(11224) INTRODUCTION TO SOFTWARE ENGINEERING

Lecture 1: Introduction to the Module

Shereen Fouad

Instructor

- Shereen Fouad
  - Teaching Fellow
  - PhD in Computer Science, The University of Birmingham.
- Room: 227
- Email: Fouadsa@cs.bham.ac.uk
- Office Hours: Mondays (2pm- 3pm)

Lectures

- 10 weeks lectures
- Every Monday from 11am to 12pm in LTA, Watson.
- Every Tuesday from 1pm to 2pm in WG5, Aston Webb.

Grading

The final module mark for the module will be calculated as follows:

- 20% of continuous assessment mark

plus

- 80% of examination mark

Continuous Assessments

- Continuous assessments for this modules will take the form of 1 submitted exercise and 2 online tests.
- **The submitted exercise (10% of the module mark)**
  - will be marked out of 50
  - designing a UML class diagram
  - groups of 3 or 4 (minimum 3 and maximum 4)
  - announced in week 4 of the semester
  - submission deadline is in week 6.

Continuous Assessments

- **The online test (each is 5% of the module mark)**
  - Marked out of 25
  - Online test on canvas
  - Duration is one hour
  - First online test is in week 7
  - Second online test is in week 9

Final Exam

- Closed-book exam.
- Duration is one hour and a half.
- Class notes will not be allowed during the exam.
- Previous years exam/solution will be released on canvas soon.

Other Module Info

- Home Pages:
  - https://canvas.bham.ac.uk/courses/9778/pages/06-11224-introduction-to-software-engineering
- Lecture notes will be available on canvas before the lecture time
- Weekly handouts will be available on canvas at the beginning of each week
- Weekly handouts will include concepts covered in the two weekly presented lectures

Components of Programming

- In the **Software Workshop module**, you are learning the Java programming language and practicing your skills in it by writing programs.
- In the **Foundations of Computer Science module**, you are learning, among other things, a collection of data structures and algorithms.
- **Software Engineering** is a further major component of programming that should be considered part of the core knowledge and skill set of a competent programmer.

What is software?

- Computer programs, libraries and their associated documentation such as requirements, design models and user manuals.
- Software products may be developed for a particular customer or may be developed for a general market.
- New software can be created by developing new programs, configuring generic software systems or reusing existing software.
- More and more systems are software controlled
- Large scale software tends to be very complex.
- Complex means that it is composed of many simple parts related to one another

Attributes of Good Software

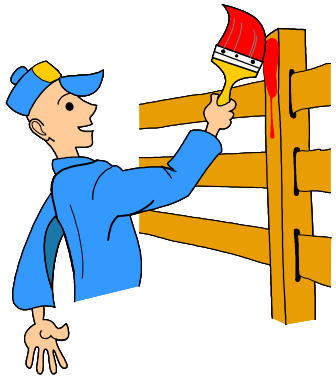
- People working together to create a robust software system that satisfies the client.
  - a high quality software (I.e., maintainable, dependable and acceptable) system
  - with a given budget
  - before a given deadline

What is software engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.

Dodgy Analogy

- To successfully build a satisfactory building that is fit for purpose team of builders, should have suitable skills with the raw materials used, namely bricklaying, joinery, plastering, plumbing, electrical wiring etc.
- These building skills correspond to the basic programming language skills of a programmer.



SOURCE: Goodaire & Parmenter, Discrete Mathematics with Graph Theory, Third Edition, Pearson Prentice Hall, 2006. [Section 11.5, p. 361]

Dodgy Analogy

- Further, one would not expect builders to produce components such as doors, windows, fuse boxes, shower units and boilers from raw materials.
- Instead we would expect them to have the knowledge and skills necessary to identify the items of this nature appropriate to the building and order and install them.
- This corresponds to the programmer's knowledge and skills with data structures and algorithms.

Dodgy Analogy

- We might even be happy for them to build a simple house for us — if they could show that they had experience in building houses of that style.
- The correspondence in this case would be to implementation of simple systems.

Dodgy Analogy

- What if we want them to design a new house of a novel style for us, or a multi-story apartment block, or a skyscraper.

