Softwaretechnik / Software-Engineering

Lecture 1: Introduction

2019-04-25

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

Albert-Ludwigs-Universität Freiburg, Germany

Engineering, Software, Software Engineering

Engineering

Engineering – The **application** of a **systematic**, **disciplined**, **quantifiable** approach to structures, machines, products, systems, or processes. **IEEE 610.12 (1990)**

Engineering – is the **application of** knowledge in the form of **science**, **mathematics**, and **empirical evidence**,

to the innovation, design, construction, operation and maintenance

of structures, machines, materials, software, devices, systems, processes, and organizations. **Wikipedia**

Non-Engineering vs. Engineering

	Non-Engineering (studio / artwork)	Engineering (workshop / techn. product)
Deadlines	**cannot be planned** due to dependency on artist's inspiration	**can usually be planned** with sufficient precision
Price / Cost	determined by **market value**, not by cost	**oriented on cost**, thus calculable
Evaluation and comparison	is only possible **subjectively**, results are disputed	can be conducted using **objective, quantified criteria**
Norms and standards	**are rare** and, if known, not respected	**exist**, are known, and are usually respected
Warranty and liability	are **not defined** and in practice hardly enforceable	are **clearly regulated**, cannot be disclaimed
Mental prerequisite	artist's **inspiration**, among others	the existing and available **technical know-how**
Author	considers the artwork as **part of him/herself**	remains **anonymous**, often lacks emotional ties to the product

(Ludewig and Lichter, 2013)

Content

- **Terminology**
  - **Engineering**, **Software**, **Software Engineering**
- **Motivation: Successful Software Development**
  - Working definition: success
  - Unsuccessful software development exists
  - Common reasons for non-success
- **Course**
  - **Content**
    - Topic areas
    - Structure of topic areas
    - Emphasis: formal methods
    - Relation to other courses
    - Literature
  - **Organisation**
    - Lectures
    - Tutorials
    - Exam

IEEE
Std 610.12-1990
(Revision and redesignation of
IEEE Std 792-1983)

IEEE Standard Glossary of Software Engineering Terminology

Sponsor

Standards Coordinating Committee of the Computer Society of the IEEE

Approved September 28, 1990

IEEE Standards Board

Abstract: IEEE Std 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology*, identifies terms currently in use in the field of Software Engineering. Standard definitions for those terms are established.
Keywords: Software engineering; glossary; terminology; definitions; dictionary

ISBN 1-55937-067-X

INTERNATIONAL STANDARD

ISO/IEC/ IEEE 24765

First edition
2010-12-15

Systems and software engineering — Vocabulary

Ingénierie des systèmes et du logiciel — Vocabulaire

Reference number
ISO/IEC/IEEE 24765:2010(E)

Software – Computer programs, procedures,
and possibly associated documentation and data
pertaining to the operation of a computer system.

See also: **application software**; **support software**; **system software**.

Contrast with: **hardware**.

IEEE 610.12 (1990)

Software –

1. all or part of the programs, procedures, rules, and associated documentation of an information processing system. [...]
2. see 610.12
3. program or set of programs used to run a computer. [...]

cf. **application software**

NOTE: includes firmware, documentation, data, and execution control statements.

IEEE 24765 (2010)

Software Engineering — This Course's Working Definition

Software Engineering –

(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

(2) The study of approaches as in (1).

IEEE 610.12 (1990)

Software Engineering –

1. the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software.
2. see IEEE 610.12 (1)

ISO/IEC/IEEE 24765 (2010)

Software Engineering–
Multi-person development of multi-version programs.

D. L. Parnas (2011)

fiff.informatik.uni-bremen.de/2001/assets/images/

Software Engineering – the establishment and use of sound engineering principles to obtain economically software that is reliable and works efficiently on real machines.

F. L. Bauer (1971)

commons.wikimedia.org (CC-by-sa 3.0)

Content

Successful Software Development

When is Software Development Successful?

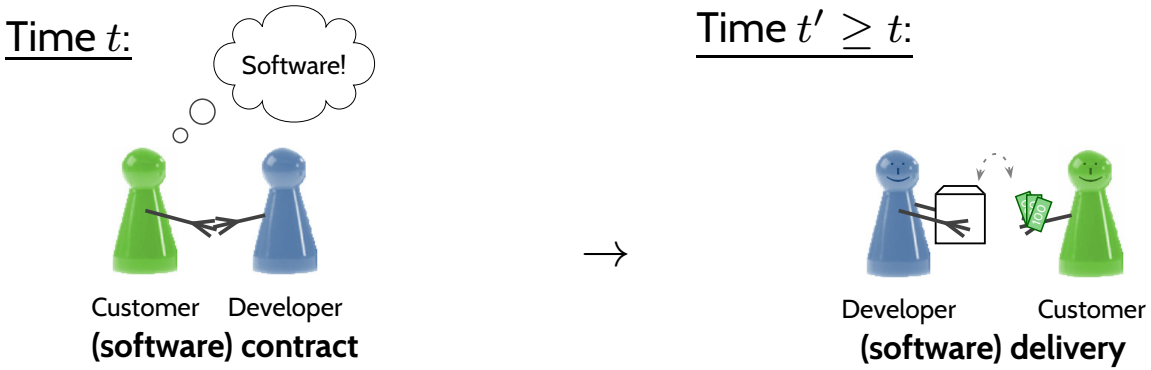
A software development project is **successful**

if and only if

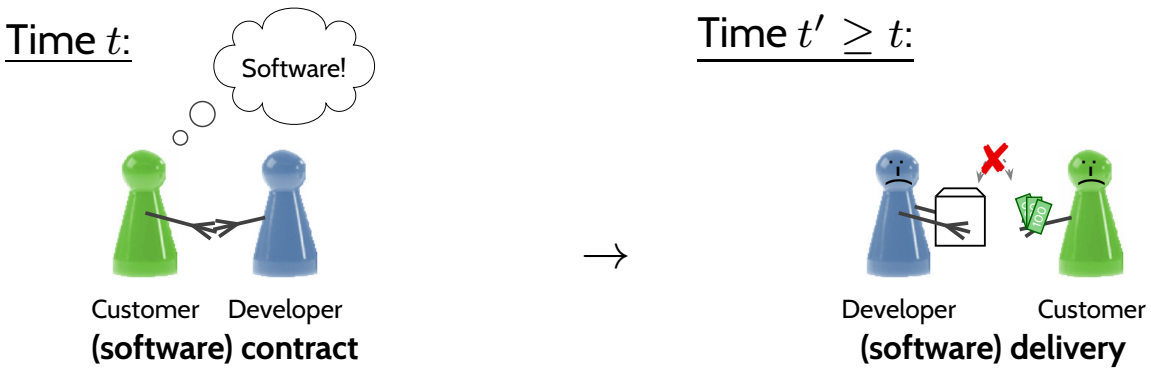
developer, customer, and user are happy with the result at the end of the project.

Which Result? Which Project?

