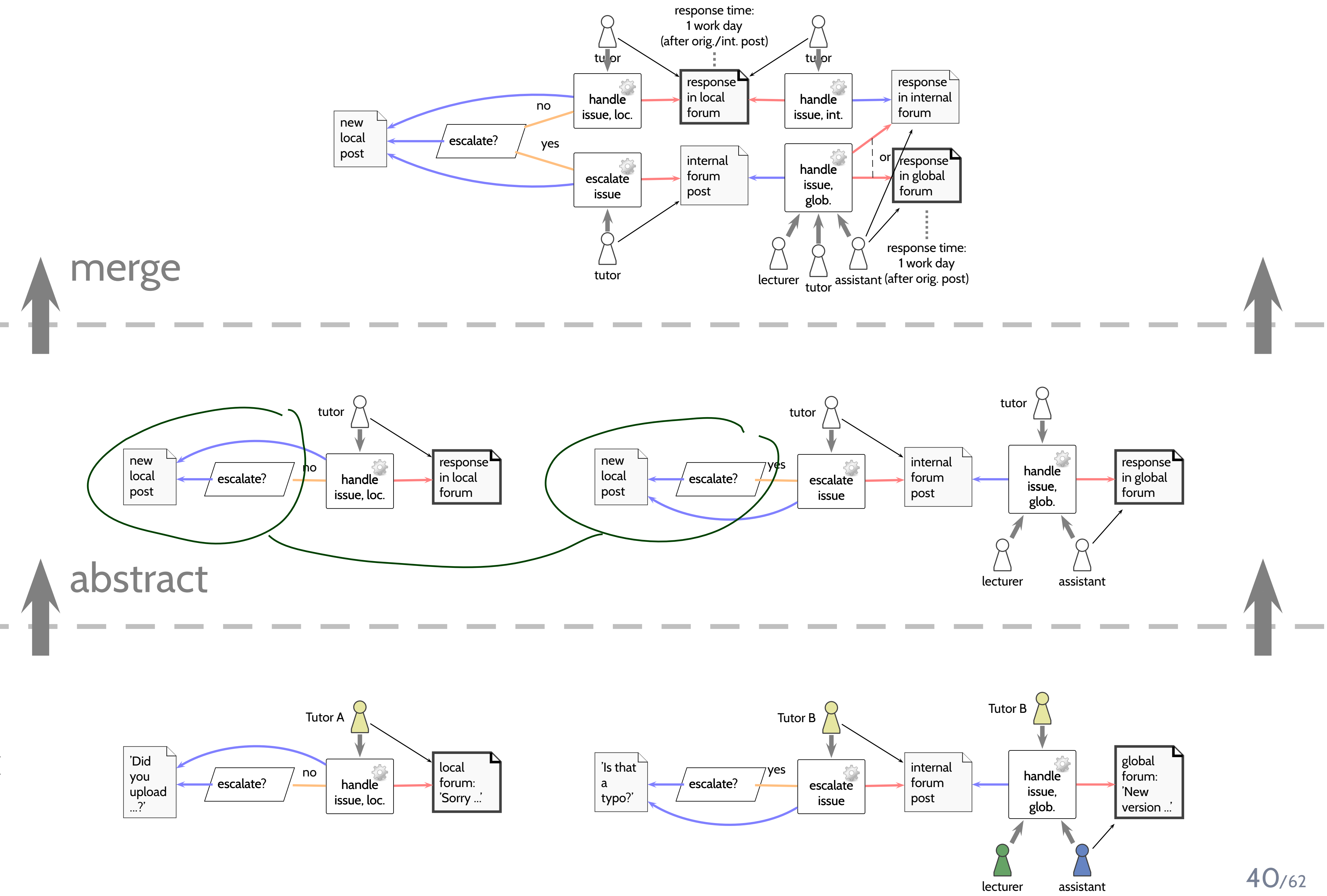


- A particular post needs to be escalated:
  - Monday, 2019-05-13, 14:01: a new post appears in the group forum: ‘Is that a typo?’
  - Tuesday, 2019-05-14, 9:59: Tutor B decides that the issues needs to be escalated.
  - Tuesday, 2019-05-14, 10:03: Tutor B writes a post to the internal forum
  - Tuesday, 2019-05-14, 12:47: Teaching Assistant contacts Lecturer
  - …
  - Tuesday, 2019-05-14, 13:59: Teaching Assistant writes a global posts ’New version is uploaded, sorry.’

