Software Engineering COMP1035

Lecture 01

Introduction

Schedule

Aims of the module.

What is the "Software" we are talking about.

The need for SE.

Overview of SE process.

Plan for the semester.

Aims of the Module

What is Module Specification?

- Summary of content: You'll be introduced to the concept of Software Engineering and will be taken through the software development process:
  - deciding exactly what should be built (**Requirements**),
  - designing how it should be built (**Software Specification/Architecture**),
  - development strategies (**Implementation & Testing**), and
  - maintaining change (**Software Evolution and Maintenance**).

- An aim of this module is to provide a general understanding of Software Engineering;
  - Typical phases of the software lifecycle with reference to practical Requirements and Specification, Software Design, and Implementation & Testing techniques.
  - To prepare you for the various software development projects undertaken throughout your studies.

COMP1035 (FSE) Aims

1. Teach you an ***overview** of the **whole** Software Engineering process.

2. Give you initial **practical experience** of work at different stages of the process.

3. **Prepare you** for Software Engineering group project in Part 1 (Y3).

(Note: Not able to cover everything in a 10 credits module)

What is the "Software" We are Talking About?

What is Software?

- Such that it needs a big process.

- You largely did not build software in semester 1 – you wrote code.

- Bigger software examples:
    - A phone game app, with a global scoreboard
    - Twitter software
    - Google Chrome
    - Universal Credit System for Benefits (for Government)

What is Software?

- Last semester, you built dedicated scripts to achieve one thing!

- A **component** of software.

- Not ready for 'real human' use.

- Probably was buggy.

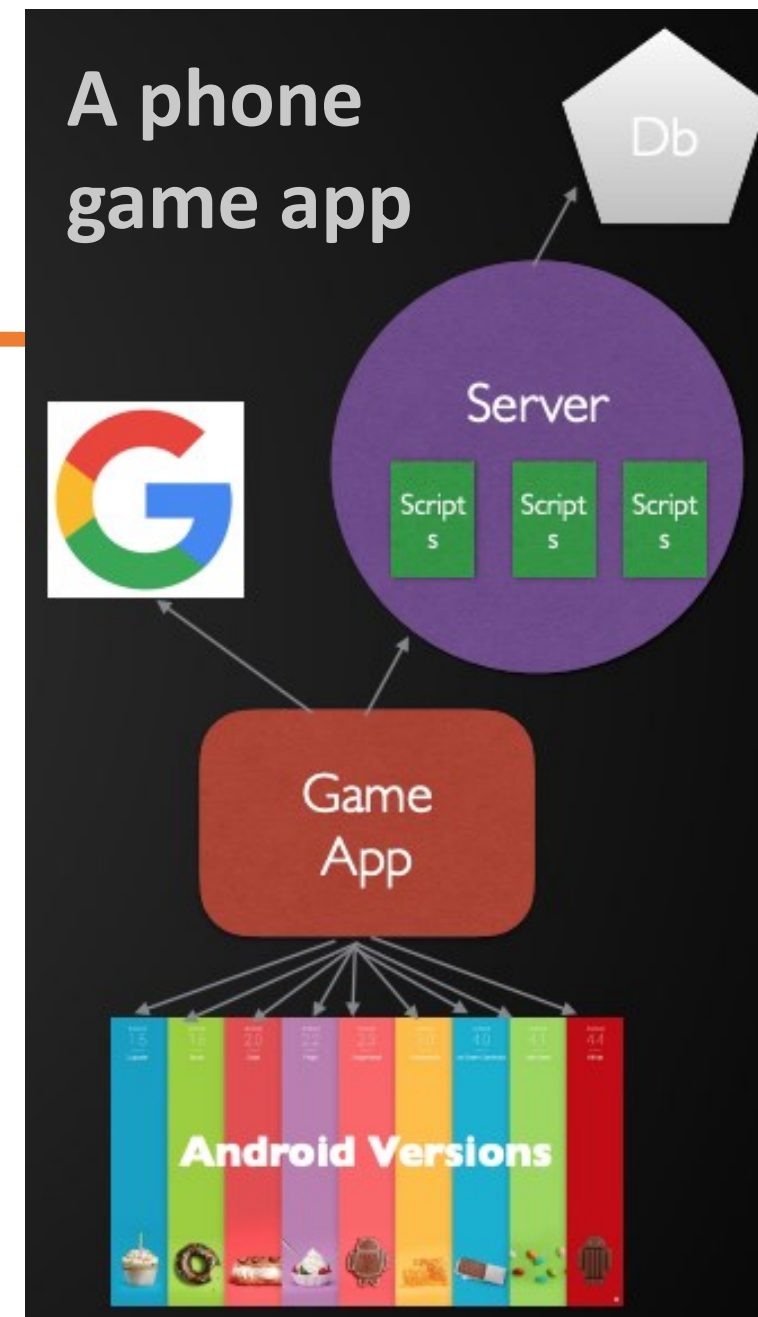- Software is something that is working, usualy for 'real humans' to use to help them do something.
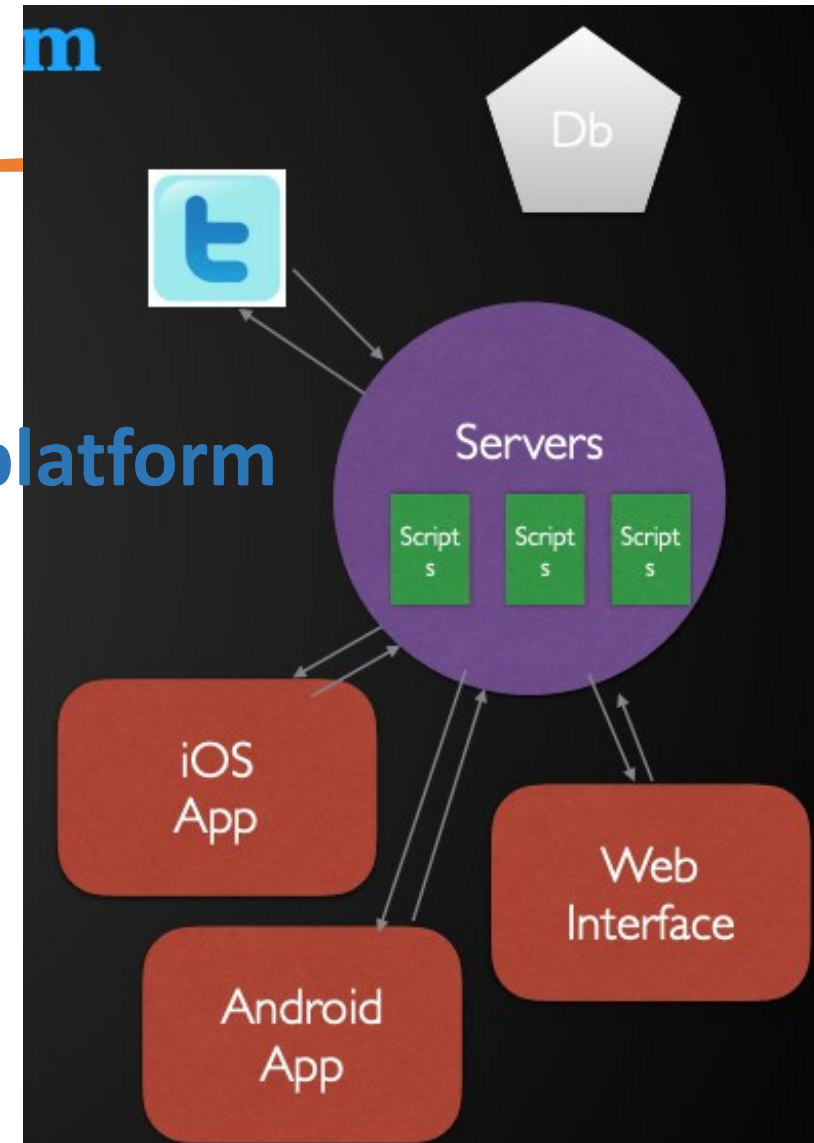
What is Software?

- Ready for real users.
- Integrates many features in the game.
- Might integrate with other services (e.g., login).
- Might have a server component.
    - Talking to a database.
- Need to push out multiple versions.

**A phone game app**

What is Software?

