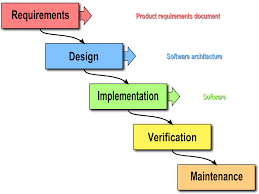
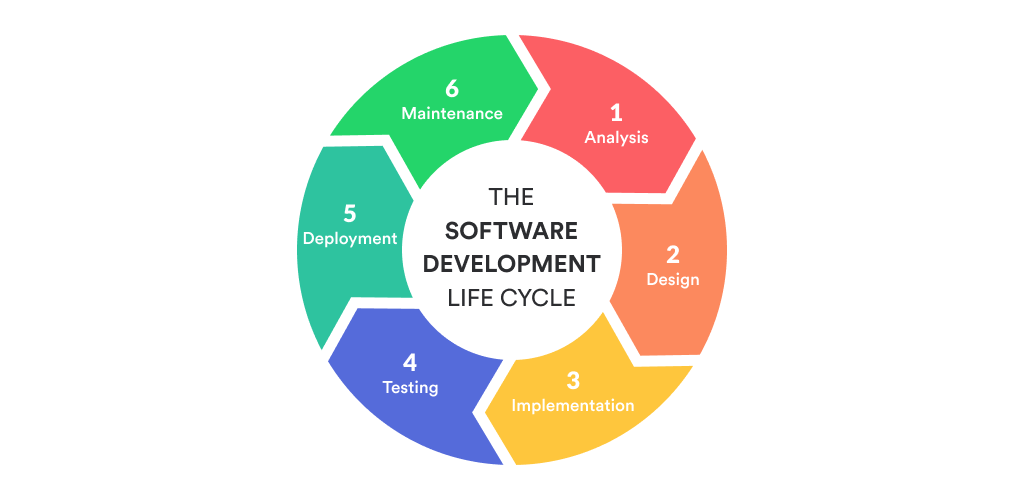
SDLC (Software Development Life Cycle)

Waterfall Lifecycle (Old School)



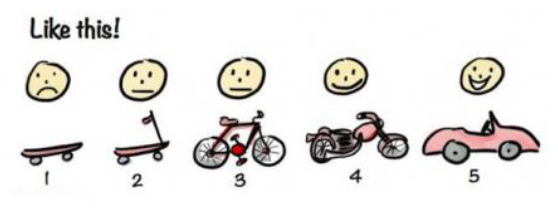
* Very regimented step by step approach to building software
* You do not move to the next level below until the current level is completely finished
* cons
  + Does not account for lessons learned while designing
  + Leads to overdesign and too much emphasis of documentation
  + Rigidly adhering to the design documentation can slow progress
  + Emphasis is on fulfilling a contract to the product owner (person/company) who wants the software
  + Leads to hierarchical software teams. Leads to inefficiency and inability to reallocate people to tasks that best fitted for.

Agile



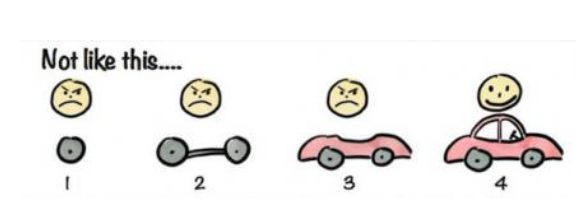
* Agile is a mindset
  + Implementations
  + Scrum
  + Kanban
* Tenets of the Agile philosophy
  + **Individuals and interactions over processes and tools**
    - Always be communicating!!!!
    - DO NOT put an emphasis on procedure and “proper means of communication”
    - Always talk to your product owner about changes and clarifications
    - Always talk to your senior and fellow developers when you have questions
  + **Working software over comprehensive documentation**
    - Always be striving to have a demo
    - Features should be added incrementally
    - DO NOT have a situation where 90% of the work is completed but you have nothing to demo
    - DO have 40% complete and have a demo showing off the 40% that works
    - DO NOT overly focus on documentation\*
      * PLEASE DO NOT READ THIS AS AN EXCUSE TO NOT HAVE DOCUMENTATION
    - Focus on code quality/readable and demos
  + **Customer collaboration over contract negotiation**
    - Always be asking the product owner for clarification whenever necessary
  + **Responding to change over following a plan**
    - Plans Are Worthless, But Planning Is Everything
    - As you code you will learn lessons about how best to design the application
    - DO NOT rigidly adhere to your designs

AGILE



Often an intermediary working version of the demo satisfies the client. And there is no need to overdevelop