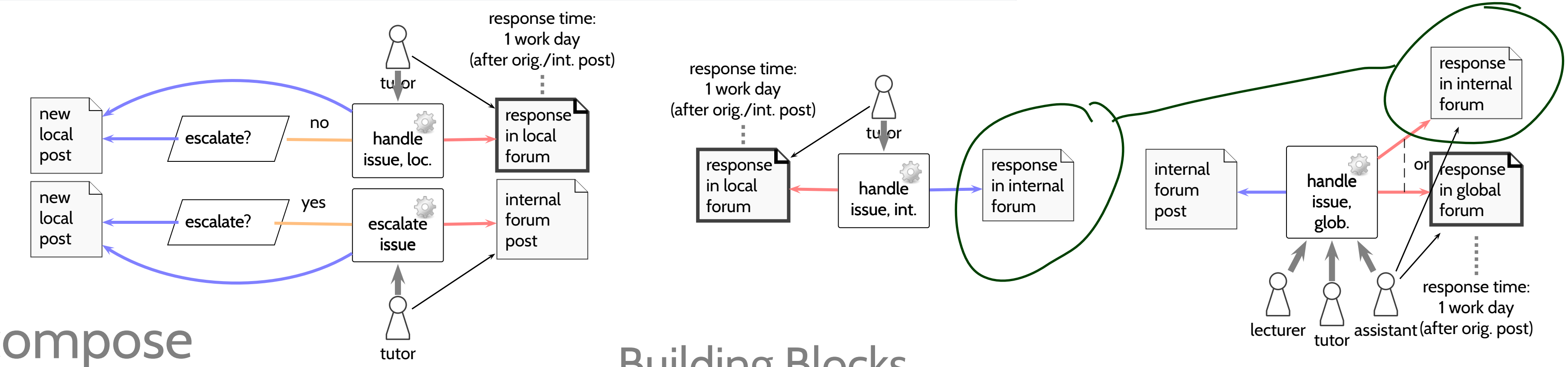


Software Project Planning: Process Modelling

From Concrete Process to Process Model



From Process Model to Concrete Process



new local post
escalate?
no
handle issue, loc.
tutor
response time: 1 work day (after orig./int. post)
response in local forum

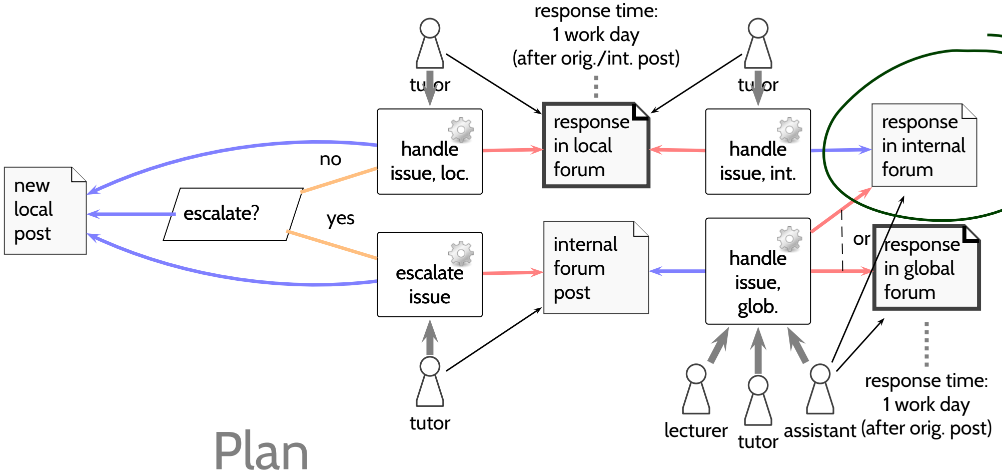
new local post
escalate?
yes
escalate issue
tutor
internal forum post