- Let's ignore issues of scale for now.
  - Pretend '1 database' is ok.
- Basically, a server driven platform.
- Has a lot interfaces to it.
  - Each is a piece of software.
  - Each one integrates functions.
  - Each piece might have different bugs.
- New features rolled out across them.

A social media platform

Interest Reading: Twitter's Own Diagram

- Video of Twitter Scale:
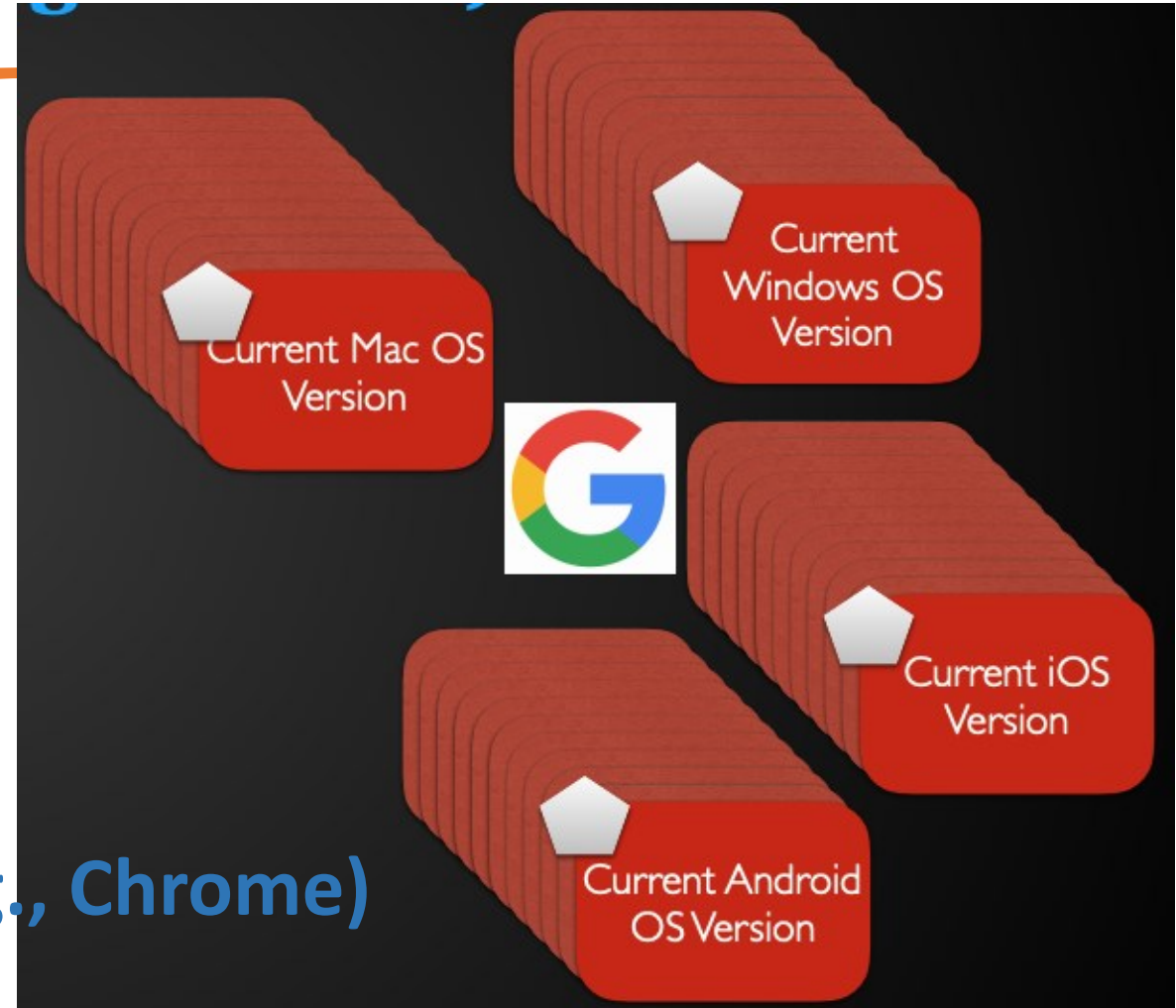https://www.infoq.com/presentations/Twitter-Timeline-Scalability/
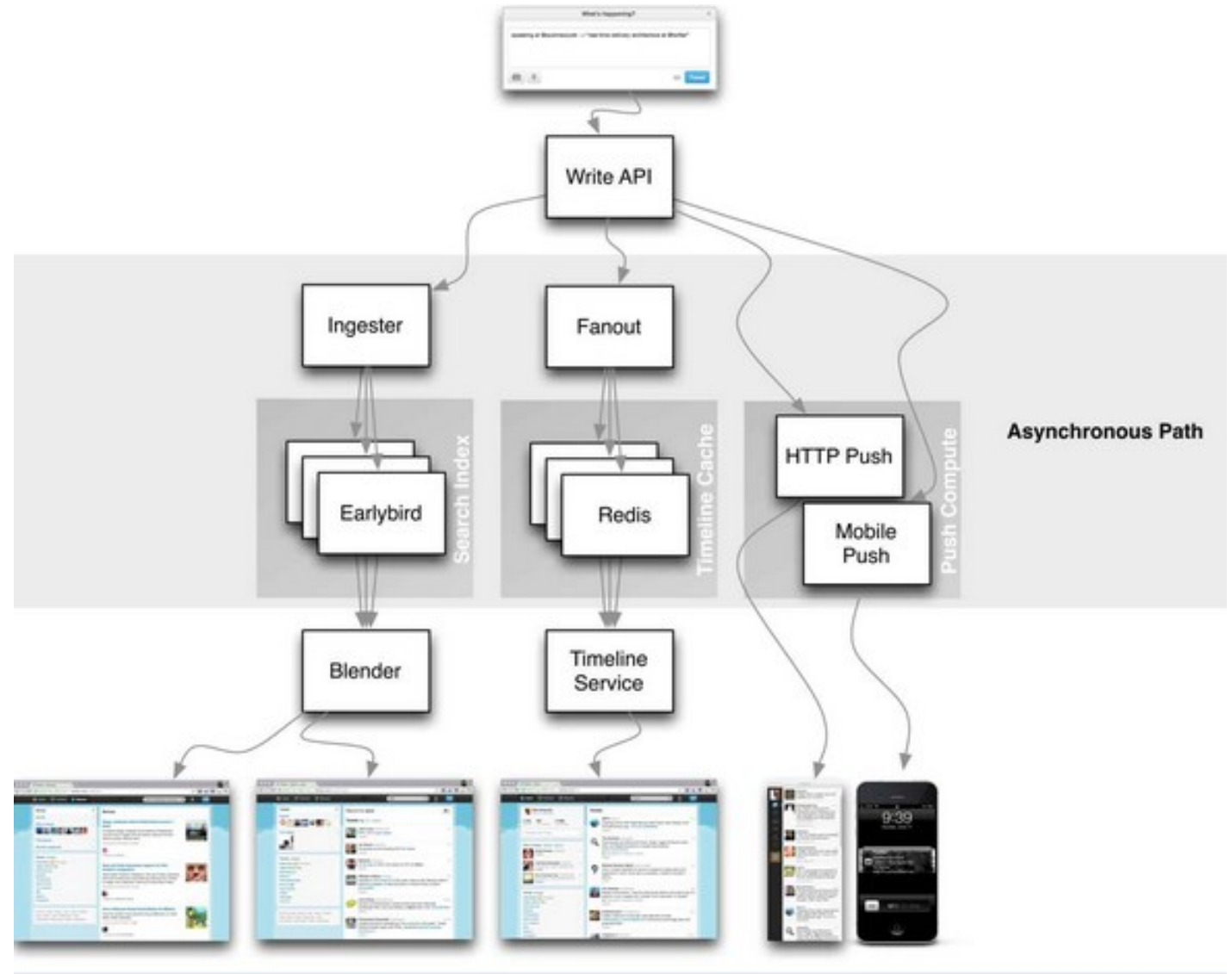
What is Software?

- Integrate multiple functions.
- Has a version history on every platform.
- New functions roll out on all platforms.
- All authenticate and share data.

A web browser (e.g., Chrome)

What is Software?

- Government service to handle benefits payments.

- Integrate 6 existing systems.

- Process millions of users.

- Be used at job centres – across UK.

