Advanced Professional Software Engineering 2024-25

Lecture 1 – Course Introduction

S Waqar Nabi
Mar 2025

Meet the team

Lecturers:

Dr S **Waqar** Nabi
syed.nabi@glasgow.ac.uk

That's it.

Sorry.

Aim of the course

The aim of this course is to help you develop and demonstrate abilities in two distinct dimensions:

1. Understanding software engineering best practices, and what makes them important.
2. Demonstrating your ability to contribute to a codebase in a team of your peers.

- To that end, you'll be
  - reading up some classic software engineering literature, and
  - developing a code base *as a group*.

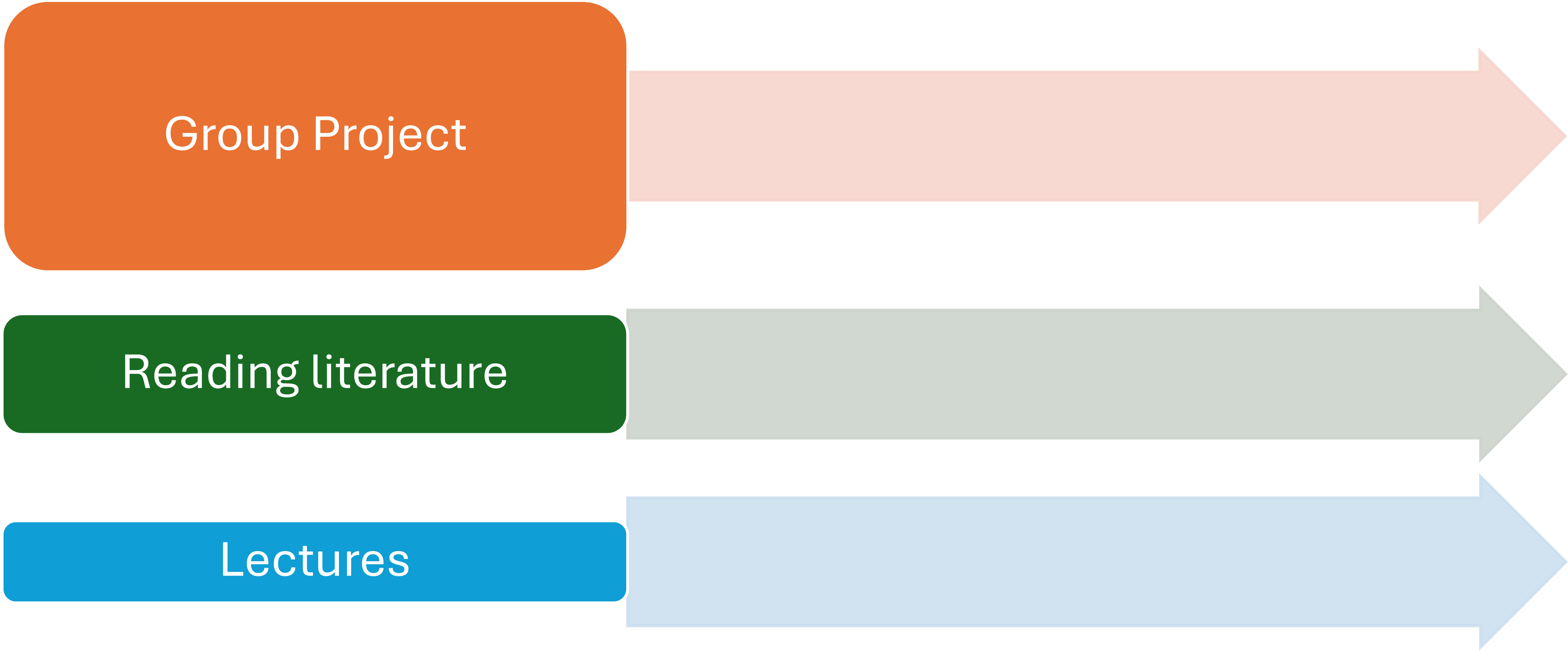
(Paraphrased) Intended Learning Outcomes

