ECE 651
Lecture 4: Process and Requirements
2 days of contents + notes
Notes Outline

- Review of facets of software engineerings
  - Requirements definition
  - Design
  - Implementation
  - Verification
  - Delivery/Deployment
  - Maintenance
- Requirements Definition Think Pair Share:
  - What went wrong?
  - Who was responsible?
  - How could things have gone better?
- Specifications
- XY problem
- Requirements Think Pair Share which are good/bad and why?
  - After a successful login, the system shall display a page which will contain a summary of the user's accounts and a navigation menu.

  - The system will be fast enough to support our users during the peak holiday season.

  - The system will have an intuitive user interface.

  - The system will use an AVL tree, keyed by the capacity of each ship to store information about the ships in the user's fleet.

  - A user shall be able to look up a student by name or student ID, and shall be able to view the matching student's current courses and previous grades.

- Characteristics of good requirements:
  - Measureable

  - Descriptive

  - Complete

  - Consistent

- Functional vs Non-functional requirements:

- Asking questions

- User Stories:

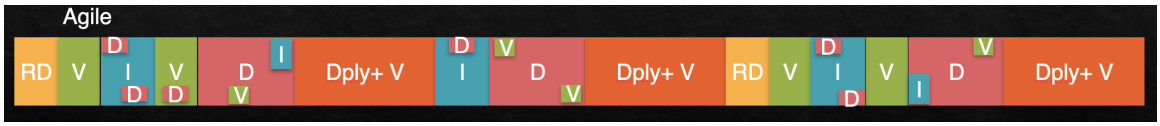
  Professor Xavier runs a school for mutants. He is very busy teaching telekinesis but needs to know which students are misusing their superpowers. Whenever a student is caught misbehaving, the system sends him a text describing the incident, where it happened, and if it is under control. If it is not under control, Prof X calls in the senior X-Men to help him handle it.

- Software Development Process

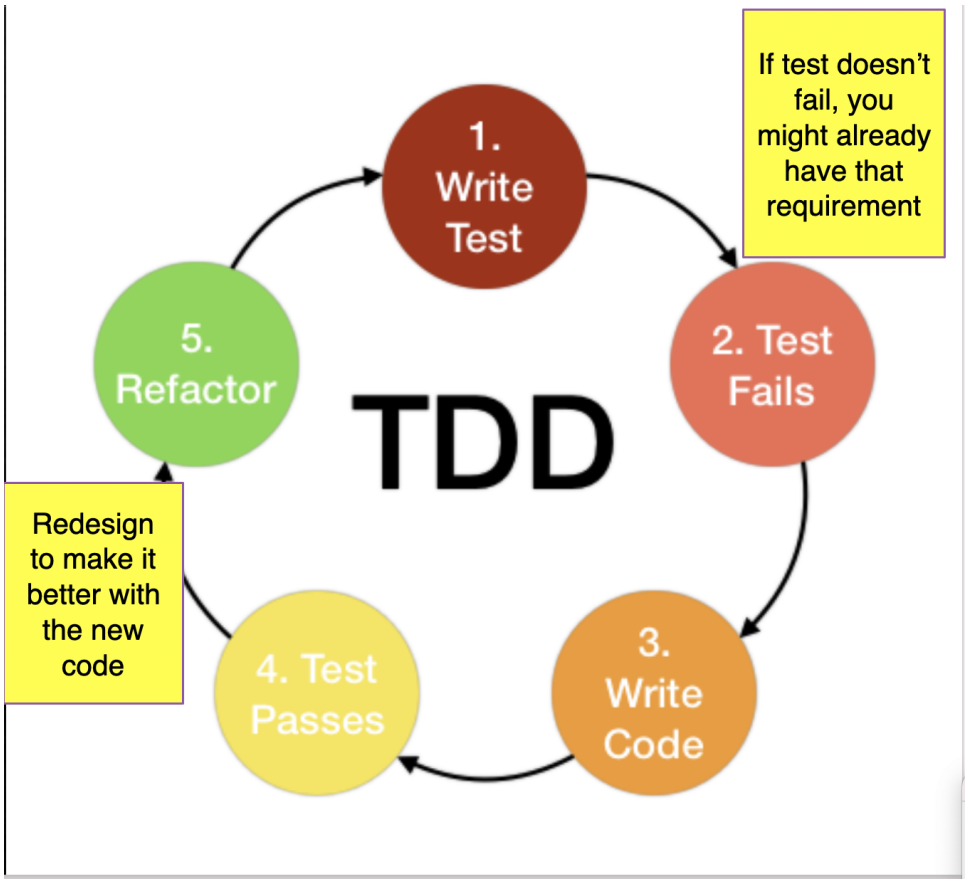
- Waterfall

- Agile Manifesto:
  - Individuals and interactions over tools and processes
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

