CS-230 Software Engineering

L01: Module Introduction and the Software Crisis

Dr. Liam O'Reilly

Semester 1 – 2020

Self Introduction

- *Me:* Hi, my name is Liam.
- *Class:* Hi Liam.
- I have a computer science problem.
  - Many of them actually.
  - Fortunately, most of them are for you.
  - Bwhahahahahah!
- I am teaching this class for the ~~first~~ ~~second~~ third time.
  - So its fortunate most of the problems are for you :-)
  - Hopefully you will get a great deal out of this module.
- A Big Thank You to Daniel Archambault, Rita Borgo, and Robert Laramee.

A Bit More About Me

- I'm Welsh.
- I did my CS BSc at Swansea – 2003-2006.
- Then I did a MPhil at Swansea – 2006-2007.
- Then I did a PhD at Swansea – 2008-2012.
- Then I ~~escaped and~~ worked at Garmin in Southampton for 3 months as a Software Engineer.
- Since 2013 I have been lecturing at Swansea.

The Research Stuff That I Do

- I mainly do teaching.
- I do a little research in Formal Methods and CS Pedagogy.
- Specifically, specifications, verification and validation of systems.

This Module

- CS-230 – Software Engineering.
- 15 Credits
- Mixture of lectures, interactive sessions, and help sessions.
  - Monday, 13:00, Zoom.
  - Wednesday, 11:00, Zoom.
  - Thursday, 10:00, Zoom.
- Please see `https://science.swansea.ac.uk` for up to date timetable information.

Office Hours and Contact

- My Office: Computational Foundry 220.
- Office Hours: Check the Collage Intranet: `https://science.swansea.ac.uk`
- For course questions, please use the forum on Canvas. Please subscribe to the forum.
- Only email me with questions or content which you think should not go on the forum. E.g.,
  - Do not post answers on the forum.
  - Do not post personal content on the forum.

Syllabus and Module Organisation

- In **CS-230** (also known as this course):
  - Review of software process and software life-cycle models.
  - Requirements engineering.
  - Software design and prototyping.
  - Software implementation.
  - Some software testing.
- Full course schedule (with dates) on Canvas.

- Software Engineering 9, by I. Sommerville, Pearson, 2010.

Syllabus and Module Organisation (2)

- Material is continued in a second module next term.
- In **CS-235** (*Software Engineering Students Only*):
  - More software engineering.
  - Further software implementation.
  - Software testing.
- Assignments in CS-230 and CS-235 are connected.
- Group work continues with the same groups.

Assessment

- **Lots of group coursework!**
- In **CS-230** (also known as this course):
  - A1 (Group Work): object-oriented (OO) design [30%]
  - A2 (Group Work): object-oriented implementation [45%]
  - Exam: Multiple-choice [25%]
- CS-230 finishes here, but **Project to be continued...** in CS-235 (for Software Engineering students).

Comments on Studying

1. Be prepared to take additional notes!
    - I tend to organise a course around lectures.
    - and use texts as support and alternate explanations.
2. Read about the subject: `http://www.freetechbooks.com/`
3. Remember: 15 credits represents roughly 150 hours of study.
4. Most of these hours spent on the assignments.

Attendance is Required

- Attendance is required.
- Material is extremely important for assignments.
- Material is needed on a timely basis for the assignments.
- Attendance facilitates (spontaneous) group meetings/group work.
- Better attendance usually means better marks.
- Exam will test content of module.