- **Appraise** workplace practices from your **career**
- **Evaluate** workplace **leadership** qualities
- **Assemble** a significant piece of **software** with your peers
- **Assess team working practices** in a team of your peers
- Break down and adequately **estimate the scale of tasks**
- Follow the selected **software development methodology**

Structure of Contact Hours

- We meet 2-4pm from Mon 3rd March to Wed 26th March
  - (excluding 13-14 March)
- A few lectures on chosen topics in Software Engineering
- Some time spent reading literature
- **Mostly** it will be you all working in teams on your projects
  - Occasionally meeting me, your "client"
- Please always come prepared with your laptops to work on your Software Engineering Project

Learning Activities



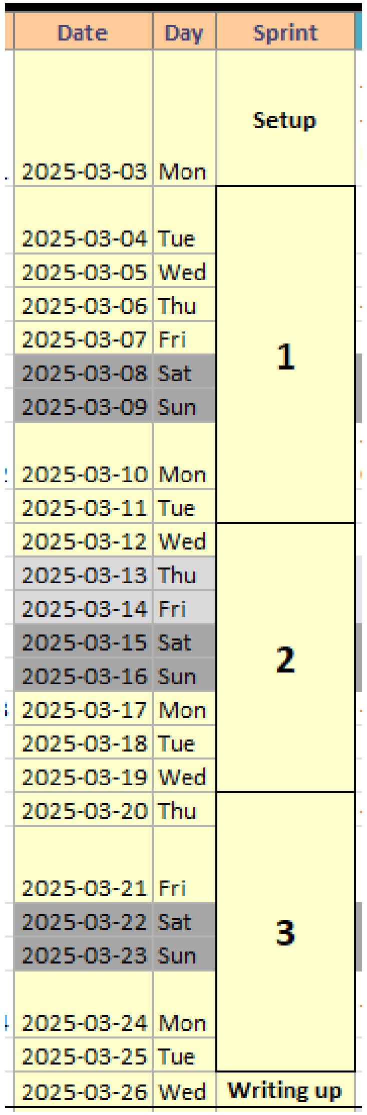
Group Project

Reading literature

Lectures

Development Timeline

- APSE is taught over a 16 day period.
- The first day (today) is reserved for creating groups, brainstorming for your projects, setting up your systems, tools, git, etc.
- The final days is reserved to give you some time for writing up.
- This leaves 14 days dedicated to product development and literature reading.



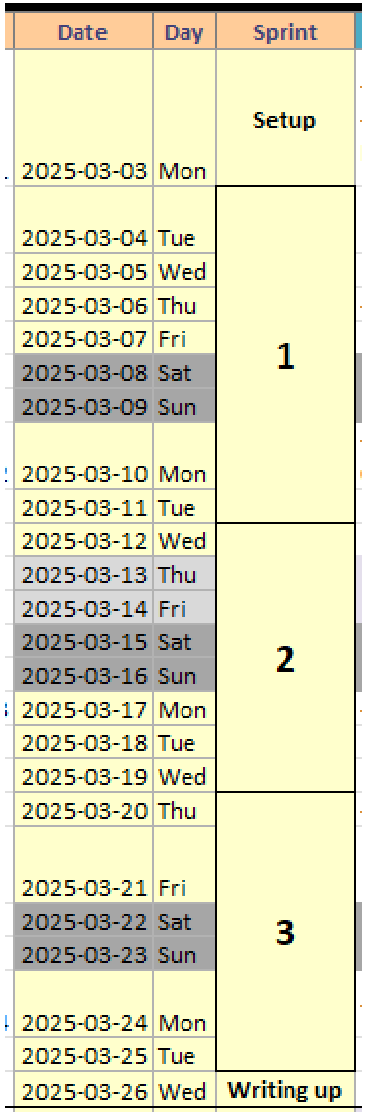
Date	Day	Sprint
2025-03-03	Mon	Setup
2025-03-04	Tue	1
2025-03-05	Wed	
2025-03-06	Thu	
2025-03-07	Fri	
2025-03-08	Sat	
2025-03-09	Sun	
2025-03-10	Mon	
2025-03-11	Tue	
2025-03-12	Wed	2
2025-03-13	Thu	
2025-03-14	Fri	
2025-03-15	Sat	
2025-03-16	Sun	
2025-03-17	Mon	
2025-03-18	Tue	
2025-03-19	Wed	
2025-03-20	Thu	3
2025-03-21	Fri	
2025-03-22	Sat	
2025-03-23	Sun	
2025-03-24	Mon	
2025-03-25	Tue	
2025-03-26	Wed	Writing up