UK Universal Credit (Benefits) Software

UoN Campus Solution (Simplified)

- In 2015 – UoN wants to buy a software package to handle student records – merge 4 different systems
  - Student records.
  - Module enrolment.
  - Admissions.
  - Course documentation.
- To use on all 3 campuses (UK, Malaysia, Ningbo China).
- They have brought this from a software company.

UoN Campus Solution (Simplified)

- **Phase 1** – Admission starts across all 3 sites in 2016.

- **Phase 2** – Went live in Malaysia in January 2017.

- **Phase 3** – Full development due January 2018 – Ningbo China goes ahead with all data managed in both old and new systems.
  - UK delayed till April 2018.
  - UK delayed till Summer 2018 (still managing data in both old & new).
  - UK delayed until Christmas 2018.
  - January 2019 – Went live in UK (data still managed in old system, just in case).

- **Phase 4** – Finance part of the system going ahead in 2019/20!

Testing of Campus Solutions

- http://blogs.nottingham.ac.uk/campussolutions/2018/10/24/overview-of-campus-solutions-testing-procedure/

Here is an insight into the comprehensive testing steps that the team are working through:

1. **Unit Testing** – In this initial test, the code, configuration, Oracle functions, data handling and security of the functionality are reviewed and tested by the developer.

2. **Functional Acceptance Testing (FAT)** – This involves analysing the original requirements and running tests to make sure each of them has been delivered by the developer.

3. **Business Acceptance Testing (BAT)** – After the functionality has passed FAT, it is given to the business user for them to run tests on how the process will be run. This is where the business checks that the functionality works as the University expects it to and nothing has been missed.

4. **End-to-end testing** – 15 different student types across the student lifecycle have been identified at the University. In this test, the functionality is tested across as many varied circumstances as possible to ensure it works with every student type. Connections and data transfers to other University systems are also tested here.

5. **Regression testing** – As new functionality is brought into the system, it is important to ensure that existing functionality continues to work. This testing stage ensures that new and existing functionality works in harmony together.

6. **Performance testing** – This assesses the performance of the functionality in terms of speed, scalability and stability. Performance testing ensures the system can cope with events such as confirmation and clearing and start of session when system usage is very high.

7. **Penetration testing** – It is important to ensure that the functionality and the system as a whole has the best security standards to withstand any malicious attacks, such as viruses or worms. The servers and network that the system will be running on are also tested. Specialist external companies are used to 'attack' the system and provide recommendations to improve security.

What is Software?

- Software includes
  - The compiled code.
  - Documentation.
  - Configuration components.
  - Installation/upgrade.
- All in an installer that 'deploys'.
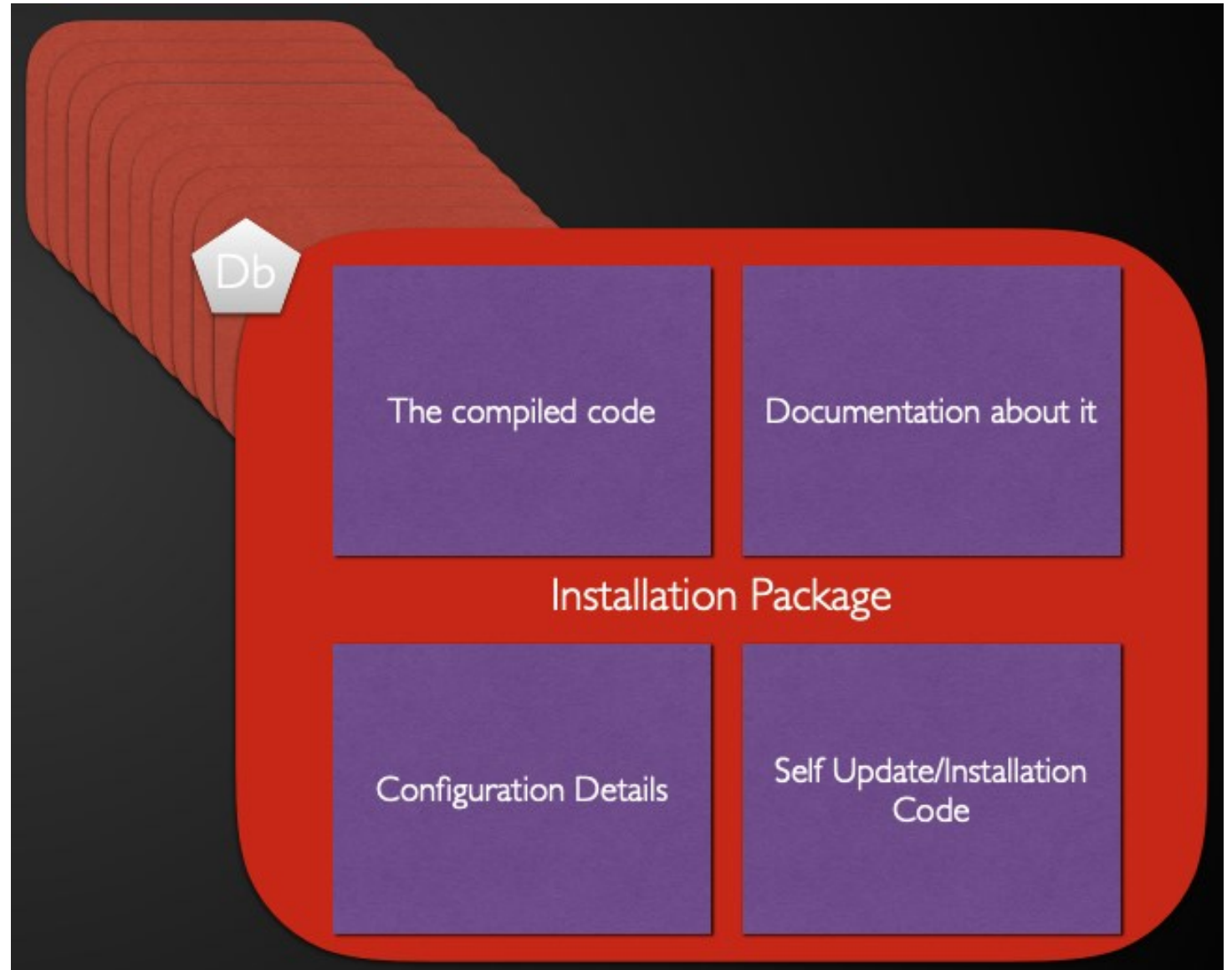
What is Software Engineering?

"The application of a **systematic**, **disciplined**, **quantifiable** approach to the **development**, **operation** and **maintenance** of software [...]; That is, the application of engineering to software"

- Wikipedia

(actually, gives a good definition, despite being a bad reference source)

The Need for Software Engineering

Bad Software is Frustrating

Bad Software is Frustrating

"Unexpected item in bagging area. Please wait for assistance."

Bad Software is the Worst

The Software Crisis (Nato, 1968)

- Software was bad, or worse.

- Software was unreliable behind schedule and cost more than expected.

- 'Software Engineering' was coined to understand the 'making' of software.

- During the 1970s/80s (and still now) people develop new processes to better 'engineer' software.

"The average customer of computing industry has been served so poorly that he expects his system to crash all the time, and we witness a massive world wide distribution of bug-ridden software for which we should be deeply ashamed."

- Dijkstra, 2001

Ongoing SE Project Failurees

- The global cost of IT failure was estimated at +6 trillion dollars in 2009. (source: Roger Sessions, The IT Complexity Crisis: Danger and Opportunity)

- Only 32% of software projects were "successfully completed" (i.e., on time, on cost, and with expected functionality) in 2009. (Source: Standish CHAOS 2009 Update)

- Only 16% of software projects were successfully completed in the UK in 2009. (Source: British Computer Society)

- "A failing industry …
  - If building engineers build buildings with the same care as software engineers build systems, the first woodpecker to come along would be the end of civilization as we know it." (Source: Paul Dorsey, Top 10 Reasons Why System Projects Fail)

Ongoing SE Project Failures

https://en.wikipedia.org/wiki/List_of_failed_and_overbudget_custom_software_projects

NEWS

Technology

US prisoners released early by software bug

🕓 23 December 2015 | Technology

US Prison Bug

- More than 3,200 US prisoners have been released early because of a software glitch.

- The bug miscalculated the sentence reductions prisoners in Washington state had received for good behavior.

- Analysis of the errors showed that, on overage, prisoners whose sentencees were wrongly calculated got out 49 days early. One prisoner had his sentence cut by 600 days.

