50002 - Software Engineering Design - Lecture 1

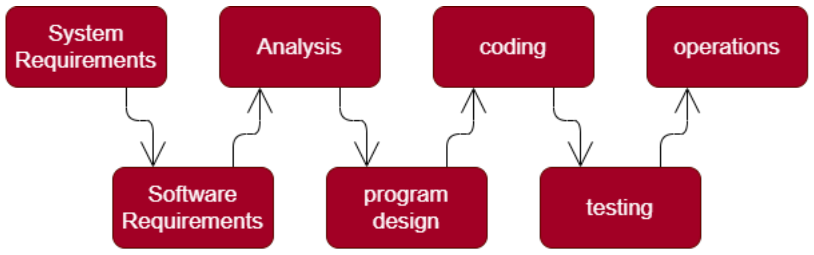
Oliver Killane

08/11/21

Lecture Recording

Lecture recording is available here

Waterfall Development



winston W. Royce's 1970 Paper "Managing Development of Large Software Systems" set out several phases of development through which a project should go. Unfortunately this paper was somewhat badly implemented & resulted in the waterfall model of software development.

Issues With Waterfall Method

- **Only one chance to get it right**
  If an issue is found with the general design/structure of the software at the testing phase, it is difficult to fix as the team that designed has already moved on to another project.

- **Separation**
  As different teams work on different phases, design decisions that could greatly aide testing or operations may not be considered by the design team.

  Phases are considered separately, so little chance to optimise each for the others.

This Module

In this module we will consider 4 parts of design:

- **Behavioural Correctness**
  Automated Testing, Test-Driven Development (TDD) & Mock Objects.

- **Design Patterns**
  Removing Duplication, Increasing Clarity, Improving extensibility, Improving testability.

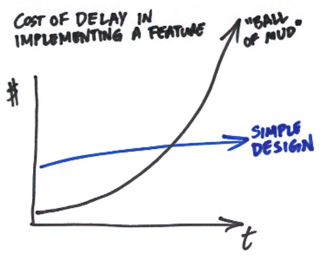
- **Software Architecture**
  Interactive Applications, Data processing applications, Integrating different services.

- **Software Delivery**
  Agile Methods, Continuous Deliuvery, Release and Deployment.

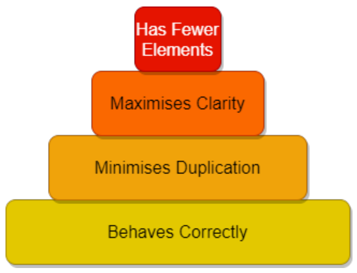
Design

In a blog post for Jetbrains J.B. Rainsberger refers to design as one of the 3 main values of software (Features, Design, Feedback).

By keeping a simple & extensible design it can become increasingly difficult to add new features for users.



Simple Design



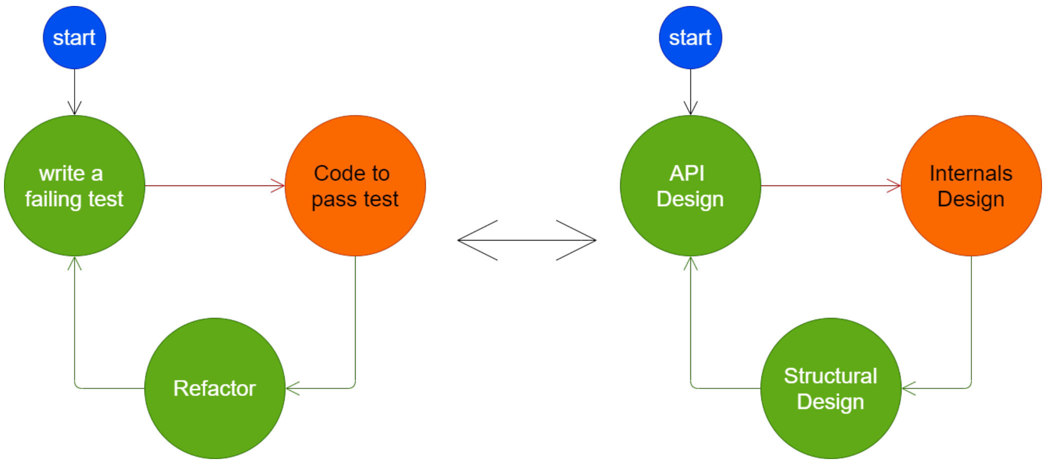
Minimising Duplication

Can be done more mechanically, often using tools provided in modern IDEs (e.g for searching, extracting functions).

Behaves Correctly

Can be done with automated test suites to run programmer written unit & integration tests.

TDD Cycle



API

Application Programming Interface defines the behaviour and usage of software, the interfacing being exposed to other software or services & keeping its internals encapsulated.

Tests are analogous to **API** design as tests make use of the **API** & check its behaviour is consistent with the **API**'s design.