Programming in the Large

+ The Project

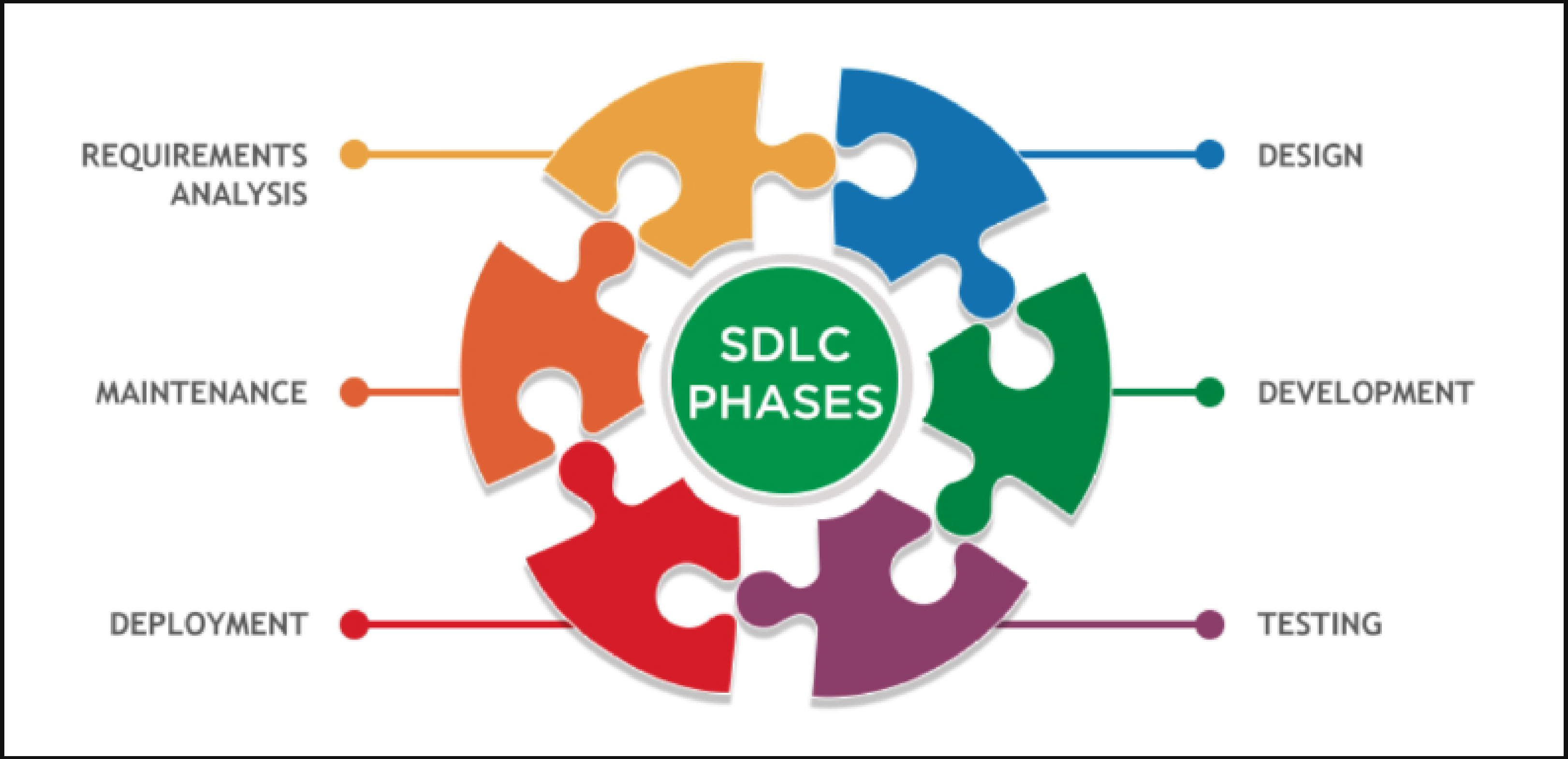
COMP2511

In this lecture

Why?