response time: 1 work day (after orig./int. post)
response in local forum
tutor
handle issue, int.
response in internal forum

internal forum post
handle issue, glob.
or
response in internal forum
response in global forum
lecturer
tutor
assistant
response time: 1 work day (after orig. post)

compose

Building Blocks

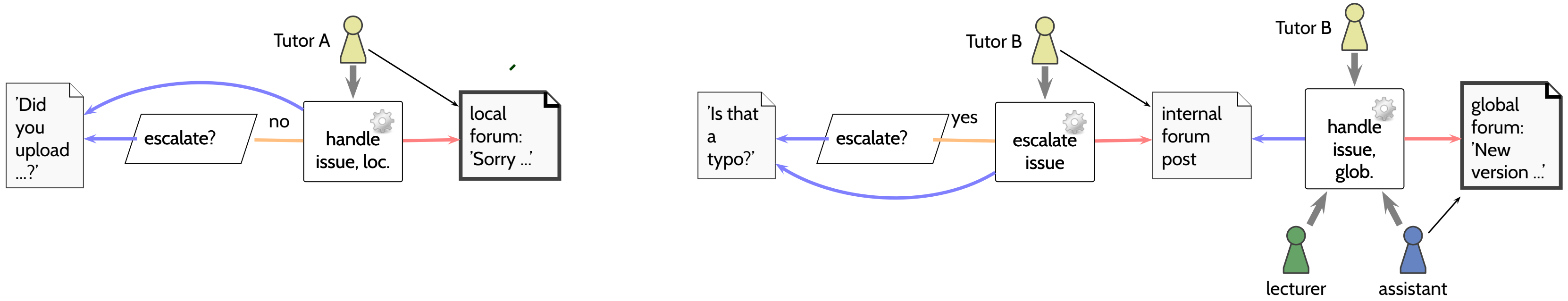


response time: 1 work day (after orig./int. post)
tutor
tutor
handle issue, loc.
no
new local post
escalate?
yes
response in local forum
handle issue, int.
response in internal forum
escalate issue
internal forum post
handle issue, glob.
or
response in global forum
tutor
lecturer
tutor
assistant
response time: 1 work day (after orig. post)

concretise

Plan

Process



Tutor A
'Did you upload ...?'
escalate?
no
handle issue, loc.
local forum: 'Sorry ...'

Tutor B
'Is that a typo?'
escalate?
yes
escalate issue
internal forum post

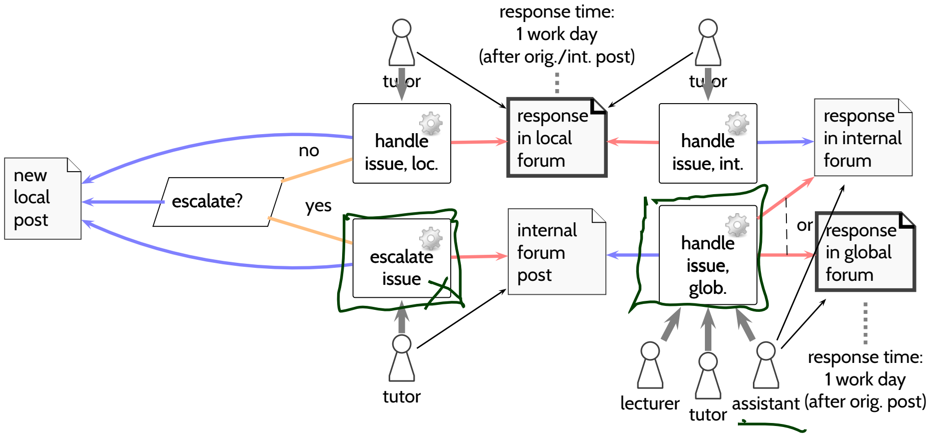
Tutor B
handle issue, glob.
global forum: 'New version ...'
lecturer
assistant

How to Read a Process Model

- A **process model** (as discussed so far) **defines dependencies**.