- **Successful:**



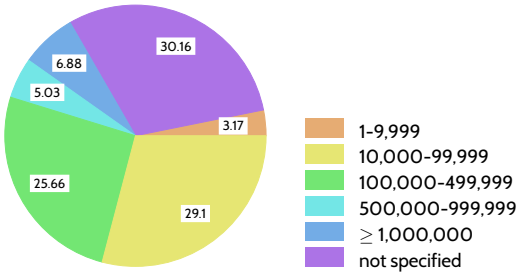
- **Unsuccessful**:



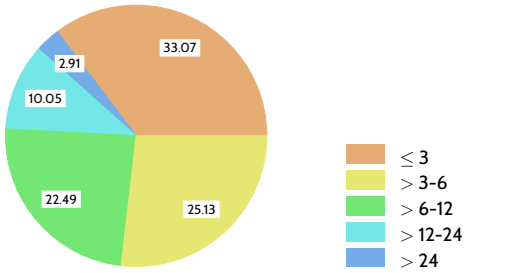
Does 'uncussessful' happen? **If yes: How can we avoid it?**

Is Software Development Always Successful?

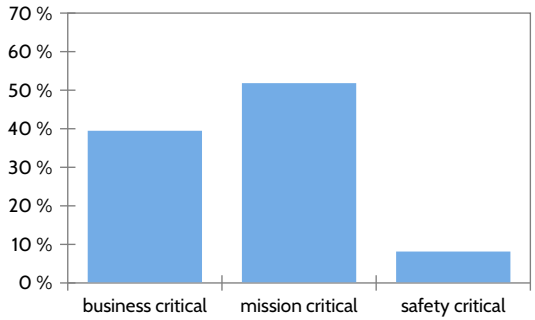
Some Empirical Findings (Buschermöhle et al. (2006))



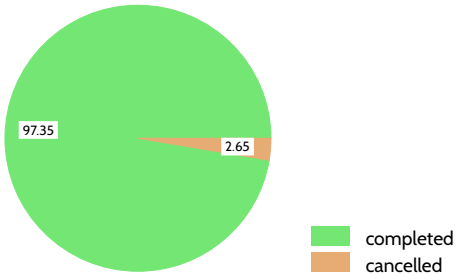
budget in € (378 responses)



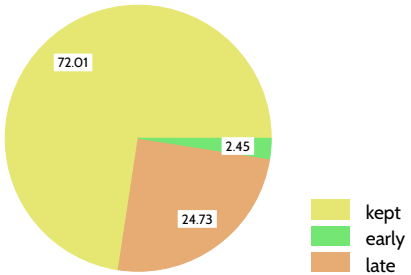
planned duration in months (378 responses)



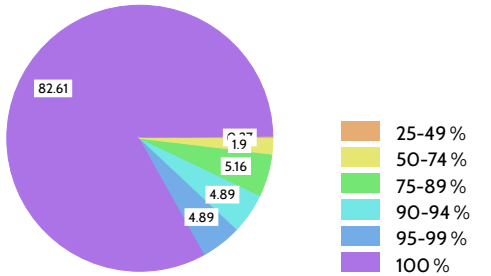
Criticality (378 responses, 30 'not spec.')



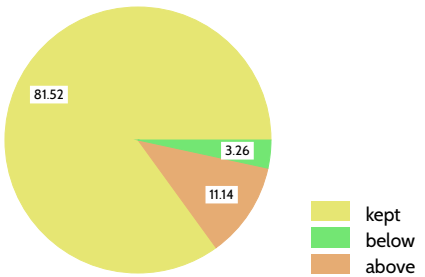
project completion (378 responses)



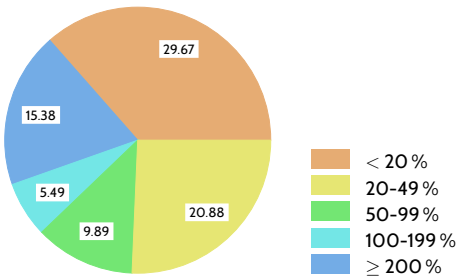
deadline (368 responses)



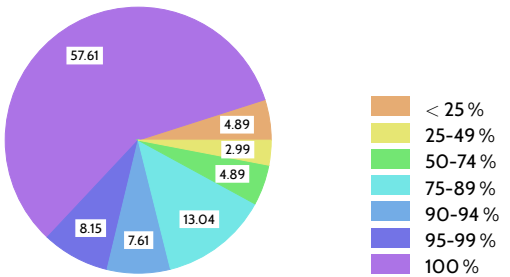
main functionality realised (368 responses)



budget (368 responses)



deadline missed by (91 responses)



secondary functionality realised (368 responses)

Causes for Unsuccessful Projects: First Approximation

Time t:

(software) contract

$\rightarrow$

Time $t' \geq t$:

(software) delivery

. . .

① Capturing Requirements

② Design

③ Implementation

④ (Code) Quality Assurance

. . .

⑤ (Software) Project Management

Possible causes (by phase):

①	②	③	④	⑤
✘	✔	✔	✔	✔
✔	✘	✔	✔	✔
✔	✔	✘	✔	✔
✔	✔	✔	✘	✔
✔	✔	✔	✔	✘

e.g. misunderstanding of requirements; contradicting requirements

e.g. non-scalable design; feature forgotten; designer misunderstood requirement

e.g. programmer misread design specification; simple programming mistake

e.g. wrongly conducted test; tester misunderstood requirement

e.g. wrong cost estimation; bad scheduling; team member was not aware of responsibilities

Causes for Unsuccessful Projects: Once Again

e.g. **misunderstanding** of requirements; **contradicting** requirements

e.g. **non-scalable** design; feature forgotten; designer **misunderstood** requirement

e.g. programmer **misread** design specification; simple programming **mistake**

e.g. **wrongly conducted** test; tester **misunderstood** requirement

e.g. **wrong** cost **estimation**; **bad** scheduling; team member was **not aware of** responsibilities

And that's this course:

- Discuss **typical Software-Engineering problems**,
  - like **communication**, **misunderstandings**, etc.
  - like **technical errors**, **quality issues**, etc.
- and (state-of-the-art) **generic mitigation approaches**
  - like **precise description languages** (e.g. for requirements),
  - like **analysis techniques** (e.g. for program correctness),

