Software Development Methodologies

Lecture 1 - Introduction

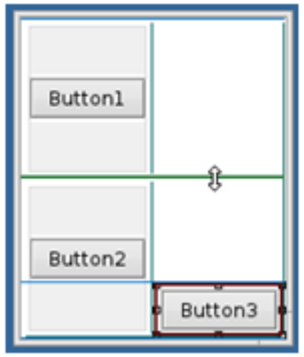
SOFTENG 750 2013-03-04

Christof Lutteroth

SE Software Engineering
The University of Auckland

- Originally from Berlin, Germany
- My research interests:
  HCI, Software tools
- Contact details:
  lutteroth@cs.auckland.ac.nz
  Phone 373-7599 84478
  Office 303 - 494 (4th level CompSci building)

- If you have questions, come to my office or email me



Features
History-based editing scheme
Manipulation of the order of history
2 key dependency cases:
- Generalisation
- Specialisation

SE750 Overview

When	What	Where
Mo 2-3pm	Lecture	Geol 1060
Wed 2-3pm	Lecture	Geol 1060
Thu 3-4pm	Lab (**5%** milestones)	UG4
At least twice every week	Your project team meeting	Up to you
Mo 29/4 7pm	Group Report **10%**	Your repository
Near the end of the course	Group Presentation **5%**	During lectures & labs
Mo 3/6 7pm	Final Report **20%**	Cecil
TBA	Exam **60%**	TBA

Workload: 10 hours per week

Introduction

Not all those who wander are lost.
(J.R.R. Tolkien)

Get the course overview from Cecil!

How to Be Successful as a Postgrad (and Beyond)

For all types of research projects

1. **Commitment**
   - Work every day, “It’s like a job”, persistence
   - Always stay in touch with your supervisors
2. **Development**
   - Good programming skills
   - Know how to use tools and best practices
   - Independent problem solving, persistence
3. **Communication Skills**
   - Willing to read & analyze a lot of related work
   - Writing skills: spelling/grammar/style/logic
   - Presentation (verbal) skills

Course Mottos

You are a Software Engineer!

This is a real project.

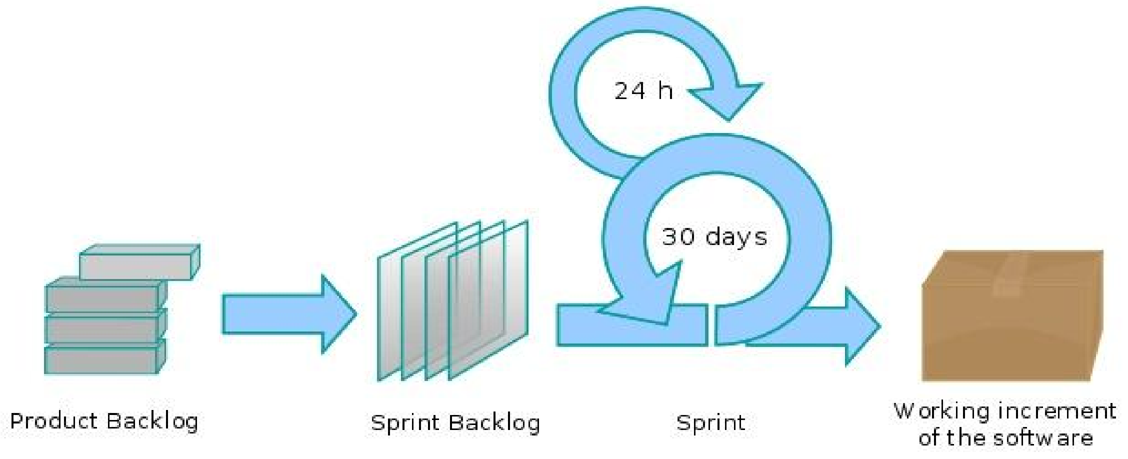
Christof and Ian are your customers.

Agile Development

- Good for small teams (such as groups of students)
- Lightweight: focus on the essential principles

SCRUM

- General product management methodology
- Using agile principles, e.g. control by feedback, iterations



eXtreme Programming (XP)

- Principles focused on code quality
- We will look at principles & practices in more detail

