

# Introduction and Process

## Lecture 1

**Ruzanna Chitchyan, Jon Bird, Pete Bennett**

TAs: Mitch Lui, Craig Barnfield, Kira Clements

# Meet the Team

## Lecturers



Ruzanna Chitchyan



Peter Bennett

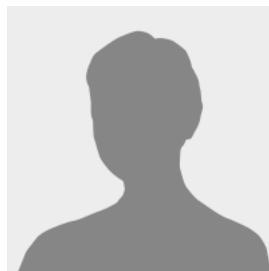


Jon Bird

## Lab Support



Mitch Lui



Craig Barnfield



Kira Clements

# Purpose of Unit

Previous units aim to improve your knowledge as computer scientists

This unit helps package this knowledge as employability skill-sets

This is achieved by targeting the following industry-relevant issues:

- Transferable skills: Management, Collaboration, Communication
- Platform skills: applying your prior programming language knowledge
- Scalability skills: Development “in the large” at industrial scale

# Software Development



But have you heard of ***Software Crisis***

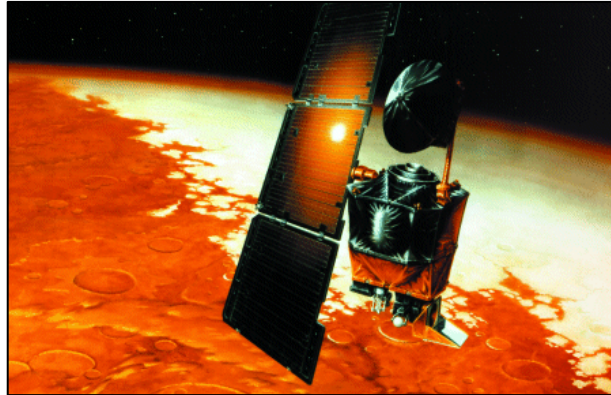
- Software is getting larger and larger
- Software is getting more complex
  - Complex domains (e.g., human behaviours)
  - Systems dependency (e.g., energy and cars)
- Time to market is shorter than ever

# And Projects Fail

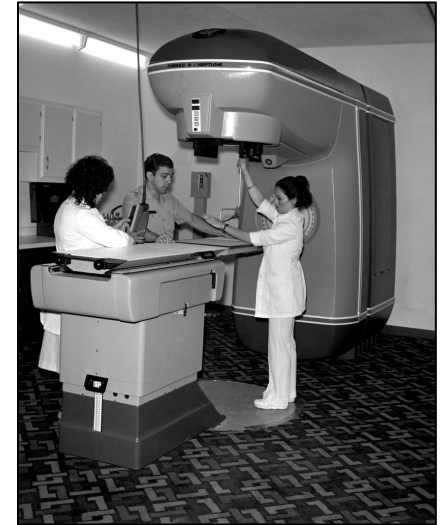
Ariane 5 Flight 501



Mars Climate Orbiter



Therac-25 Radiotherapy



# Why Do Software Projects Fail?

- Bad developers – not the main problem!
- Over budget
  - Poor requirements
  - Over-ambitious requirements
  - Unnecessary requirements
- Contract management
- End-user training
- Operational management



# Software Engineering

- Software Engineering
  - Engineering: **cost-effective solutions** to practical problems by applying **scientific knowledge** to building **things** for **people**
    - Cost-effective solutions: process and project management, contracts...
    - Scientific knowledge: modelling, proofs, testing, simulation, patterns...
    - Things=software
    - People: customers and end users

OK, so how do we do *that* ?



# Collaboration Tools and Techniques

- UML Designs: Diagramming to reach a consensus on design
- GitHub: Sharing of documents for effective collaboration
- Kanban: Task allocation and progress monitoring
- Test-driven development: Stop other people breaking your code !
- Other techniques are encouraged: Feel free to experiment and explore

# Weekly Themes

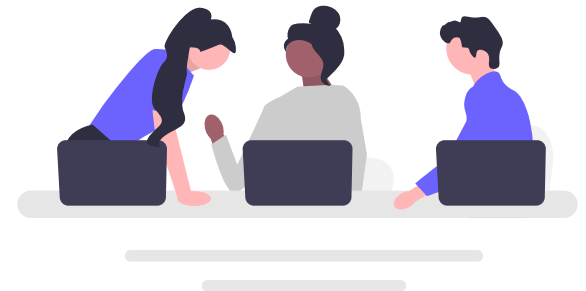
1. Introduction + Overview
2. Agile Development
3. Requirements Engineering
4. Design
5. Implementation
6. [Reading Week]
7. Project Management
8. Human Computer Interaction: Qualitative Analysis
9. Human Computer Interaction: Quantitative Analysis
10. [Easter break: 3 weeks]
11. Advanced Topics in Software Engineering
12. Module Closure

