Inf2: Software Engineering and Professional Practice
Lecture 1: Overview

Cristina Adriana Alexandru

School of Informatics
University of Edinburgh

Why do this course?

Because software engineering is fascinating

a blend of human and technical challenges

fast-moving

important

It could help you get a job!

For many of you, it's one of the most job-relevant courses you'll take in your Informatics degree.

Course aims

The Inf2-SEPP course has several important aims:

- Give you an overview of what software engineering is
- Show you industry standard tools and techniques of the trade
- Discuss different processes for developing software
- Raise awareness of professional considerations surrounding software engineering
- Allow you to practice engineering software hands-on
- Encourage you to interact with technical documentation
- Develop your planning, organization, reflection, reviewing and teamwork skills
- Show you the reality of being a software engineer

Overall, take you **beyond programming to engineering** software

Learning outcomes

On completion of this course, you will be able to:

1. Explain the modern techniques used in the design and development of large-scale software systems
2. Apply, evaluate and reflect on these techniques in a small-scale, but realistic scenario
3. Analyse the professional and ethical implications of software engineering decisions and propose solutions
4. Comfortably read and write technical documentation
5. Constructively engage in interaction with peers

Components of the course

Overall, there are 2 parts to this course:

- The software engineering (SE) part- focused on activities, processes, tools and techniques for engineering software.
- The professional practice (ProP) part- focused on larger professional considerations surrounding software engineering

They go hand-in-hand, with ProP meant to complement SE.

Meet the lecturers!

Dr Cristina Adriana Alexandru

Course organiser

Lecturer on Software Engineering part

Dr Michael Glienecke

Lecturer on Professional Practice part

Lectures

- **All Tuesdays, Thursdays and Fridays @ 14:10-15:00**
  - 2 of them on SE (Cristina)
  - One on ProP (Michael) **OR**
  - One guest lecture from experienced industry profesionals
- Most relevant for the coursework
- Essential/recommended resources provided on slides and online at the start of each week
- Self-study of course topics and Java essential!

Unassessed Wooclap quizzes used during SE lectures to check understanding and engagement

Tutorials

- In **Week 2 for ProP**
- In **Weeks 3, 5, 6, 8, 10 for SE**
- Task sheets provided the week before, solutions provided Friday of the week
- Opportunity to practice concepts needed for the coursework
- Preparation before tutorials is desirable and expected by tutors
- Attendance will be taken

Labs

- **Tuesdays, Thursdays, Fridays @16:00-17:30 starting wk 2**
- Drop in as often as you like
- Focused on working on coursework (SE and ProP) and asking demonstrators questions about it
- Coming prepared with questions is desirable
- Other activities in labs:
  - Lab 1 and first lab for each new CW: on team development
  - During CW1 and CW2: opportunity to interview demonstrators
  - During CW3: labs also used for code review

Q&A

- Run:
  - **During Tuesday lab on SE part (Cristina)**
  - **During Friday lab on ProP part (Michael)**
- Expectation for you to formulate questions

Separate Q&A sessions can be organised if motivated by amount of questions in labs and Piazza

Formative assessment

- ProP formative coursework, starting in week 2
  - Individual work
  - Writing an argumentative essay on professional issues
  - Receiving feedback from the markers

Summative assessment (marks that count!)

1) A software development project worth 100%, in 3 parts:

- CW1: Requirements engineering (SE part 1, worth 15%)
- CW2: Design (SE part 2, worth 22%)
- CW3: Construction, testing (63%: SE part 3 38%, ProP 25%)

SE parts done in your 3-4 student teams; practical work, justification, reflection/self-assessment.

ProP part individual, consisting of the writing of an essay on professional issues surrounding the project.

Marking a mix of additive and criteria-based, with high-level marking scheme provided in advance.

2) 2 bonus marks if answering to 4 teamwork questionnaires

Teamwork and teamwork research

Teamwork skills very sought after by the IT industry

- We aim to help you develop them through:
  - Realistic situation: quasi-randomly assigned team for SE parts
  - Written guide with resources, guest lecture on teamwork
  - Team development activities in some of the labs
  - Realistic amount of support for team difficulties
  - Requirement to reflect on teamwork in CW1-3
  - For marking fairness: declaration of % of work contributed by each team member in CW1-3, and marks adjusted accordingly
- Online questionnaires will be run in week 2 and after CW1-3 deadlines to evaluate above approach (bonus awarded); visible only by course organiser, answers do not influence other marks
- Optional possibility to allow for questionnaire answers to be used for research

Sources of support

Labs, tutorials

Piazza online discussion forum, handled Mondays, Wednesdays, Fridays @ 16:00-17:00. There is a confidential channel for lecturers.

Email (Cristina.Alexandru@ed.ac.uk for SE questions, Michael.Glienecke@ed.ac.uk for ProP questions)

Resources for SE part

Essential books (all provided for free electronically by library):

- Sommerville, *Software Engineering* (10th Ed)
  - Large, classic. Comprehensive on SE, but limited on UML and Java.
- Sommerville, *Engineering Software Products: An Introduction to Modern Software Engineering*
  - Excellent on agile processes, product engineering, modern architectures
- Stevens with Pooley, *Using UML* (2nd Ed)
  - Covers basic SE, does UML thoroughly.

Other resources will be provided as external links, or recommended

Getting started in the course

- Check the course webpage on Learn for general course information, later assessment instructions and marks
- Check the opencourse webpage for course schedule, materials, readings, resources, tutorial solutions
- Go onto the Piazza forum for the course (linked from both websites), start interacting
- Check your allocated tutorial slot and fill in the time change form if its time does not work for you
- Check who your coursework teammates are and get in touch with them
- Start reading

Why is SE still hard?

Easy (or at least routine) projects

small systems (up to c. 100k LOC),

- without hard timescales or budgets,
- without requirement for very high reliability,
- without complex interfaces or legacy requirements [...]

Hard projects

everything else. Projects with *all* the above challenges, and more.

Statistics

The Standish Group produce annual CHAOS reports on software development projects of medium-large organisations.

Surveying on average 50k projects each year.

Projects categorised 3 ways:

- Succeeded
  - on time, on budget, with a satisfactory result
- Challenged
  - delivered something but maybe late, over budget, not satisfactory to the customer or user
- Failed
  - cancelled without delivering anything

Standish CHAOS trends

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

Taken from `https://standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf`

2011-2015 latest report freely available, but similar in references to 2020 report: 31% successful, 50% challenged, 19% failed.

How are we going to approach the software engineering (SE) topic?

1. After brief intro to SE, delve into overview of main software development process types (plan-driven, agile)
2. Take each SE activity in plan-driven vs. agile processes
3. Go into some UML modelling
4. Look in detail at some examples of software deveopment processes (e.g. Waterfall, Spiral for plan-driven; XP, Scrum, Kanban for agile)
5. Finish with a detailed consideration of quality attributes, influencing the whole process

Reading

Recommended: Google CHAOS Standish reports, find e.g.

- Standish Group: ENDLESS MODERNIZATION: How Infinite Flow Keeps Software Fresh
- Standish Group: CHAOS Report 2015