# MTAT.03.094
# Software Engineering

## Lecture 14:

## Course wrap-up, review and exam preparation


Dietmar Pfahl

email: dietmar.pfahl@ut.ee

Fall 2013

# Schedule of Lectures

Week 01: Introduction to SE

Week 02: Requirements Engineering I

Week 03: Requirements Engineering II

Week 04: Analysis

Week 05: Development Infrastructure I

Week 06: Development Infrastructure II

Week 07: Architecture and Design

Week 08: Refactoring

Week 09: Quality Management

Week 10: Verification and Validation I

Week 11: Verification and Validation II

Week 12: Agile/Lean Methods

Week 13: Measurement and Process Improvement

Week 14: Course wrap-up, review and exam preparation

Week 15: no lecture

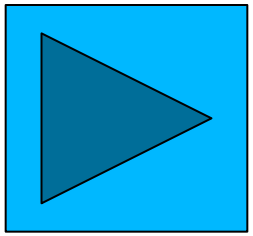
Week 16: no lecture

# Structure of Lecture 14

- Software Engineering as a Profession
- Review and Exam Preparation

# Professional and ethical responsibility

- Software engineering involves wider responsibilities than simply the application of technical skills.
- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.
- Ethical behaviour is more than simply upholding the law, it's about:
  - Competence
  - Confidentiality
  - Intellectual property rights (IPR)
  - Computer misuse



# Issues of professional responsibility

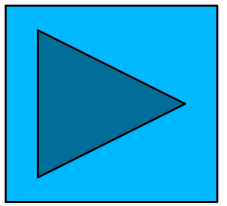
- Competence
  - Engineers should not misrepresent their level of competence.
  - They should not knowingly accept work which is outwith their competence.
- Confidentiality
  - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.

# Issues of professional responsibility

- Intellectual property rights
  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc.
  - They should be careful to ensure that the intellectual property of employers and clients is protected.
- Computer misuse
  - Software engineers should not use their technical skills to misuse other people's computers.
  - Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

# ACM/IEEE Code of Ethics

- The professional societies in the US have cooperated to produce a code of ethical practice.
- Members of these organisations sign up to the code of practice when they join.
- The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers,
  - including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.



# ACM/IEEE Code of Ethics - Principles

PUBLIC

- Software engineers shall act consistently with the public interest.

CLIENT AND EMPLOYER

- Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.

PRODUCT

- Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

# ACM/IEEE Code of Ethics - Principles

JUDGMENT

- Software engineers shall maintain integrity and independence in their professional judgment.

MANAGEMENT

- Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

PROFESSION

- Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

# ACM/IEEE Code of Ethics - Principles

COLLEAGUES

- Software engineers shall be fair to and supportive of their colleagues.

SELF

- Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

# Structure of Lecture 14

- Software Engineering as a Profession
- Review and Exam Preparation

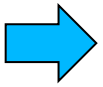
# Course Information/Overview

- Level: Advanced course at master's level (in English)
- Credits: 6 ECTS, 4 CP
- Prerequisite: MTAT.03.130 Object-oriented Programming (6 ECTS, 4 CP)
- Work load (per student):
  - Lectures: 26 hours
  - Lab work (incl. independent work): 14 x (2 + 5) = 98 hours
  - Exam preparation: 32 hours
- Assessment:
  - 7 Lab Assignment (team work) – 70% of grade
  - 1 Exam (written) – 30% of grade
- Grade scale: A (90%+), B(80%+), C(70%+), D(60%+), E(50%+), F

# Assessment (1)

- Labs – 70% of total grade
- Exam – 30% of total grade

- Proposed Exam Dates:
  - Exam 1: Friday 10-Jan-2014
  - Exam 2: Friday 17-Jan-2014
  - (Retake: ~~Wednesday 22-Jan-2014~~ Friday 24-Jan-2014)



You can take either exam 1 **or** 2
You must register!

  - Room: lecture hall (111) – for Exams 1 & 2
  - Time: 9.15 – 11.45

# Assessment (2)

**Labs – Practical Assessment**

10 points per lab session. Total = 70 points.

If you get less than 30 out of 70 points in the practical assessment, you will get a grade of 'F' in your first examination.

