Part A: Development of Correct Software

1. **Introduction**
2. Implementation
3. Testing
4. Software Changes
5. Version Control

Part B: Development in Schedule & Budget

6. Project Management
7. Development Process

Part C: Development of Needed Software

8. Requirements
9. System Modeling
10. Software Architecture
11. Software Design
12. Software Reuse
13. Summary

1a. The Impact of Software

Software (Products)
Characteristics of Software
Application and System Software
Where Does Software Run?
Downloads of Android Apps
Properties of Software
Summary

1b. Software and Its Engineering

The Project Cartoon
Software Engineering
Software and System Engineering
(Software) Engineering
Relevance of Software for This Course
Summary

1c. Course Overview

Structure of This Course
About This Course
The Lectures
The Exercises
Literature for This Course
Summary

1. Introduction

1. Introduction

Technische Universität Braunschweig

ISF

Software Engineering 1

Part A: Development of Correct Software

1. **Introduction**
2. Implementation
3. Testing
4. Software Changes
5. Version Control

Part B: Development in Schedule & Budget

6. Project Management
7. Development Process

Part C: Development of Needed Software

8. Requirements
9. System Modeling
10. Software Architecture
11. Software Design
12. Software Reuse
13. Summary

1. Introduction

1a. The Impact of Software

1b. Software and Its Engineering

1c. Course Overview

About The Lecturers

Thomas (Thüm)

- (avoid Prof./Mr. Thüm please)
- Started programming in 1998
- Contributing to FeatureIDE since 2007 (Eclipse project with 200,000 LOC)

About The Lecturers

Thomas (Thüm)

- (avoid Prof./Mr. Thüm please)
- Started programming in 1998
- Contributing to FeatureIDE since 2007 (Eclipse project with 200,000 LOC)
- 2004–2010: studied Computer Science (minor subject Mathematics)
- 2010–2015: PhD student in Magdeburg
- 2015–2019: PostDoc at ISF

About The Lecturers

Thomas (Thüm)

- (avoid Prof./Mr. Thüm please)
- Started programming in 1998
- Contributing to FeatureIDE since 2007 (Eclipse project with 200,000 LOC)
- 2004–2010: studied Computer Science (minor subject Mathematics)
- 2010–2015: PhD student in Magdeburg
- 2015–2019: PostDoc at ISF
- 2020–2024: Professor in Ulm
- 2024–2024: Professor in Paderborn
- since October 2024: Professor at ISF

About The Lecturers

Sebastian (Krieter)

- Started programming in 2006
- Contributing to FeatureIDE since 2012 (Eclipse project with 200,000 LOC)

About The Lecturers

Sebastian (Krieter)

- Started programming in 2006
- Contributing to FeatureIDE since 2012 (Eclipse project with 200,000 LOC)
- 2010–2015: studied Computer Science
- 2015–2022: PhD student in Magdeburg
- 2022–2024: PostDoc in Ulm
- 2024–2024: PostDoc in Paderborn
- since October 2024: PostDoc at ISF

About The Lecturers

Student Staff

- B.Sc. Linek Phil Höhn
- B.Sc. Niclas Kleinert
- B.Sc. Jassin Mohammedi
- B.Sc. Lennart Pape
- Malcom Schulz
- Felix Schumacher
- Joschka Weikum

About The Lecturers

Student Staff

- B.Sc. Linek Phil Höhn
- B.Sc. Niclas Kleinert
- B.Sc. Jassin Mohammedi
- B.Sc. Lennart Pape
- Malcom Schulz
- Felix Schumacher
- Joschka Weikum

Roles in This Course

- Lectures: Thomas, Sebastian
- Interactive Tasks in Lectures: Joschka
- Exercises: student staff

1. Introduction

1a. The Impact of Software

1b. Software and Its Engineering

1c. Course Overview

Software (Products)

Software [adapted from Sommerville]

Software stands for one or several computer programs and all associated documentation, libraries, support websites, and configuration data that are needed to make these programs useful.

Explanation

The term program is used in a broader sense here. Software may also include source code, software models, or binaries.

Software (Products)

Software [adapted from Sommerville]

Software stands for one or several computer programs and all associated documentation, libraries, support websites, and configuration data that are needed to make these programs useful.

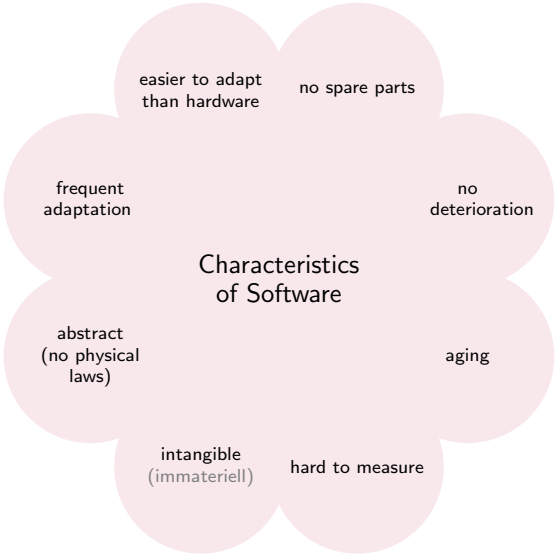
Explanation

The term program is used in a broader sense here. Software may also include source code, software models, or binaries.

Software Product and Professional Software

A **software product** is a software that can be sold to a customer. **Professional software** is software intended for use by someone apart from its developer and it is usually developed by teams rather than individuals. [adapted from Sommerville]

Characteristics of Software



Application and System Software

Application Software or Application

Software that is designed for end users and applied for certain purposes. (Anwendungssoftware oder Anwendung)

Examples

web browsers, media players, email or chat clients, text or photo editors, games

Application and System Software

Application Software or Application

Software that is designed for end users and applied for certain purposes. (Anwendungssoftware oder Anwendung)

Examples

web browsers, media players, email or chat clients, text or photo editors, games

System Software

Software that is not application software and typically being designed to provide a platform for other software.

Examples

operating systems, firmware, basic input/output system (BIOS), device drivers, game engines, GUI frameworks

Application and System Software

Application Software or Application

Software that is designed for end users and applied for certain purposes. (Anwendungssoftware oder Anwendung)

Examples

web browsers, media players, email or chat clients, text or photo editors, games

System Software

Software that is not application software and typically being designed to provide a platform for other software.