Class Ground Rules

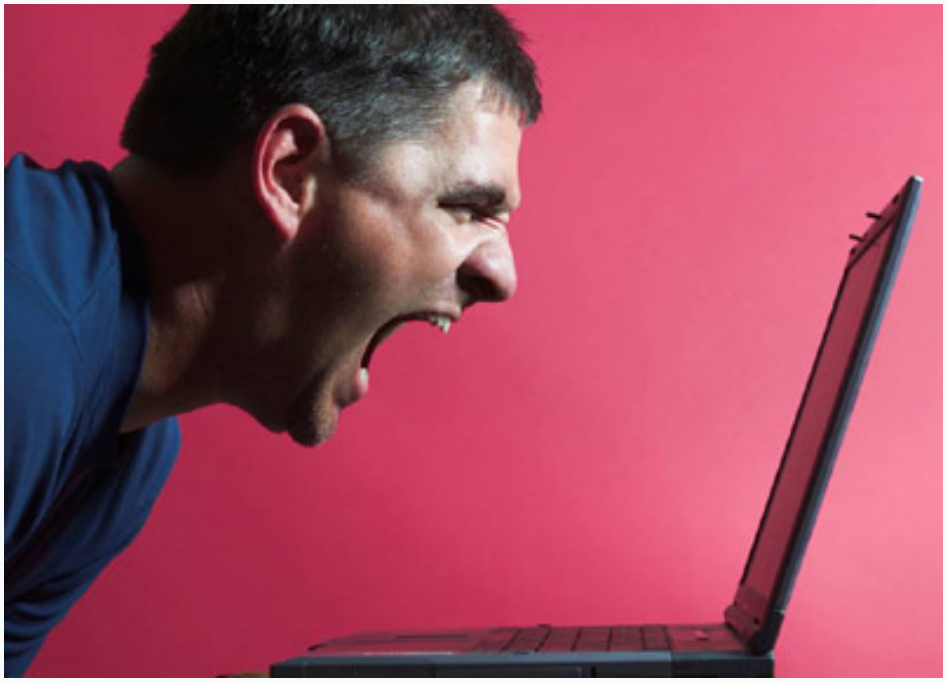
1. Treat everyone in the class with respect:
    - That includes your fellow students.
    - That includes me.
2. There is no such thing as a stupid question.
    - Never be afraid to ask a question.
3. Give your best effort on every assignment.
    - The worst thing you can do, in my mind, is not submit
    - or demonstrate very little effort.
4. Be positive and have fun!

Cheating

- It sucks.
- Don't do it.
- Not even unintentionally.
  - For individual assignments:
    - Do collaborate on white board or laptop.
    - Before end of session, erase all recorded data from the session.
  - For group assignments:
    - Talk to other groups about the problem but...
    - Before end of session, erase all recorded data from the session.
- Don't exchange code or copy text from each other.
- Obviously, no copying of code from the Internet...

Software in Crisis!

Engineering Long Lasting Software



Who Likes....

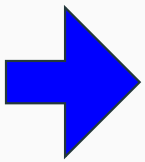
??

This is Facebook

- Lots of users all over the world.
- Pretty complex piece of software, eh?

Back in CS-110 and CS-115...

- You learned how to write Java code.
- `.java` files, containing your program compile to `.class`
- Ok, let's now implement Facebook in Java...



1 file, Gazillion Lines of Code

In: `public static void main ()`

- Good way to implement Facebook?
- Probably not...

What is Software Engineering?

"Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use." (Sommerville, 2012)

- Intuitively - How do we handle large software projects?
- Engineering discipline:
  - Using appropriate theories and methods to solve problems.
  - Bearing in mind both organisational and financial constraints.
  - Includes all aspects of software development including project management.

What is Software Engineering? (2)

Software Engineering involves teams. Large, useful software applications generally require a team.

- What if you want to build a dog house (kennel)?
  - Just yourself, some wood and some nails. Maybe some paint.
- What if you want to build a house?
  - Architects, carpenters, electricians, plumbers, designers, construction workers etc.
- What if you want to build a larger work building?
  - Very extensive requirements specification and design (planning!) phases, costs estimates, safety requirements etc.
- What if you want to build an operating system?
  - Different... but similar in some ways.

What is Software Engineering? (3)

Software Engineering is the construction and management of large software projects, i.e. software project management.