- An update that applies the correct formula for calculating sentence cuts is due to be place by 7 January.

Hawaii Incident

- A test of "incoming nuclear bomb" message was accidentally distributed as a live message to citizens.

- A human error, but the software has been blamed for faciliting it.

- Requiring as little as 3 clicks either way, with the same UI
    - The interaction for test vs live is the same – with the same popups.
    - Depending on a user 'noticing' their error.

Ariane 5 Rockets

- "Ariane 5's first test flight (Ariane 5 Flight 501) on 4 June 1996 failed, with **the rocket self-destructing 37 seconds after launch because of a malfunction in the control software**. A data conversion from 64-bit floating point value to 16- bit signed integer value to be stored in a variable representing horizontal bias caused a processor trap (operand error) because the floating-point value was too large to be represented by a 16-bit signed integer.

- The software was originally written for the Ariane 4 where efficiency considerations (the computer running the software had an 80% maximum workload requirement) led to four variables being protected with a handler while three others, including the horizontal bias variable, were left unprotected because it was thought that they "physically limited or that there was a large margin of error".

- The software, written in Ada, was included in the Ariane 5 through the reuse of an entire Ariane 4 subsystem despite the fact that the particular software containing the bug, which was just a part of the subsystem, was not required by the Ariane 5 because it has a different preparation sequence than the Ariane 4."
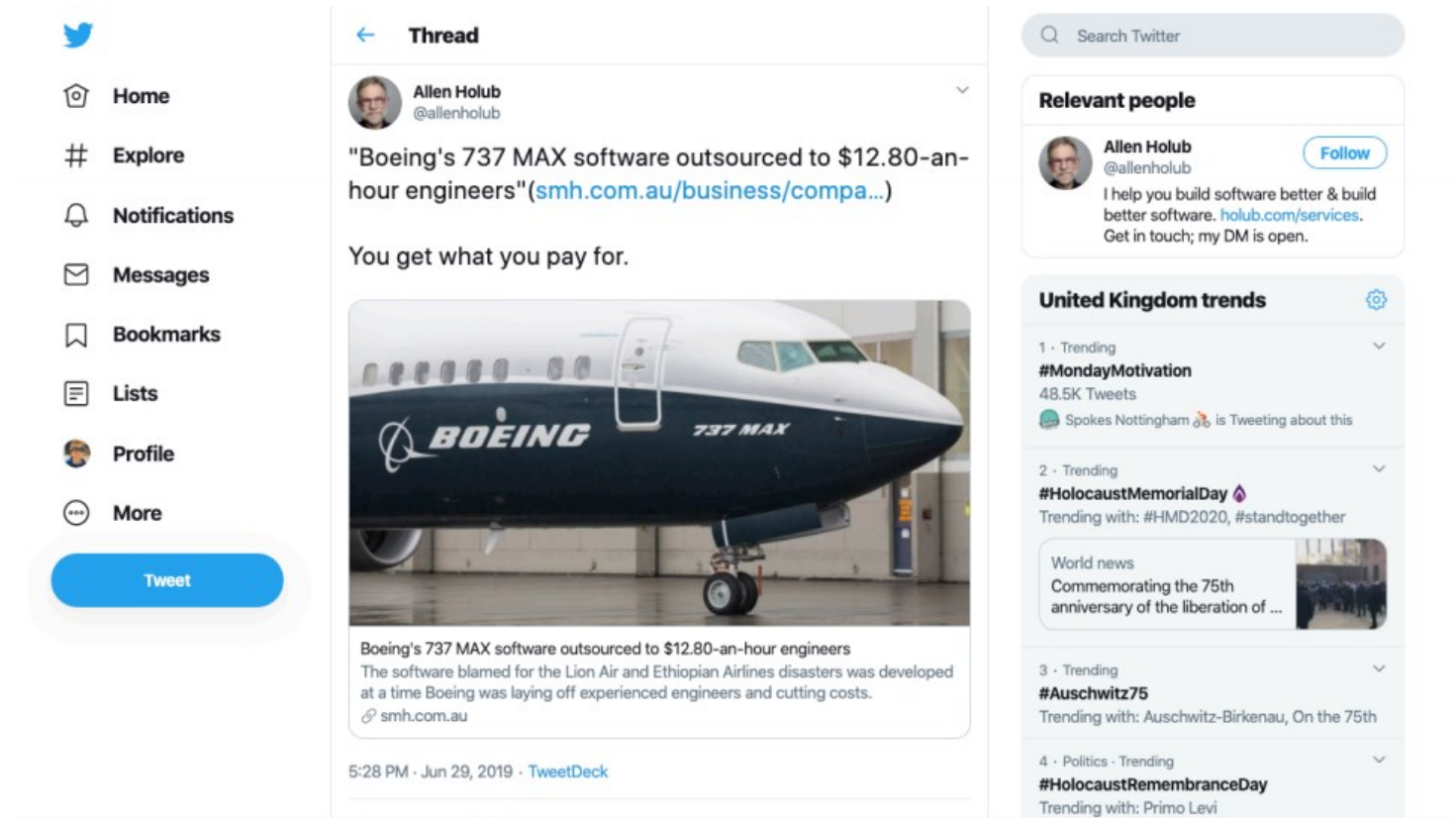
Coding & Testing

Software Maintenance

Design & Test Planning

Change Maintenance

More Contemporary Example

- https://twitter.com/allenholub/status/1145006228348657664?s=12

Software Maintenance

Top 10 SE Mistakes

1. Presume 'good code' is the only thing that matters.
2. Aim to finish at the delivery deadline.
3. Don't design a data model, let the code produce data it needs.
4. Use a Technical Lead that has never built a similar system, rather than choose/hire someone who has.
5. Hire forty developers to make coding go faster.
6. Build the system in the language you know best, rather than 'the best tool for the job'.
7. Hire a junior developer to handle the migration.
8. Skip the testing phase because the project is behind schedule.
9. Change code, without planning, to meet newly discovered requirements.
10. Buy a commercial, off-the-shelf package and customise it … a lot.

Dorsey's Conclusions

1. Don't cut corners, methodogically. In the long run, this result in system failure or an inadequate system that doesn't meet the users' needs.

2. **Audit each major deliverable and step along the way for accuracy and correctness**.

3. Carefully monitor top management support for the project. Make sure that managers are aware of the progress of the team.

4. Secure the correct technical lead for the project.

"There is no free lunch in software engineering. If you insist on **keeping costs low** and **hurrying the project along**, then **quality will be low** or the **risk of failure will be high** no matter how well the project is managed."

- Paul Dorsey

Overview of SE Process
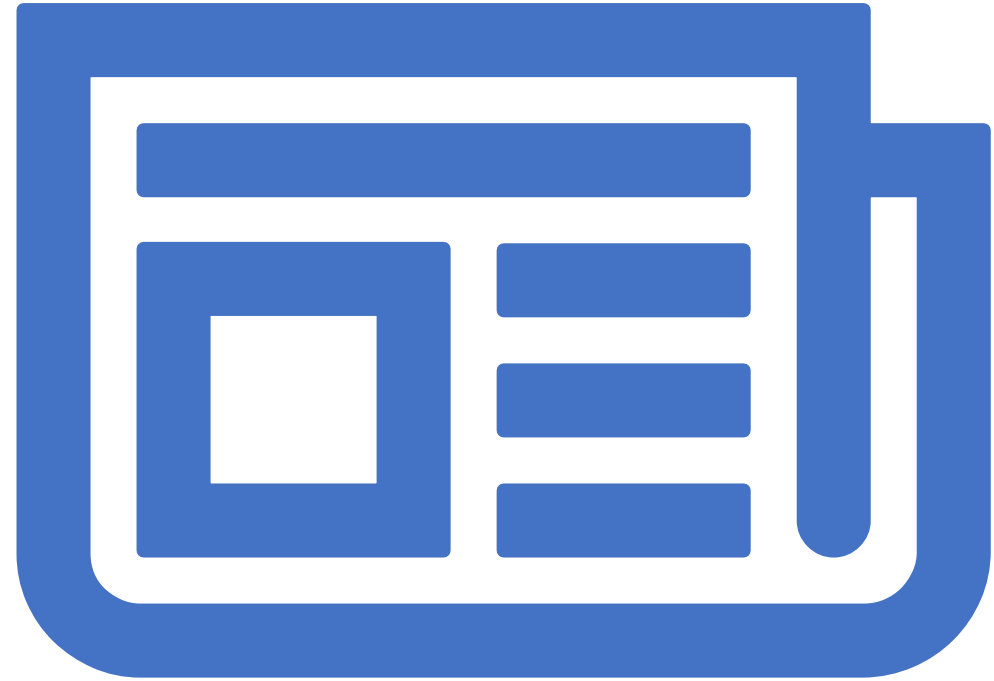
Software Engineering Process

- Software engineering is an engineering discipline that is concerned with **all aspects** of software production.