Examples

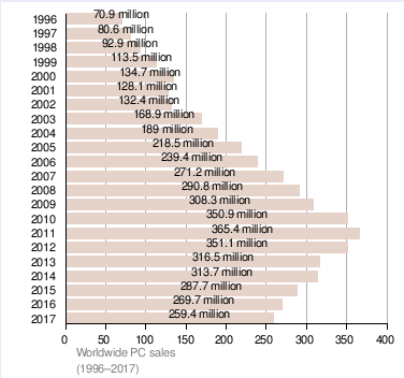
operating systems, firmware, basic input/output system (BIOS), device drivers, game engines, GUI frameworks

Classification Not Always Unique

e.g., web browsers and chat clients take over more and more features of operating systems

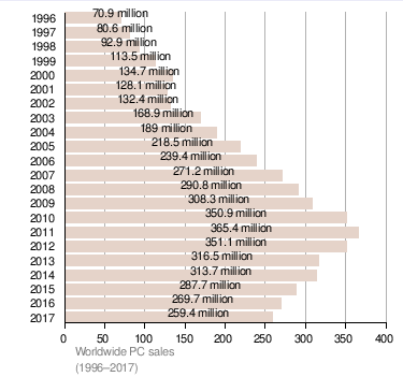
Where Does Software Run?

World-Wide PC Sales

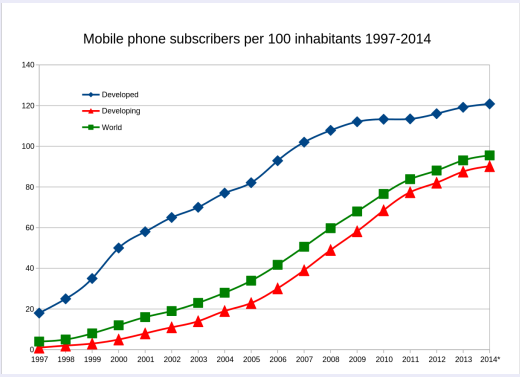


Where Does Software Run?