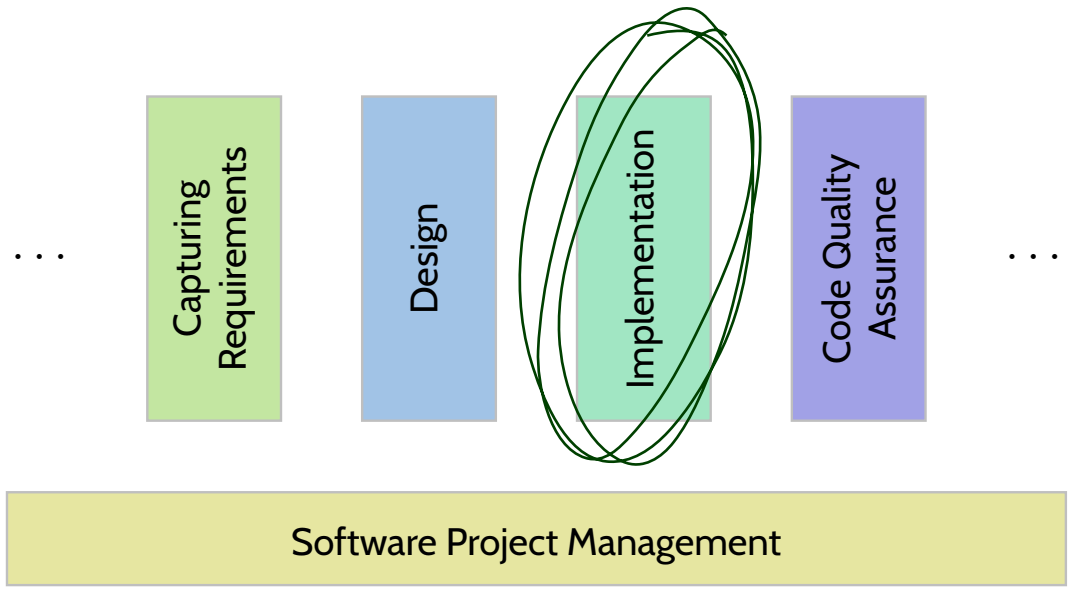
  by development phase (Requirements, Design, etc.).

Content

- **Terminology**
  - **Engineering**, **Software**, **Software Engineering**
- **Motivation: Successful Software Development**
  - Working definition: success
  - Unsuccessful software development exists
  - Common reasons for non-success
- **Course**
  - **Content**
    - Topic areas
    - Structure of topic areas
    - Emphasis: formal methods
    - Relation to other courses
    - Literature
  - **Organisation**
    - Lectures
    - Tutorials
    - Exam

Course: Content

Course Content (Tentative)

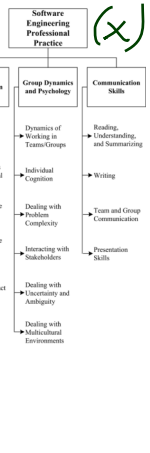


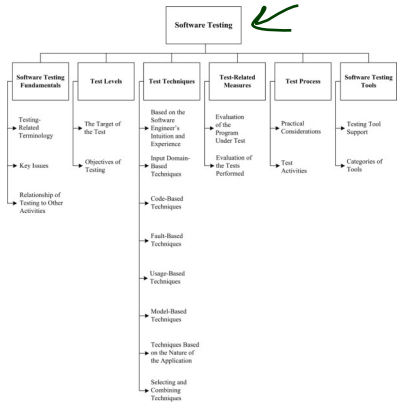
	-	22.4., Mon
Introduction	L 1:	25.4., Thu
Metrics, Costs, Development Process	L 2:	29.4., Mon
	L 3:	2.5., Thu
	L 4:	6.5., Mon
	T 1:	9.5., Thu
Requirements Engineering	L 5:	13.5., Mon
	L 6:	16.5., Thu
	L 7:	20.5., Mon
	T 2:	23.5., Thu
	L 8:	27.5., Mon
	-	30.5., Thu
	L 9:	3.6., Mon
	T 3:	6.6., Thu
	-	10.6., Mon
	-	13.6., Thu
Arch. & Design,	L10:	17.6., Mon
	-	20.6., Thu
Software-	L 11:	24.6., Mon
	T 4:	27.6., Thu
Modelling, Patterns	L 12:	1.7., Mon
	L 13:	4.6., Thu
QA	L 14:	8.7., Mon
	T 5:	11.7., Thu
(Testing, Formal Verification)	L 15:	15.7., Mon
	L 16:	18.7., Thu
Wrap-Up	L 17:	22.7., Mon
	T 6:	25.7., Thu

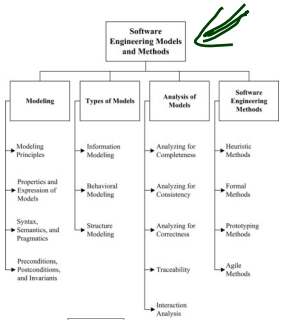
http://www.antarctica.gov.au (Janet Shelley)

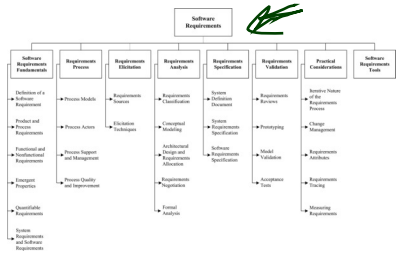
SWEBOK v3.0

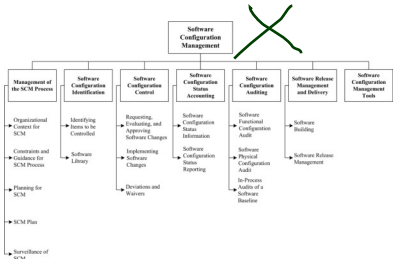
Bourque and Fairley (2014)