- "Software engineering is concerned with *principles* and *methods* for *specifying, designing, implementing* and *maintaining* large software systems." R.Sirewalt 2004

- Software engineers should adopt a **systematic and organized approach** to their work and **use appropriate tools** and techniques **depending on the problem** to be solved, the development contstraints and the resources available.

Software Engineering Process

- You are not building something **for** yourself
  - you need to understand/agree what someone else wants
- You are not building something **by** yourself
  - you need to work with others - in different countries even
- You are not building it for people **like** yourself
  - it has to work for 'real humans', not computer experts
- You are not working on code you **wrote** yourself
  - you need to figure out how the hell is this stupid code working, who wrote this? an octopus? its the worst code I've ever seen. were they drunk? There's no comments.

Programming Wisdom
@CodeWisdom

Following

"Any code of your own that you haven't looked at for six or more months might as well have been written by someone else." - Eagleson's law

6:00 PM - 4 Jan 2019

Examples of SE Processes

- This depends on what you are doing.

- Structured stages for large industrial projects.

- Flexible processes for small teams.

- Processes for international collaboration.

- However, they all have same kinds of activities in them.

Core SE Process Stages

- Generic activities in all software processes are:
  - **Specification** – what the system should do and its development constraints.
  - **Development** – production of the software system.
  - **Validation** – checking that the software is what the customer wants.
  - **Evolution** – changing code in response to demands.

Waterfall Model

- Developed in 70s
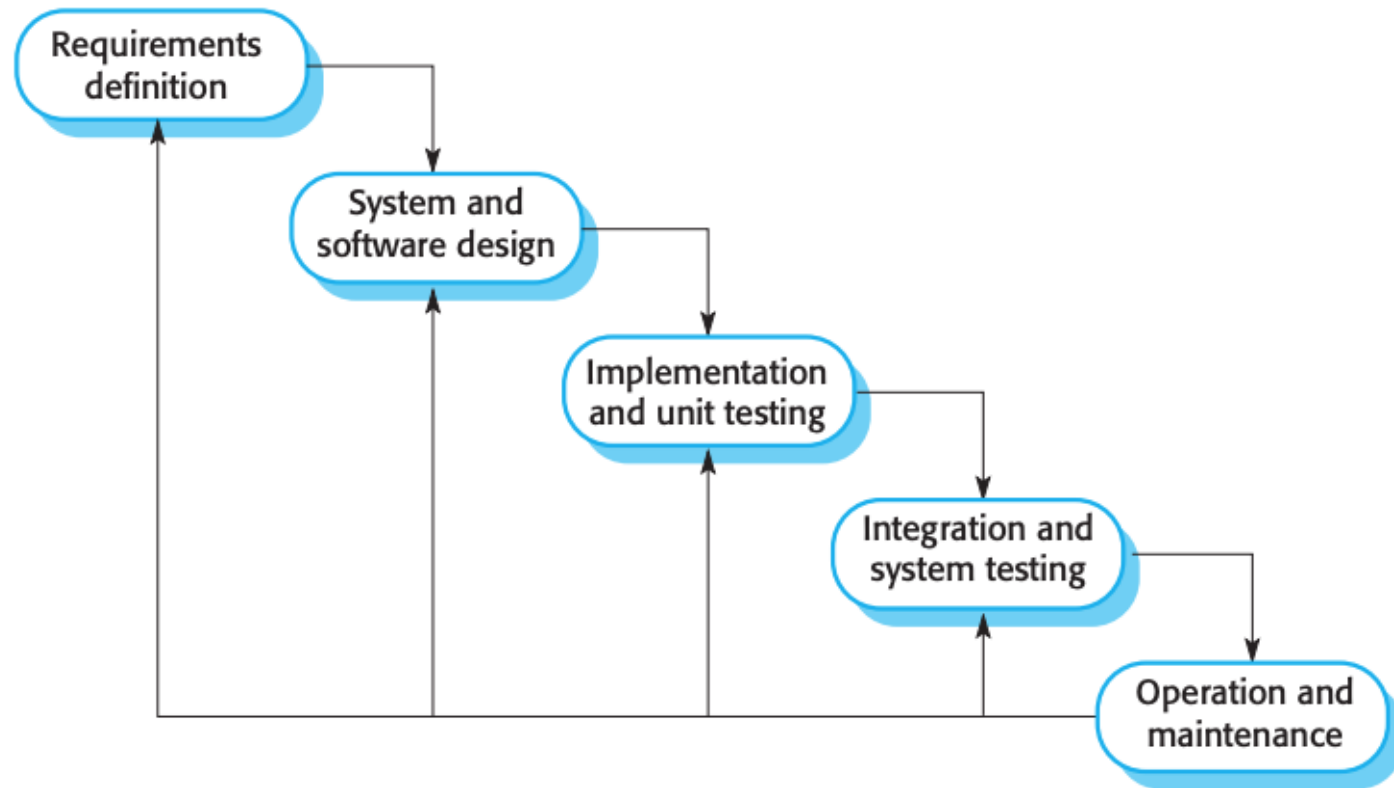
V-Model

- http://sqa.fyicenter.com/FAQ/Software-DevelopmentModels/Software_Development_Models_V_Model.html

V- Model

Developer's life Cycle

Tester's Life Cycle

Verfication Phase

Validation Phase

- Business req. Specification → Acceptance Testing
- System Req. Specification → System Intergration Testing
- High level Design → Component Testing
- Low level Design → Unit Testing
- Coding
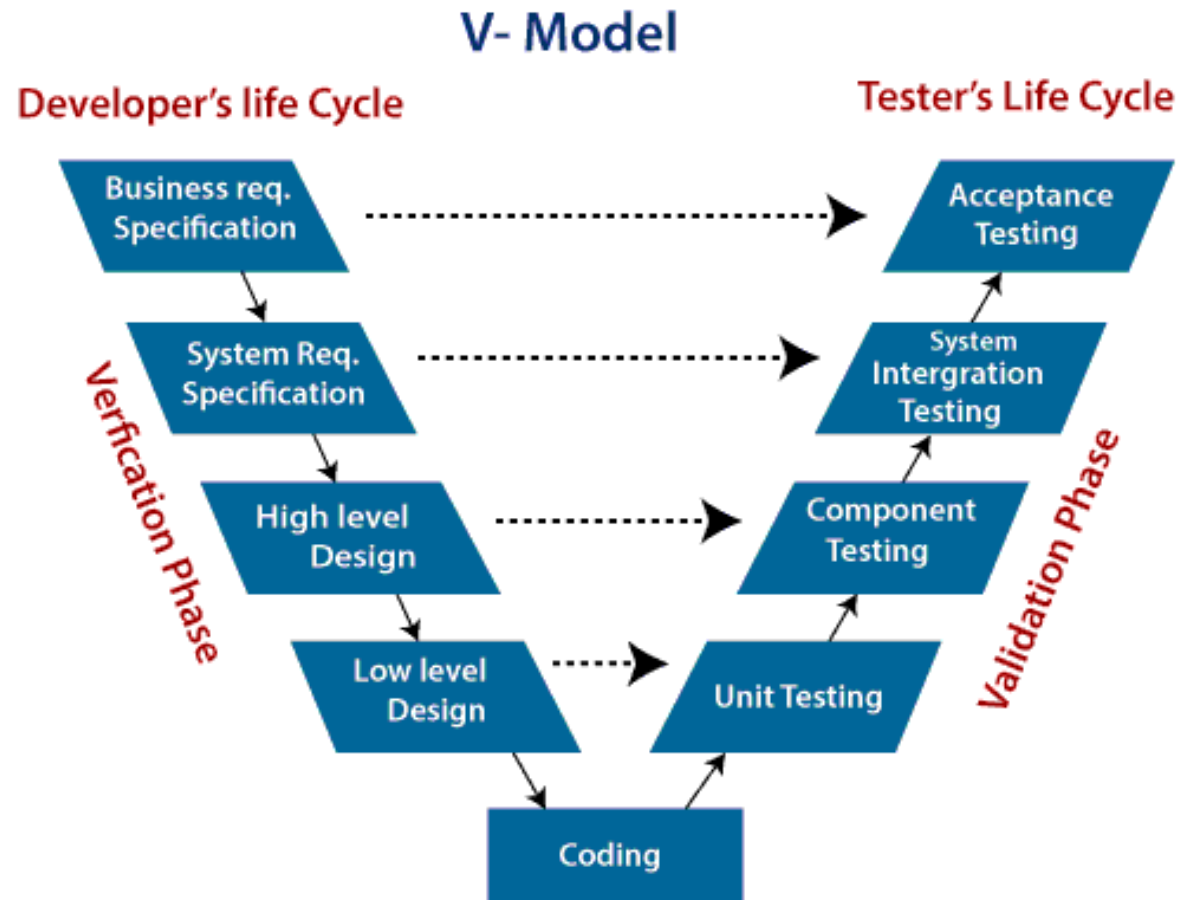
Problems with Waterfall & V

- Needs stable and "perfect" requirements.