World-Wide PC Sales



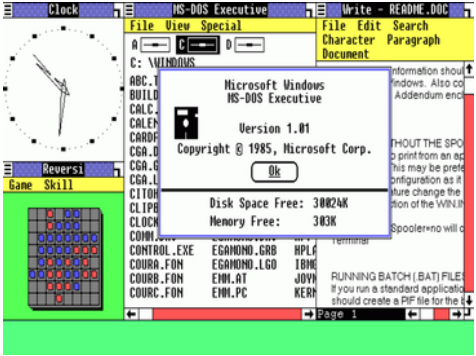
World-Wide Mobile Phone Subscriptions



Application Software

Desktop Application or Desktop App

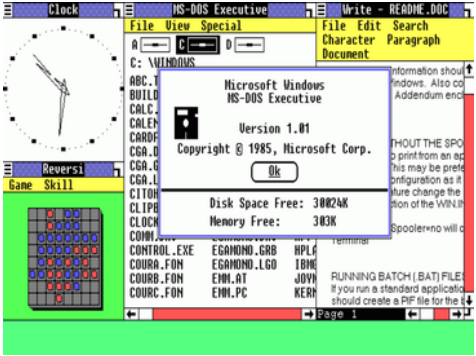
Windows 1.0 released in 1985



Application Software

Desktop Application or Desktop App

Windows 1.0 released in 1985

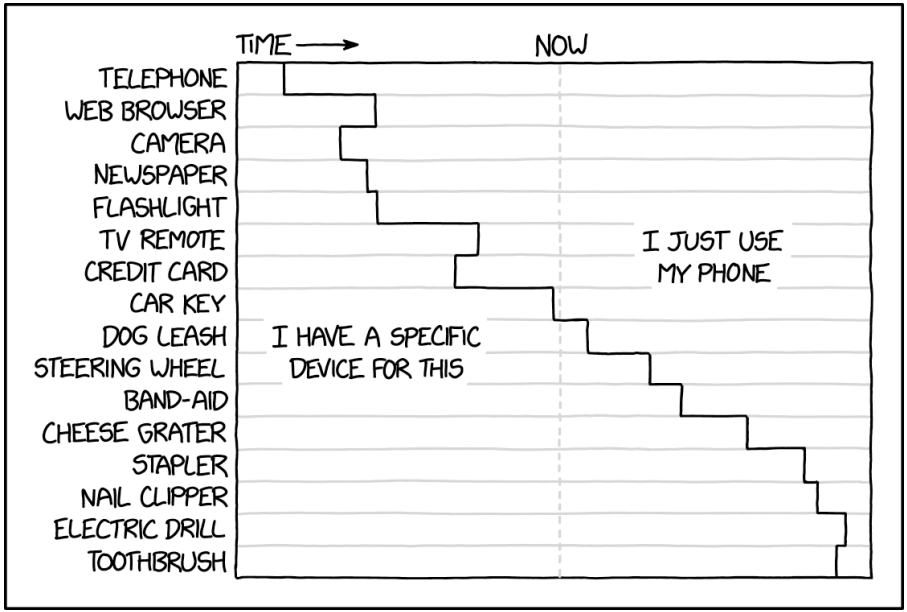


Web Application or Web App

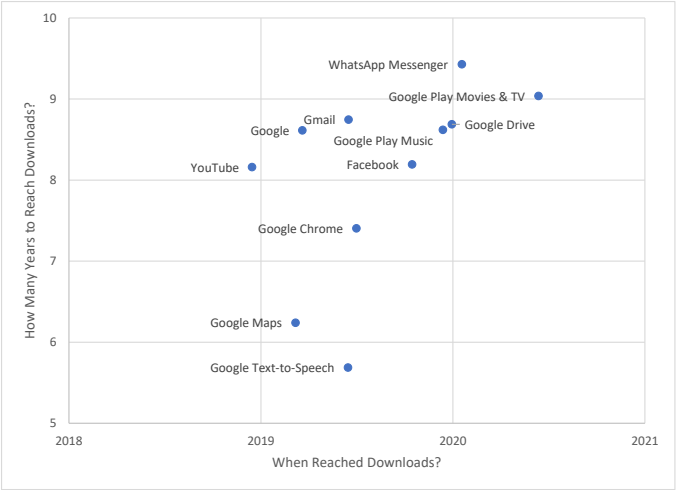
Ebay was born in 1995

Mobile Application or Mobile App or App

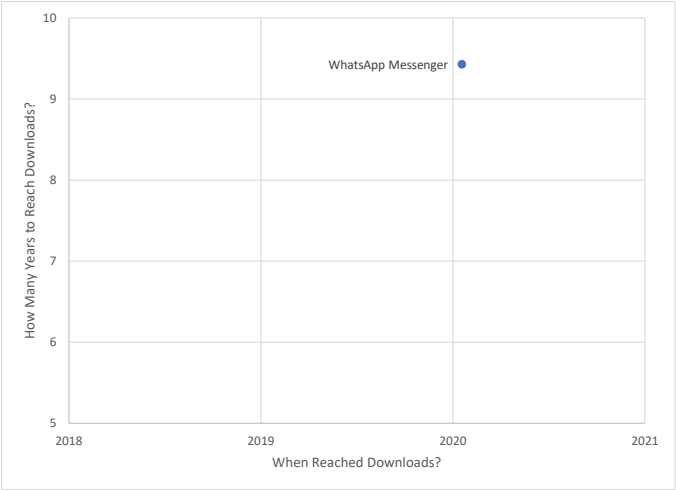
First iPhone released in 2007


TIME
NOW
TELEPHONE
WEB BROWSER
CAMERA
NEWSPAPER
FLASHLIGHT
TV REMOTE
CREDIT CARD
CAR KEY
DOG LEASH
STEERING WHEEL
BAND-AID
CHEESE GRATER
STAPLER
NAIL CLIPPER
ELECTRIC DRILL
TOOTHBRUSH
I HAVE A SPECIFIC DEVICE FOR THIS
I JUST USE MY PHONE

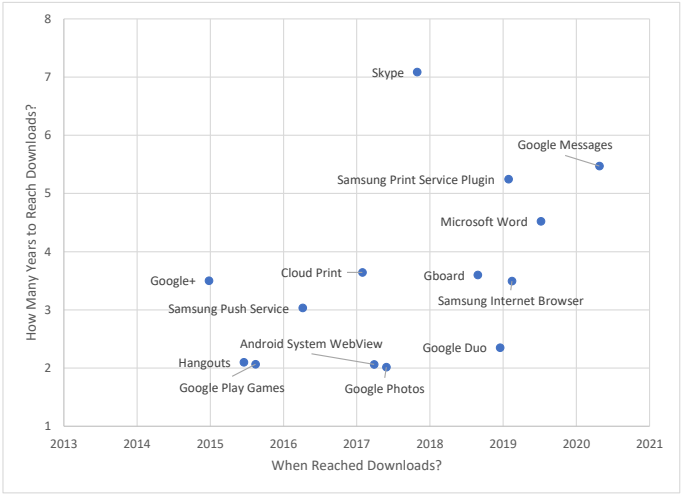
Android Apps with 5 Billion Downloads (5 Milliarden)



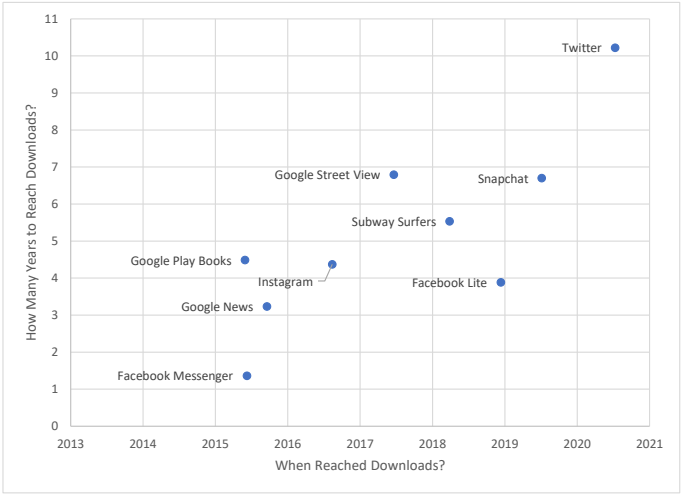
Android Apps with 5 Billion Downloads (5 Milliarden)



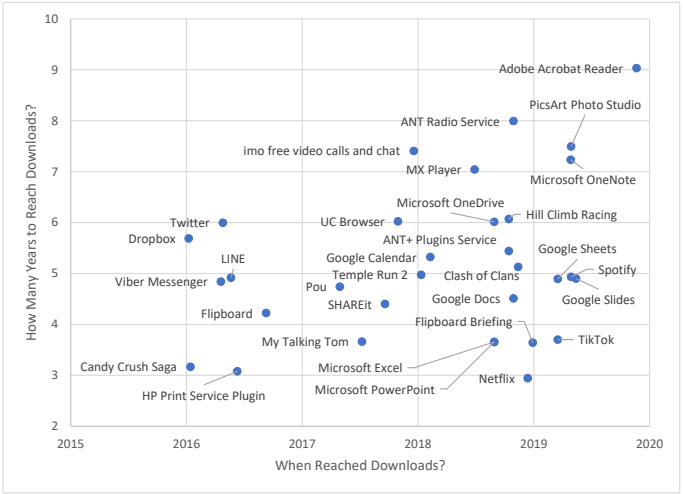
Android Apps with 1 Billion Downloads (1 Milliarde)



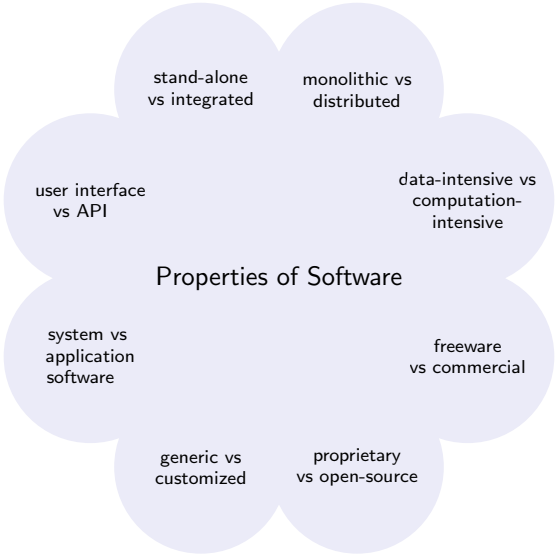
Android Apps with 1 Billion Downloads (1 Milliarde)



Android Apps with 500 Million Downloads



Properties of Software



The Impact of Software – Summary

Lessons Learned

- What is software?
- What is the difference between program, software product, professional software, desktop/web/mobile app?
- What are characteristics and properties of software?
- What is the impact of software?
- Next: What can go possibly wrong with software?