Week	Date	Lecture Monday 11:00-12:00 PHYS BLDG G44 FRANK	Workshop Monday 13:00-15:00 MVB 2.11 PC	Groupwork
1	23/01/22	Introduction and Process <a href="#">[slides]</a> <a href="#">[materials]</a>	Teams, Waterfall Method and Project Brief <a href="#">[slides]</a> [case study] <a href="#">[project brief]</a>	Research games, create list on team repo. Install Processing
2	30/01/22	Agile Methodology and User Centred Design	Intro to Processing, Agile Techniques	Decide on two game ideas
3	06/02/22	Requirements Engineering	Paper Prototyping, Requirements, Ideas Clinic	Collect requirements. Decide on final idea
4	13/02/22	Object Orientated Design	Classes Activity	Add requirements section to report
5	20/02/22	Implementation	Case Study, Spring Prep, Continuous Integration	Develop a working prototype over reading week!
6	28/02/22	READING WEEK	GAMES JAM	
	06/03/22	Project Management	IN CLASS TEST (assessing lectures 1-4)	Define team roles
8	13/03/22	HCI - Qualitative	HCI Qualitative Task	Add qualitative assessment (of your choice) to report
9	20/03/22	HCI - Quantitative	HCI Quantitative Task	Add quantitative assessment (of your choice) to report
	27/03/22	EASTER week 1	SPRINT 1	
	03/04/22	EASTER week 2	SPRINT 2	
	10/04/22	EASTER week 3	SPRINT 3	
10	17/04/22	Software Engineering Extended	IN CLASS TEST (assessing lectures 5-9)	Develop Game
11	24/04/22	Coursework Feedback		Finish Report
12	01/05/22	Bank Holiday Monday (no class)	Demo Day Weds/Thurs (tbc)	Submit Report

- 25% in class test A
- 25% in class test B
- 25% project report
- 25% project presentation

# Coursework

- You get to work on one big project – for the whole term
- Apply all of the knowledge you have gained previously...to a significantly sized, real project
- Work on the project is structured, We won't just leave you to get on with it !
- Each week you will be given a structured activity, and support will be provided to complete that work
- You will be working as part of an integrated development team...



# Groupwork

- We can only hope to tackle a real-world scale application...

....if we work in teams towards the end goal

- Working together is also a key transferable skill
- In any non-trivial industrial project, you'll be working in a team (maybe summer project!)
- It's not easy – which is why it is essential to practice now!

- Teams of 5 (or 6), assigned randomly



# “Tasting Stick” Metaphor

- All of the previously named techniques are large and complex
- We don't expect you to become experts in everything
- The aim is to give you a “taster” of everything
- So you can at least be able to say that you have tried them
- One way to think of this unit is as a “tasting stick”
- Working in groups, individuals are likely to specialise in some aspects



# What's Happening This Week?

This week's theme - ***Introduction and Overview!***

**Lecture:** Grounding of the SE Discipline and Historic View

Introduction to Software Engineering  
Waterfall Life Cycle Model

**Workshop:**

Teams, Waterfall Method and Project Brief

# Software Development Life Cycle Processes:

Which tasks are to be done and in what order?



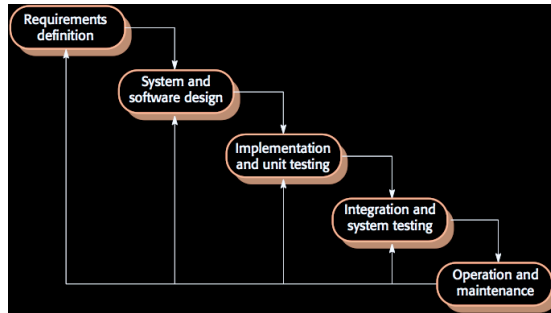
# Software Development Tasks

- Requirements Analysis
- Planning
- Design: high level and detailed
- Development
- Testing
- Deployment
- Operation and Maintenance

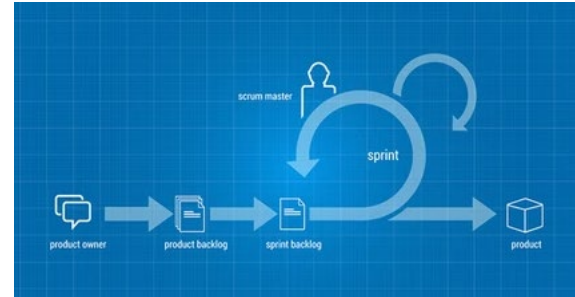
These to-does are combined in various sequences, making up different Software Development Life Cycle Processes