- Agile Project Management



- Test Driven Development



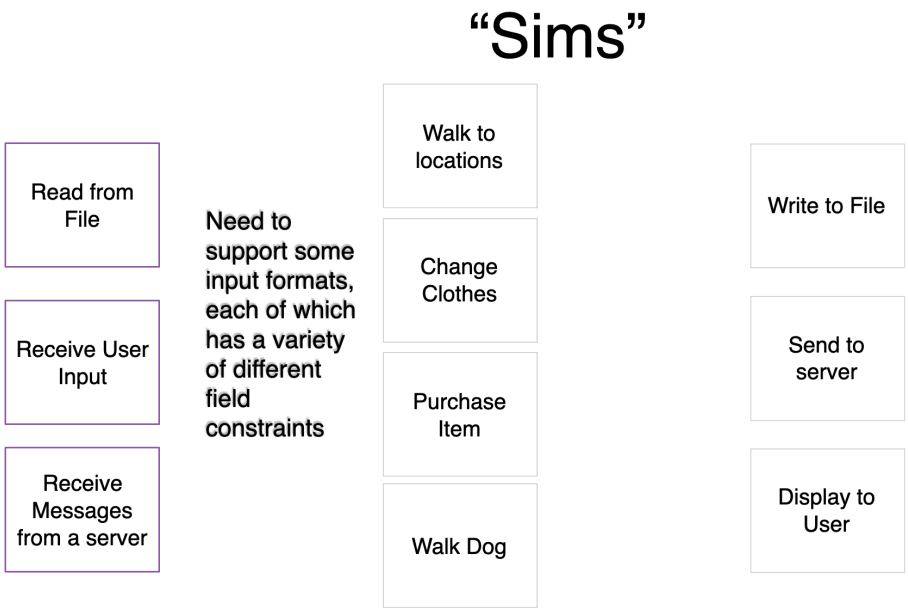
- Continuous Integration/Continuous Deployment

- Scrum

- Agile Myths

- Task Breakdown

- "Sims" example



  - Bad first task choice

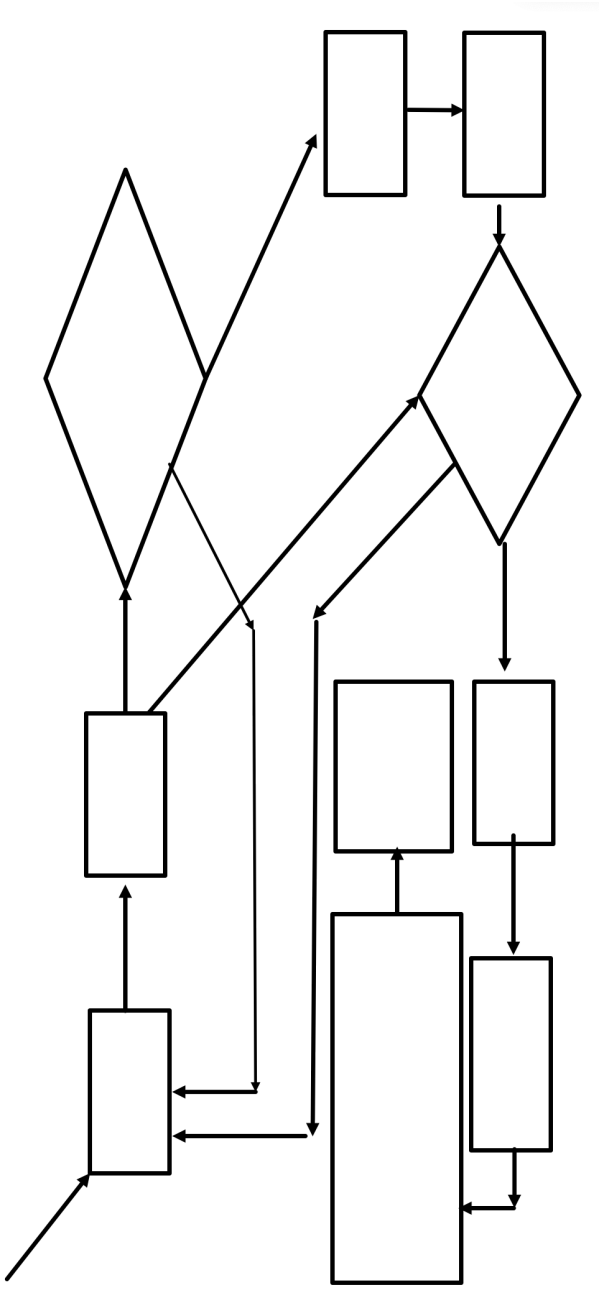
  - State of program

  - Smallest state possible

  - Minimal end-to-end system

– Why is minimal-end-to-end system important?

- Getting started on any software project:



- Applying this to battleship
    1.
    2.
    3.
    4.
    5.
    6.
    7.
    8.
    9.
    10.
- Class, Responsibility, Collaborator (CRC) cards
- Task Estimation
    - Naive approach
    - Developer's and estimation
    - Evidence Based Scheduling
        * https://www.joelonsoftware.com/2007/10/26/evidence-based-scheduling/

- Think/Pair/Share
  - Why not just wait until someone finishes, then reassign tasks?