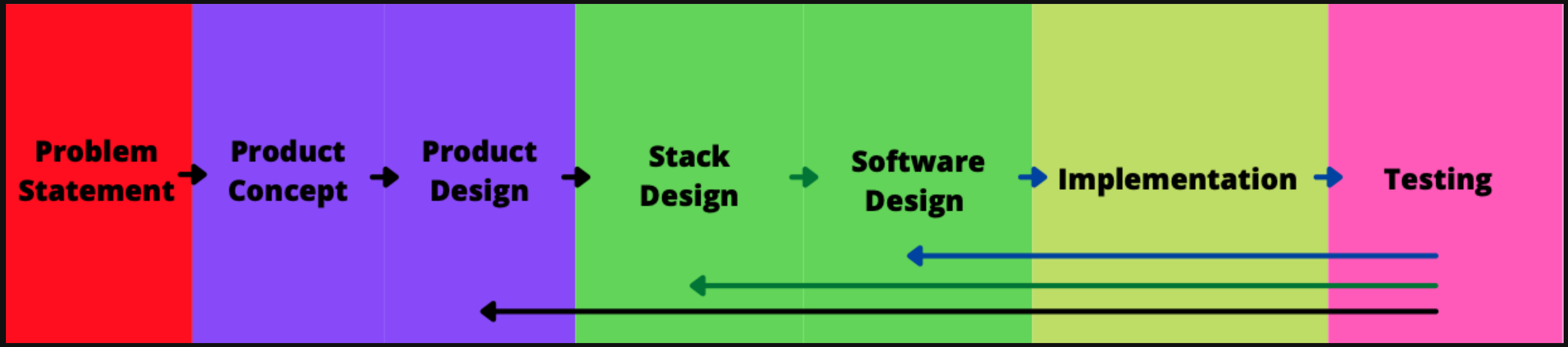
- Understand the broader context of the project and software design in the SDLC
- Compare sequential and iterative design
- Introduce the project
- Discuss industry best-practice approaches for developing software

The Software Development Lifecycle



Where does it all fit in?

- Product concept: How can we solve the problem with software?
- Product design: Epics, user stories, acceptance criteria, UI design
- Stack design: frontend, backend, data layer, integrations
- Software design (OO): Entities, objects, relationships (UML diagram)

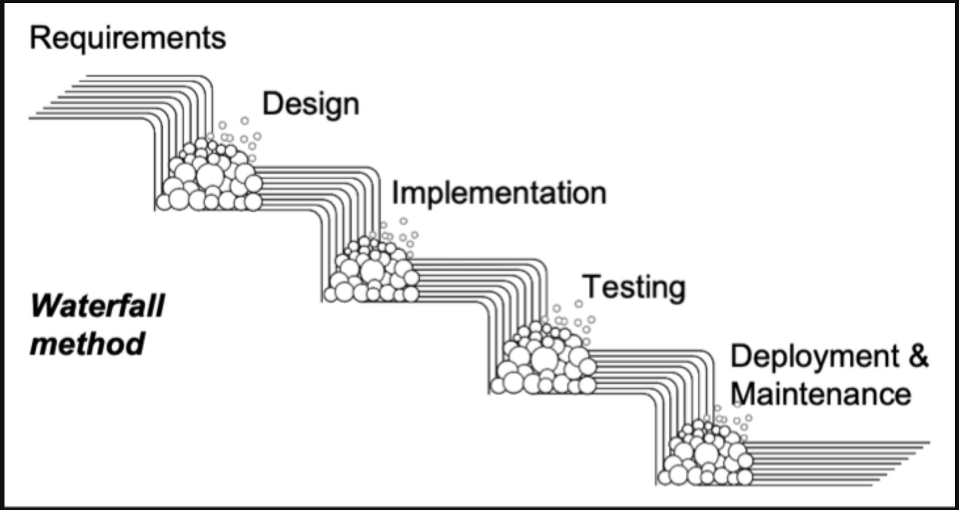


Project Intro

The Big Design Up-Front vs Incremental Design

"Traditional Engineering"

- One step at a time
- Ensure the current step is perfect before moving onto the next one
- A big design up front
- Project can take months-years to complete



Problems with Designing Up-Front

1. The game changes
   - Changing market
   - Changing client expectations
   - Changing technical world
2. Evolution of Requirements
3. Too many unknown unknowns

Unknowns

Two types of unknowns:

1. **Known unknowns** - we know that it exists, but we don't know what it is
2. **Unknown unknowns** - that which we had never even thought to consider