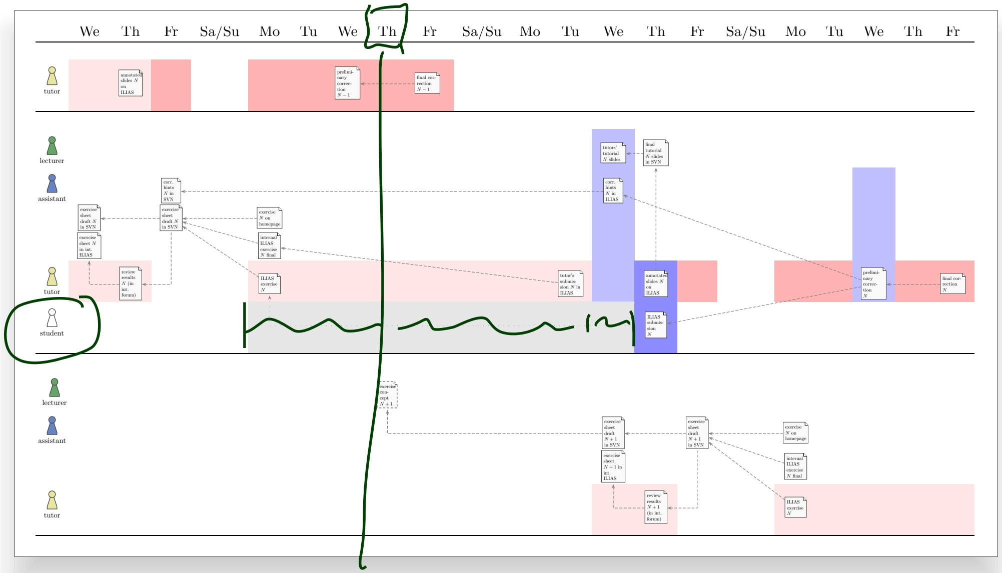
  → which artefacts needs to be available **before starting** which activity.
- A **process model does not**
  - define when (date/time) an activity starts.
  - say that Activity A must be completed before (depending) Activity B.

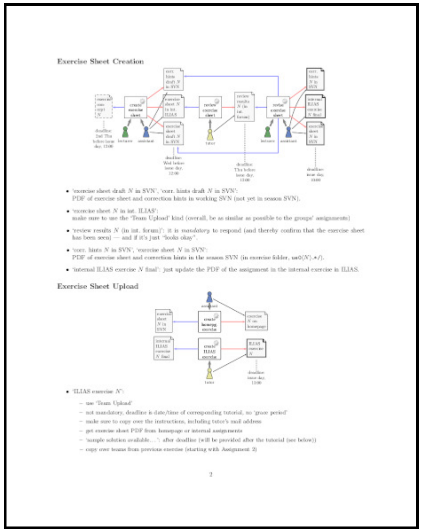
Example:



- Tuesday, 2019-05-14, 10:03: Tutor B writes a post to the internal forum:
  "This is what I know so far. I'll get back to the students and post more information later."
  → Activity '**escalate issue**' **started** (and **continues**)
- Tuesday, 2019-05-14, 12:47: Teaching Assistant contacts Lecturer
  → Activity '**handle issue glob.**' **started** (and **continues**)
- Tuesday, 2019-05-14, 12:54: Tutor B posts further information
  → Activity '**escalate issue**' **continues** (Tutor B is available for further questions)
- Tuesday, 2019-05-14, 13:03: Teaching Assistant writes to Tutor B: "Okay, thanks, we got it."
  → Activity '**escalate issue**' **completed**.