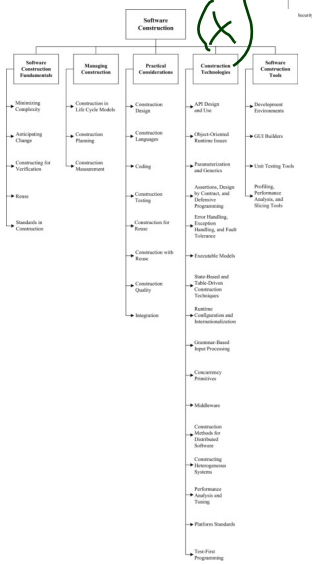


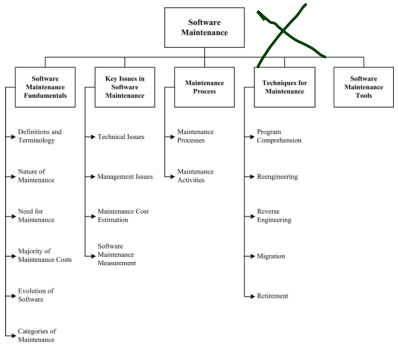


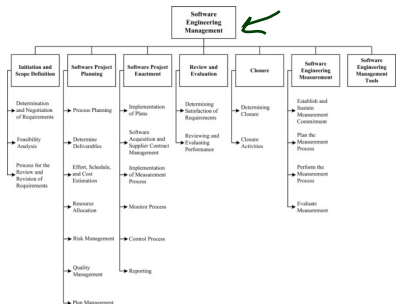


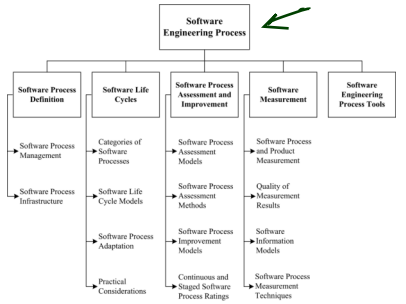


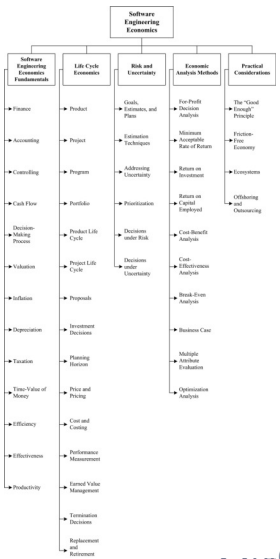


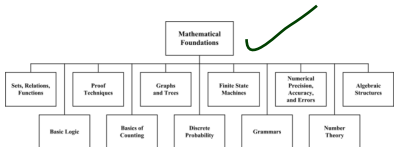


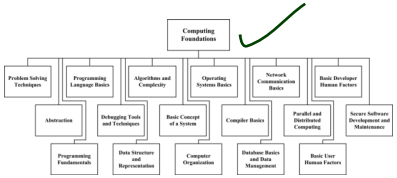


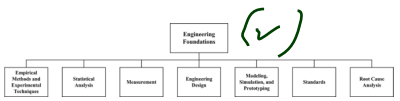




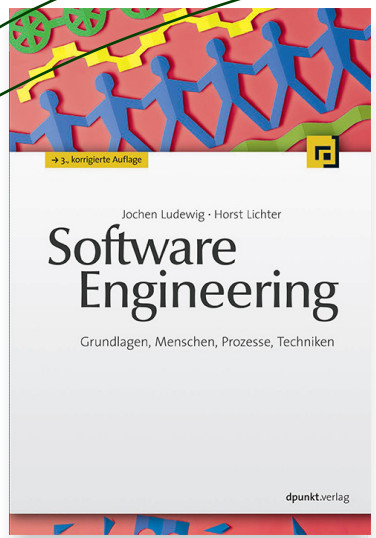


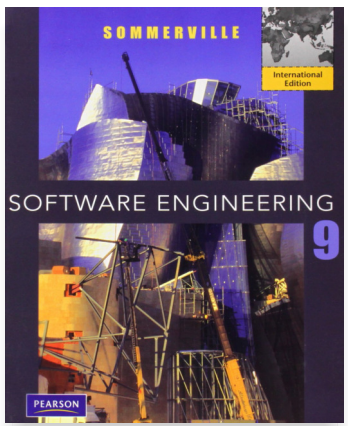






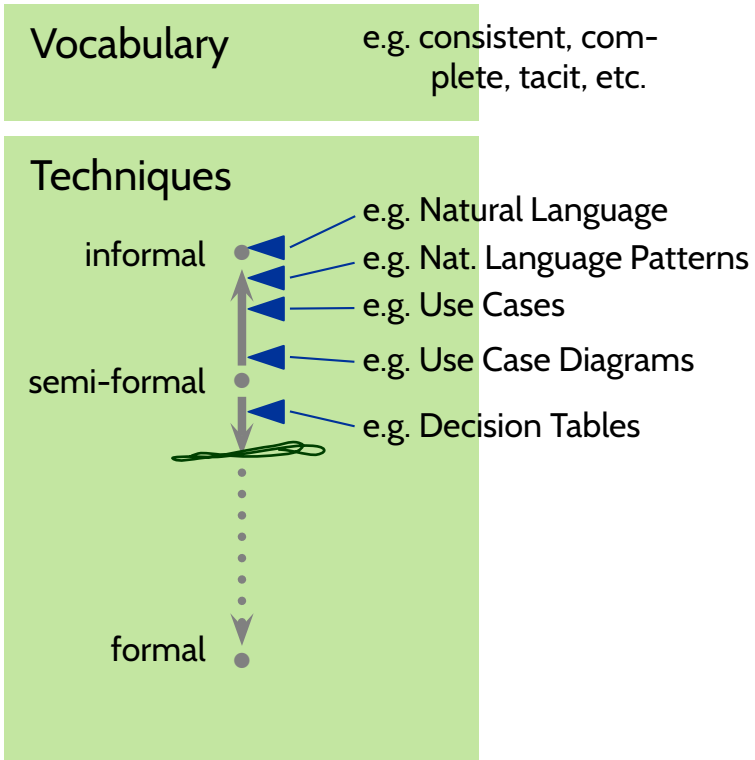
Literature (Preview)



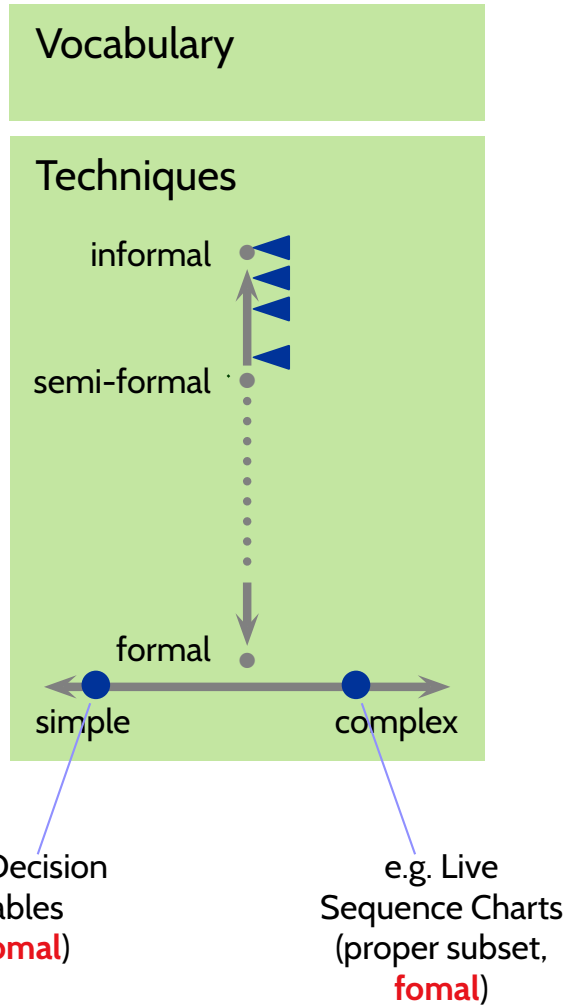


Structure of Topic Areas (Example: Requirements Eng.)

Sommerville (2010),
Balzert (2009),
Ludewig and Lichter (2013),
etc.:



This course:

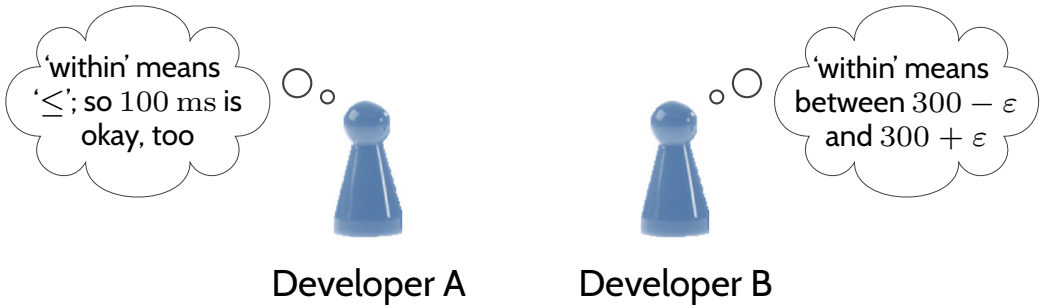


Excursion: Informal vs. Formal Techniques

Example: Requirements Engineering, Airbag Controller

Requirement specification, informal:

Whenever a crash is detected, the airbag has to be fired within $300\,\mathrm{ms}$ $(\pm\varepsilon)$.