The Impact of Software – Summary

Lessons Learned

- What is software?
- What is the difference between program, software product, professional software, desktop/web/mobile app?
- What are characteristics and properties of software?
- What is the impact of software?
- Next: What can go possibly wrong with software?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

The Impact of Software – Summary

Lessons Learned

- What is software?
- What is the difference between program, software product, professional software, desktop/web/mobile app?
- What are characteristics and properties of software?
- What is the impact of software?
- Next: What can go possibly wrong with software?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Introduce yourselves to each other (5 min)
3. Share your (recent) experiences with software (5 min)

The Impact of Software – Summary

Lessons Learned

- What is software?
- What is the difference between program, software product, professional software, desktop/web/mobile app?
- What are characteristics and properties of software?
- What is the impact of software?
- Next: What can go possibly wrong with software?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Introduce yourselves to each other (5 min)
3. Share your (recent) experiences with software (5 min)
4. Survey: What is your experience with software?

The Impact of Software – Summary

Lessons Learned

- What is software?
- What is the difference between program, software product, professional software, desktop/web/mobile app?
- What are characteristics and properties of software?
- What is the impact of software?
- Next: What can go possibly wrong with software?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Introduce yourselves to each other (5 min)
3. Share your (recent) experiences with software (5 min)
4. Survey: What is your experience with software?

A Warst du jemals von einer Softwarepanne betroffen?

The Impact of Software – Summary

Lessons Learned

- What is software?
- What is the difference between program, software product, professional software, desktop/web/mobile app?
- What are characteristics and properties of software?
- What is the impact of software?
- Next: What can go possibly wrong with software?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Introduce yourselves to each other (5 min)
3. Share your (recent) experiences with software (5 min)
4. Survey: What is your experience with software?

A Warst du jemals von einer Softwarepanne betroffen?

B Hast du dich im letzten Semester über Softwarefehler geärgert?

The Impact of Software – Summary

Lessons Learned

- What is software?
- What is the difference between program, software product, professional software, desktop/web/mobile app?
- What are characteristics and properties of software?
- What is the impact of software?
- Next: What can go possibly wrong with software?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Introduce yourselves to each other (5 min)
3. Share your (recent) experiences with software (5 min)
4. Survey: What is your experience with software?

A Warst du jemals von einer Softwarepanne betroffen?

B Hast du dich im letzten Semester über Softwarefehler geärgert?

C Hast du selbst Programmiererfahrung?

The Impact of Software – Summary

Lessons Learned

- What is software?
- What is the difference between program, software product, professional software, desktop/web/mobile app?
- What are characteristics and properties of software?
- What is the impact of software?
- Next: What can go possibly wrong with software?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Introduce yourselves to each other (5 min)
3. Share your (recent) experiences with software (5 min)
4. Survey: What is your experience with software?

A Warst du jemals von einer Softwarepanne betroffen?

B Hast du dich im letzten Semester über Softwarefehler geärgert?

C Hast du selbst Programmiererfahrung?

D Hast du bereits Software getestet?

The Impact of Software – Summary

Lessons Learned

- What is software?
- What is the difference between program, software product, professional software, desktop/web/mobile app?
- What are characteristics and properties of software?
- What is the impact of software?
- Next: What can go possibly wrong with software?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Introduce yourselves to each other (5 min)
3. Share your (recent) experiences with software (5 min)
4. Survey: What is your experience with software?

A Warst du jemals von einer Softwarepanne betroffen?

B Hast du dich im letzten Semester über Softwarefehler geärgert?

C Hast du selbst Programmiererfahrung?

D Hast du bereits Software getestet?

E Hast du schon mal Programmierfehler verursacht?

1. Introduction

1a. The Impact of Software

1b. Software and Its Engineering

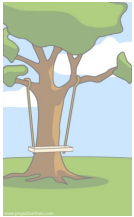
1c. Course Overview

The Project Cartoon

how the customer explained it

The Project Cartoon

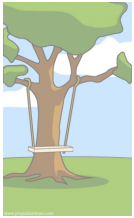
how the customer explained it



how the project leader understood it

The Project Cartoon

how the customer explained it



how the project leader understood it

how the analyst designed it

The Project Cartoon

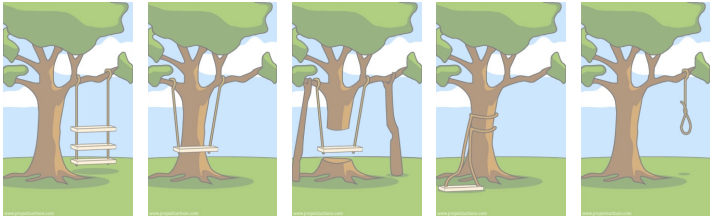
how the customer explained it

how the project leader understood it

how the analyst designed it

how the programmer implemented it

The Project Cartoon



how the customer explained it | how the project leader understood it | how the analyst designed it | how the programmer implemented it | what the beta testers received

The Project Cartoon

how the customer explained it

how the project leader understood it

how the analyst designed it

how the programmer implemented it

what the beta testers received

how the business consultant described it

The Project Cartoon

how the customer explained it

how the project leader understood it

how the analyst designed it

how the programmer implemented it

what the beta testers received

how the business consultant described it

how the customer was billed

The Project Cartoon

how the customer explained it

how the project leader understood it

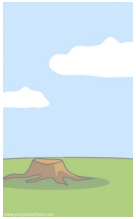
how the analyst designed it

how the programmer implemented it

what the beta testers received

how the business consultant described it

how the customer was billed



how it was supported

The Project Cartoon

how the customer explained it

how the project leader understood it

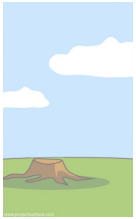
how the analyst designed it

how the programmer implemented it

what the beta testers received

how the business consultant described it

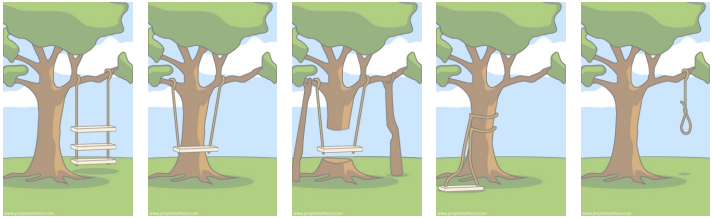
how the customer was billed



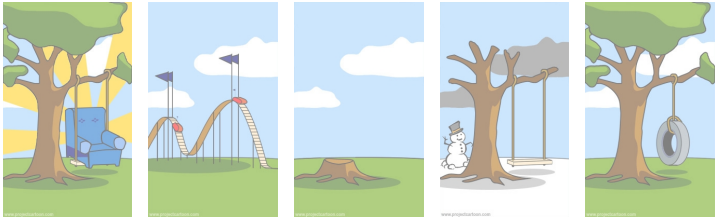
how it was supported

when it was delivered

The Project Cartoon



how the customer explained it | how the project leader understood it | how the analyst designed it | how the programmer implemented it | what the beta testers received



how the business consultant described it | how the customer was billed | how it was supported | when it was delivered | what the customer really needed

The Project Cartoon

Why Do Software Projects Fail?