Sprints

Your time is divided up into three sprints:

1. A six-day sprint
2. A four-day sprint
3. Another four-day sprint

- **At the end of every sprint**, you must have a working software
- At the end of every sprint a **retrospective** needs to be documented
  - read the retrospective section on moodle for this
- Every session should start with a **5 minute stand-up** among yourselves so that everyone gets up-to-speed about the project.



Date	Day	Sprint
2025-03-03	Mon	Setup
2025-03-04	Tue	1
2025-03-05	Wed	
2025-03-06	Thu	
2025-03-07	Fri	
2025-03-08	Sat	
2025-03-09	Sun	
2025-03-10	Mon	
2025-03-11	Tue	
2025-03-12	Wed	2
2025-03-13	Thu	
2025-03-14	Fri	
2025-03-15	Sat	
2025-03-16	Sun	
2025-03-17	Mon	
2025-03-18	Tue	
2025-03-19	Wed	
2025-03-20	Thu	3
2025-03-21	Fri	
2025-03-22	Sat	
2025-03-23	Sun	
2025-03-24	Mon	
2025-03-25	Tue	
2025-03-26	Wed	Writing up

Assessments

Assessment	Weightage	Deadline
Assessed Exercise 1 - **Critical Reflection on Literature**	30%	19/03
Assessed Exercise 2 - **Portfolio of Contributions to Group Project**	30%	25/03
Assessed Exercise 3 - **A Process Improvement Report**	30%	02/04
Class Tests x 4	10%	Continuous

Assessed Exercise 1 –
Critical Reflection on Literature (30%)

- This assessed exercise will allow you engage with some classic literature in Software Engineering, reflect on your practices in your current group, as well as appraise workplace practices from your career.

- This assessment consists of two components:

(a) Reading notes on the provided literature, and

(b) An essay based on one of the provided papers.

Literature for Assessed Exercise 1

1. Naur’s “Programming as Theory Building”, which investigates what we mean when we talk about “programming” (and building software systems in industry teams in general).
2. Royce’s “Managing the Development of Large Software Systems” is an early and influential essay on what the process of building software ought to be. Waterfall software engineering is believed to have been first adopted en-masse due to a misinterpretation of this essay.
3. Heddleston’s “The Null Process” is written by an industry engineer and describes issues with undefined processes in software engineering.
4. Conway’s “How do Committees Invent?” is probably the most abstract paper in this set. It describes how the structure of a “design team” (a software engineering team, for our purposes) can influence what that team produces, and actually arrives at the conclusion that the structure of a team’s communication is the structure of their products. We find increasing amounts of empirical evidence that this is true.
5. Hamilton’s “What the Errors Tell Us” is an essay written by the woman who coined the term “software engineering”, and reflects on what she learned fighting errors from the early days of industrial programming. It assessed where bugs come from, and how planning software products can mitigate these.
6. Raymond’s “The Cathedral and the Bazaar” is a very influential essay investigating why open source software engineering works as well as it does, written by an author who believed all software must be planned, observed Linux’s success despite a total lack of planning, and proceeded to test the open source model himself. This essay investigates what he learned. This is the longest of the required reading for this course, but it’s very accessibly written and a great read. If you use, say, Firefox or Wikipedia, you use the software this essay inspired

Assessed Exercise 2 – **Portfolio of Contributions to Group Project (30%)**

- A portfolio of your contributions to the group project, submitted on Mahara.