Requirement specification, formal:

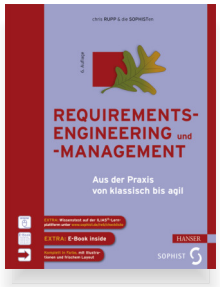
- Fix observables: **crashdetected** : Time $\to \{0,1\}$ and **fireairbag** : Time $\to \{0,1\}$
- Formalise requirement:

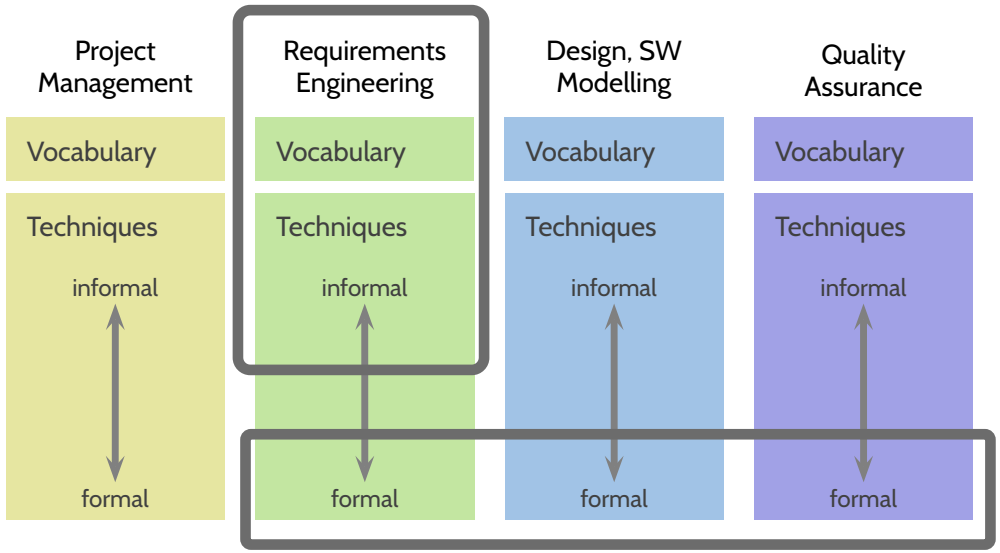
$$\forall\, t, t' \in \text{Time} \bullet \textbf{crashdetected}(t) \land \textbf{airbagfired}(t') \implies t' \in [t + 300 - \varepsilon, t + 300 + \varepsilon]$$

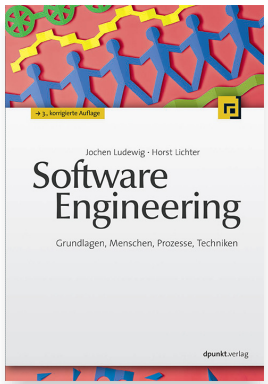
$\to$ no more misunderstandings, sometimes **tools** can **objectively** decide: requirement satisfied yes/no.

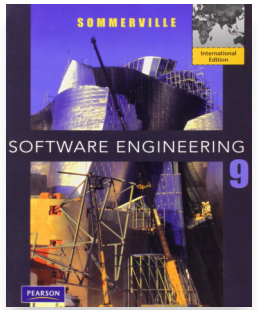

Sign says:
Welcome to formal methods paradise:
No more misunderstandings!
Let tools decide things objectively!

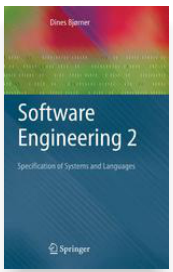
Literature

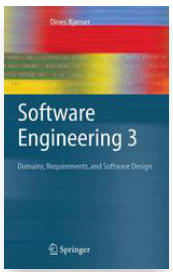












…more on the course homepage.

