Five years of a SE course at HU: experience, conclusions, didactic principles

Klaus Bothe

1. Changing course contents
2. Which textbook is suitable?
3. Motivation as an integral part of the course
4. Interactivity as an element of better comprehension
5. On the usage of slides
6. The organization of assignments
7. On the role of case studies
8. Tools
9. Examinations: form and contents
10. Guest lectures
11. Web presentation

1. Changing course contents

- **new subjects:**

  refactoring, extreme programming,
  cleanroom software engineering ...

- **subjects move to basic studies:**

  (e. g. to 1st semester)

  - principles of object orientation:
    classes, inheritance, class diagrams ...

  - style guides:
    naming conventions, layout of programs ...

  reason: Modula-2 --> Java

Software Engineering Course: C o n t e n t s ?

- **Theses: Each course in a computer science curriculum has to do with SE**

- **Theses: SE principles should be taught from the very beginning**

1st semester:
Java and OOP

3rd semester:
compiler construction
(SW project)

advanced studies:
→ SE course
OOA & AAD: UML
formal methods
SDL
SW testing
data base

- **Theses: There may be an overview course on SE which should be detailed in seperate courses**

2. Which textbook is suitable?

Sommerville (6th edition)

Pressman (5th edition)

- internationally recognized textbooks
- rather textual
- long tradition: fixed to traditional structures
- recommended as secondary literature

Braude: Software Engineering - an object-oriented perspective

- first version in 2001
- more modern perspective
- problems: not an ideal textbook
  (figures, exercises, still too textual ...)
- suitable secondary literature

(2. Which textbook is suitable? - cont.)

Balzert:

- disadvantage: in German ...
- ideal textbook:
  - a lot of excellent pictures
  - ppt slides available
  - a useful case study throughout the book:
    requirements document, cost estimation,
    OOA model, Structured Analysis Model
  - very systematic: classification of concepts
    (basic concepts)
  - very comprehensive (all important subfields covered)
  - exercises with answers
  - up-to-date

$\longrightarrow$ our way in project:
English slides
+ secondary reading
+ remarks for the instructor

3. Motivation is an integral part of the course

student's view:

software development is equal to programming

SE:

software development is much more than programming

$\longrightarrow$ topic 1 (chapter 1): give motivation

each chapter is accompanied with motivation
(e. g. examples from practice ...)

4. Interactivity as an element of better comprehension

- interactive lectures:

  dialogue with students,

  tutorial character

  pose questions to the students

  do not present complete or definitive solutions

SE means:

the subject is in move / development,
the answers to problems are nor unique,
software development depends also on subjectivity,

$\longrightarrow$ discuss with the students advantages and disadvantages of techniques, methods, tools ...

$\longrightarrow$ our slides will reflect this interactivity:
questions included ...
... and answers too
(problem: reuse of presentation over the years:
answers become known)

6. On the organization of assignments (exercises)

- SE without exercises is frustrating.
- The best SE exercise is a project.
- Educational project management takes too much time.

Examples of exercises:

- Review of requirement documents
  (to learn about such a document,
  to work in groups assessing documents)

- Apply function point method to a given
  requirements document (cost estimation)

- Derive UML class diagram from requ. documents

- Apply tools, e. g.
  Mc Cabe: calculate metrics for a given C++ program

- Determine test cases for a problem (cte tool)

7. On the role of case studies: projects

- Understanding the necessity of SE principles deeply requires real projects

- SE course: 2 case studies

  - commercial (book) example
    (provider for extension studies / training)

  - technical system: use case of XCTL

    -> review documents

    -> derive UML class diagram

    -> determine test cases

8. Tools

- Together: OOA-OOD (UML)
- McCabe Tool:
  metrics, reengineering, analysis of SW
- Tessy, cte:
  test system (test case determination)
  (DaimlerChrysler)
- Microsoft project
- cvs: version management (XCTL project)

10. Guests: the view of practice

- *Test methodology and Test tools*
  (J. Wegener, R. Pitschinetz, DaimlerChrysler)

- *Formal software specifikation with Z*
  (A. Fett, DaimlerChrysler):