- Until now you have experienced small programs, analogous to a post-it note.
- How do you organise many post-it notes?
- How do you organise a book?
  - title, sentences, paragraphs, chapters, ToC, index, publishers, formatting, printing, copyright, etc.
- How do you organise a shelf of books?
- How do you organise a library with millions of books?
  - Think about searching.
- How do you organise every line of code in MS Word?

Ariane 5 – Arithmetic Overflow Condition

- **Show Video**
- European Space Agency – 10 years and 7 billion USD to build.
- So... what happened? Software failure...
- Working code for Ariane 4 rocket is reused in the Ariane 5.
- But Ariane 5 is more powerful – faster!
- Domino Effect:
  - Guidance system tried to convert the sideways velocity of the rocket from a 64-bit format to a 16-bit format - overflow error.
  - Guidance system shut down, it passed control to an identical, redundant unit which then crashed in the same way.
  - The data (error messages) from the crashed guidance system looked like flight data – bizarre and impossible flight data.
  - Rocket tried to compensate for a wrong turn that never occurred. It made an abrupt course change.
  - Self-destruction was triggered automatically due to boosters being ripped from the rocket.

Software vs Hardware Engineering

- Engineering software is different from hardware.
- **Q**: Why so many SW disasters and few HW disasters?
  - Ariane 5 rocket explosion.
  - Therac-25 lethal radiation overdose.
  - Mars Climate Orbiter disintegration.
  - FBI Virtual Case File project abandonment.
- Wired Magazine, History's Worst Software Bugs, 2005, `http://www.wired.com/software/coolapps/news/2005/11/69355`

Independent Products vs. Continuous Improvement

- An advantage/disadvantage of software is its mutability
- Consider the cost of an upgrade...
- Hardware $\approx \infty$
  - HW designs must be finished before manufactured and shipped.
  - Bugs in HW = Recall of HW (substantial loss of revenue).
- Software ≈ 0
  - We expect SW to improve over time.
  - Bugs in SW = wait for an upgrade (little loss in revenue).

How do we go About Engineering Software?

- *Successful software applications require a plan!*
- All useful approaches involve some elements of the following:
  - Software Specification, (requirements, customer-oriented planning).
  - Software Design (developer-oriented planning, organisation).
  - Software Development (implementation).
  - Software Validation (testing) .
    - Verification is starting to be more commonly used for critical systems.
  - Documentation.
  - Software Maintenance/Evolution.
    - Software is modified to reflect changing customer and market requirements.

The Software Crisis

- A term used to describe...
  - Impact of rapid increases in computer power.
  - Increase in complexity of problems which can be tackled.
- Refers to difficulty of writing correct, understandable, and verifiable computer programs.

"The major cause of the software crisis is that machines have become several orders of magnitude more powerful! As long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem."

– Edsger Dijkstra, The Humble Programmer (EWD340), Communications of the ACM, 1972

The Software Crisis (2)

- Various methodologies to "tame" software crisis, with varying degrees of success.
- However, it is widely agreed that there is no "silver bullet".
  - i.e., no single approach prevents project overruns and failures in all cases.
- In general, software projects are large, complicated, poorly-specified, involve unfamiliar aspects, and are still vulnerable to large, unanticipated problems.
- `http://en.wikipedia.org/wiki/Software_crisis`

Why is Software Engineering is Important?

- With greater frequency, society relies on software systems.
- We need to be able to produce these systems reliably, economically, and quickly.
    - It is cheaper to use these techniques rather than to hack it together. Really – it is!
- It better be well coded, because maintenance dominates most lifecycles of deployed software.

Conclusions and Motivations

- The software crisis is...
  - impact of rapid increases in computer power.
  - increase in complexity of problems which can be tackled.
- We will learn how to tackle large software projects in this course.
- It's best that your 10^6 book library isn't organised as a pile of post-it notes.
- And please...
  - Don't put the source of Facebook all in `public static void main ()`.
  - Corollary: Don't make a single function and put all of Facebook in it either!