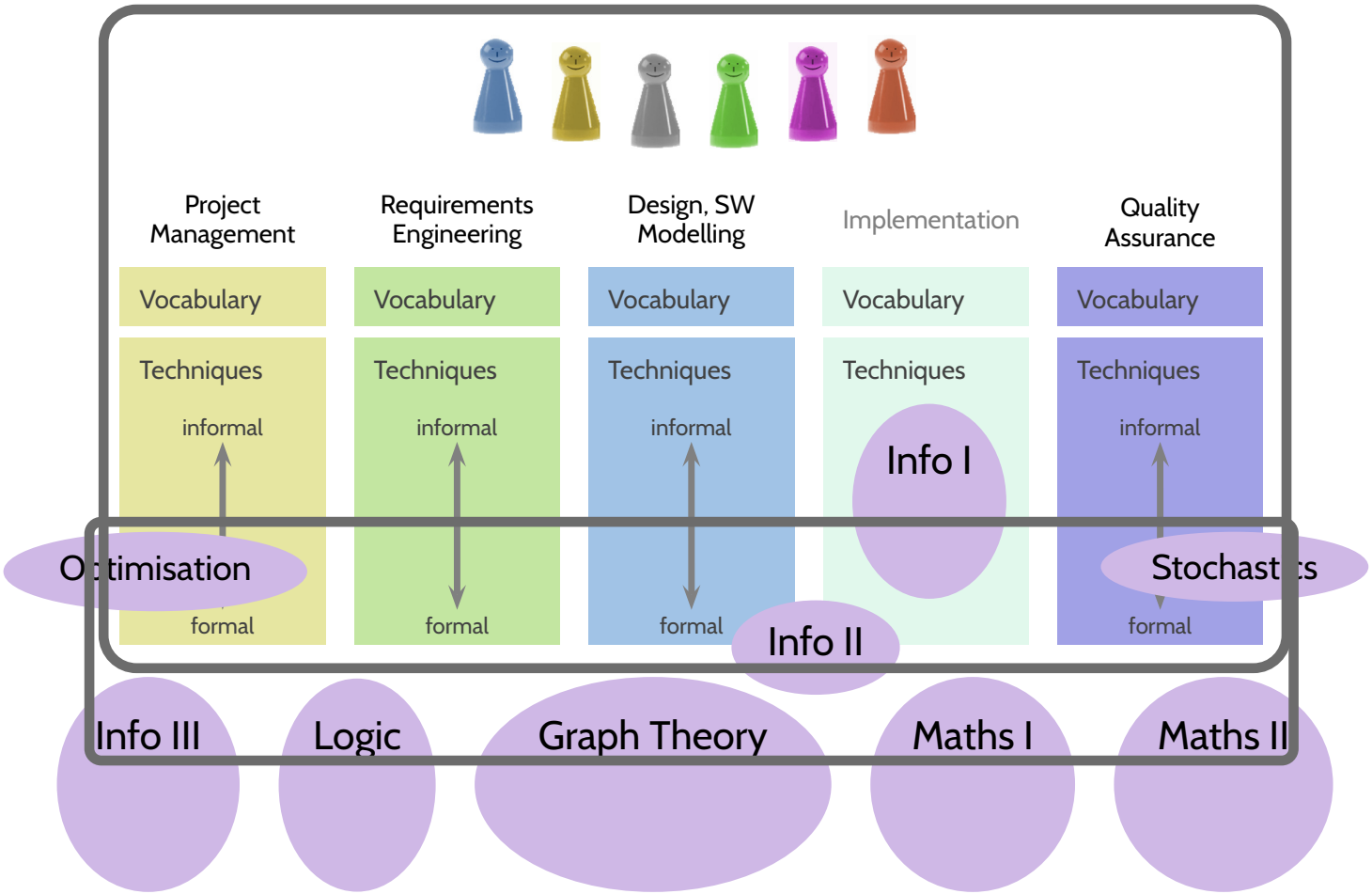
Content

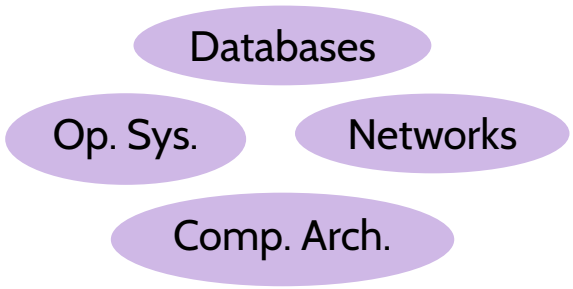
- **Terminology**
  - **Engineering**, **Software**, **Software Engineering**
- **Motivation: Successful Software Development**
  - Working definition: success
  - Unsuccessful software development exists
  - Common reasons for non-success
- **Course**
  - **Content**
    - Topic areas
    - Structure of topic areas
    - Emphasis: formal methods
    - Relation to other courses
    - Literature
  - **Organisation**
    - Lectures
    - Tutorials
    - Exam

Course Software-Engineering vs. Other Courses

The lecturer points out connections to other topic areas (e.g. research, praxis).

totally agree ✔ ○ ○ ○ ○ strongly disagree



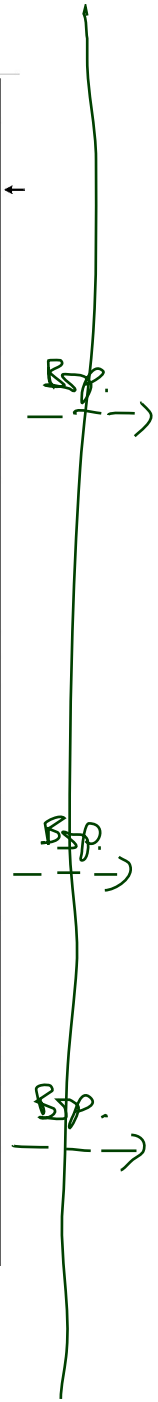


Course Software-Engineering vs. Softwarepraktikum

Agreement between ‘Fachschaft’ and the chair for software engineering: strong(er) **coupling between both courses**.

Zeitplan

Woche	Organisation	Entwurf	MS 01	MS 02	MS 03	MS 04	MS 05	Was?	Wann und Wo?
0	✓	✗	✗	✗	✗	✗	✗	• Vorlesung "Organisation und Prozess" (Einführungsveranstaltung) • Vorlesung "Game Design Document (GDD)" • Gruppeneinteilung abwarten	• Vorlesung: 20.04., 14:00 - max. 18:00, 101-00-006 • Fragebogen: Bis 20.04. 23:59 • Gruppeneinteilung: Am 23.04. online
1	✓	✓	✗	✗	✗	✗	✗	• Vorlesung "Grundlagen Softwarearchitektur" • Abgabe Hausaufgabe	• Vorlesung: 27.04., 14:00 - max. 18:00, 101-00-006 • Abgabe: 30.04. bis 23:59
2	✗	✓	✓	✗	✗	✗	✗	• Vorlesung "Architektur von Videospielen" • Abgabe GDD (beta)	• Vorlesung: 04.05., 14:00 - max. 18:00, 101-00-006 • Abgabe: 07.05. bis 23:59
3	✗	✓	✓	✗	✗	✗	✗		
4	✗	✓	✓	✓	✗	✗	✗	• MS01 erreicht (Spielobjekt in der Welt bewegbar, interaktive Kamera, Karte laden/speichern, Soundausgabe) • Präsentation des aktuellen Stands • Abgabe Architektur (beta)	• Präsentation: 18.05., 14:00 - max. 18:00, 101-00-010/14 • Abgabe: 21.05. bis 23:59
5	✗	✓	✗	✓	✗	✗	✗		
6	✗	✓	✗	✓	✓	✗	✗		
7	✗	✓	✗	✓	✓	✗	✗	• MS02 erreicht (Mehrere Spielobjekte bewegen, Interaktionen zwischen Spielobjekten, Pathfinding, Screen-Management, Menü, HUD, Musik)	
8	✗	✓	✗	✗	✓	✗	✗	• Präsentation Programm (beta) • Abgabe Programm (beta)	• Präsentation: 15.06., 14:00 - max. 18:00, 101-00-006 • Abgabe: 18.06. bis 23:59
9	✗	✓	✗	✗	✓	✓	✗	• Abgabe GDD (final) • MS03 erreicht (KI, primäre Interaktionen vorhanden, Sieg-/Niederlagebedingungen, vollständiges Pathfinding, Inhalte, Grafik/Soundeffekte)	• Abgabe: 25.06. bis 23:59
10	✗	✗	✗	✗	✗	✓	✓		
11	✗	✗	✗	✗	✗	✓	✓		
12	✗	✗	✗	✗	✗	✓	✓	• MS04 erreicht (finale Version vorhanden) • Abgabe Architektur (final)	• Abgabe: 16.07. bis 23:59
13	✗	✗	✗	✗	✗	✗	✓	• MS05 erreicht (Fehlerbehebung & Balancing) • Präsentation Programm (final) • Abgabe Programm (final)	• Präsentation: 20.07., 14:00 - max. 18:00, 101-00-006 • Abgabe: 23.07. bis 23:59



	-	22.4., Mon
Introduction	L 1:	25.4., Thu
Metrics, Costs, Development Process	L 2:	29.4., Mon
	L 3:	2.5., Thu
	L 4:	6.5., Mon
	T 1:	9.5., Thu
Requirements Engineering	L 5:	13.5., Mon
	L 6:	16.5., Thu
	L 7:	20.5., Mon
	T 2:	23.5., Thu
	L 8:	27.5., Mon
	-	30.5., Thu
	L 9:	3.6., Mon
	T 3:	6.6., Thu
	-	10.6., Mon
	-	13.6., Thu
Arch. & Design,	L10:	17.6., Mon
	-	20.6., Thu
Software-	L 11:	24.6., Mon
	T 4:	27.6., Thu
Modelling, Patterns	L 12:	1.7., Mon
	L 13:	4.6., Thu
QA	L 14:	8.7., Mon
	T 5:	11.7., Thu
(Testing, Formal Verification)	L 15:	15.7., Mon
	L 16:	18.7., Thu
Wrap-Up	L 17:	22.7., Mon
	T 6:	25.7., Thu

Any Questions So Far?