> In this case, you will be given a second chance to improve your practical assessment (lab) score.
>
> Condition: You have not received 0 marks in two or more labs due to non-submission

If your score after the second try is at least 30 out of 70, you will become eligible for a retake exam (korduseksam).

**Exam – Conceptual Assessment**

The Conceptual Assessment will consist of an exam worth 30 points.

Students who get less than 10 out of 30 in this exam, will get a grade of 'F', regardless of their Practical Assessment score.

This same rule will apply for the retake exam (korduseksam).

# Project Tasks (Labs)

- Week 01: no labs
- Weeks 02-05: Task 1: Requirements gathering
- Weeks 03-06: Task 2: Formalizing, modeling, planning
- Weeks 05-08: Task 3: Development infrastructure
- Weeks 07-10: Task 4: Realization - I
- Weeks 09-12: Task 5: Realization - II
- Weeks 11-14: Task 6: Automatic testing and refactoring
- Weeks 13-16: Task 7: Verification and validation

Remember that assessment/evaluation of lab task 7 will be in week 16 (Dec 16-20).

**Don’t miss the last lab!**

# Project Schedule

| Week | Task 1 | Task 2 | Task 3 | Task 4 | Task 5 | Task 6 | Task 7 |
|---|---|---|---|---|---|---|---|
| 2 | Assigned | | | | | | |
| 3 | Consult | Assigned | | | | | |
| 4 | Submit* | Consult | | | | | |
| 5 | Assess | Submit* | Assigned | | | | |
| 6 | | Assess | Consult | | | | |
| 7 | | | Submit* | Assigned | | | |
| 8 | | | Assess | Consult | | | |
| 9 | | | | Submit* | Assigned | | |
| 10 | | | | Assess | Consult | | |
| 11 | | | | | Submit* | Assigned | |
| 12 | | | | | Assess | Consult | |
| 13 | | | | | | Submit* | Assigned |
| 14 | | | | | | Assess | Consult |
| 15 | | | | | | | Submit* |
| 16 | | | | | | | Assess |

* = submit before midnight of the day before Lab

# Schedule of Lectures – relevant for Exam

Week 01: Introduction to SE

Week 02: Requirements Engineering I

Week 03: Requirements Engineering II

Week 04: Analysis

Week 05: Development Infrastructure I

Week 06: Development Infrastructure II

Week 07: Architecture and Design

Week 08: Refactoring

Week 09: Quality Management

Week 10: Verification and Validation I

Week 11: Verification and Validation II

Week 12: Agile/Lean Methods

Week 13: Measurement and Process Improvement

Week 14: Course wrap-up, review and exam preparation

Week 15: no lecture

Week 16: no lecture

# Topic: Introduction to SE

- Know how to define ’Software’, ’Engineering’, ’Software Engineering’
- Know how to explain the difference to ’Hardware Engineering’ and the special challenges of ’Software Engineering’
- Know the elements of a software development process
- Know different kinds of development methods/processes
- Know the various characteristics (attributes) of software quality

# Topic: Requirements Engineering I + II

- Know how to describe/define ’Requirements Engineering’ (RE)
- Know about the difficulties eliciting and describing software requirements
- Know examples of conflicting requirements and how to deal with them
- Know about various types of RE contexts
  - types of project-settings, kinds of stakeholders, etc.
- Know the difference between ’user requirements’ and ’systems requirements’
- Know different ways of defining/specifying requirements
  - i.e., user stories and use cases
- Know the elements of a use case description

# Topic: Requirements Engineering I + II (cont'd)

- Know the difference between functional and non-functional requirements (aka 'quality requirements')
- Know the activities involved in the RE process (elicitation, analysis, specification, validation)
- Know the difficulties of requirements elicitation
- Know how to draw a use case diagram
- Know how to do effort estimation with user stories / with use case points
- Know the elements of UML class diagrams

# Topic: Analysis

- Know types of domain classes
  - business objects (e.g., order, receipt), real world objects (e.g., materials), actors/workers/persons (e.g., controller, customer), events (e.g., sale, payment)
- Know how to identify domain classes
  - E.g., by identifying nouns in domain descriptions
- Know how to identify attributes of domain classes
  - E.g., simple data types (like ’order number’)
- Know how to create a domain model
  - Outside-in approach (boundary, control, entity concepts)
  - Setting-up an enterprise approach (’workers’ vs. ’things’, ’doing’ vs. ’knowing’ concepts)

