CSCB07 - Software Design

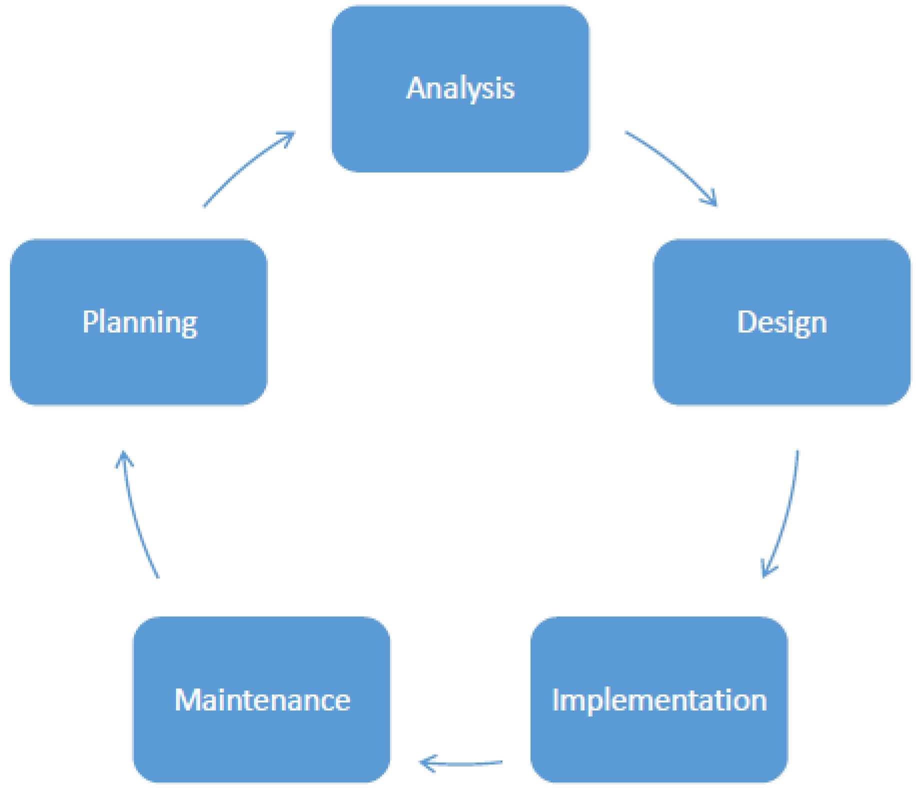
Software Development Life Cycle

Based on the slides by Joe Bettridge

Summer 2022

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

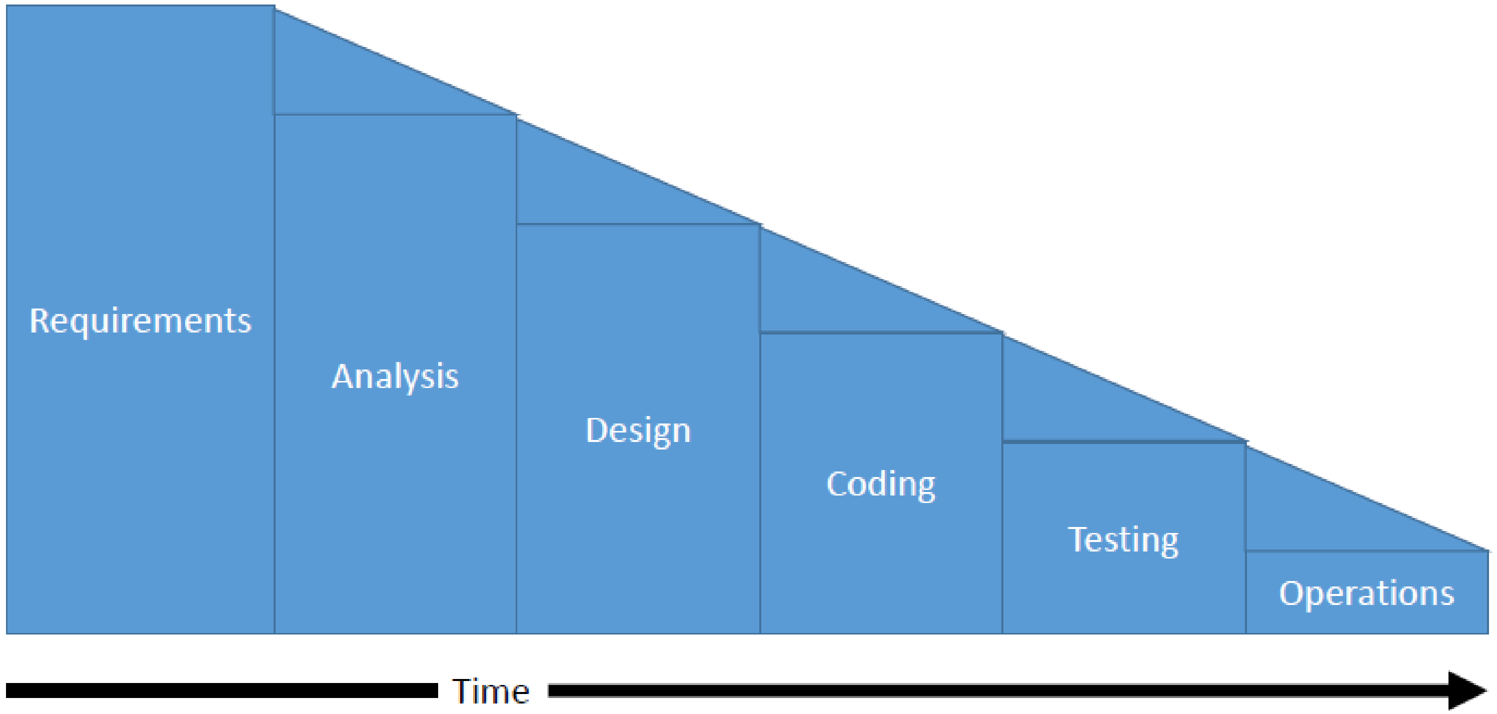


Different SDLC implementations

- Rigid timeline / budget (Waterfall)
