B07 Nov 8 Lec 1 Notes

Software Development Life Cycle (SDLC)

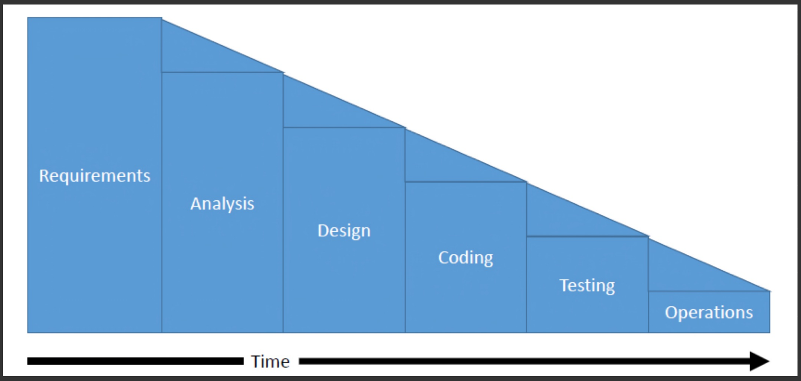
↳ Planning - develop a plan for creating the concept or evolution of the concept.
↳ Analysis - analyze the needs of those using the system. Create detailed requirements.
↳ Design - Translate the detailed requirements into detailed design work.
↳ Implementation - Complete the work of developing and testing the system.
↳ Maintenance - Complete any required maintenance to keep the system running.

Different SDLC implementations

↳ Rigid timeline / Budget (Waterfall)
↳ Risk Adverse (Spiral)
↳ Quality Deliverables / Less Management

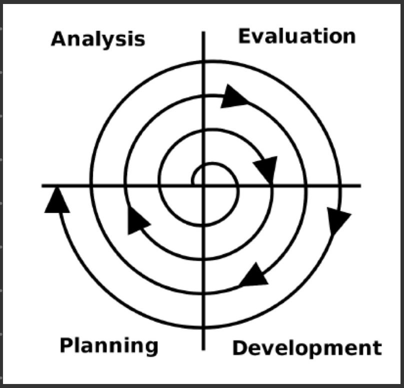
Waterfall

↳ A sequential (non-iterative) model
↳ Involves a large amount of upfront work, in an attempt to reduce the amount of work done in later phases of the project.



Spiral

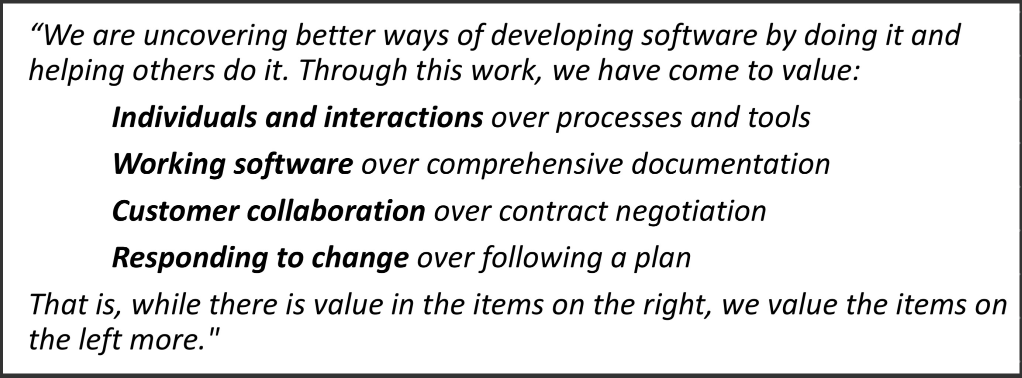
↳ Risk-driven model
↳ More time is spent on a given phase based on the amount of risk that phase poses for the project.



Agile

- Issues with Waterfall
  - Inappropriate when requirements change frequently
  - Time gets squeezed the further into the process you get.
- Agile Methodologies
  - Extreme Programming (XP)
  - Scrum
  - Test-driven Development (TDD)
  - Feature-driven Development (FDD)
  - etc.