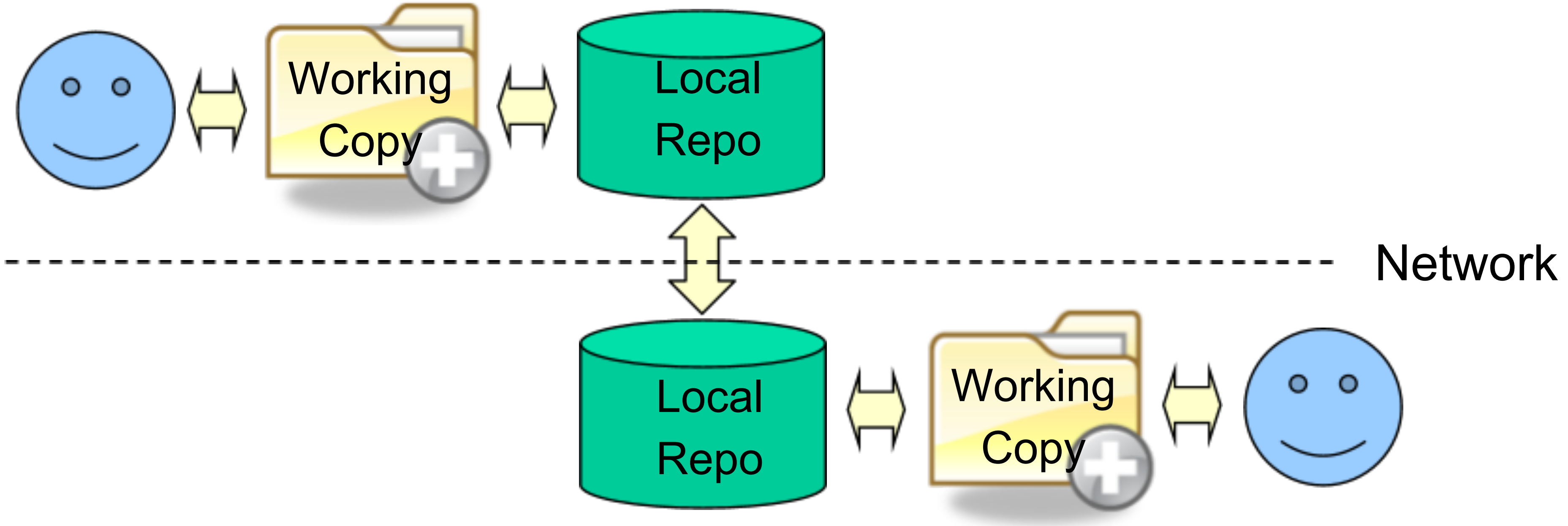
Version Control

Centralized Version Control Systems (CVCSs)

- All developers work with the same repository
- E.g. SVN

Distributed Version Control Systems (DVCSs)

- Every developer has their own local repository
- Versions are exchanged between repositories
- E.g. Mercurial, Git



Implementation Part: Android Mobile App

Aims:

- Gain some "real" team development experience
- Learn how to target real users
- Learn to evaluate and present your work
- Learn a new technology

How to get there

- Team project (group of 4)
- Weekly project milestones (in the labs)
- Interim group report & final individual report
- Final group presentation
- Lectures about Android

User Studies

How do we know your software is useful?

- Software Engineers are biased: we like our own software
- Your friends are also biased

Solution: conduct a user study

1. Define **hypotheses**
2. Choose a **methodology** to investigate them
3. Recruit participants and collect **data** from them
4. Interpret your data to verify if hypotheses are true

Without a user study you cannot be sure whether your software is great or not.

Academic Writing

- It's hard, but if you understand the principles it becomes much easier
- Getting and learning from feedback is the key
- Writing happens on different levels:
  1. **Spelling & Grammar**: learn it from a proof reader
  2. **Style**: learn it from other authors
  3. **Structure**: learn it here