- Risk Adverse (Spiral)
- Quality Deliverables / Less management (Agile)

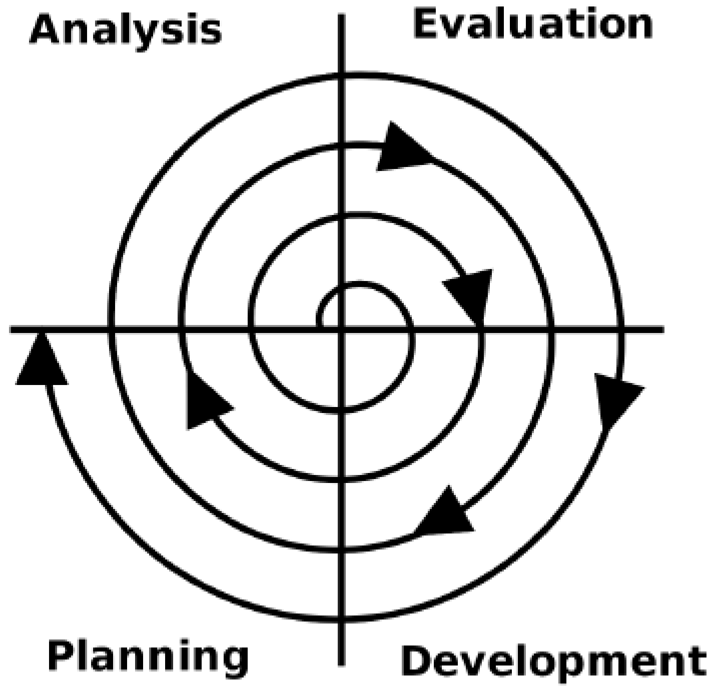
Waterfall

- 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."

Beck, K., et al. (2001) Manifesto for agile software development.

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)