System Complexity

- Systems become exponentially more complex as they grow in size
- Complexity leads to unknown unknowns
- This is why things go wrong
- Learn to deal with unknowns gracefully as they arise

Iterative Design

- Work in sprints, iterations, milestones
- 'Agile' software development
- Many variants - eXtreme programming, Rapid Application Development, Kanban, Scrum
- Design incrementally
    - Adapt to changes in requirements
    - Discover and deal with problems in design as they arise

Problems with Incremental Design

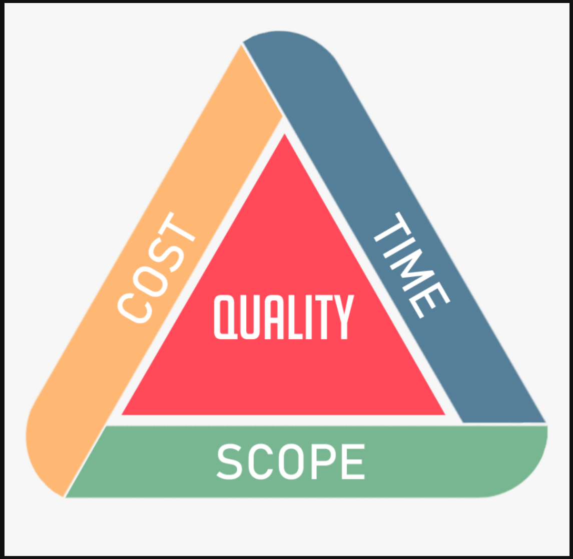
- No clear sense of direction/trajectory
- In poorly designed systems, adaptations to new requirements become smaller-scale 'workarounds' - limit functionality/decrease maintainability
- Tendency to 'make it up as we go along'

A solution?

- **High Level Design**
  - Design a broad overview up-front
  - A framework to begin development
  - Set the trajectory and boundaries of work at the start
- Adapt and change the design during development as needed
- Design up-front a solution that is open for extension, reusable, etc.
- Complete work in **small increments** and **improve iteratively**
- Milestone 1

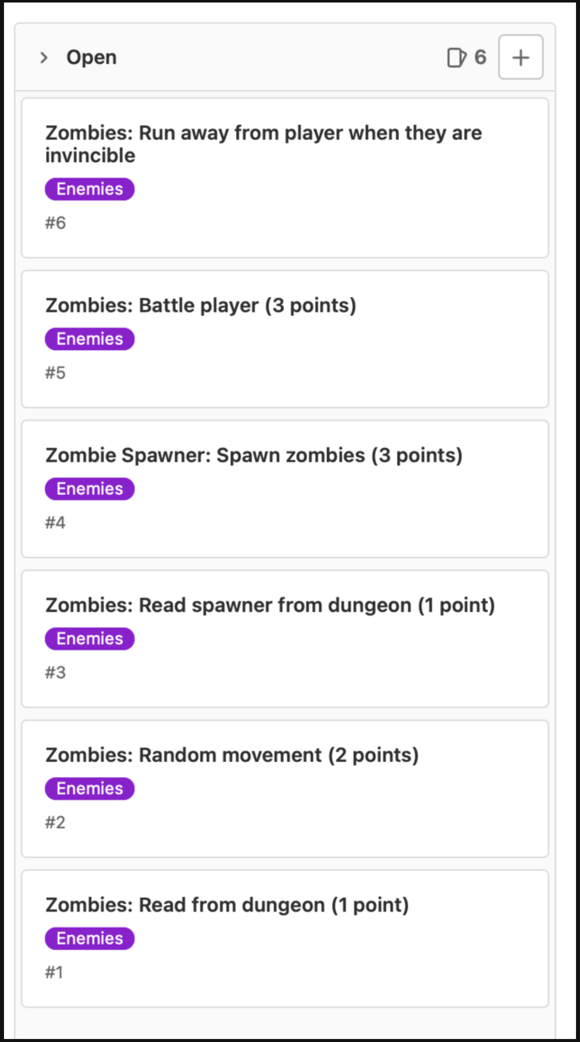
Project Management

- There is always too much work and not enough time and resources in Software Engineering
- **Prioritisation** - what's the most important?
- Work to complete is a **Priority Queue**
- Incremental development
  - Start with the most basic working app
  - **Minimum Viable Product**