# Topic: Refactoring

- Know what ’refactoring’ is
- Know examples of ’code smells’ and examples of ’refactorings’
- Work through Martin Fowler’s introductory example on refactoring

# Topic: Verification and Validation I + II

- Know the difference between ’verification’ and ’validation’
- Know what the ’testing paradox’ is
- Know the difference between ’error’, ’fault’, ’failure’
- Know the terms ’test case’, ’test suite’, ’test oracle’, ’test verdict’
- Know the difference between ’white-box’ and ’black-box’ testing, and know why they are complementary
- Know how to apply the various ’black-box’ and ’white-box’ testing techniques
- Know how to construct a control-flow graph
- Know test coverage criteria (and how to use them)

# Topic: Verification and Validation I + II (cont'd)

- Know the elements of reviews and formal document inspections
- Know how to apply simple defect estimation models (i.e., simple capture-recapture models)
- Know how to test non-functional (quality) requirements, e.g., performance and usability (cf. Lecture 13 – measurement)

# Topic: Agile/Lean Methods

- Know the spirit of the ’agile manifesto’ and know how to explain/interpret it
- Know examples of agile software development methods
- Know elements of XP and Scrum (e.g., TDD, Planning Poker, Pair Programming)
- Know the difference between XP and Scrum
- Know the contents and purpose of a ’burndown chart’ (i.e., know how to read it)
- Know the difference between (pure) Scrum board and (pure) Kanban board
- Know the origin and the main goal of lean methods
- Know examples of ’waste’ in software development

# Topic: Measurement

- Know the definition of ’measurement’ and ’measure’
- Know the differences between measurement scale types and why it’s important on what scale type a measure is defined
- Know typical challenges of measurement in software engineering
- Know the measurement terminology
  - Qualitative – quantitative, direct – indirect, subjective – objective, reliability
- Know examples of measures in software engineering (i.e., measures of product, process, resource attributes)
- Know the difference between ’time’ and ’effort’

# Exam – ‘Open Book’

**MTAT.03.243 – Software Engineering Management**

Written Exam – 12 June 2013

Important Notes:

- The exam is open-book and open-laptop. Web browsing is allowed, but you are not allowed to use e-mail clients nor Instant Messaging clients, nor to share any information “live” with anybody inside or outside the exam room.
- At the end of the exam you must submit both the question sheets and your answer sheets. To avoid that any of your solutions get lost, make sure to write your name and student ID on each sheet of paper that you submit. Also, please number the pages on your answer sheets.
- Write clearly. Answers that are illegible cannot be counted as correct answers. Only answers written in English will be marked.
- Total marks: 30 (equivalent to 30% total grade). You must get at least 10 marks in this exam to not fail the course.

==========

# Exam Structure

- Part 1 (10 marks): Multiple Choice – 30 min
- Part 2 (10 marks): Open Questions – 45 min
- Part 3 (10 marks): Problems where a solution needs to be developed (no coding) – 75 min

- There may be up to 5 bonus marks (not yet decided)

# Exam – Part 1

Example (Exactly one correct answer):

- You execute a program and you see the following message on the screen: ’Program aborted due to division by 0’. Which of the following terms describes your experience best:
    A. I made an error
    B. I detected an error
    C. I detected a fault
    D. I triggered a failure

# Exam – Part 2

Examples:

- Briefly describe the three elements of a user story
- Explain the difference between ’extend’ and ’include’ relationships between use cases.
- Explain why it is important to distinguish between ’error’, ’fault, and ’failure’

# Exam – Part 3

Example:

- Two persons independently review the same piece of code. Person A detects 6 defects. Person B detects 8 defects. 2 of the defects found by B are also in the set of the 6 defects found by A. Estimate how many defects are still in the code after all defects found by A and B have been corrected. (Assume that no new defects are introduced when corrections are made.)

# Exam – Part 3 (cont'd)

Example:

- Describe as detailed as possible how you would test the following performance requirement:
    - In standard usage, CPU usage shall be less than 50%, leaving 50% for background jobs.

# Next Lecture

- Date/Time:
    - No more lectures

- For you to do:
    - Submit Lab Task 7 in time!
    - Remember that all team members must attend the lab assessment session in week 16

Thanks for coming to the lectures!

**All the best for the exam!**