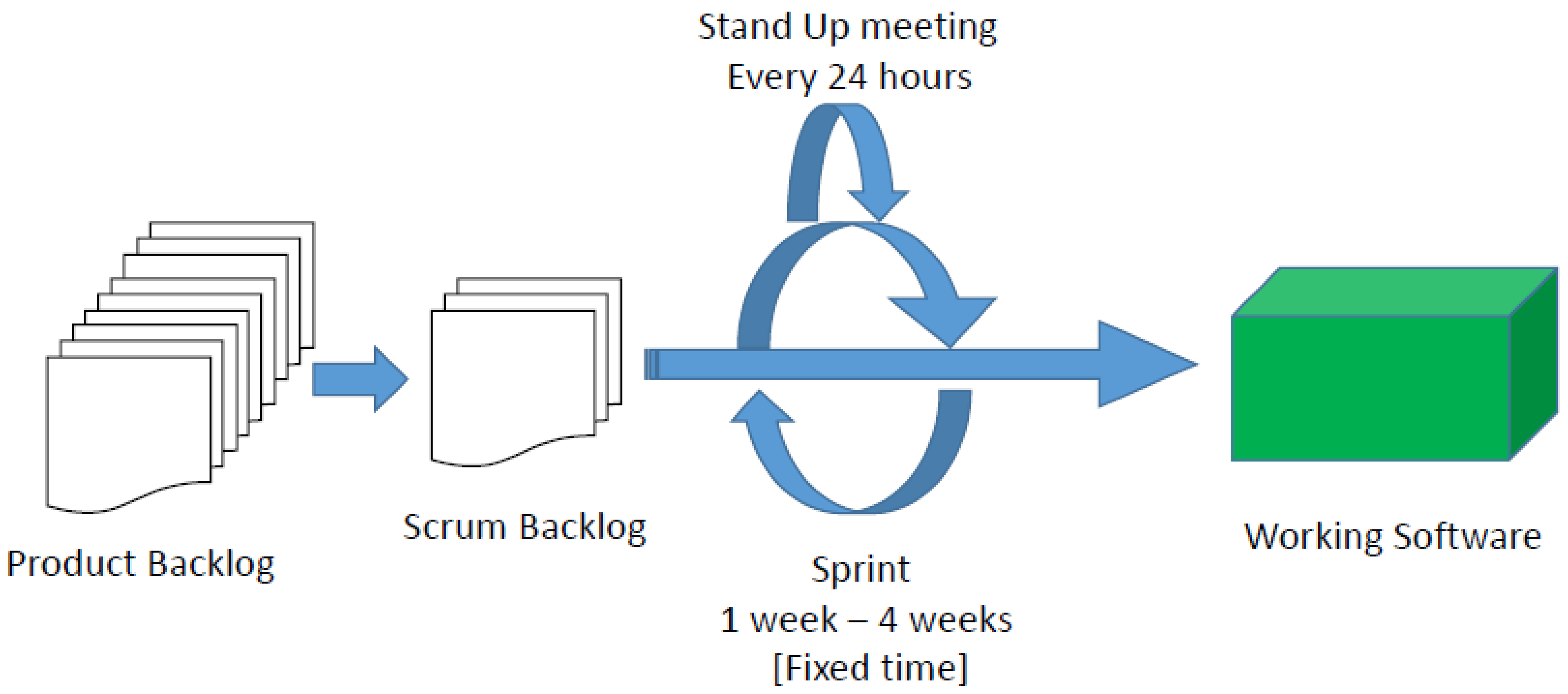
- One of the most rigorous forms of Agile
- Involves building a series of feedback loops, which are used to help guide when change can occur and allow for changes to be quickly integrated into the plan for development
- Built on the idea that you can reduce the cost of developing software, and build 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.

Code Review

- Although pair programming has gone out of vogue along with XP, it is important to note a practice that has become common place that was born from this idea –Code Review.
- A code review is a session in which you **must sit down with another developer** from the team and walk them through your implementation line-by-line in order to get advice and feedback.
- This process has been shown to lead to better code, through finding bugs earlier, and an increased amount of collaboration on difficult concepts.

Scrum

- Scrum is currently one of the most widely used methodologies of software development



Scrum - Roles

- Product Owner
  - ➢ Responsible for delivering requirements and accepting demos
  - ➢ Involved in planning session
- Scrum Master
  - ➢ Responsible for removing impediments
- Team members
  - ➢ No one has a fixed role other than the scrum master and product owner
  - ➢ Everyone takes on tasks, and completes them based on what they are most comfortable with

Scrum - Sprint

- The sprint is a fixed time to deliver a working set of features, that are reviewed in a demonstration to the product owner
- Tasks in Scrum are broken into “User Stories”
- In a sprint, a team agrees at the beginning to take on a certain number of user stories
- Sprints are usually between 1 and 4 weeks in length
- At the end of each sprint, teams hold a “retrospective” which is a meeting where the past sprint is discussed, and chances for improvement for the next sprint are raised

Scrum – User Stories

- User stories are similar to requirements. They are written in the following format:

  As a {ACTOR/OBJECT} I want to {ACTION} so that {RESULT}

Scrum – Planning Poker

- In scrum, 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 (usually: 1, 2, 3, 5, 8, etc.) 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 and highest numbers to state their case.
- The process is repeated until everyone ends up at the same number.

Scrum – Planning Session

- Planning sessions happen at the start of each sprint.
- They usually take a few hours. During this time, the team decides how much work it will take on, and discusses any major technical challenges they expect to face.
- Usually, Product Owners are available for at least a portion of this meeting, to help with prioritization. They are only there to assist in this regard, and not to dictate what the team will complete.

Scrum – The Standup Meeting

- Happens EVERY SINGLE day that you are working
- The goal is to make sure people are doing alright
- Shouldn’t be longer than 15 minutes
- Answer three questions:
    1. What did I finish since the last standup?
    2. What am I going to finish by the next standup?
    3. What is stopping me / what impediments am I facing?

