Principles of Software Design
Introduction

Robert Lukoťka
lukotka@dcs.fmph.uniba.sk
www.dcs.fmph.uniba.sk/~lukotka

M-255

What is the difference between real world project and your typical class assignment?

Class assignment.

- The underlying reason on why to do the assignment is not your concern.
- The assignment is clearly stated and complete.
- There are no changes about your assignment during your work.
- You work on your own.
- You work on it for short amount of time, then you forget about.
- The resulting product is small, easy to comprehend by a single person.
- ...

What we will do here

The aim is to give you a basic idea about software development.

1. We discuss various areas of software development.
2. How to actually write code under typical real life circumstances.
3. Basics of several useful technologies.

Areas of knowledge [1]

- Software requirements
- Software design
- Software construction
- Software testing
- Software maintenance
- Software configuration management
- Software engineering management
- Software engineering process
- Software engineering models and methods
- Software quality
- Software engineering professional practice
- Software engineering economics

Areas of knowledge [1]

- Computing foundations
- Mathematical foundations
- Engineering foundations

Supporting:

- Computer engineering
- Systems engineering
- Project management
- Quality management
- General management
- Computer science
- Mathematics

Areas of knowledge

- The previous courses only touched some of these areas.
- We give a very basic overview of most of these areas.

What to expect an what should you focus on

- No theorem or proofs.
- Various approaches how to do something.
- Strong and weak sides of each approach, when given approach is appropriate.
- Good practices within chosen approach:
  - amplify the strong sides of the approach
  - manage the weak side
- How given practice affects other areas of software development.

How to program

You will encounter many programming languages in this course.

- Most examples will be mostly in Python.
- Assignments will be in Java |(this changes)

A small introduction to Python as part of this course.

How to program

You learnt how to program. But to make complex software you need reasonable design. You need to split yout problem into small tasks.

- In such a way that each task is easy to handle.
- In such a way that they do not affect each other too much.
- In such a way that each task may be tested well.
- In such a way that that we can get rapid feedback.

Small is way smaller then I typically see in student projects.

Technologies

Several technologies:

- Git
- Makefile
- Unit testing frameworks
- ...

Grading

- Two evaluated assignments, 2×25 points. Tentative deadlines:

Assignment 1	22.10., 29.10.
Assignment 2	19.11., 26.11.

- Oral examination 50 points (three open questions from a published set of questions).

Grading

- You have to use Git and Java to complete your assignments.
- You will have to read manuals and search for information to complete the assignments
- After your are done you send entire repository to me and one of your colleagues (first deadline)
- During the next week your colleague reviews the code and you improve it accordingly.
- You send me the whole repository again (second deadline)
- Send the assignments to lukotka.pts@gmail.com. Use this e-mail address just to sent the assignments and maybe for some immediate discussion after I send you my feedback.

Grading

- Grading:

$$\begin{array}{ll} A & \langle 100, 90\rangle \\ B & (90, 80\rangle \\ C & (80, 70\rangle \\ D & (70, 60\rangle \\ E & (60, 50\rangle \end{array}$$

Resources

On this frame you usually find materials to learn from. You should read this stuff to prepare for the examination (especially if you missed the course or were sleeping during my presentation :)).

References I

Sometimes I feel it is necessary to give you the source of some claim (I should do that more often then I do). This will appear here. You don't have to read the content of it.

 SWEBOK V3 -TOC