COMP201

Software Engineering I

Lecture 1 – Welcome to the Module

Lecturer: **T. Carroll**
Ashton Building, Room G.14
E-mail: ***Thomas.Carroll2@liverpool.ac.uk***

See Vital for all notes

Module Information

Module Delivery

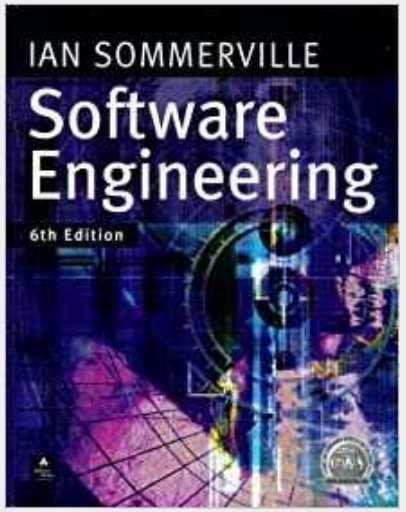
- **3 lectures per week**
  - Monday (check your timetable)
  - Wednesday 10:00-11:00 SHER-LT2
  - Thursday 17:00-18:00 CHAD-ROTBLATT
    - All lecture slides will be on Vital
    - Recordings of lectures on Vital soon after the lecture
- **1 hour lab session per week**
  - Use to do coursework
  - **LABS START IN WEEK 3**
- **Independent reading and study**

Assessment

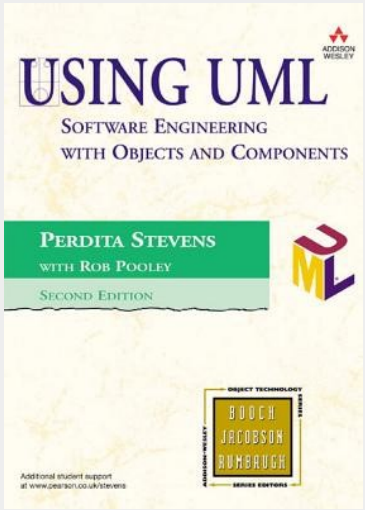
- **60% Exam**
  - 2 hours long
  - January 2020 exam season
- **40% Coursework**
  - 2 assignments, both worth 20%
  - First assignment given out in Week 3
  - Assignments will be distributed on Vital

Reading List – All available in Library as real book and e-book

- **I. Sommerville (2001,2004, 2007)**
  - Software Engineering 6th ,7th or 8th Edition,
  - Addison-Wesley, Harlow, Essex, UK



- **P. Stevens with R. Pooley (2000)**
  - Using UML: Software Engineering with Objects and Components, 1st or 2nd Edition,
  - Addison-Wesley, Harlow, Essex, UK



Syllabus Outline

- Introduction to Software Engineering
- Software models
- Software requirements
- Formal Specification
- Software Design and Implementation
- UML (Unified Modelling Language)
- Software verification, validation and testing
- Management of Software Projects & Cost Estimation

What is Software Engineering?

What is Software Engineering?

- **ENGINEERING**
  - Making stuff
  - In a structured and disciplined manner!
  - Using tried and tested approaches

- **SOFTWARE**
  - Code (instructions)
  - Data designs (data base schemas)

Where It All Began...

- Software Engineering was first introduced in 1968 during a conference about the “software crisis”
- Early approaches were based on informal methodologies leading to:
- Delays in software delivery
- Higher costs than initially estimated
- Unreliable, difficult to maintain software
- Need for new methods and techniques to manage the production of complex software.

Why Software Engineering?

- **Software development is hard !**
- **Important to distinguish :**
  - “easy” systems (one developer, one user, experimental use only)
  - “hard” systems (multiple developers, multiple users, products)
- **Experience with “easy” systems is misleading**
- **Single person techniques do not scale up**

Analogy with Bridge Building

- Plank over stream is a relatively easy, single person job
- Bridge over the Mersey…. Maybe not as simple!

- Techniques used for the smaller project do not scale!

Why Software Engineering ?

- **The problem is complexity**
- **Many sources of complexity, but size is key:**
  - The Linux kernel v.3.10 contains >15 million lines of code
  - Windows XP contained >40 million lines of code
- **Software engineering is about managing this complexity.**

Why Software Engineering ?

- **Software failure can be very serious**
  - Software controls safety critical systems
  - Software protects sensitive data
  - Software is involved in systems which handle money
- **Software Engineering has to:**
  - Produce software which has a very low chance of faulting
  - Be able to demonstrate/proof that software has very low chance of fault
    - Testing or program proving

Software Engineering Tasks

- **Define requirements**
  - What should it do?
- **Design the product**
  - Design how the product should look and be constructed
  - UI design, software module design, data design (what data?)
- **Implement and test**
  - Coding, testing and validation
- **Managing the process**
  - Software project management

Software Engineering

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development.
- Some software can be classified as critical (air traffic control, medical software, nuclear reactor control software..).
-

Software Engineering

- Software costs often dominate computer system costs. (The costs of software on a PC are often greater than the hardware costs.)
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.
- Software engineering is concerned with cost-effective software development.
- Critical Systems must be verifiably reliable to avoid significant environmental, human or financial costs.