- many stakeholders with different background each
- miscommunication
- implicit or wrong assumptions
- time pressure
- high complexity
- ...
- numerous reasons specific to certain phases and roles

how the customer explained it

how the project leader understood it

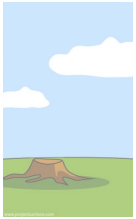
how the analyst designed it

how the programmer implemented it

what the beta testers received

how the business consultant described it

how the customer was billed



how it was supported

when it was delivered

what the customer really needed

The Project Cartoon

Why Do Software Projects Fail?

- many stakeholders with different background each
- miscommunication
- implicit or wrong assumptions
- time pressure
- high complexity
- ...
- numerous reasons specific to certain phases and roles

software engineering aims to reduce such problems

how the customer explained it

how the project leader understood it

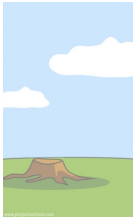
how the analyst designed it

how the programmer implemented it

what the beta testers received

how the business consultant described it

how the customer was billed



how it was supported

when it was delivered

what the customer really needed

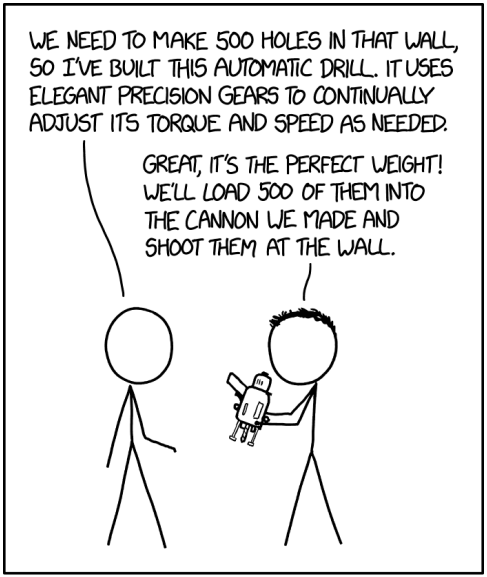
Software Engineering

Software Engineering [Sommerville]

"Software engineering is an engineering discipline that is concerned with all aspects of software production from initial conception to operation and maintenance. [...] Software engineering is not just concerned with the technical processes of software development. It also includes activities such as software project management and the development of tools, methods, and theories to support software development."

Software Engineering vs Programming

what is needed besides programming will be motivated and shown throughout this course



HOW SOFTWARE DEVELOPMENT WORKS

Software Engineering vs Computer Science

SE vs CS [Sommerville]

"Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software. [...] Computer science theory, however, is often most applicable to relatively small programs. Elegant theories of computer science are rarely relevant to large, complex problems that require a software solution."

Software and System Engineering

System Engineering [Sommerville]

"System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process."

(Software) Engineering

Engineering [Sommerville]

"Engineering is about getting results of the required **quality** within **schedule** and **budget**. [...] Engineers make things work. They apply theories, methods, and tools where these are appropriate. However, they use them selectively and always try to discover solutions to problems even when there are no applicable theories and methods. Engineers also recognize that they must work within organizational and financial constraints, and they must look for solutions within these constraints."

Relevance of Software for This Course

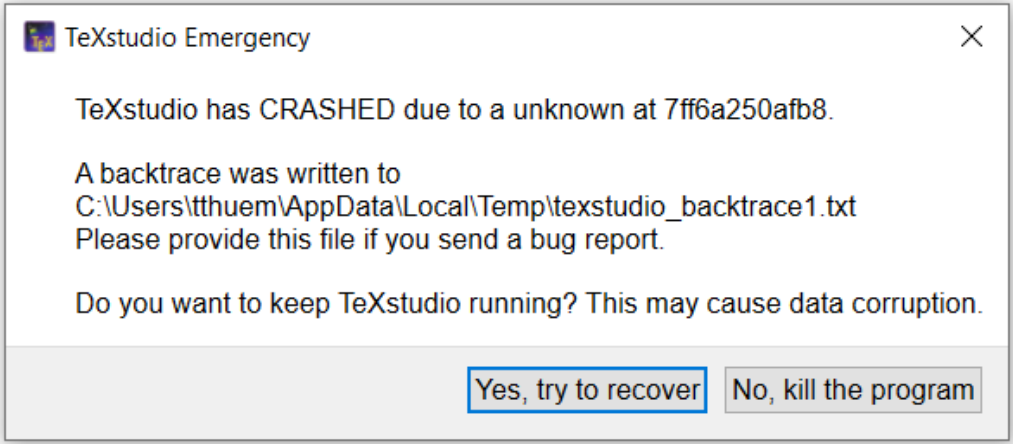
Google

500. That's an error.

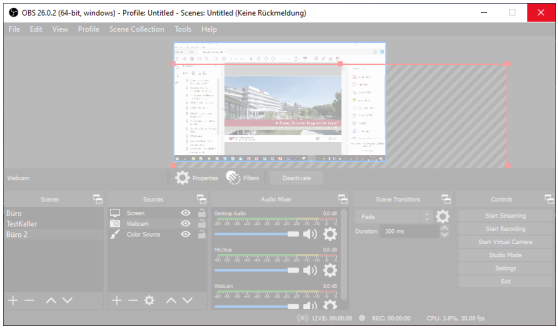
The server encountered an error and could not complete your request.

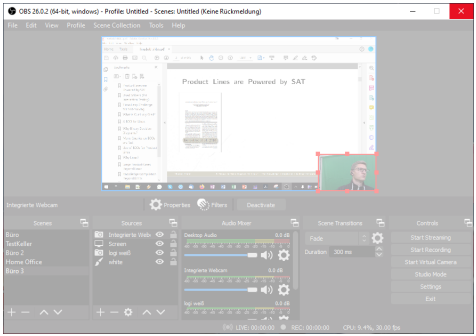
If the problem persists, please report your problem and mention this error message and the query that caused it.
That's all we know.