Content

- **Terminology**
  - **Engineering**, **Software**, **Software Engineering**
- **Motivation: Successful Software Development**
  - Working definition: success
  - Unsuccessful software development exists
  - Common reasons for non-success
- **Course**
  - **Content**
    - Topic areas
    - Structure of topic areas
    - Emphasis: formal methods
    - Relation to other courses
    - Literature
  - **Organisation**
    - Lectures
    - Tutorials
    - Exam

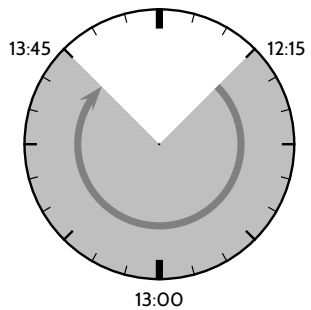
Course: Organisation

Organisation: Lectures

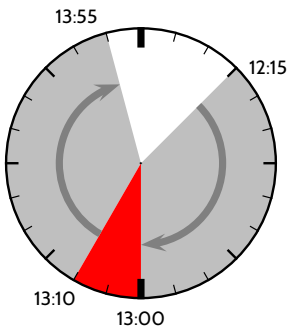
- **Homepage:** `http://swt.informatik.uni-freiburg.de/teaching/SS2019/swtv1`
- **Course language:** German (since we are in an odd year)
- **Script/Media:**
  - slides without annotations on homepage with beginning of lecture the latest
  - slides with annotations on homepage typically soon after the lecture
  - recording on ILIAS (stream and download) with max. 2 days delay (cf. link on homepage)
- **Schedule:** topic areas à three 90 min. lectures, one 90 min. tutorial (with exceptions)

- **Interaction:** absence often moaned; but it takes two, so please ask/comment immediately.
- **Questions/comments:**
  - "online": ask immediately or in the break
  - "offline": (i) try to solve yourself
    (ii) discuss with colleagues
    (iii) a) Exercises: ILIAS (group) forum, contact tutor
    b) Everything else: contact lecturer (cf. homepage)
    or just drop by: Building 52, Room 00-020

- **Break:** we'll have a 5-10 min. break
  in the middle of each lecture (from now on),
  unless a majority objects now.

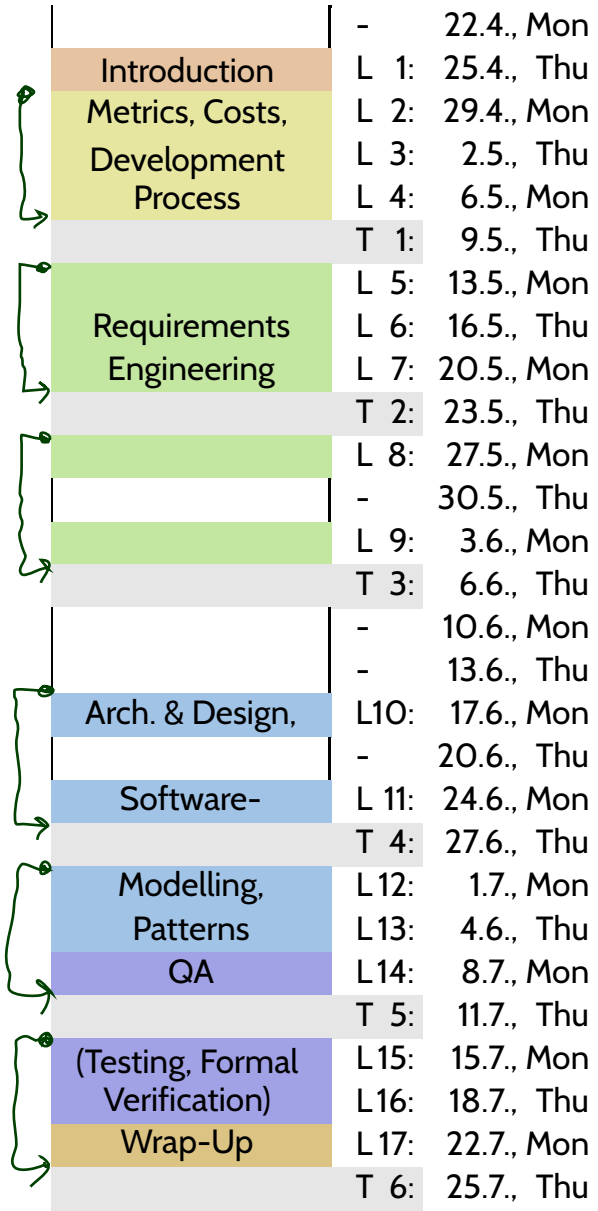


vs.



Organisation: Exercises & Tutorials

- **Schedule/Submission:**
  - exercises **online** (**homepage** and **ILIAS**) with first lecture of a block,
  - **early submission** 24h before tutorial
    (usually Wednesday, 12:00, local time),
  - **regular submission** right before tutorial
    (usually Thursday, 12:00, local time).
  - please submit **electronically** via **ILIAS**
  - should work in teams of **2–3 people**, clearly give **names** on submission



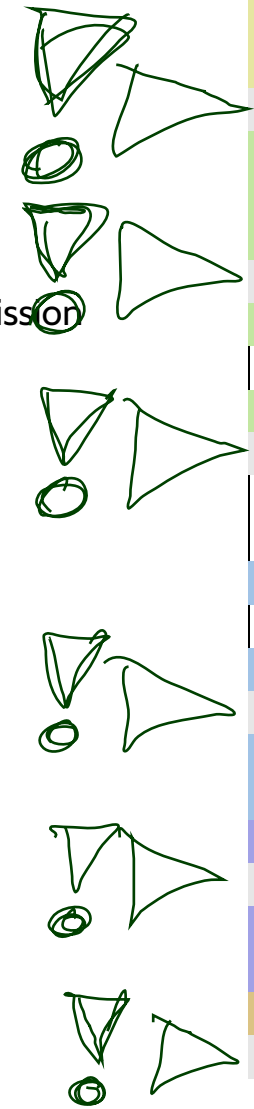
Organisation: Exercises & Tutorials

- **Schedule/Submission:**
  - exercises **online** (**homepage** and **ILIAS**) with first lecture of a block,
  - **early submission** 24h before tutorial (usually Wednesday, 12:00, local time),
  - **regular submission** right before tutorial (usually Thursday, 12:00, local time).
  - please submit **electronically** via **ILIAS**
  - should work in teams of **2–3 people**, clearly give **names** on submission

- **Grading system:** "**most complicated grading system ever**"
  - **Admission points** (good-will rating, upper bound)
    ("reasonable grading given student's knowledge **before** tutorial")
  - **Exam-like points** (evil rating, lower bound)
    ("reasonable grading given student's knowledge **after** tutorial")

  10% **bonus** for **early** submission.