Example: Process Model of Tutorials


We
Th
Fr
Sa/Su
Mo
Tu
We
Th
Fr
Sa/Su
Mo
Tu
We
Th
Fr
Sa/Su
Mo
Tu
We
Th
Fr
tutor
annotated slides N on ILIAS
preliminary correction $N-1$
final correction $N-1$
lecturer
assistant
corr. hints N in SVN
exercise sheet draft N in SVN
exercise sheet draft N in SVN
exercise sheet N in int. ILIAS
exercise N on homepage
internal ILIAS exercise N final
tutors' tutorial N slides
final tutorial N slides in SVN
corr. hints N in ILIAS
tutor
review results N (in int. forum)
ILIAS exercise N
tutor's submission N in ILIAS
annotated slides N on ILIAS
preliminary correction N
final correction N
student
ILIAS submission N
lecturer
exercise concept $N+1$
assistant
exercise sheet draft $N+1$ in SVN
exercise sheet draft $N+1$ in SVN
exercise sheet $N+1$ in int. ILIAS
exercise N on homepage
internal ILIAS exercise N final
tutor
review results $N+1$ (in int. forum)
ILIAS exercise N



Exercise Sheet Creation

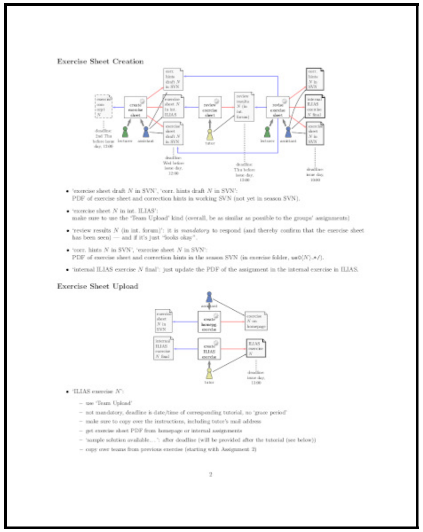
exercise concept N
create exercise sheet
corr. hints draft N in SVN
exercise sheet N in int. ILIAS
exercise sheet draft N in SVN
review exercise sheet
review results N (in int. forum)
revise exercise sheet
corr. hints N in SVN
internal ILIAS exercise N final
exercise sheet N in SVN
lecturer
assistant
tutor
deadline: 2nd Thu before issue day, 12:00
deadline: Wed before issue day, 12:00
deadline: Thu before issue day, 12:00
deadline: issue day, 10:00

- 'exercise sheet draft N in SVN', 'corr. hints draft N in SVN':
  PDF of exercise sheet and correction hints in working SVN (not yet in season SVN).
- 'exercise sheet N in int. ILIAS':
  make sure to use the 'Team Upload' kind (overall, be as similar as possible to the groups' assignments)
- 'review results N (in int. forum)': it is *mandatory* to respond (and thereby confirm that the exercise sheet has been seen) — and if it's just "looks okay".
- 'corr. hints N in SVN', 'exercise sheet N in SVN':
  PDF of exercise sheet and correction hints in the season SVN (in exercise folder, `ue0`(N)`_*/`).
- 'internal ILIAS exercise N final': just update the PDF of the assignment in the internal exercise in ILIAS.

Exercise Sheet Upload

- 'ILIAS exercise N':
  - use 'Team Upload'
  - not mandatory, deadline is date/time of corresponding tutorial, no 'grace period'
  - make sure to copy over the instructions, including tutor's mail address
  - get exercise sheet PDF from homepage or internal assignments
  - 'sample solution available…': after deadline (will be provided after the tutorial (see below))
  - copy over teams from previous exercise (starting with Assignment 2)