- *Architectural skills to do the design.*
- *Engineering skills to ensure that structural elements are safe, that the design works, and to translate the architects' designs to the detailed plans and instructions necessary for the builders.*
- *Management skills to ensure that the large number of people and resources involved in the project are properly coordinated to minimise cost, ensure safety and provide a satisfactory result on time.*

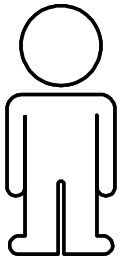
Software Engineering

- The closest corresponding collective term for all these skills is Software Engineering.

Software Engineering

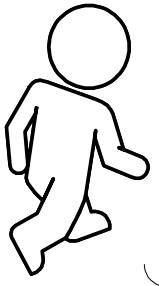
- Software engineering comprises many techniques that have proved effective, practical and useful, and have made huge steps in pushing back the boundaries of the complexity and sophistication of the programming tasks that we have been able to master because of them.

The Role of Software Engineering

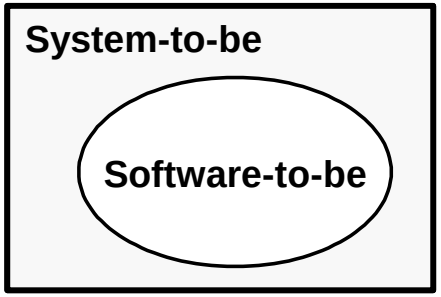


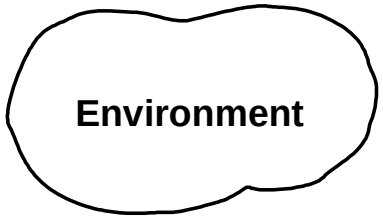
Customer:

Requires a computer system to *achieve some business goals* by user interaction or interaction with the environment in a specified manner



User



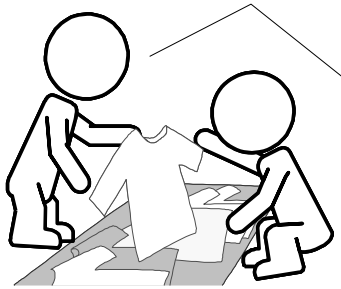


Environment

Software Engineer's task:

To ***understand how*** the system-to-be needs to interact with the user or the environment so that customer's requirement is met and ***design*** the software-to-be

May be the same person



Programmer's task:

To ***implement*** the software-to-be designed by the software engineer

What is the difference between software engineering and computer science?

- Computer science is concerned with computer science theory and fundamentals;
- Software engineering is concerned with the practicalities of developing/designing and delivering useful software.

Importance of Software Engineering

- Recently, individuals and society rely heavily on advanced/complex software systems.
- We aim to produce reliable and efficient systems economically and quickly.
- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project.
- For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

This Module

- Consider issues of software engineering that apply mostly to small programs
- Address some of the techniques and approaches that can help to lift your programming skills to a more disciplined, and better engineered, level than you have yet achieved.
- Discuss how to deal with some design issues when tackling a programming problem, how to document, describe, explain and discuss a program with colleagues, supervisors, and subordinates.
- Address managing code development, debugging, testing and deployment of your programs.

Essential Contents in this Module

- Software Process Models
- Software Testing
- Software Design - UML Class Diagrams
- Structured Design
- Use CASES
- Agile Process Models
- State Machine Diagrams
- Sequence Diagrams
- Refactoring

Learning Outcomes

- On successful completion of this module you will be able to:
  - demonstrate facility with basic strategies of program design
  - demonstrate facility with recording and communicating program designs
  - understand and be able to apply software engineering approaches in the small

A major problem in learning this topic

- While many of the techniques involved can help even when writing relatively small programs, they often only become necessary, and demonstrate their true value, when tackling large programs — typically larger than is feasible to undertake until at least your final year project.

References

- A number of slides in this talk is based on:
  - Alan P. Sexton hand-outs (Introduction to Software Engineering. The University of Birmingham. Spring Semester 2014)
  - SOFTWARE ENGINEERING 9 Ed. by Ian Sommerville
  - Goodaire & Parmenter, Discrete Mathematics with Graph Theory, Third Edition, Pearson Prentice Hall, 2006. [Section 11.5, p. 361]

Thank You ☺