Software Engineering FAQs

- What is software?
- Is Software Engineering just like Computer Science?
- What are the attributes of good software?
- What does a good Software Engineer do?

In groups,
discuss these questions
and let's see what you think

What is Software?

- Computer programs (apps, embedded systems, etc...) and associated documentation (user manuals, designs, etc...)
- Software products may be developed for a particular customer or may be developed for a general market
- Software products may be
  - ***Generic*** - developed to be sold to a range of different customers
  - ***Bespoke*** (custom) - developed for a single customer according to their specification

Is Software Engineering Just Like Computer Science?

- Software engineering is an engineering discipline which is concerned with all aspects of software production
- Computer Science is concerned with theory and fundamentals
- Software Engineering is concerned with developing and delivering good software
- CS gives a foundation for the practical aspects of SE

What Are The Attributes of Good Software?

- **The delivered product should provide the required functionality, and have the following characteristics:**
  - Maintainability
    - Software must be easily evolvable to meet changing needs
  - Depndability
    - Software must be trustworthy (work with all data)
  - Efficiency
    - Software should not make wasteful use of system resources
  - Usability
    - Software must be usable by the users for which it was designed

What Does A Good Software Engineer Do?

- Predicts their own productivity
- Works well in teams
- Documents their work
- Adopt a systematic and organised approach to their work
- Use appropriate tools and techniques depending on:
  - the problem to be solved,
  - development constraints
  - resources available
- Produce code which can be fixed/modified easily by others

What Is A Software Process?

- **A Software Process is a set of activities whose goal is the development or evolution of software**
- **Fundamental activities in all software processes are:**
  - Specification - what the system should do and its development constraints
  - Development - production of the software system (design and implementation)
  - Validation - checking that the software is what the customer wants
  - Evolution - changing the software in response to changing demands

What Is A Software Process Model?

- A simplified representation of a software process, from a particular perspective
- **Examples of software process models:**
  - Waterfall
  - Evolutionary development
  - Formal transformation
  - Integration from reusable components
- **Examples of perspectives:**
  - Work-flow
  - Data-flow
  - Role/action perspective

The Costs of Software Engineering

- $150bn worldwide spend on SE
- Costs depend on the system requirements
- Approx 60% development costs, 40% testing costs
- Cost ratio can depend on the model used

Computer Aided Software Engineering (CASE)

- **Systems which are intended to provide automated support for software process activities, such as:**
  - Requirements analysis
  - System modelling
  - Debugging and testing
- **Upper CASE supports early process requirements and design**
- **Lower CASE supports later activities, such as programming, debugging and testing**

Challenges of Modern Software Engineering

Software engineering in 21st century faces 4 main challenges:

- **Legacy systems**
  - Old, valuable systems must be maintained and updated
- **Heterogeneity**
  - Systems are distributed and include a mix of hardware and software
- **Delivery**
  - There is increasing pressure for faster delivery of software
- **Trust**
  - Developing techniques that demonstrate that software can be trusted by its users

When Things Go Wrong...

Major Software Failures

- **Therac-25 (1985-1987)**
  - six people overexposed during treatments for cancer
- **Taurus (1993)**
  - the planned automatic transaction settlement system for London Stock Exchange cancelled after five years of development
- **Ariane 5 (1996)**
  - rocket exploded soon after its launch due error conversion (16 floating point into 16-bit integer leading to an exception)
- **The Mars Climate Orbiter**
  - assumed to be lost by NASA officials (1999): different measurement systems (Imperial and metric)

Recent Failures

- **Healthcare.gov (US government project)**
  - Coding was started before requirements completed in detail
  - You could actually see errors in code by merely looking at Javascript…
  - Testing was squeezed to get site delivered in time
  - There was no manual backup on launch
- **TSB bank (April 2018)**
  - Users locked out of accounts
  - Users logged in could see others user's account details

Important Progress Has Been Made

- Ability to produce more complex software has increased
- New technologies have led to new SE approaches
- A better understanding of the activities involved in software development
- Effective methods to specify, design and implement software have been developed
- New notations and tools have been produced

Ethics and Professionalism in Software Engineering

Professional and Ethical Behaviour

- 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.