Planning

- Breakdown of tasks
  - Use High Level Design to break work down into tickets / tasks
  - Highlight logical dependencies between tickets
  - Create tickets at the smallest possible feature level
- Determine priorities for each ticket (high, medium, low)
- Determine **story points** for each ticket
  - Assign points based on relative complexity



Open 6

Zombies: Run away from player when they are invincible
Enemies
#6

Zombies: Battle player (3 points)
Enemies
#5

Zombie Spawner: Spawn zombies (3 points)
Enemies
#4

Zombies: Read spawner from dungeon (1 point)
Enemies
#3

Zombies: Random movement (2 points)
Enemies
#2

Zombies: Read from dungeon (1 point)
Enemies
#1

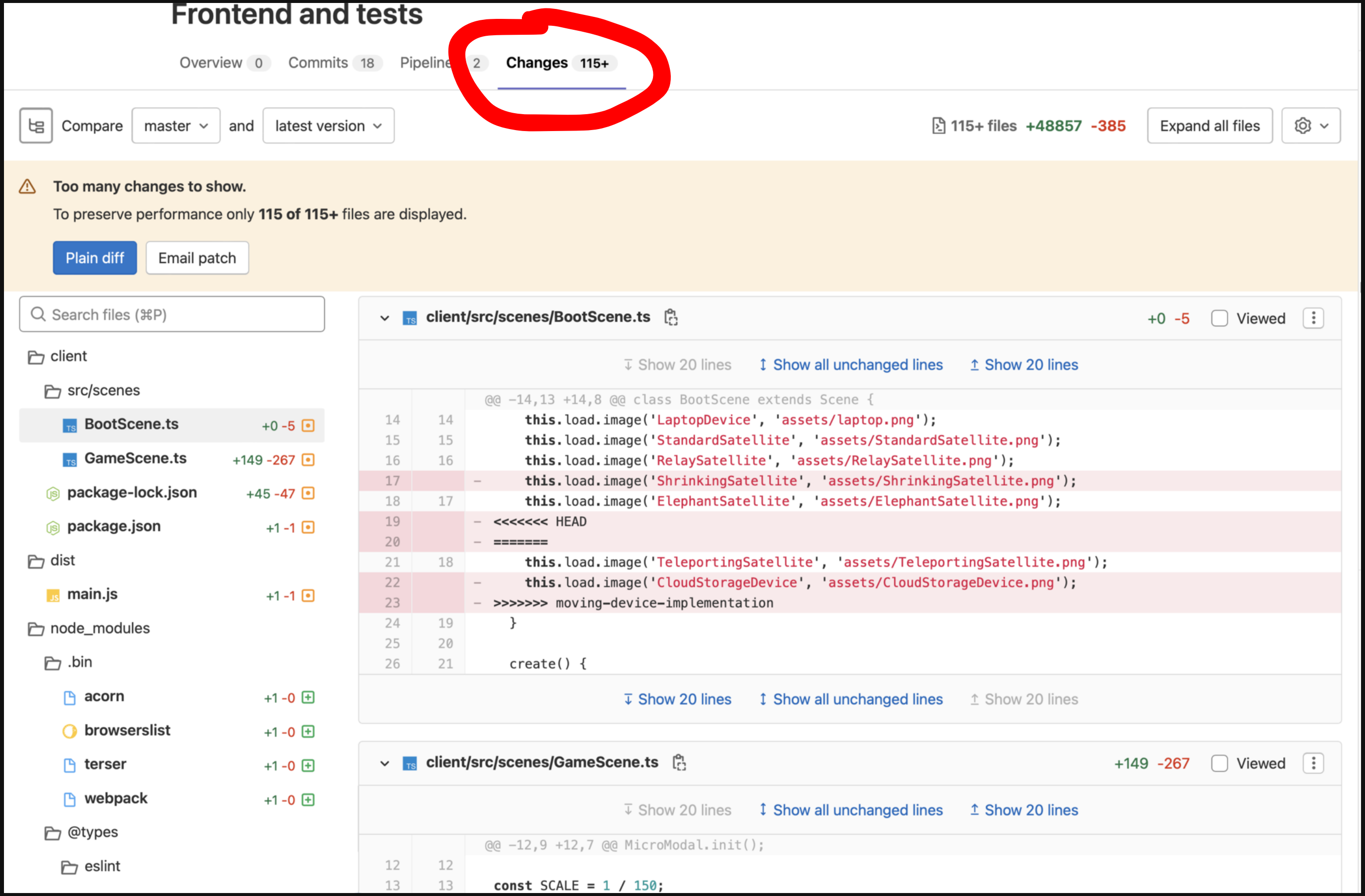
Software Delivery

- We are emphasising how you **deliver your software** rather than how you **manage your project**
- Process to follow for each ticket (See Section 12.3.1 of the spec)
- Slower today, faster forever

Best Practice PRs

- A good pull request / merge request:
  - Touches as few files as it can
  - Has a clear title and description, or links to documentation
- A bad pull request:
  - Contains irrelevant changes
  - Contains accidentally committed files
  - Contains a very large number of changes (should split PR - into sections, or work your way up the dependency tree)

