Review of Software Engineering Design

Software Engineering Design
CITS 2220
Lecture 22

SW Process models
- **Software lifecycle**: All activities and work products necessary for the development of a software system, including
  - Requirements
  - Design
  - Implementation
  - Test
- **Software life cycle model**: An abstraction representing a SW life cycle for the purpose of understanding, monitoring or controlling a SW life cycle

V-Model: Distinguishes between Development and Verification Activities

Comparing Process Models
- Managers love waterfall models
  - Nice milestones
  - No need to look back (linear system), one activity at a time
  - Easy to check progress: 90% coded, 20% tested
- Different stakeholders need different abstractions
  - V-Model, Sawtooth and sharks tooth
- Software development is iterative
  - During design problems with requirements are identified
  - During coding, design and requirement problems are found
  - During testing, coding, design & requirement errors are found
  - Spiral Model
The Agile Manifesto
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Agile methods vs plan driven methods
- Better for small teams, experienced developers, low criticality and rapid change
- Worse for junior developers, remote clients/management, highly critical project.

Examples of agile methods
- Pair programming, Scrum, Test driven design

Test Driven Design
- "TDD completely turns traditional development around. When you first go to implement a new feature, the first question that you ask is whether the existing design is the best design possible that enables you to implement that functionality. If so, you proceed via a TFD approach. If not, you refactor it locally to change the portion of the design affected by the new feature, enabling you to add that feature as easily as possible."

Scott Ambler

Course Review: Requirements
- Use Cases and Use Case Diagrams
- Object Identification methods and Class Diagrams
- Boundary, Controller and Entity Objects
- Sequence Diagrams and State Charts
- Types of Documentation: RAD, SDD, Problem Statement
- Function vs Non-functional Requirements

Course Review: Software Design
- Design Constraints and NFR’s
- Rationale and Rationale Driven Design
- Software Architectures:
  - Repositories, Layered, Pipe and Filter
- Design Patterns:
  - Bridge, Adapter, Façade, Abstract Factory,...
- Non-Object Oriented Design Methods

Course Review: Testing
- The anatomy of a test:
  - Purpose, assumptions, evidence, deduction.
- Black box versus white box tests
- Code Inspections and static testing
- Equivalence class testing
- Path testing
- Conformance Testing
- Use Case Testing