Relevance of Software for This Course

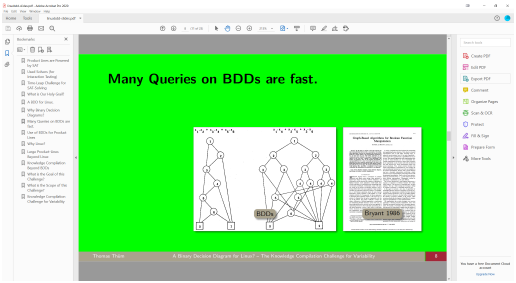


Relevance of Software for This Course

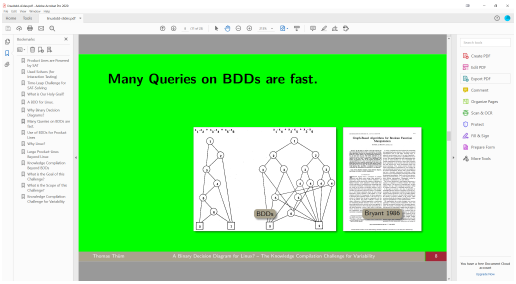


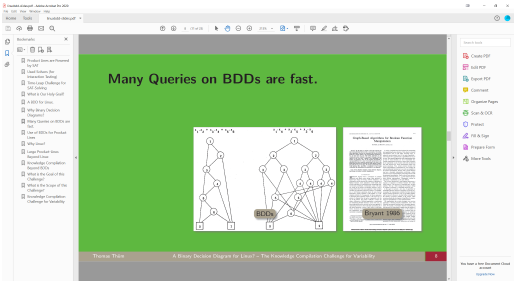


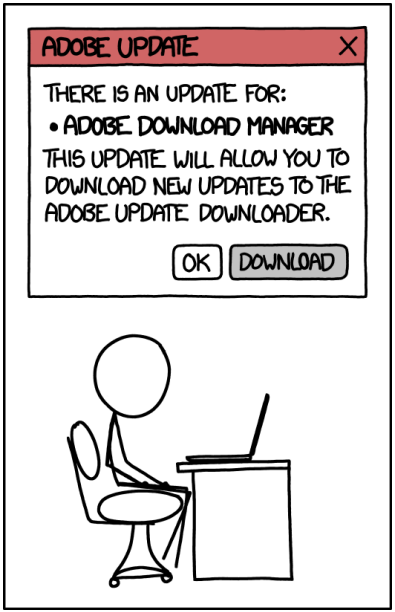
Relevance of Software for This Course



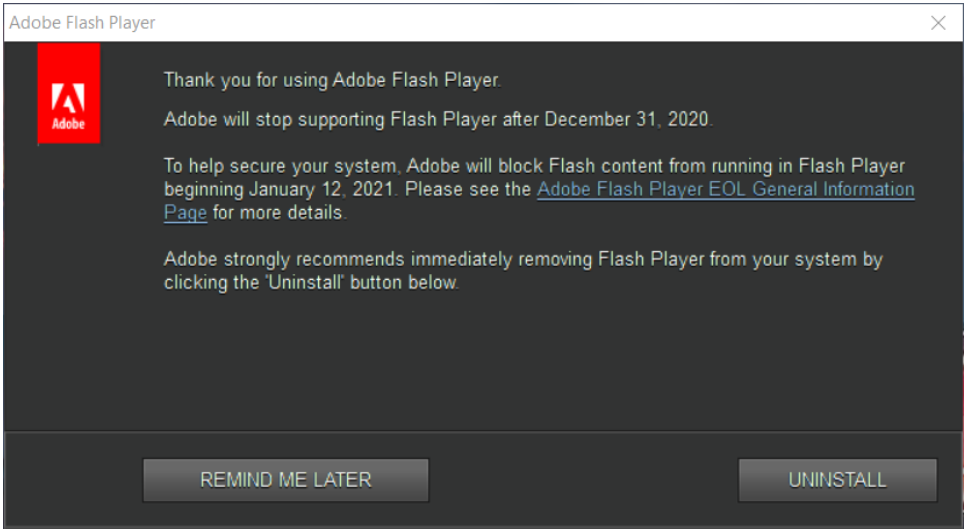
Relevance of Software for This Course



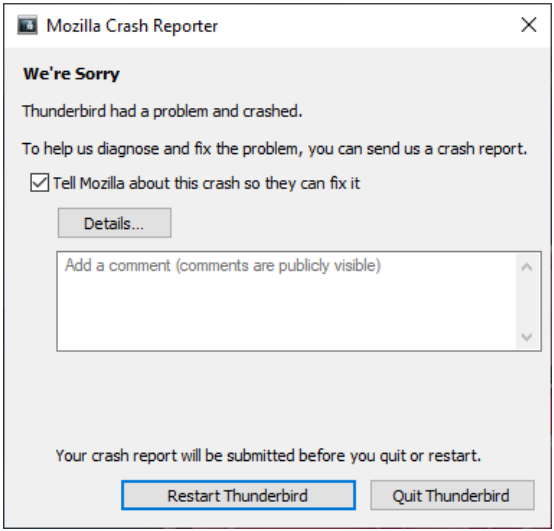



ADOBE UPDATE
THERE IS AN UPDATE FOR:
• ADOBE DOWNLOAD MANAGER
THIS UPDATE WILL ALLOW YOU TO DOWNLOAD NEW UPDATES TO THE ADOBE UPDATE DOWNLOADER.
OK
DOWNLOAD

Relevance of Software for This Course



Relevance of Software for This Course



Software and Its Engineering – Summary

Lessons Learned

- What is software engineering?
- Which trade-off is crucial to software engineering?
- Next: What is expected from you during this term?

Software and Its Engineering – Summary

Lessons Learned

- What is software engineering?
- Which trade-off is crucial to software engineering?
- Next: What is expected from you during this term?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Software and Its Engineering – Summary

Lessons Learned

- What is software engineering?
- Which trade-off is crucial to software engineering?
- Next: What is expected from you during this term?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Discuss: What will you do for a living in 10 years? What will be your connection to software? (5 min)
3. Survey: What is your connection to software in 10 years?

Software and Its Engineering – Summary

Lessons Learned

- What is software engineering?
- Which trade-off is crucial to software engineering?
- Next: What is expected from you during this term?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Discuss: What will you do for a living in 10 years? What will be your connection to software? (5 min)
3. Survey: What is your connection to software in 10 years?

A Wirst du in 10 Jahren Software entwickeln?

Software and Its Engineering – Summary

Lessons Learned

- What is software engineering?
- Which trade-off is crucial to software engineering?
- Next: What is expected from you during this term?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Discuss: What will you do for a living in 10 years? What will be your connection to software? (5 min)
3. Survey: What is your connection to software in 10 years?

A Wirst du in 10 Jahren Software entwickeln?

B Wirst du in 10 Jahren Software testen?

Software and Its Engineering – Summary

Lessons Learned

- What is software engineering?
- Which trade-off is crucial to software engineering?
- Next: What is expected from you during this term?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Discuss: What will you do for a living in 10 years? What will be your connection to software? (5 min)
3. Survey: What is your connection to software in 10 years?