Bad PR Example



Frontend and tests

Overview 0 Commits 18 Pipelines 2 **Changes 115+**

Compare master and latest version — 115+ files +48857 -385 — Expand all files

Too many changes to show.
To preserve performance only **115 of 115+** files are displayed.

Plain diff | Email patch

- client
  - src/scenes
    - BootScene.ts +0 -5
    - GameScene.ts +149 -267
  - package-lock.json +45 -47
  - package.json +1 -1
- dist
  - main.js +1 -1
- node_modules
  - .bin
    - acorn +1 -0
    - browserslist +1 -0
    - terser +1 -0
    - webpack +1 -0
  - @types
    - eslint

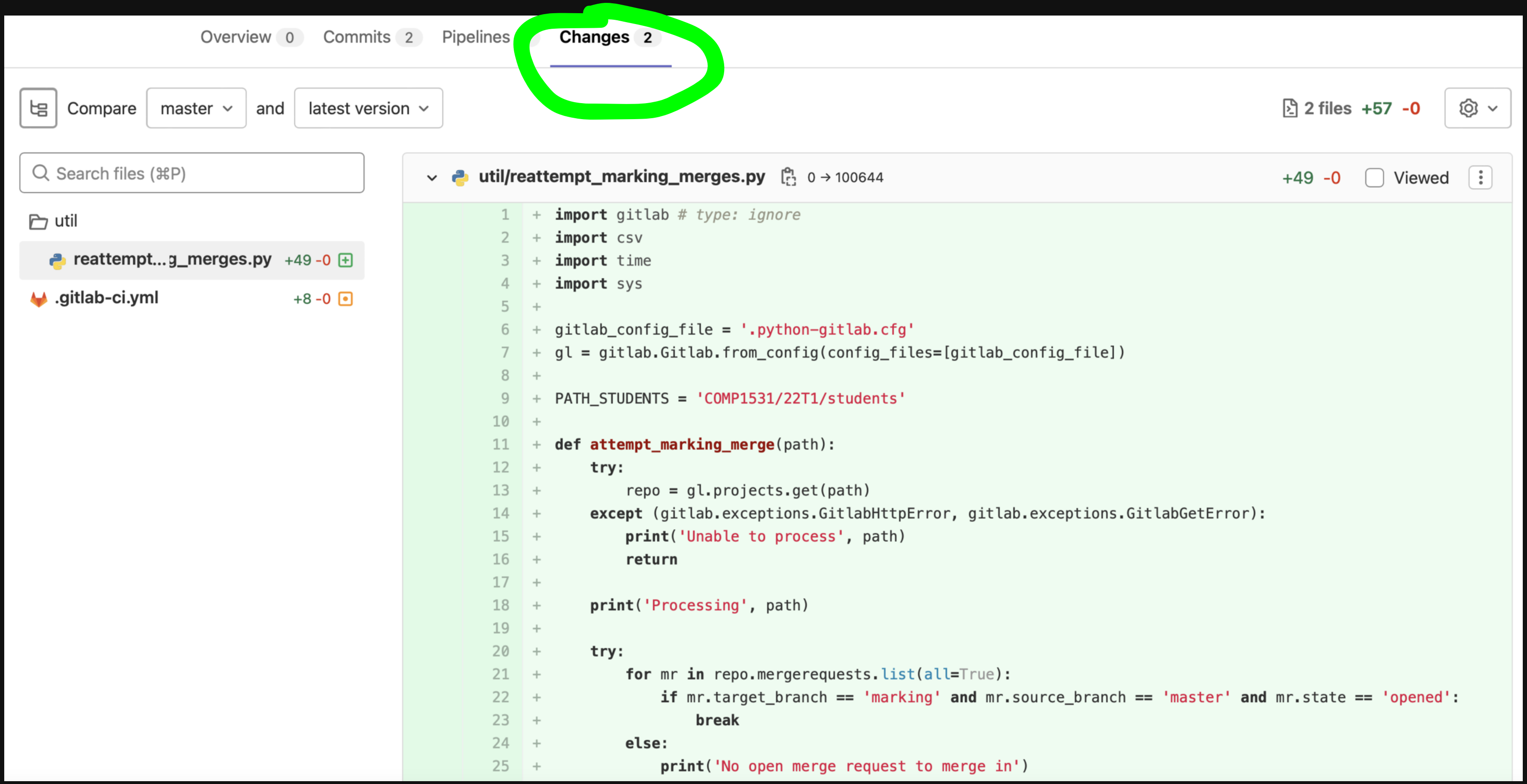
client/src/scenes/BootScene.ts +0 -5

```
@@ -14,13 +14,8 @@ class BootScene extends Scene {
      this.load.image('LaptopDevice', 'assets/laptop.png');
      this.load.image('StandardSatellite', 'assets/StandardSatellite.png');
      this.load.image('RelaySatellite', 'assets/RelaySatellite.png');
-     this.load.image('ShrinkingSatellite', 'assets/ShrinkingSatellite.png');
      this.load.image('ElephantSatellite', 'assets/ElephantSatellite.png');
- <<<<<<< HEAD
- =======
      this.load.image('TeleportingSatellite', 'assets/TeleportingSatellite.png');
-     this.load.image('CloudStorageDevice', 'assets/CloudStorageDevice.png');
- >>>>>>> moving-device-implementation
    }

    create() {
```