2



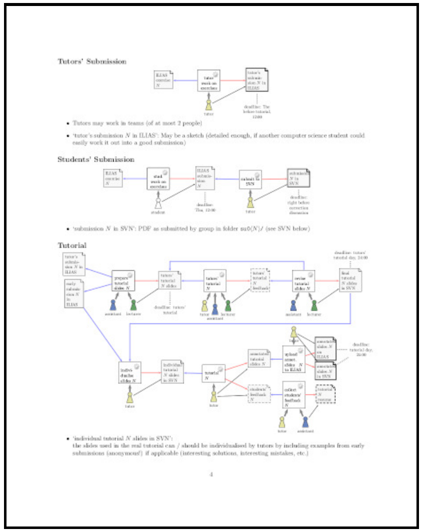
Exercise Sheet Creation

- 'exercise sheet draft N in SVN', 'corr. hints draft N in SVN':
  PDF of exercise sheet and correction hints in working SVN (not yet in season SVN).
- 'exercise sheet N in int. ILIAS':
  make sure to use the 'Team Upload' kind (overall, be as similar as possible to the groups' assignments)
- 'review results N (in int. forum)': it is *mandatory* to respond (and thereby confirm that the exercise sheet has been seen) — and if it's just "looks okay".
- 'corr. hints N in SVN', 'exercise sheet N in SVN':
  PDF of exercise sheet and correction hints in the season SVN (in exercise folder, `ue0`(N)`_*/`).
- 'internal ILIAS exercise N final': just update the PDF of the assignment in the internal exercise in ILIAS.

Exercise Sheet Upload

- 'ILIAS exercise N':
  - use 'Team Upload'
  - not mandatory, deadline is date/time of corresponding tutorial, no 'grace period'
  - make sure to copy over the instructions, including tutor's mail address
  - get exercise sheet PDF from homepage or internal assignments
  - 'sample solution available…': after deadline (will be provided after the tutorial (see below))
  - copy over teams from previous exercise (starting with Assignment 2)

2



Tutors' Submission

ILIAS exercise N
tutor work on exercises
tutor's submission N in ILIAS
tutor
deadline: Tue before tutorial, 12:00

- Tutors may work in teams (of at most 2 people)
- 'tutor's submission N in ILIAS': May be a sketch (detailed enough, if another computer science student could easily work it out into a good submission)

Students' Submission

ILIAS exercise N
stud. work on exercises
ILIAS submission N
submit to SVN
submission N in SVN
student
tutor
deadline: Thu, 12:00
deadline: right before correction discussion

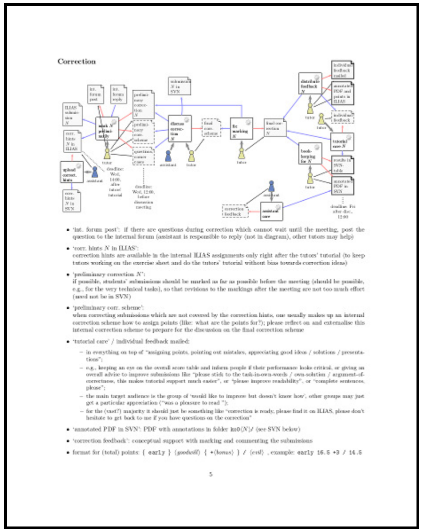
- 'submission N in SVN': PDF as submitted by group in folder `su0`(N)`/` (see SVN below)

Tutorial

tutor's submission N in ILIAS
early submission N in ILIAS
prepare tutorial slides N
tutors' tutorial N slides
tutors' tutorial N
tutors' tutorial N feedback
revise tutorial slides N
final tutorial N slides in SVN
assistant
lecturer
tutor
deadline: tutors' tutorial
deadline: tutors' tutorial day, 24:00
individualise slides N
individual tutorial N slides in SVN
tutorial N
annotated tutorial slides N
upload annot. slides N to ILIAS
annotated slides N on ILIAS
annotated slides N in SVN
students' feedback N
collect students' feedback N
tutorial N resume
deadline: tutorial day, 24:00

- 'individual tutorial N slides in SVN':
  the slides used in the real tutorial can / should be individualised by tutors by including examples from early submissions (anonymous!) if applicable (interesting solutions, interesting mistakes, etc.)