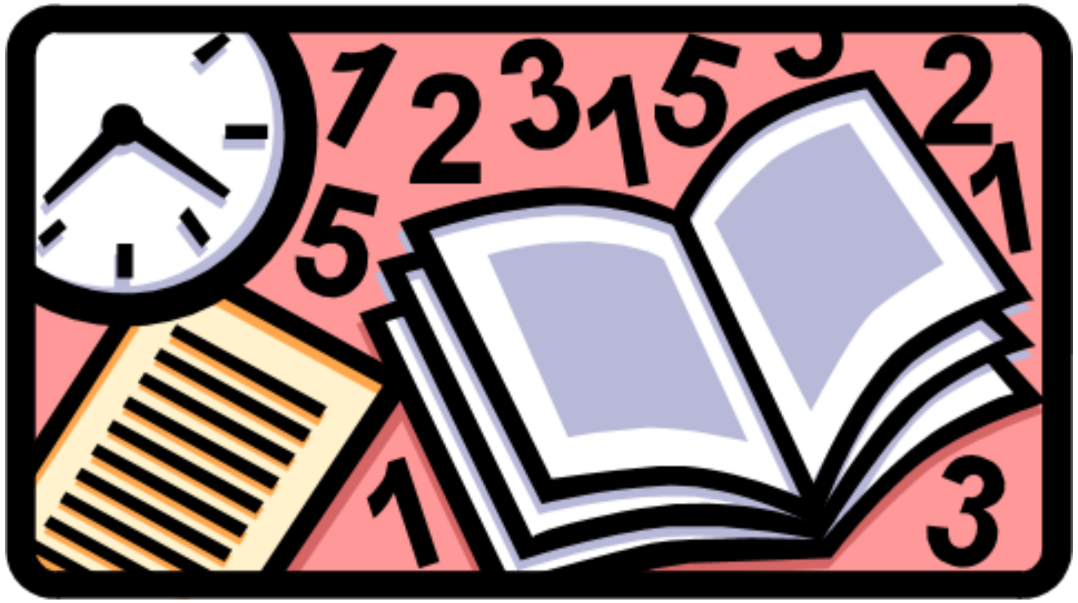
How do you convince a reader that your research results are...
- **Useful**: motivation, e.g. with applications
- **New**: filling a gap in the scientific literature
- **True**: logical arguments & empirical evaluation

Learning Outcomes

After the course, you will be able to...

1. Apply some principles of **agile development** processes.
2. Use some modern software **tools** such as VCSs.
3. Program **Android apps**.
4. Apply some **project management** principles.
5. Perform a **user study** to evaluate your work critically.
6. Get better at **writing** well-founded, clear technical reports.
7. Get better at **presenting** your work to an audience.

Project Guidelines



Team Project

- Form a **team of 4**
- Propose project idea in an **abstract**, get it approved: What? Why? For whom? How? (100-200 words)
  - Must be an Android app
  - Must be well-motivated (realistic)
  - Must have a non-trivial user interface
- Each week there will be a project **milestone**, to be delivered to the lab (5% of final mark)
- May use 3rd-party components, but must have a **substantial own code contribution**
  - VCS repository logs will be used to check activity
  - Logs will be used to check individual contributions

Lab Milestones (Tentative)

1. Team registration & project abstract
2. Initial low-fi prototype for intended app (paper-based)
3. Refined low-fi prototype (paper-based)
4. Set-up of version control system with evidence of use
5. Development of Android app (evidence in repo)
6. Development of Android app (evidence in repo)
7. Development of Android app (evidence in repo)
8. User study plan
9. Initial usability results, to be iterated
10. Final report outline

Interim and Final Reports

Interim Report (Deadline: Monday 29/4 7pm, 10%)

- Only one report per team, everyone must contribute
- 4 pages IEEE style, including figures & bibliography
- Similar to part 4 intermediate report
- Graded by marker

Final Report (Deadline: Monday 3/6 7pm, 20%)

- Individually written: may share figures/diagrams/tables but **must not share any text**
- 5-8 pages IEEE style, including figures & bibliography
- Graded by marker

Interim Report
Grading Schedule

Approx. **4 pages** (including figures) IEEE style

1. **Introduction**: Introduce and motivate the project.
2. **Requirements:** What is the project trying to achieve?
3. **Related Work:** What have others done? Compare it with your project.
4. **Initial Design:** Software architecture (e.g. class diagram)? User interface (e.g. screen diagram)?
5. **Future Work:** What comes next?
6. **Conclusion**: Conclusions / lessons so far?

Final Report
Grading Schedule

Approx. **5 pages** (including figures) IEEE style

1. **Introduction**: Introduce and motivate the project.
2. **Requirements:** What is the project trying to achieve?
3. **Related Work:** What have others done? Compare it with your project.
4. **Design:** Software architecture (e.g. class diagram)? User interface (e.g. screen diagram)?
5. **Implementation:** What have *you* implemented?
6. **Evaluation:** How have you evaluated your app? What are the results?
7. **Methodology/Management** of your team in the project.
8. **Conclusion**: Conclusions? Lessons? Future work?

Today's Summary

SE750 will cover various aspects of practical software development: processes, VCSs, user studies, technical reports, mobile development...

References:

- Course information sheet & slides on on Cecil
- Lecture videos on Christof's home page: http://www.cs.auckland.ac.nz/~lutteroth/teachings.html

Milestone 1 (Deadline: Lab on Thursday)

1. Form a team of 4
2. Decide on a project together
3. Email group member names & UPIs and project abstract to Christof

Quiz

1. How can you be a successful postgraduate student?
2. What would be a cool project for your team?
3. How can you manage your team and use the right tools so that your project will be successful?

International Obfuscated C Code Contest (ioccc.org)