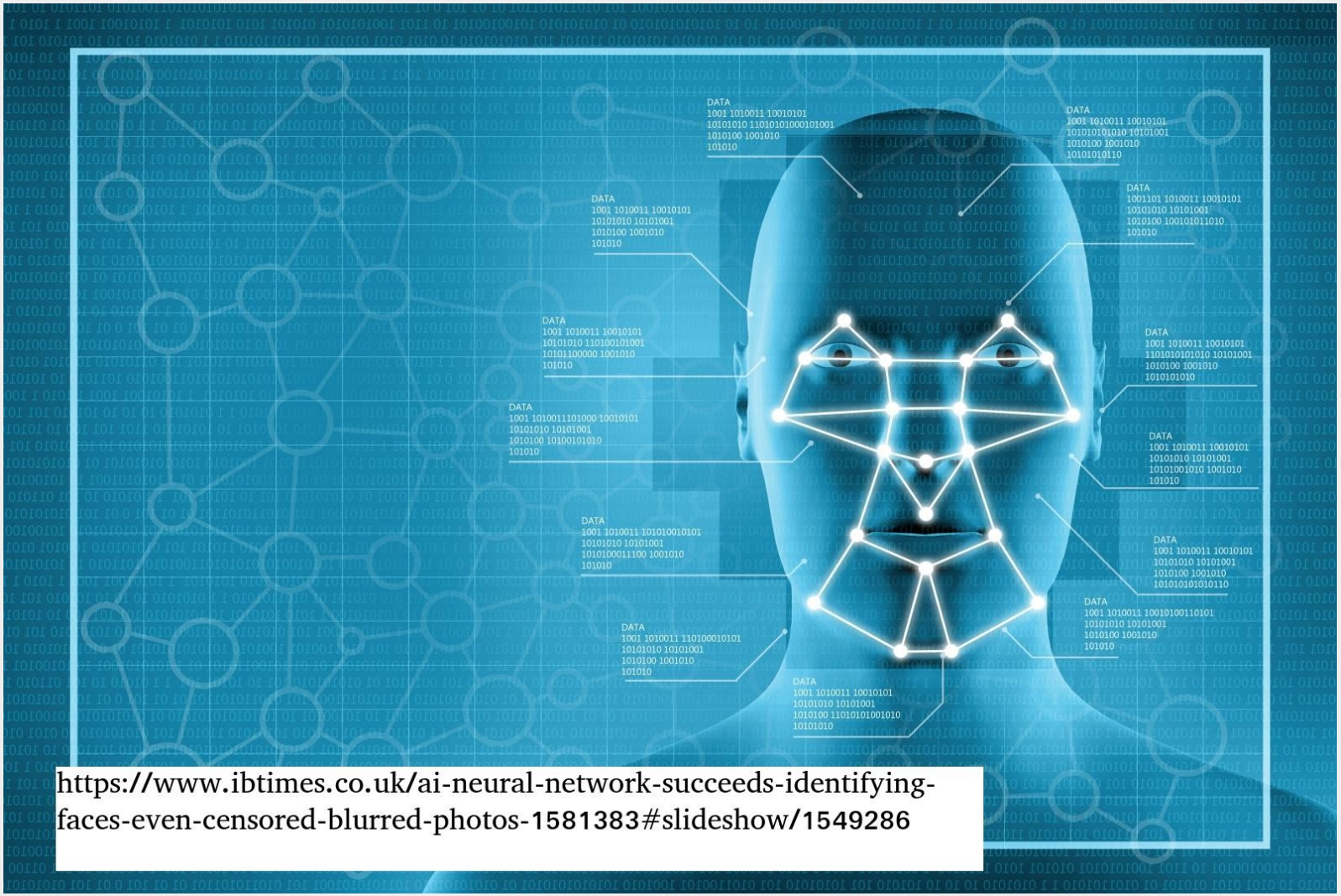
Professional Responsibility

- **Confidentiality**
  - Engineers should respect the confidentiality of their employers or clients without a formal confidentiality agreement.
- **Competence**
  - Engineers should not misrepresent their level of competence.
  - They should not knowingly accept work which is beyond their competence.

Professional Responsibility

- **Intellectual property rights**
  - Engineers should be careful to ensure that the intellectual property of employers and clients is protected and know the local laws governing IP.
- **Computer misuse**
  - Software engineers should not use their technical skills to misuse other people's computers
- **GDPR**
  - Engineers should create their products in such a way that respects users privacy, respects the laws on data protection, and that protects data in a way that makes it difficult to access for attackers.

What If…?


https://www.ibtimes.co.uk/ai-neural-network-succeeds-identifying-faces-even-censored-blurred-photos-1581383#slideshow/1549286

What if….?

- **A robot, that you designed AI for, hurts someone.**
- **Who is responsible?**
  - You?
  - The owner of the robot?
  - The robot?
- **What if it identified the wrong person?**
- **What if a robot you design and produce, designs and produces another robot?**

Recap

Recap

- SE is an engineering discipline concerned with all aspects of software development
- Software products consists of both the actual program and the associated documentation
- We have seen reasons for requiring solid SE principles
- Software engineers must act in an ethical and professional manner at all times