- Three artefacts to be submitted, which can be either of:
  - pull requests / feature implementations,
  - documentation, or
  - code review on other team members' code.

- You should aim to include one from each category (will boost your marks, see details on moodle)

Assessed Exercise 3 –
A Process Improvement Report (30%)

- You will submit a short report describing how your group's software development process could have improved.
- This report will be marked out of 100 and is worth 30% of your APSE grade.
- More details of this will be shared on moodle.

Assessment # 4 –
4 x Class Tests (10%)

- You will have 4 light-touch class tests based on content we cover in lectures.

- They will be announced in advance, they will be open-book, timed, with single attempt.

Assessments

Assessment	Weightage	Deadline
Assessed Exercise 1 - **Critical Reflection on Literature (30%)**	30%	19/03
Assessed Exercise 2 - **Portfolio of Contributions to Group Project**	30%	25/03
Assessed Exercise 3 - **A Process Improvement Report**	30%	02/04
Class Tests x 4	10%	Continuous

Many more details are on Moodle.
Please give yourself the time to go through them deliberately and carefully.

Feedback & Course changes

- You will get **feedback** in the following ways:
  - 1-1 and group feedback from course instructor during class
    - Most contact hours dedicated to tutorial/lab style interaction
  - Peer feedback during classes
  - Individual coursework grades with detailed breakdown of marks, and comments where appropriate
  - Overall, general commentary on courseworks
  - Automated feedback in class tests
  - Office hours (by request)
- **Course changes**
  - 3 instead of 4 assessed exercises
  - Class tests added

Selecting a Project

- You can choose to make one of the sample projects already described on moodle

- Or you can come up with your own idea through brainstorming
  - Please pass it by me before finalizing

- Better to go for a project that works well with your combined strengths

Software Development Approach

- It is probably obvious already, but for our practical project, we will follow the Agile Methodology, and specifically, the Scrum method.

- For your essay writeup(s), you should feel free to comment and criticize this approach, and if you feel a different approach would have worked better.

How to Brainstorm

- Everyone sits isolated and thinks of a project idea for some allocated time (say 15 minutes)
- All the points good or bad are written down
- After the time is over, everyone comes together and tells their idea in a given time slot in round robin fashion
- No one interrupts the idea or passes judgment, however questions maybe asked in the end
- Once all the ideas have been discussed, everyone votes anonymously on other’s ideas to pick the most voted one

Tooling

- It is strongly recommended that you make use of the school's gitlab servers, at stgit.dcs.gla.ac.uk, to manage your project. You will need to use the VPPN to access these servers. Everybody has an account on stgit already. (If you can't get in, let me know.)

- Gitlab is strongly recommended because it provides all of the tooling necessary to easily meet the requirements of this course. In particular, I recommend you use:
    - a gitlab repo everybody commits to as the hub of your team's activity
    - the repo's wiki for your documentation
    - branching and merge requests to separate work and to support code review during merge
    - gitlab's issue management system to manage tickets
    - (the kanban view for these is very helpful for project management!)
    - The wiki to keep retrospective notes
    - The wiki to document team structure

- While I suggest you use the school's stgit servers, any project management system which works for your team is acceptable.

Now…

- Take some time to divide yourself up into groups of 4-5.
- Please let me know by email your group composition (and a name if you like) before the start of class tomorrow (Tuesday).
- Then: Brainstorm to choose one of the projects already on moodle, or come up with a new one.
- Start setting up the tools of the trade, see instructions on Moodle.
- Take some time to review the course moodle