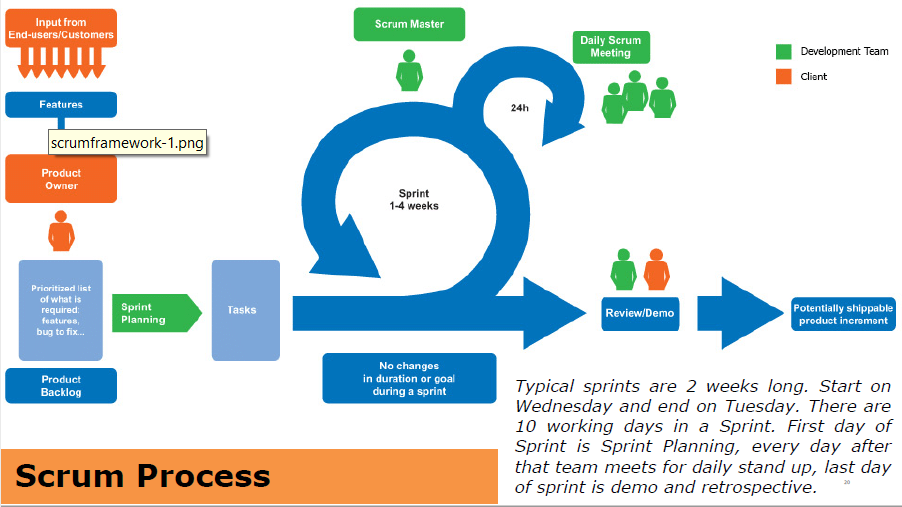
Waterfall



Scrum (An implementation of Agile)

Roles in a Scrum implementation of Agile

* Product Owner
  + Person ultimately responsible for presenting the end client with a working product
  + Most face to face time with the stakeholders
  + Act as a go between the stakeholders and team members
* Scrum Master
  + Head of the team in charge of building the software
  + Jack of all trades
  + Make the team work as efficiently as possible
    - Allocating people to the tasks they are best suited for
    - Remove any blockers regarding development
      * I.e. buy a software license in case it is needed to develop the application
      * Expand your cloud resources
  + Easily approachable
  + Cheerleader
    - Keep people motivated and engaged in the project
  + Mentor
* Users
  + People who will be using the application
  + I.e Accounting software would have accountants as users
* Team Members (scrumlings\*\* not a real term)
  + Developers/Testers
  + Business Analysts
    - People who help guide development usually through user stories and communicating with the client
* Stakeholders
  + Who you are building the software for
  + E.g. Department of Defense, Joe’s Pizza Chain



**Sprint**

* An incremental unit of work
* Typically last about 2 weeks
* At the end of each sprint you should have something to demo

**Sprint Phases**

* Start (day 1)
  + Initial planning
    - Create user stories
    - And assign points to them
  + Create a sprint backlog
    - Tasks to be completed this sprint
    - Asana todo tasks
  + Assign members to work on those tasks
  + Assess how much work can be accomplished in a sprint
    - Reasonable goals
* Throughout (day 2 – day 14)
  + Daily standup
    - Everyone will communicate to the entire team how they are progressing on their current task
      * Ask for additional help
      * Talk about any problems you are having
      * Mention if you have extra capacity because you completed the assignment
      * Task should be reassigned because you cannot do it
  + Development/testing
    - Test as you develop
    - TDD where you write your tests first and then code to pass those tests
* End (Last Day)
  + Sprint retrospective
    - What got accomplished and what did not
    - Sprint report/metrics
      * Burndown charts
      * Velocity
    - Lesson learned
      * Are there designs we should change etc…
      * Should we use a different technology etc…
    - Prepare a demo
  + You will use what you have learned and accomplished in the retrospective to plan your next Sprint

User Story

* Short descriptions of a feature from the perspective of a user
  + As a <user>
  + I want to <some kind of activity>
  + So that <some reason>
* User Stories are often only a sentence or two
* Walmart Online User story
  + As a unregistered user
  + I want to create an account
  + So that I can make online purchases
* P0 user story
  + As a registered user
  + I want to deposit money into an account
  + So that I can use the money whenever Necessary
* P0 user story
  + As a registered user
  + I want to name my accounts
  + So that I can manage my money more easily

Agile Story Board or product grooming

* At the start of a sprint the product owner and the Scrum master (some other senior developers)
* Take the pending user stories and see which ones are the most critical and pressing to implement
* They will break user stories into tasks to be handed out to individuals and teams

UML

* Unified Modeling Language
* Visual documentation of your application
  + Quickly bring people up to speed
* Two main types of UML diagrams
  + Structural
    - Technical architecture of your application
    - Classes, interfaces and how they interact with each
    - Can include things like libraries
      * Class diagram
        + Passive classes are classes which mostly store information
        + Active classes usually perform operations and modify other objects
  + Behavioral
    - How users interact with the software
    - How users interact with each
      * Use case diagram