Software Tools
Introduction to Part II

Part II - Lecture 1

Christof Lutteroth

- Originally from Berlin, Germany
- My research interests: HCI, Software Engineering
- 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
- A good time to see me is directly after the lectures

Part II Timetable

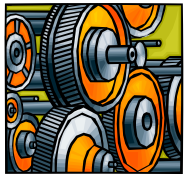
When		What	Where
Monday	13-14	Lecture	279
Tuesday	13-14	Lecture/Tutorial	279
Thursday	13-14	Lab	GCL
At least twice every week		Your project team meeting	You decide
1st June	7pm	Assignment 2 (25%)	ADB
TBA		Exam (50%)	TBA

Introduction to Part II

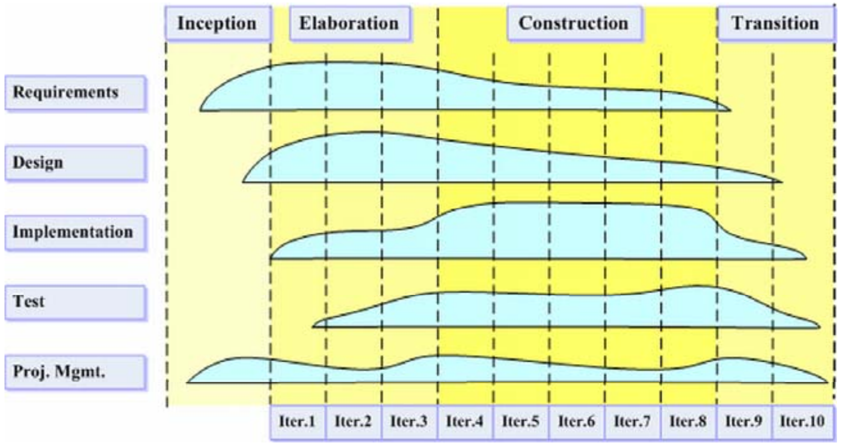
Software Tools

- Humans are necessary for creative, intelligent tasks
- Tools can **support** such tasks
  - Increase productivity with useful functionality
  - Guide the developer (e.g. context help)
  - Avoid defects

- Humans are not necessary for highly repetitive, routine work
- Tools can **automate** such tasks
  - Increase productivity; more time for creative work
  - Avoid defects introduced by the human factor



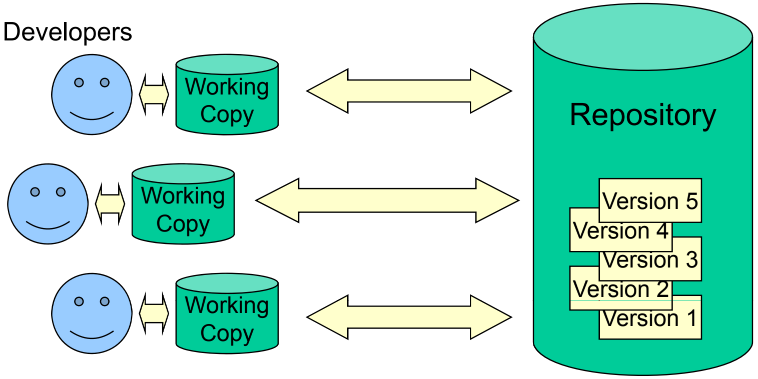
Development Processes (Example: RUP)



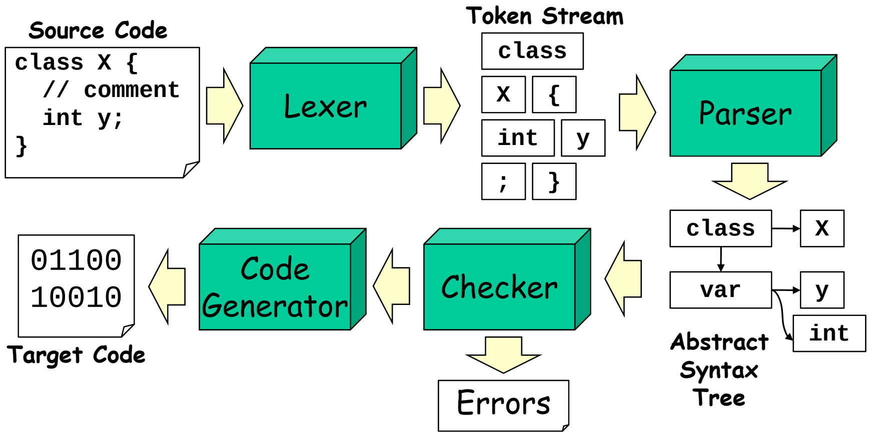
2006 Giles Lewis

Version Control Systems

- Technology to manage changes that several developers do on a common repository
- Changes create new version of the changed files
- Old versions are always accessible



Compilers



```
class X {
  // comment
  int y;
}
```

- Lexer chops the source code into tokens
- Parser constructs the syntactic relations between the tokens (abstract syntax tree, AST)

Type Systems

Type Checking: detect potential runtime errors in source code

```
int m(String s) {
   int y = s + 1;
   m(y,3);
   return s;
}
```