A Wirst du in 10 Jahren Software entwickeln?

B Wirst du in 10 Jahren Software testen?

C Wirst du in 10 Jahren Software beauftragen?

Software and Its Engineering – Summary

Lessons Learned

- What is software engineering?
- Which trade-off is crucial to software engineering?
- Next: What is expected from you during this term?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Discuss: What will you do for a living in 10 years? What will be your connection to software? (5 min)
3. Survey: What is your connection to software in 10 years?

A Wirst du in 10 Jahren Software entwickeln?

B Wirst du in 10 Jahren Software testen?

C Wirst du in 10 Jahren Software beauftragen?

D Wirst du in 10 Jahren Softwareanforderungen ermitteln?

Software and Its Engineering – Summary

Lessons Learned

- What is software engineering?
- Which trade-off is crucial to software engineering?
- Next: What is expected from you during this term?

Further Reading

- Sommerville, Chapter 1.1, pp. 19–28

Practice

1. Form groups of 2-3 students
2. Discuss: What will you do for a living in 10 years? What will be your connection to software? (5 min)
3. Survey: What is your connection to software in 10 years?

A Wirst du in 10 Jahren Software entwickeln?

B Wirst du in 10 Jahren Software testen?

C Wirst du in 10 Jahren Software beauftragen?

D Wirst du in 10 Jahren Softwareanforderungen ermitteln?

E Wirst du in 10 Jahren die Entwicklung von Software leiten?

1. Introduction

1a. The Impact of Software

1b. Software and Its Engineering

1c. Course Overview

Structure of This Course

Part A: How to develop correct software?

A common problem of software is its **insufficient quality**. In the first part we will investigate the basics in software engineering to improve the quality of software. First, we start with the right **choice of a programming language** and the available tools to support the development. Second, we give an overview on software quality and the two orthogonal **testing techniques**, namely white-box and black-box testing. Third, we illustrate the role of **changes to software** and their impact on software quality. Finally, we discuss how to **keep track of versions** and how to detect problems with software faster.

Structure of This Course

Part A: How to develop correct software?

A common problem of software is its **insufficient quality**. In the first part we will investigate the basics in software engineering to improve the quality of software. First, we start with the right **choice of a programming language** and the available tools to support the development. Second, we give an overview on software quality and the two orthogonal **testing techniques**, namely white-box and black-box testing. Third, we illustrate the role of **changes to software** and their impact on software quality. Finally, we discuss how to **keep track of versions** and how to detect problems with software faster.

Part B: How to develop software in schedule and budget?

Developing high quality software is typically conflicting with the available budget and the time available for development. Good software engineering cannot concentrate only on quality, but also needs to focus on how software projects meet their schedule and budget. Here it is not enough to consider a single project in isolation, but instead maintain a well functioning teams in the long run. First, we start with a basic introduction into **project management**, which is keeping track of the progress of a software project and adjusting the development where required. Second, we give an overview on different **process models** such as scrum, which split the development into dedicated phases.

Structure of This Course

Part A: How to develop correct software?

A common problem of software is its **insufficient quality**. In the first part we will investigate the basics in software engineering to improve the quality of software. First, we start with the right **choice of a programming language** and the available tools to support the development. Second, we give an overview on software quality and the two orthogonal **testing techniques**, namely white-box and black-box testing. Third, we illustrate the role of **changes to software** and their impact on software quality. Finally, we discuss how to **keep track of versions** and how to detect problems with software faster.

Part B: How to develop software in schedule and budget?

Developing high quality software is typically conflicting with the available budget and the time available for development. Good software engineering cannot concentrate only on quality, but also needs to focus on how software projects meet their schedule and budget. Here it is not enough to consider a single project in isolation, but instead maintain a well functioning teams in the long run. First, we start with a basic introduction into **project management**, which is keeping track of the progress of a software project and adjusting the development where required. Second, we give an overview on different **process models** such as scrum, which split the development into dedicated phases.

Part C: How to develop needed software?

Even if software is of high quality and its development has met its time and budget constraints, it does not necessarily mean that the software is useful at all. Software is supposed to solve a problem of its users and it is far from trivial to develop software that is actually needed. The project cartoon illustrates common problems, which we distinguish into whether the problem to be solved is understood (**requirements** and **system modeling** aka. analysis) and whether the envisioned solution fits its purpose (**software architecture** and **software design** aka. design). Finally, software engineering is not only about developing software from scratch but also about **reusing existing software** where appropriate, with the potential to drastically reduce costs and time-to-market.

Structure of This Course

Part A: Development of Correct Software

1. **Introduction**
2. Implementation
3. Testing
4. Software Changes
5. Version Control

Part B: Development in Schedule & Budget

6. Project Management
7. Development Process

Part C: Development of Needed Software

8. Requirements
9. System Modeling
10. Software Architecture
11. Software Design
12. Software Reuse
13. Summary

About This Course

Software Engineering 1

- Abbreviation: SE1
- Credits: 5 ECTS
- Semester hours: 2+1
- Courses of studies:
  - B. Sc.: Informatik, Wirtschaftsinformatik, ...
  - M. Sc.: ...

Disclaimer

Passing this course is required to participate in the SEP (Softwareentwicklungspraktikum) in summer term!

About This Course

Software Engineering 1

- Abbreviation: SE1
- Credits: 5 ECTS
- Semester hours: 2+1
- Courses of studies:
  - B. Sc.: Informatik, Wirtschaftsinformatik, ...
  - M. Sc.: ...

Disclaimer

Passing this course is required to participate in the SEP (Softwareentwicklungspraktikum) in summer term!

Course Organization

- 2 hours: typically one lecture per week, see schedule in Stud.IP (Lecture)
- 1 hour: one exercise (2 hours) every two weeks, see schedule in Stud.IP (Exercise)
- Written exam at the end of the term

The Lectures

The Lectures

- weekly, Tuesday 11:30–13:00
- lecture follows the Sandwich model
- three lecture parts
- interactive tasks in between (some require smartphone, tablet, notebook)
- lecture recordings planned (but without any guarantee)

The Lectures

The Lectures

- weekly, Tuesday 11:30–13:00
- lecture follows the Sandwich model
- three lecture parts
- interactive tasks in between (some require smartphone, tablet, notebook)
- lecture recordings planned (but without any guarantee)

Adam Osborne

"The most valuable thing you can make is a mistake – you can't learn anything from being perfect."