- **Tutorial:** **Four groups** (central assignment), hosted by tutor.
  - Starting from discussion of the early submissions (anonymous), develop **one** good proposal together,
  - tutorial notes provided via **ILIAS**.



	-	22.4., Mon
Introduction	L 1:	25.4., Thu
Metrics, Costs, Development Process	L 2:	29.4., Mon
	L 3:	2.5., Thu
	L 4:	6.5., Mon
	T 1:	9.5., Thu
Requirements Engineering	L 5:	13.5., Mon
	L 6:	16.5., Thu
	L 7:	20.5., Mon
	T 2:	23.5., Thu
	L 8:	27.5., Mon
	-	30.5., Thu
	L 9:	3.6., Mon
	T 3:	6.6., Thu
	-	10.6., Mon
	-	13.6., Thu
Arch. & Design,	L10:	17.6., Mon
	-	20.6., Thu
Software-	L 11:	24.6., Mon
	T 4:	27.6., Thu
Modelling, Patterns	L 12:	1.7., Mon
	L 13:	4.6., Thu
QA	L 14:	8.7., Mon
	T 5:	11.7., Thu
(Testing, Formal Verification)	L 15:	15.7., Mon
	L 16:	18.7., Thu
Wrap-Up	L 17:	22.7., Mon
	T 6:	25.7., Thu

Organisation: Exam

- **Exam Admission:**

  Achieving **50%** of the **regular admission points** of Exercise Sheets 0–3
  and **50%** of the **regular admission points** of Exercise Sheets 4–6
  is sufficient for admission to exam.

  5 + 15 regular admission points on sheets 0 and 1, and
  20 regular admission points on exercise sheets 2–6

  $\rightarrow$ 120 **regular** admission points for 100%.

  (plus plenty of **admission bonus points** in both blocks, 0–3 and 4–6)

- **Exam Form:**
  - **written** exam
  - date, time, place: tba
  - permitted exam aids: one A4 paper (max. 21 x 29.7 x 1 mm) of notes, max. two sides inscribed
  - scores from the exercises **do not** contribute to the final grade.
  - example exam available on **ILIAS**

One Last Word on The Exercises...

- Every exercise task is **a tiny little scientific work**!
- Basic rule for high quality submissions:
  - **rephrase** the task in your own words,
  - **state** your solution,
  - **convince** yourself and your tutor of the correctness of your solution (at best: prove it).

Example:

Task: What is the length of the longest line inside the square with side length $a = 19.1$?

Submission A:

27

Submission B:

The length of the longest straight line fully inside the square with side length $a = 19.1$ is 27.01 (rounded).

The longest straight line inside the square is the diagonal. By Pythagoras, its length is $\sqrt{a^2 + a^2}$. Inserting $a = 19.1$ yields 27.01 (rounded).

One Last Word on The Exercises...

- Every exercise task is **a tiny little scientific work**!
- Basic rule for high quality submissions:
  - **rephrase** the task in your own words,
  - **state** your solution,
  - **convince** yourself and your tutor of the correctness of your solution (at best: prove it).

Example:

Task: What is the length of the longest line inside the square with side length $a = 19.1$?

Submission A:

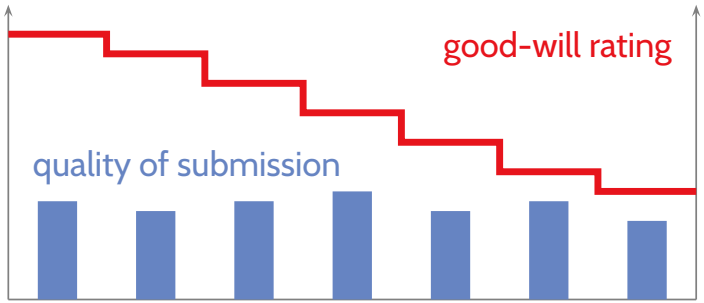
Submission B:

The length of the longest straight line fully inside the square with side length $a = 19.1$ is 27.01 (rounded).

The longest straight line inside the square is the diagonal. By Pythagoras, its length is $\sqrt{a^2 + a^2}$. Inserting $a = 19.1$ yields 27.01 (rounded).

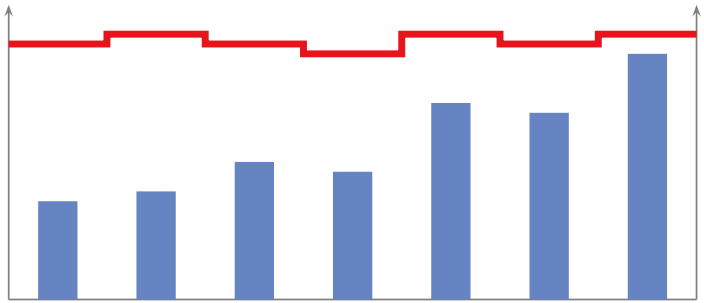
One Last Word on The Exercises. . .

- Every exercise task is **a tiny little scientific work**!
- Basic rule for high quality submissions:
  - **rephrase** the task in your own words,
  - **state** your solution,
  - **convince** yourself and your tutor of the correctness of your solution (at best: prove it).



I have improved my skills in scientific problem solving.

totally agree ○ ○ ○ ○ ✗ strongly disagree



I have improved my skills in scientific problem solving.

totally agree ✓ ○ ○ ○ ○ strongly disagree

Tell Them What You've Told Them...

- **Basic vocabulary**:
  - software, engineering, software engineering,
  - customer, developer, user,
  - successful software development

  → **note:** some definitions are neither formal nor universally agreed
- **(Fun) fact**: software development is not always successful
- **Basic activities of (software) engineering**:
  - gather requirements,
  - design,
  - implementation,
  - quality assurance,
  - project management

  → motivates content of the course – for the case of software
- **Formal (vs. informal) methods**
  - avoid misunderstandings,
  - enable objective, tool-based assessment

→ **note:** still, humans are at the heart of software engineering.

- **Course content and organisation**

Any (More) Questions?

References

References

Balzert, H. (2009). *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. Spektrum, 3rd edition.

Bauer, F. L. (1971). Software engineering. In *IFIP Congress (1)*, pages 530–538.

Bourque, P. and Fairley, R. E. (2014). *Guide to the Software Engineering Body of Knowledge, Version 3.0*. IEEE Computer Society. `www.swebok.org`.

Buschermöhle, R., Eekhoff, H., and Josko, B. (2006). success – Erfolgs- und Misserfolgsfaktoren bei der Durchführung von Hard- und Softwareentwicklungsprojekten in Deutschland. Technical Report VSEK/55/D, OFFIS.

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).

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

Parnas, D. L. (2011). Software engineering: Multi-person development of multi-version programs. In Jones, C. B. et al., editors, *Dependable and Historic Computing*, volume 6875 of *LNCS*, pages 413–427. Springer.

Sommerville, I. (2010). *Software Engineering*. Pearson, 9th edition.