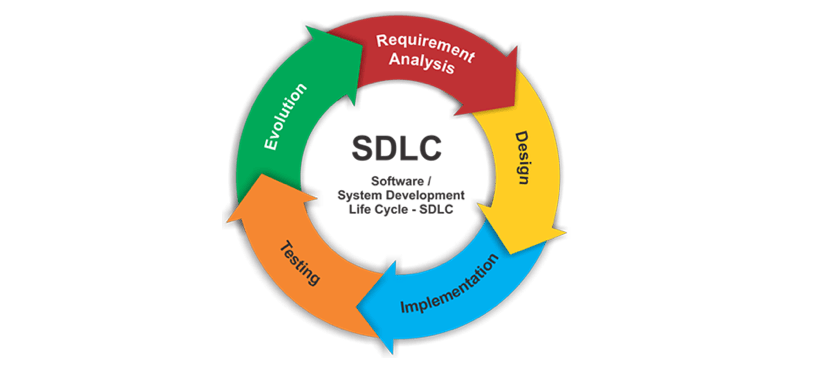
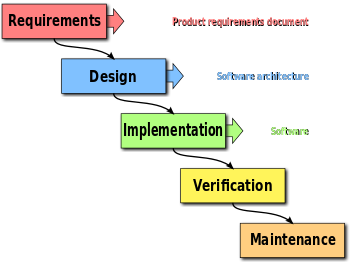
SDLC

Software Development Lifecycle

* 
* Requirement Analysis/ Assessing the current situation
  + Decide what have you to work on.
  + What are your priorities.
  + What is the current state of the application.
* Design
  + Work out how to achieve your goals.
  + Set up of your application.
* Implementation.
  + Coding the application
* Testing
  + I feel this blurs with implementation.
* Evolution/monitoring/feedback
  + Find out what bugs people are reporting.
  + See any pain points or requested features

waterfall approach



* Old School Approach
* Follow a rigid one way path to completing the application.
  + Requirements document.
    - Labels everything needed in the application.
      * Features
      * Technologies to be used
      * How it is going to be tested
      * Specific dates on features.
  + Design
  + Develop
  + Test
  + Deploy/maintenance
* You never skip a step.
* You never go back a step.
* Software engineering is not your typical engineering.
  + It can be very easy to change past decisions.
    - A building cannot just redo the concrete foundation.
  + It is possible to work on many different parts at once.
    - You can’t install windows into a building until you have walls.
  + Software can be more flexible than other types of engineers.
  + Software is constantly changing.
    - You dependency you are using may be discovered to be a security risk.
    - The cost of a certain piece of software got a lot more expensive.

Agile

* It is a paradigm of thinking.
* Not a specific implementation or set of rules.
  + More like guidelines
* The core principle is that you should be agile and be willing to adapt and change. Do not stick to a rigid structure.
  + Tenets
    - Individuals over processes
    - Demos of documentation.
      * It is much better to have a non-complete demo than a word document describing what was done.
    - Customer collaboration over Contracts
      * Rather than a requirements document it is better to speak with the client every few days about progress and clarification on features.
    - Adapting to change over following plan.
      * Software changes and you should learn from the project to do it better.

Scrum

An implementation of Agile

* Roles
  + Stakeholder(s)
    - Whoever hired you to make the application.
    - For whom you are building the application for.
  + Project owner
    - They represent the stakeholder and ultimately responsible for the application.
    - A person you are likely to go to seek clarification on issues.
  + Scrum Master (lead)
    - The most senior developer on a team.
    - He will be in charge of giving programmers tasks.
      * Part senior developer
      * Part cheerleader
      * Part Manager
    - Their ultimate goal is to remove any barriers and make development as efficient as possible.
  + Scrum Members (Scrumlings)
    - Developers on a team.
* Sprint
  + 2-3 week iteration on a project
  + Applications should be incrementally built by these sprints
* Scrum Ceremonies
  + Day 1
    - User stories and story pointing
    - Assigning user stories.
  + Everyday
    - Team standup
    - Every person/representative of each time give a quick update on their progress and asks for help if necessary.
  + Day 14
    - Sprint Retrospective
      * Analyze what went good and bad.
      * Go over ways to improve for the next sprint.