The Exercises

The Exercises

- goal: deeper understanding of lecture topics, preparation for the exam
- bi-weekly, concrete dates in Stud.IP (Exercise)
- starting in **third** week (special exercise in **second** week)
- at most **15 participants** in each exercise
- assignment to time slots via Stud.IP (Exercise)
- switching between or attending multiple exercises is **not allowed**
- holidays: alternative dates discussed in affected exercises

The Exercises

The Exercises

- goal: deeper understanding of lecture topics, preparation for the exam
- bi-weekly, concrete dates in Stud.IP (Exercise)
- starting in **third** week (special exercise in **second** week)
- at most **15 participants** in each exercise
- assignment to time slots via Stud.IP (Exercise)
- switching between or attending multiple exercises is **not allowed**
- holidays: alternative dates discussed in affected exercises

Voting System (Votierungssystem)

- 7 sheets with 5-8 tasks each
- participants prepare solutions for the tasks **before** the exercise
- in the 15 minutes before the exercise, participants can vote (votieren) tasks they prepared in a dedicated list (Votierungsliste)
- during the exercise one or multiple prepared participants are selected to present their solution at the whiteboard (or beamer)
- at least 50 % votes (Votierungspunkte) and at least 3 presentations (Vortragspunkte) required to pass the exercise (Studienleistung)
- students may prepare solutions in groups but every member needs to be able to explain the solution if voted

The Exercises

The Exercises

- goal: deeper understanding of lecture topics, preparation for the exam
- bi-weekly, concrete dates in Stud.IP (Exercise)
- starting in **third** week (special exercise in **second** week)
- at most **15 participants** in each exercise
- assignment to time slots via Stud.IP (Exercise)
- switching between or attending multiple exercises is **not allowed**
- holidays: alternative dates discussed in affected exercises

Voting System (Votierungssystem)

- 7 sheets with 5-8 tasks each
- participants prepare solutions for the tasks **before** the exercise
- in the 15 minutes before the exercise, participants can vote (votieren) tasks they prepared in a dedicated list (Votierungsliste)
- during the exercise one or multiple prepared participants are selected to present their solution at the whiteboard (or beamer)
- at least 50 % votes (Votierungspunkte) and at least 3 presentations (Vortragspunkte) required to pass the exercise (Studienleistung)
- students may prepare solutions in groups but every member needs to be able to explain the solution if voted

Exception Handling

- tasks without prepared participants will not be discussed
- unprepared/cheating participants risk to loose one or all votes for the current sheet (incl. presentation points)

The Exercises

Special Exercises in Second Week

- following rules only apply to the second week!
- special offer for students with limited or no programming skills
- good preparation for later lecture topics on programming and testing
- come unprepared (i.e., no voting)
- participation not necessary, but recommended
- time slot of both alternating weeks merged

How to Find Answers

1. Ask questions during the lecture (e.g., during interactive parts)

The Exercises

Special Exercises in Second Week

- following rules only apply to the second week!
- special offer for students with limited or no programming skills
- good preparation for later lecture topics on programming and testing
- come unprepared (i.e., no voting)
- participation not necessary, but recommended
- time slot of both alternating weeks merged

How to Find Answers

1. Ask questions during the lecture (e.g., during interactive parts)
2. Check information in StudIP
   - Check already answered questions
   - Ask your own questions and answer questions of fellow students

The Exercises

Special Exercises in Second Week

- following rules only apply to the second week!
- special offer for students with limited or no programming skills
- good preparation for later lecture topics on programming and testing
- come unprepared (i.e., no voting)
- participation not necessary, but recommended
- time slot of both alternating weeks merged

How to Find Answers

1. Ask questions during the lecture (e.g., during interactive parts)
2. Check information in StudIP
   - Check already answered questions
   - Ask your own questions and answer questions of fellow students
3. Ask questions in your exercise

The Exercises

Special Exercises in Second Week

- following rules only apply to the second week!
- special offer for students with limited or no programming skills
- good preparation for later lecture topics on programming and testing
- come unprepared (i.e., no voting)
- participation not necessary, but recommended
- time slot of both alternating weeks merged

How to Find Answers

1. Ask questions during the lecture (e.g., during interactive parts)
2. Check information in StudIP
   - Check already answered questions
   - Ask your own questions and answer questions of fellow students
3. Ask questions in your exercise
4. Meet Thomas in his consultation hour: Wednesday 2pm in IZ 348 (preregistration useful)

The Exercises

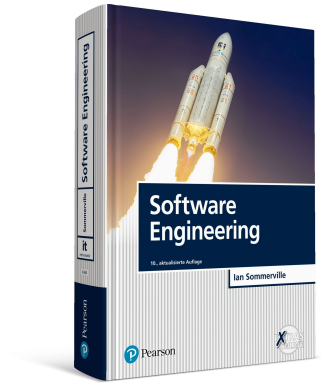
Special Exercises in Second Week

- following rules only apply to the second week!
- special offer for students with limited or no programming skills
- good preparation for later lecture topics on programming and testing
- come unprepared (i.e., no voting)
- participation not necessary, but recommended
- time slot of both alternating weeks merged

How to Find Answers

1. Ask questions during the lecture (e.g., during interactive parts)
2. Check information in StudIP
   - Check already answered questions
   - Ask your own questions and answer questions of fellow students
3. Ask questions in your exercise
4. Meet Thomas in his consultation hour: Wednesday 2pm in IZ 348 (preregistration useful)
5. Slowest option: Contact us via e-mail

Literature for This Course



[Sommerville]

- Ian Sommerville. Software Engineering, 10. Edition, Pearson, 2018.
  - German, English, and earlier versions
  - Videos by Ian Sommerville and others available online
- More literature announced in each lecture

Course Overview – Summary

Lessons Learned

- How is this course organized?
- Next Lecture: Why do programming languages matter?

Course Overview – Summary

Lessons Learned

- How is this course organized?
- Next Lecture: Why do programming languages matter?

Further Reading

- Main Book: Ian Sommerville. Software Engineering, 10. Edition, Pearson, 2018.

Course Overview – Summary

Lessons Learned

- How is this course organized?
- Next Lecture: Why do programming languages matter?

Further Reading

- Main Book: Ian Sommerville. Software Engineering, 10. Edition, Pearson, 2018.

Practice

- Any questions?

Software Engineering 1

Part A: Development of Correct Software

1. **Introduction**
2. Implementation
3. Testing
4. Software Changes
5. Version Control

Part B: Development in Schedule & Budget

6. Project Management
7. Development Process

Part C: Development of Needed Software

8. Requirements
9. System Modeling
10. Software Architecture
11. Software Design
12. Software Reuse
13. Summary