Type System: Formalize type checking by using rules that describe correct programs

(Expr Plus)

$$\frac{\Gamma \vdash E_1 : Nat \qquad \Gamma \vdash E_2 : Nat}{\Gamma \vdash E_1 + E_2 : Nat}$$

(Expr NotEq)

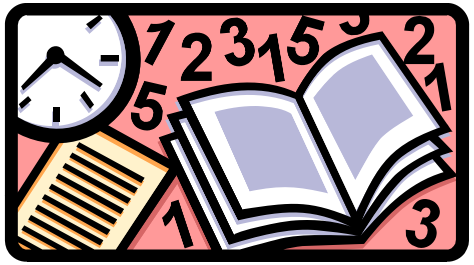
$$\frac{\Gamma \vdash E_1 : Nat \qquad \Gamma \vdash E_2 : Nat}{\Gamma \vdash E_1 \text{ not= } E_2 : Bool}$$

Learning Outcomes

After the course you should be able to…

- Apply some of the **eXtreme Programming** practices
- Describe main concepts of **version control systems**
- Use **Subversion** to efficiently work in a team
- Use ANTLR to create your own **lexers and parsers**
- Do simple type derivations using formal **type systems**
- Write simple academic **research papers**

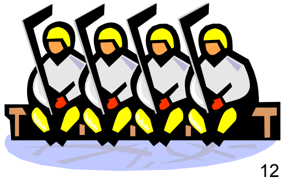
Assignment 2



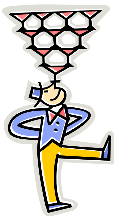
Assignment 2

25% in total for one team research project

- Teams of **four**
- Choose a given topic or come up with your own
- Every project has a **mentor**
  - Mentor is there to guide the project
  - Mentor has regular meetings with team
  - Team's responsibility to ask mentor
- Submit **individual report**
  - 5 pages IEEE style
  - Text, figures, bibliography
  - Will be graded by marker
  - 1st June Assignment Dropbox

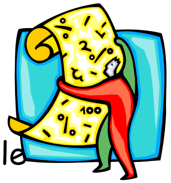


Project Expectations



- The project is flexible and scalable Expectations:
  - Work as a **team** (you can work with other teams as well!)
    - Have a project group meeting every week
    - If you are stuck, ask! (your teammates, mentor, other teams, lecturer)
    - Come to the lectures/labs (you will learn what you need to do a good project)
  - **20 hours** for development (commit to the repo)
  - **10 hours** for report (IEEE style)
  - Only the project report has to be written individually (everything else can be teamwork)

Report Grading Schedule



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

1. **Introduction**: Have you introduced the project and its aims?
2. **Related Work**: Project background? Have you cited and described academic related work (≥4 **citations**)?
3. **Requirements**: What needed to be done? Why?
4. **Design**: How did you design your solution? Why?
5. **Implementation**: How did you implement it? What did you contribute? How did the team work? Challenges?
6. **Conclusion**: Achievements? Conclusions? Lessons? Future/unfinished work?

Suggested Projects

PDStore Projects

- *A database system developed here at Uni*
- Cool features:
  - Versioning
  - Structured/unstructured data
  - Compression
  - Merging
- Comes with tools:
  - Visual data editor PDEdit
  - Data access layer generator PDGen
- *Group of Masters & PhD students working on it*

PDStore Projects

1. Configurable shapes for PDEdit (Ted)
2. Automatic layout for PDEDit (Ted)
3. Performance engineering (bottleneck analysis, optimizations, caching, Btrees…) (Daniel)
4. Extraction of Wikipedia into PDStore (Lian)
5. Port PDStore to Python (Danver)
6. Flexible object persistence in Python (Danver)
7. SPARQL for PDStore (Mark)

Daniel (zden011), Danver (dbra072),

Lian (llee058), Mark (gsun014), Ted (tyeu008)

Auckland Interface Model (AIM)

- Cross-platform customization system for GUIs
- Allows you to change a GUI while it is running
- E.g. to make it easier, better, more personal

Projects:

1. Widget customization in AIM for Java (using latest PDStore) (Clemens)
2. Layout editing in AIM for Flash (Ted)
3. AIM for C# (Clemens)
4. AIM table widget (Ted)

Clemens (clemens.zeidler@googlemail.com),

Ted (tyeu008)

Haiku

- Novel open-source operating system
- Modular, coherent design
- Friendly and active community
- C++

Projects:

1. PDStore for Haiku
2. PDEdit for Haiku
3. AIM for Haiku
4. Text view with C++ code completion (using CLANG)

Mentor: Clemens (clemens.zeidler@googlemail.com)

Summary

Today's summary

Part 2 will cover various types of tools and techniques:

1. Development processes (eXtreme Programming)
2. Version Control Systems (Subversion, git)
3. Compilers
4. Type systems
5. Academic Writing (Assignment 2 report)

1. Form a team (max 4)
2. Choose a project together
3. Attend the lab to discuss your project with Christof (Thursday 1pm in GCL)!!!

Quiz

1. How can software tools help with repetitive routine tasks?
2. How can software tools help with creative tasks?
3. Name four of the five main topics covered in part 2.