# Software Development Life Cycle Examples

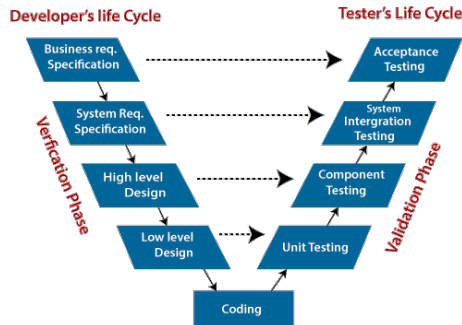
Waterfall



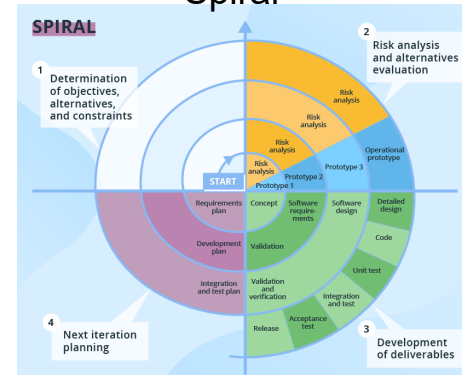
Agile



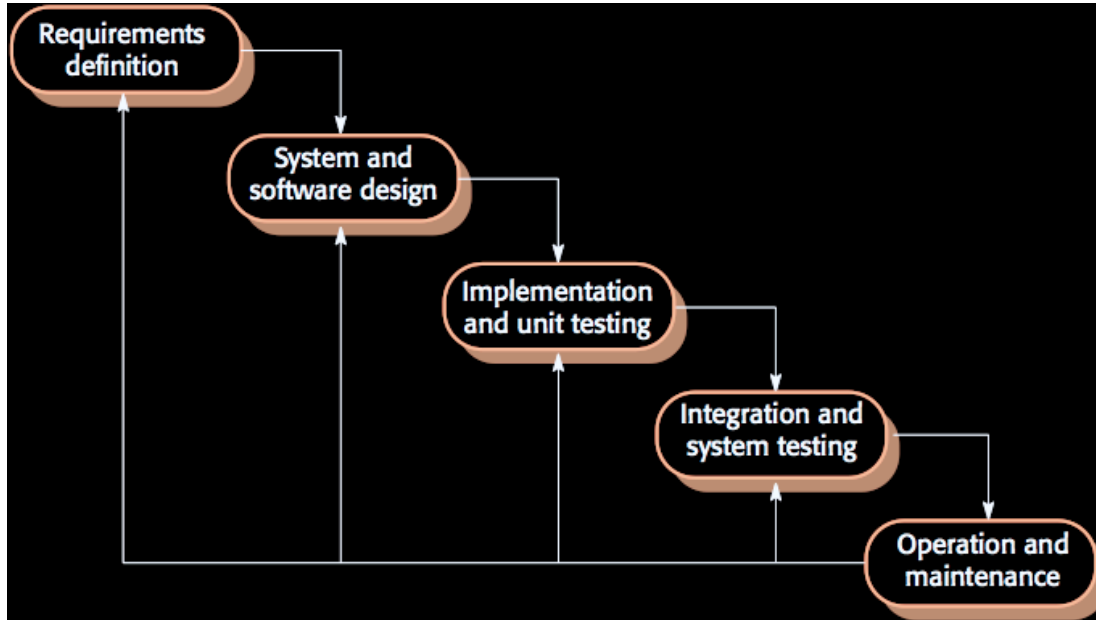
V-Model



Spiral



# Waterfall Software Development Life Cycle



# Requirements: What to Implement?

- What should the system do ?
- What are the needs of the users ?
- What are the needs of the host organisation ?
- What are interoperability needs of existing systems

We need to think about functional elements: what the system should do?

And also non-functional elements: quality properties of system operation, e.g., security, ease of use, response time etc.

# Design: How to Structure Software?

- What objects, databases, servers, services etc. should we create?
- How are these elements structured into a system?

Why to design?

- Helps to decide where to place requirements
- How the parts of the system will be interacting
- Divide up work between team members

# Implementation

Not just programming, but also...

- Parallel working and code sharing
- API partitioning and “firewalling”
- Versioning, integration and config management
- Development environments and automated build
- Automated testing
- Docs and training material

# Verification and Validation

- Verification: check if the software/service complies with a requirements, constraints, and regulations. ("Are you building it right?")
  - Demonstrates that system meets specifications
- Validation: does this meet the needs of the customers/ stakeholders? ("Are you building the right thing?")
  - Demonstrate that system meets user needs
- Can pass verification but fails validation: if built as per the specifications yes the specifications do not address the user's needs.
- Involves checking, reviewing, evaluating & testing

# Waterfall SDLC: Advantages and Disadvantages

- Simple to use and understand
- Every phase has a defined result and process review
- Development stages go one by one
- Perfect for projects where requirements are clear and agreed upon
- Easy to determine the key points in the development cycle
- Easy to classify and prioritize tasks
- The software is ready only after the last stage is over
- High risks and uncertainty
- Misses complexity due to interdependence of decisions
- Not suited for long-term projects where requirements will change
- The progress of the stage is hard to measure while it is still in the development
- Integration is done at the very end, which does not give the option of identifying the problem in advance



# Review

- What is Software Engineering about?
- What is Software Development Life Cycle?
- Can you name any Software Development Life Cycles?
- What are the specific characteristics of Waterfall SDLC?