- Can't always anticipate what you are going to have to do.

- Does not account for revision or refactoring.

- Too inflexible and static.

- Depends on getting each stage exactly right
  - Changes can have many knock-on effects.

Iterative Model

Concurrent activities

Outline description → Development

Specification → Initial version

Development → Intermediate versions

Validation → Final version
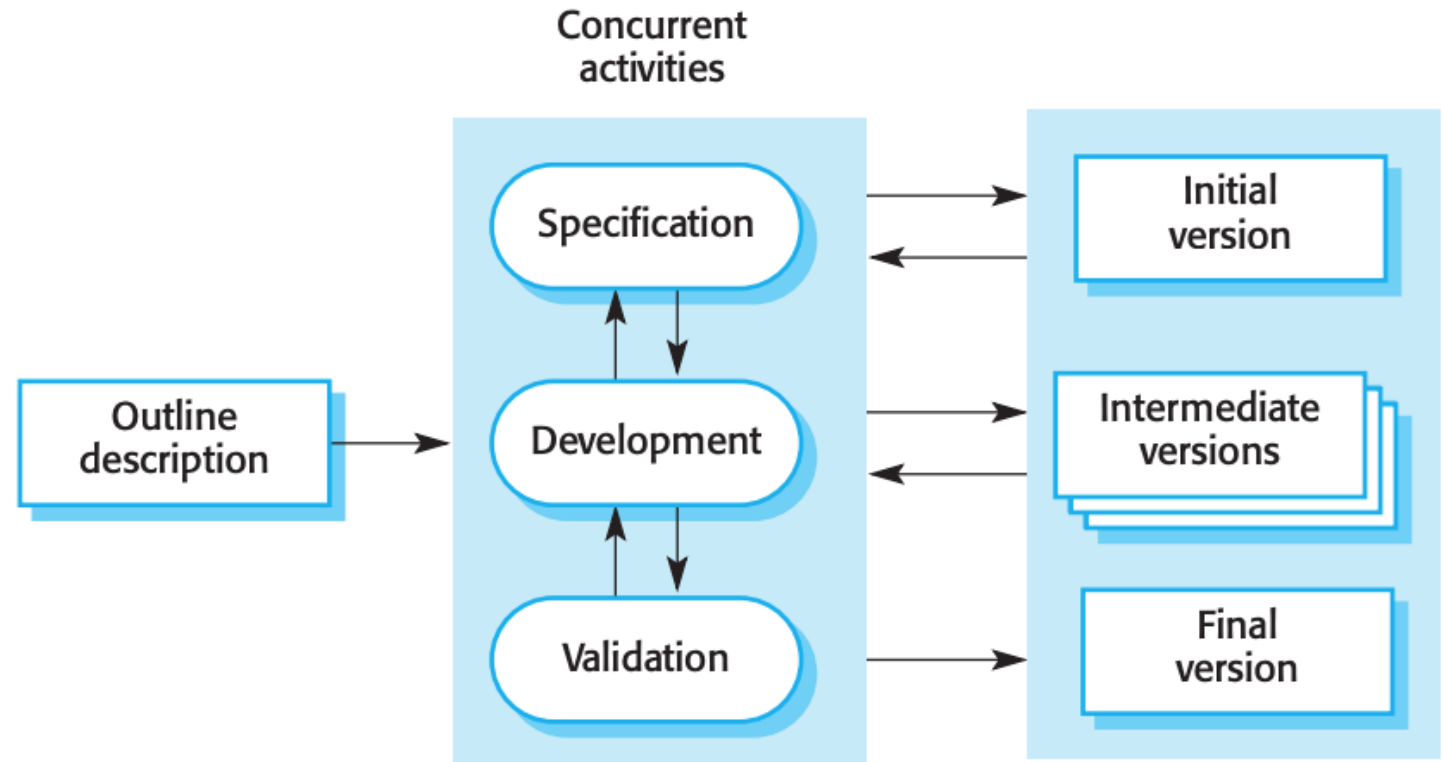
The iterative process model is a software development life cycle (SDLC) approach where development begins with clearly defined initial requirements. Additional features are then incorporated into the base software product through iterative cycles until the final system is completed.

Iterative Model – Boehm's Spiral

Developed in late 80s
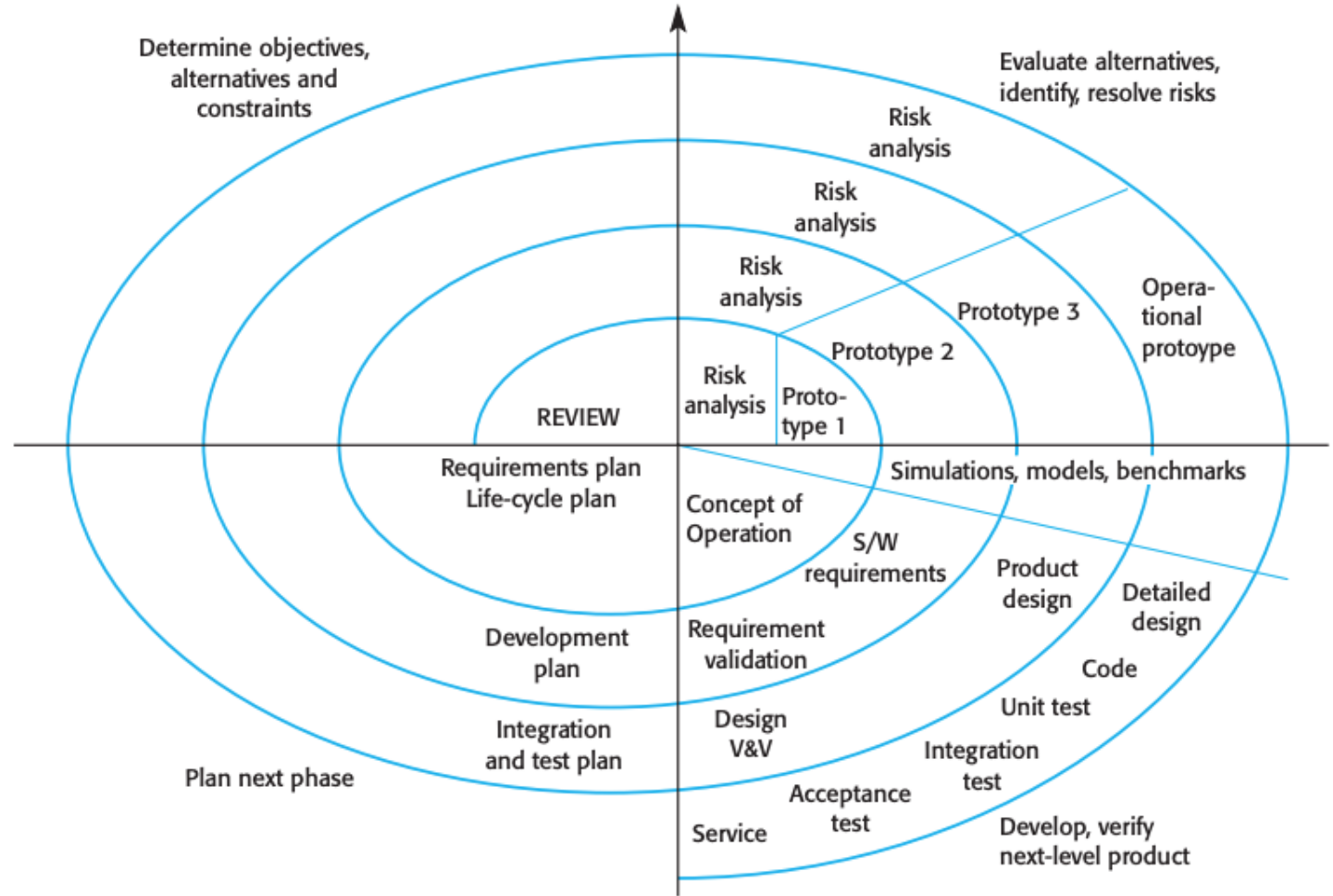
Problems with Iterative Model

- Lack of process visibility.
  - Systems are often poorly structured.
- Too much doing and not enough planning.
- Special skills (e.g., in languages for rapid prototyping) may be required.
- Lightweight documentation taken to mean no documentation.