Agile Manifesto

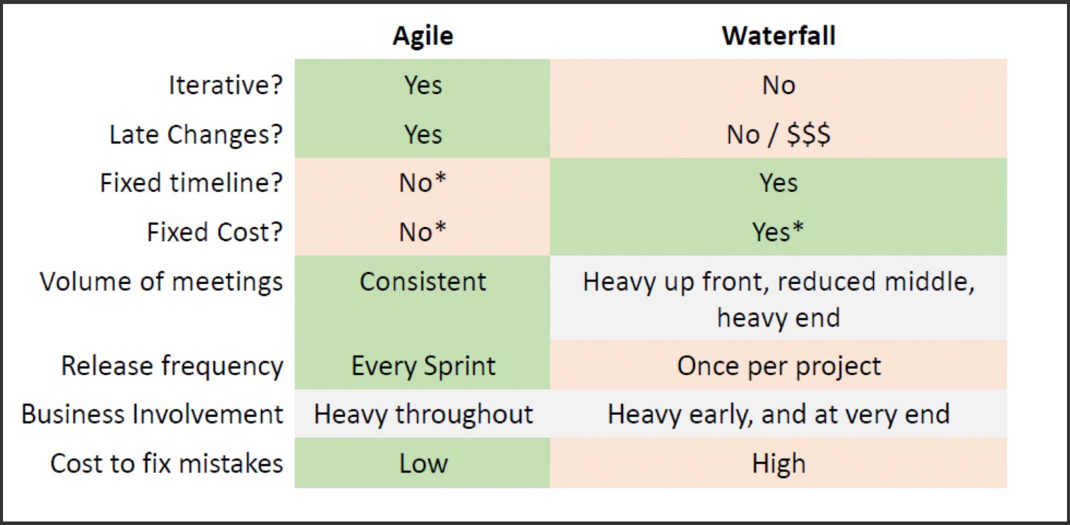


"We are uncovering better ways of developing software by doing it and helping others do it. Through this work, we have come to value:

> ***Individuals and interactions*** *over processes and tools*
>
> ***Working software*** *over comprehensive documentation*
>
> ***Customer collaboration*** *over contract negotiation*
>
> ***Responding to change*** *over following a plan*

That is, while there is value in the items on the right, we value the items on the left more."

Agile vs Waterfall



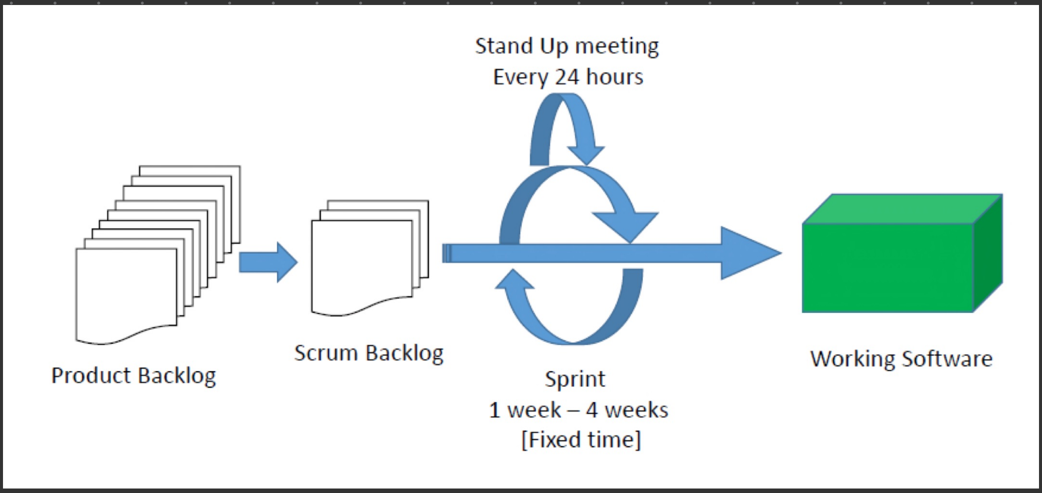
	Agile	**Waterfall**
Iterative?	Yes	No
Late Changes?	Yes	No / $$$
Fixed timeline?	No*	Yes
Fixed Cost?	No*	Yes*
Volume of meetings	Consistent	Heavy up front, reduced middle, heavy end
Release frequency	Every Sprint	Once per project
Business Involvement	Heavy throughout	Heavy early, and at very end
Cost to fix mistakes	Low	High

eXtreme Programming (XP)

- One of the most rigorous forms of Agile.
- Involves building a series of feedback loops, which are used to help guide when change can occur and allow for changes to be quickly integrated into the plan for development.
- Built on the idea that you can reduce the cost of developing software, and build better software, by having goals.
- XP requires that everything that can be unit tested is unit tested, that everyone works in pairs, and that these pairs change frequently.