4



Correction

- 'int. forum post': if there are questions during correction which cannot wait until the meeting, post the question to the internal forum (assistant is responsible to reply (not in diagram), other tutors may help)
- 'corr. hints N in ILIAS':
  correction hints are available in the internal ILIAS assignments only right after the tutors' tutorial (to keep tutors working on the exercise sheet and do the tutors' tutorial without bias towards correction ideas)
- 'preliminary correction N':
  if possible, students' submissions should be marked as far as possible before the meeting (should be possible, e.g., for the very technical tasks), so that revisions to the markings after the meeting are not too much effort (need not be in SVN)
- 'preliminary corr. scheme':
  when correcting submissions which are not covered by the correction hints, one usually makes up an internal correction scheme how to assign points (like: what are the points for?); please reflect on and externalise this internal correction scheme to prepare for the discussion on the final correction scheme
- 'tutorial care' / individual feedback mailed:
  - in everything on top of "assigning points, pointing out mistakes, appreciating good ideas / solutions / presentations";
  - e.g., keeping an eye on the overall score table and inform people if their performance looks critical, or giving an overall advise to improve submissions like "please stick to the task-in-own-words / own-solution / argument-of-correctness, this makes tutorial support much easier", or "please improve readability", or "complete sentences, please";
  - the main target audience is the group of 'would like to improve but doesn't know how', other groups may just get a particular appreciation ("was a pleasure to read ");
  - for the (vast?) majority it should just be something like "correction is ready, please find it on ILIAS, please don't hesitote to get back to me if you have questions on the correction"
- 'annotated PDF in SVN': PDF with annotations in folder `ko0`(N)`/` (see SVN below)
- 'correction feedback': conceptual support with marking and commenting the submissions
- format for (total) points: `[ early ]` ⟨*goodwill*⟩ `[ +`⟨*bonus*⟩ `] /` ⟨*evil*⟩ , example: `early 16.5 +3 / 14.5`

5

Tell Them What You've Told Them…

- Cost Estimation
  - It's about **experience** (and based on data obtained with metrics), and often a **well-kept business secret**.
  - **Algorithmic Cost Estimations** "just" **shift** the estimation.
  - Cost estimation is **everywhere** (→ tutorials).
- **Project**: has (among others)
  - **project owner** and **leader**; **goals** (Excursion: **Risk**)
  - **process** – each project has one
- A **process model** relates
  - **roles**, **artefacts**, **activities**, **decision points**
  - relations: **responsibility**, **dependency**, **creation/modification**.
- **Use** process models
  - **descriptive** ("we did it like that"), or
  - **prescriptive** ("please do it like that")
- A process model can allow us to (→ exercises)
  - devise a **schedule** ('who does what when')
  - estimate and control **phases** and **deadlines**.
- Distinguish **process** and **procedure model**.

References

References

Boehm, B. W. (1981). *Software Engineering Economics*. Prentice-Hall.

Boehm, B. W., Horowitz, E., Madachy, R., Reifer, D., Clark, B. K., Steece, B., Brown, A. W., Chulani, S., and Abts, C. (2000). *Software Cost Estimation with COCOMO II*. Prentice-Hall.

IEEE (1990). *IEEE Standard Glossary of Software Engineering Terminology*. Std 610.12-1990.

ISO/IEC/IEEE (2010). *Systems and software engineering – Vocabulary*. 24765:2010(E).

Knöll, H.-D. and Busse, J. (1991). *Aufwandsschätzung von Software-Projekten in der Praxis: Methoden, Werkzeugeinsatz, Fallbeispiele*. Number 8 in Reihe Angewandte Informatik. BI Wissenschaftsverlag.

Ludewig, J. and Lichter, H. (2013). *Software Engineering*. dpunkt.verlag, 3. edition.

Noth, T. and Kretzschmar, M. (1984). *Aufwandsschätzung von DV-Projekten, Darstellung und Praxisvergleich der wichtigsten Verfahren*. Springer-Verlag.

Rosove, P. E. (1967). *Developing Computer-based Information Systems*. John Wiley and Sons.

Thayer, R. H. (1997). *Tutorial – Software Engineering Project Management*. IEEE Society Press, revised edition.

Wheeler, D. A. (2006). Linux kernel 2.6: It's worth more!