Effort in SE Processes

- 60% in designing and building the software.

- 40% in testing/delivering the software.

- BUT – Maintaining/Evolving software often costs **more** than producing it in the first place.

Why Do We Need SE Process?

1. Coding is one stage, out of e.g., 5 or more stages.

2. Having an appropriate SE Process/Methodology is really important.

3. Quality means getting the right process - and process right.

4. Quality means getting every stage done right, not fast.

Summary

- Several real-world cases of SE software failures.

- Importances of the SE methodologies.

- Brief introduction of the overview of SE models.

- 4 stages of SE activities:
  - Requirements and Specification
  - Development (will be covered in other programming modules)
  - Validation
  - Evolution (will be covered in COMP2059 Developing Maintainable Software)

Optional Readings

- Major Software Fails – Wikipedia

- Universal Credit Fail – 2013

- Universal Credit Fail – 2016

- Facebook Software Architecture

- Video Lecture – How Twitter is Built (c2013)

Plan for the Module

SE Book

Software Engineering
TENTH EDITION
Ian Sommerville
GLOBAL EDITION
ALWAYS LEARNING
PEARSON
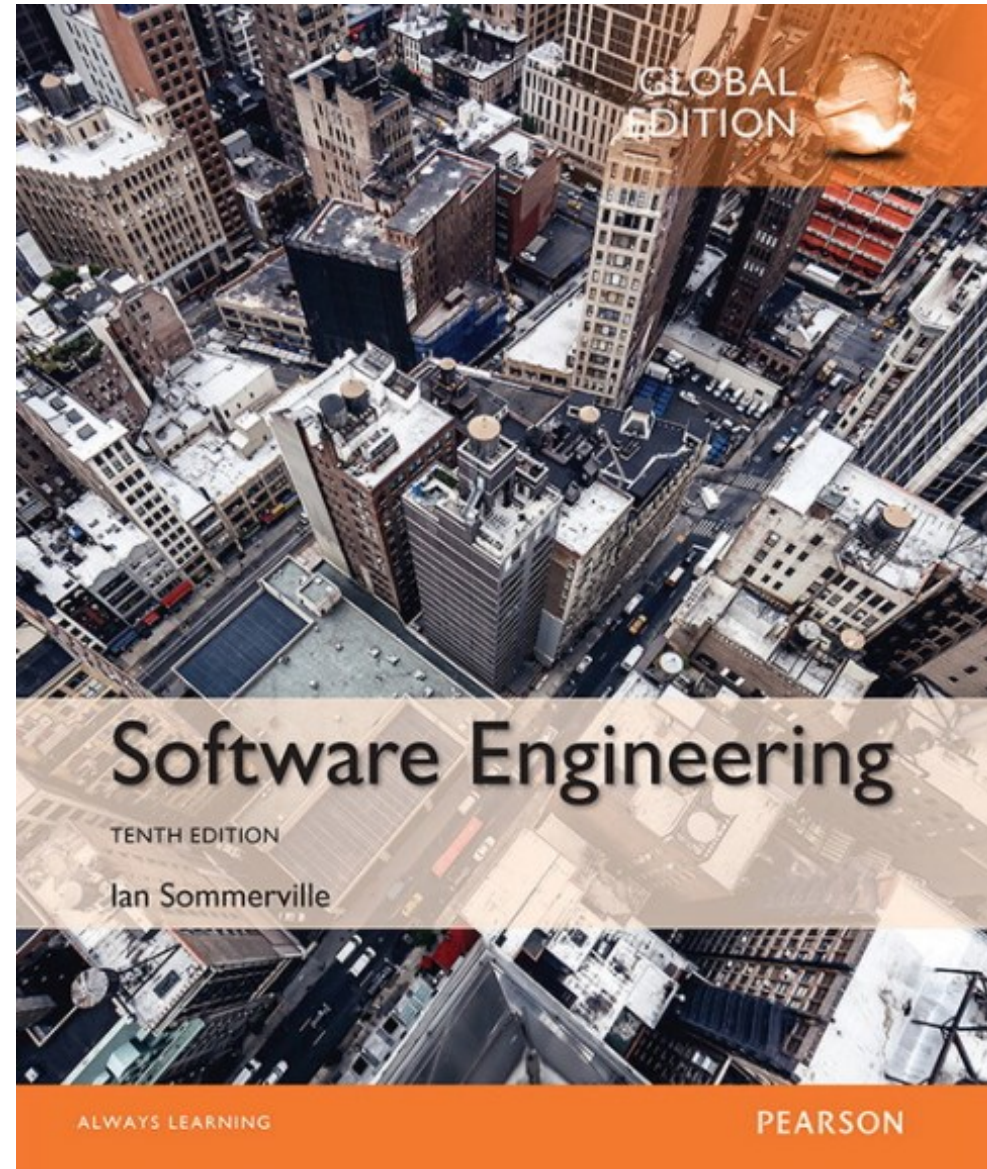
Lectures/Labs for Each Stage

Requirements definition — 4 Lecs + 1.5 Labs

System and software design — 3 Lecs + 1.5 Labs

Implementation and unit testing — 4 Lecs + 3 Labs

Integration and system testing — 2 Lecs + 1 Lab

Operation and maintenance — 3 Lecs + 1 Labs

Project Management — 2 Lecs
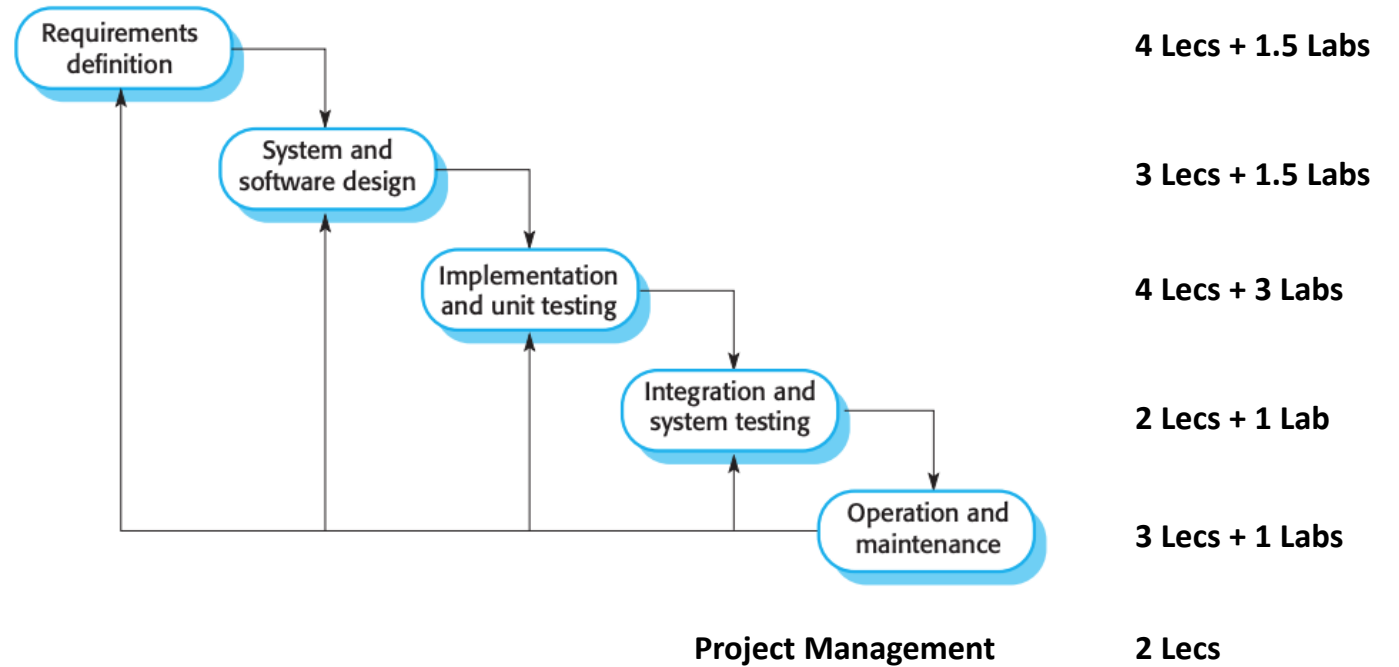
CAL Week	SEM Week	Week Beginning Date	Lectures	Computing Labs
			Tue 14:00 - 16:00	Thu 09:00 - 11:00
24	1	19-Feb	Intro to SE, Using Git as a Team	Git and Markdown
25	2	26-Feb	Req Engineering & Gathering	Persona & Use Cases
26	3	04-Mar	Req Modelling & Validation	UML Diagram I
27	4	11-Mar	Specifications & Prototyping	UML Diagram II
28	5	18-Mar	OO Design, Test Plans & Debugging	CW1 (20%): Reqs & Specs
29	6	25-Mar	No Class	Invited Talk: Journey to GRP with Dave Towey
30	7	01-Apr	Types of Testing, JUnit Testing	JUnit
31	8	08-Apr	Continuous Integration	CW2 (30%): TDD Kickoff
32	9	15-Apr	Agile Methodologies & Software Quality	CW2 (30%): TDD I/O
33	10	22-Apr	Risk Management & Project Planning	PERT/Gantt Chart
34	11	29-Apr	Revision	No Class
35	12	06-May	No Class	
36	Exam	13-May	EXAM Week Begins	