Scrum

- Currently most widely used methodogies



Scrum - Roles

- Product Owner
  - Responsible for delivering requirements and accepting demos
- Scrum Master
  - Responsible for removing impediments (schedule meetings)
- Team member
  - No one has a fixed role
  - Everyone takes on tasks.

Scrum - Sprint

- The sprint is a fixed time to deliver a working set of features, that are reviewed in a demonstration of the product owner.
- Tasks in scrum are broken into "User Stories"

Scrum - User Stories

- User stories are similar to requirements. Format:
  - As a {Actor/Object} I want to {Action} so that {Result}.

Scrum - Planning Poker

- We do not assign time to tasks, but assign arbitrary points. This is a form of estimation that helps gauge how much work something will take to complete.
- Planning Poker takes a set of pre-determined numbers and gets you to estimate how much work something will be relative to a known task.
- After discussing the Story at hand, everyone selects a card. Then, the cards are turned over simultaneously. Usually time is given for those who had the lowest or highest numbers to state their case.
- The process is required until everyone ends up at the same number.

Scrum - Planning Session

- Planning sessions happen at the start of each sprint.
- The team decides the work load.

Scrum - The Standup Meeting

↳ Happens every single day

Scrum - Working Agreement

↳ A series of statements that everyone on the team agrees to about how the team will work.

Scrum - Definition of Done

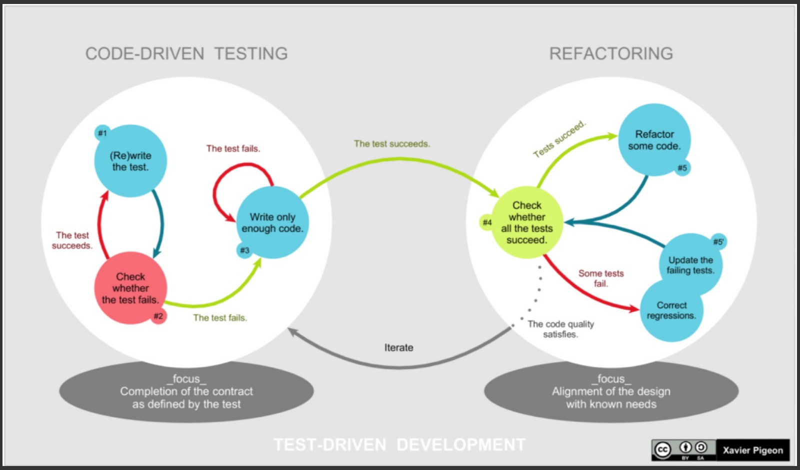
↳ A formal agreement

Test Driven Development (TDD)

↳ TDD is a way to develop software that revolves around writing test cases.

↳ Write the unit tests needed to be passed for a feature to be considered working. Then code to the tests.

↳ Once working, you review and refactor.



Feature Driven Development

↳ Based on the idea of building a focused model for the project, and then iterating on the features needed:

↳ Splits development into 5 major pieces:

(i) Develop overall model

(ii) Build feature list

(iii) Plan by feature

(iv) Design by feature

(v) Build by feature

Android

Android

- ↳ A platform consisting of three components
  - ↳ An operating system
  - ↳ A framework for developing applications.
  - ↳ Devices that run the android OS and the applications created for it.
- ↳ Android SDK
  - ↳ A collection of libraries and tools needed for developing android applications.

Android App Basics

- ↳ An android app is a collection of screens, and each screen is comprised of a layout and an activity.
  - ↳ Layout: describes the appearance of a screen (written in XML).
  - ↳ Activity: responsible for managing user interaction with the screen (written in Java).
- ↳ An activity transitions between different states during its lifecycle:
  - ↳ Created
  - ↳ Started
  - ↳ Resumed
  - ↳ Paused
  - ↳ Stopped
  - ↳ Destroyed