client/src/scenes/GameScene.ts +149 -267

```
@@ -12,9 +12,7 @@ MicroModal.init();

const SCALE = 1 / 150;
```

Good PR Example



Overview 0
Commits 2
Pipelines
Changes 2
Compare master and latest version
2 files +57 -0
Search files (⌘P)
util
reattempt...g_merges.py +49 -0
.gitlab-ci.yml +8 -0
util/reattempt_marking_merges.py 0 → 100644
+49 -0
Viewed

```python
+ import gitlab # type: ignore
+ import csv
+ import time
+ import sys
+
+ gitlab_config_file = '.python-gitlab.cfg'
+ gl = gitlab.Gitlab.from_config(config_files=[gitlab_config_file])
+
+ PATH_STUDENTS = 'COMP1531/22T1/students'
+
+ def attempt_marking_merge(path):
+     try:
+         repo = gl.projects.get(path)
+     except (gitlab.exceptions.GitlabHttpError, gitlab.exceptions.GitlabGetError):
+         print('Unable to process', path)
+         return
+
+     print('Processing', path)
+
+     try:
+         for mr in repo.mergerequests.list(all=True):
+             if mr.target_branch == 'marking' and mr.source_branch == 'master' and mr.state == 'opened':
+                 break
+         else:
+             print('No open merge request to merge in')
```

Communication

- Communicating design is difficult, especially when the requirements are complex
- Pair up on tickets - developer, reviewer
- Real teamwork and collaboration - you can't slice up the pie
- Agile practices
  - Standups & "communication saturation"
  - Keep Kanban up to date

Assessment

- Four key areas:
  - Correctness
  - Design
  - Testing (Wednesday lecture)
  - Delivery
- Quality over quantity

Teamwork

- Everyone needs to write code and contribute to documentation (PM, UML, etc.)
- Tutor & project check-ins - mentoring & guidance
- Dealing with teamwork problems:
  - Make an active effort to resolve internally
  - Speak to your tutor
  - Email cs2511@cse.unsw.edu.au
- Individual blogging

Advice

- Please be patient
- Start today