Labs & Assessments

- 50% Exam + 50% Coursework.

- 2-hours Lecture per week on **Tuesday at 2pm to 4pm**.

- 2-hours Lab session per week on **Thursday 9am to 11am**.
  - 8 Skill Labs – to prepare you for subsequent assessment.

- Two coursework: CW1 (20%) and CW2 (30%)

Module Convenors

- Dr. Boon Giin, Lee (Bryan)
    - **Email**: boon-giin.lee@nottingham.edu.cn
    - **Office**: PMB424
    - **Consultation Hours**: Wednesday @ 11:00 to 13:00

- Dr. Tianxiang Cui
    - **Email**: tianxiang.cui@nottingham.edu.cn
    - **Office**: PMB426
    - **Consultation Hours**: Wednesday @ 14:00 to 16:00

What Type of Learning are We Doing in this Module?
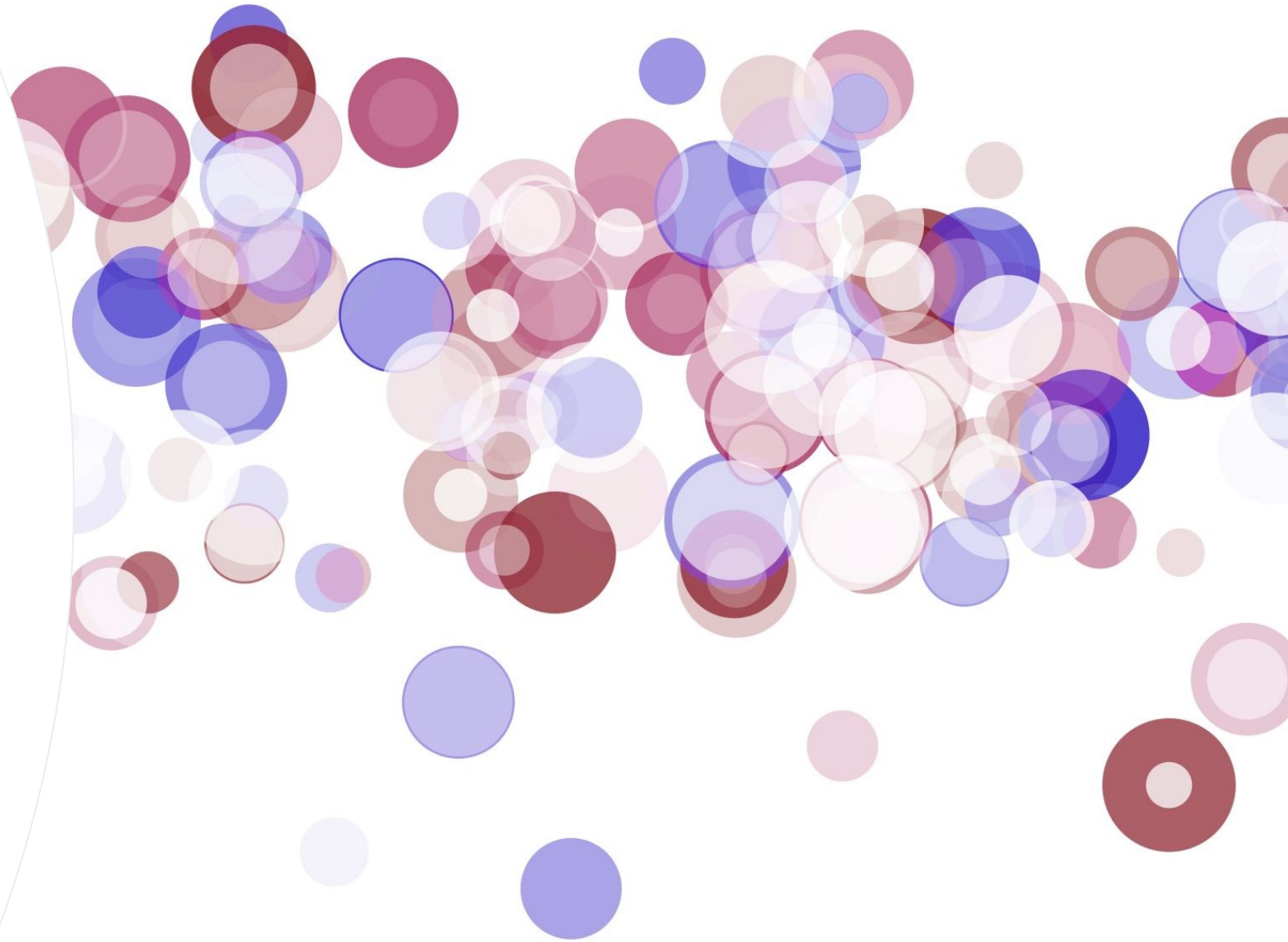
Why are We Working in Group?

Working in Group

- Software Engineering happens in groups.
  - Y3 software engineering group project (GRP) is done in groups.
- You'll learn more by discussing these topics.
  - Because they are all about intellectual comprehension.
- Every lab is groupwork:
  - Peer-assess each others contribution, at end of each lab.
  - And your own contribution.

Working in Group

- Group of FIVE or SIX students.

- You can submit preferred/agreed group application using link in Moodle.
  - By **Wednesday 21st Feb. 2023 at 12 noon**.
  - Final group list will be released before Lab 1.

- Small groups with less than FIVE or SIX students will be merged with other small groups.

Time Allocation

- 20 hours of lectures (including revision).
- 20 additional hours of extra reading (e.g. from the book).
- 20 hours for labs + 10 hours extra for last coursework.
- 10 hours of preparation for labs and peer making.
- 20 hours of revision & taking exam.

Using Moodle

- The schedule of classes is on Moodle.

- The class materials will be on Moodle.

- Q&A Forum for questions.
    - Moodle discussion forum.
    - Q&A via Microsoft Forms.

- Coursework will be released on Moodle.

- We'll be using announcement mechanism in Moodle to provide updates to you.

Labs

- Labs are mandatory.
- There are tasks to prepare you for the coursework.
- Encourage all groupmates to work on the lab exercises together.

Using CSProjects (GitLab)

- You will create a GitLab repository FOR YOUR GROUP.

- You will have a template repository forked in it. (Lab 01)

- All submitted work will be the group's work.

- We will mark based on your group's work.

- Your group repository is basically what you are "managing" for the semester.

THANK YOU