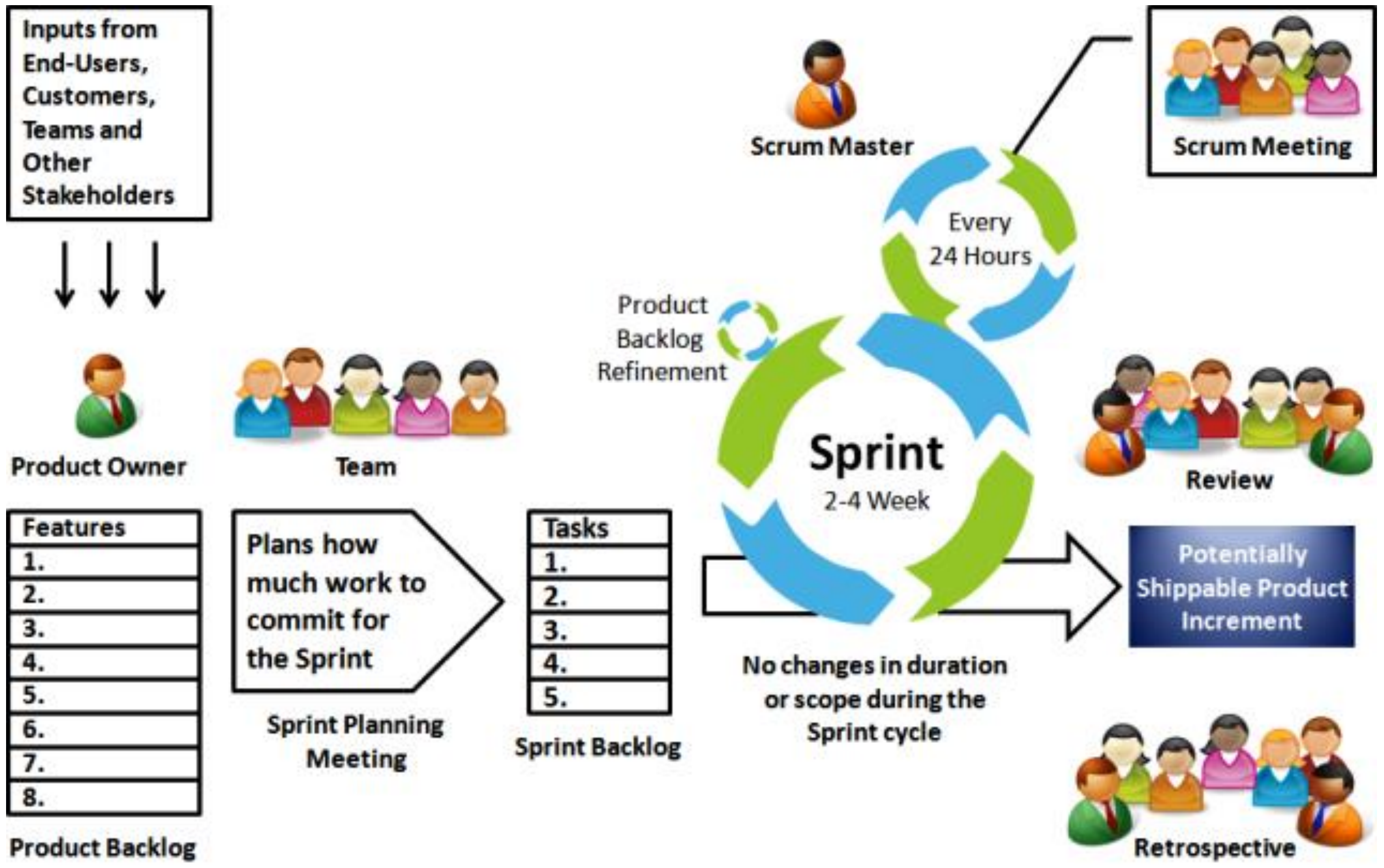
Scrum – Working Agreement

- A series of statements that everyone on the team agrees to about how the team will work
- Things in working agreements may include:
  - ➢ The standup will occur at 1:00 pm every day, and last 15 minutes
  - ➢ We will not speak during the standup, unless it is our turn to speak
  - ➢ Our meetings will take place in the lobby of the IC building
  - ➢ All code must be peer-reviewed
  - ➢ We will submit all code 24 hours prior to the due date

Scrum – Definition of Done

- A formal agreement of when work is considered complete.
- For example, a story can be marked as done when:
  - ➢ It has been fully unit tested
  - ➢ It successfully integrated with the rest of the code
  - ➢ It has been peer reviewed
  - ➢ It is fully commented
  - ➢ Etc.
- It is important that team comes to an agreement on this definition before they start work.

Scrum – overall mechanism

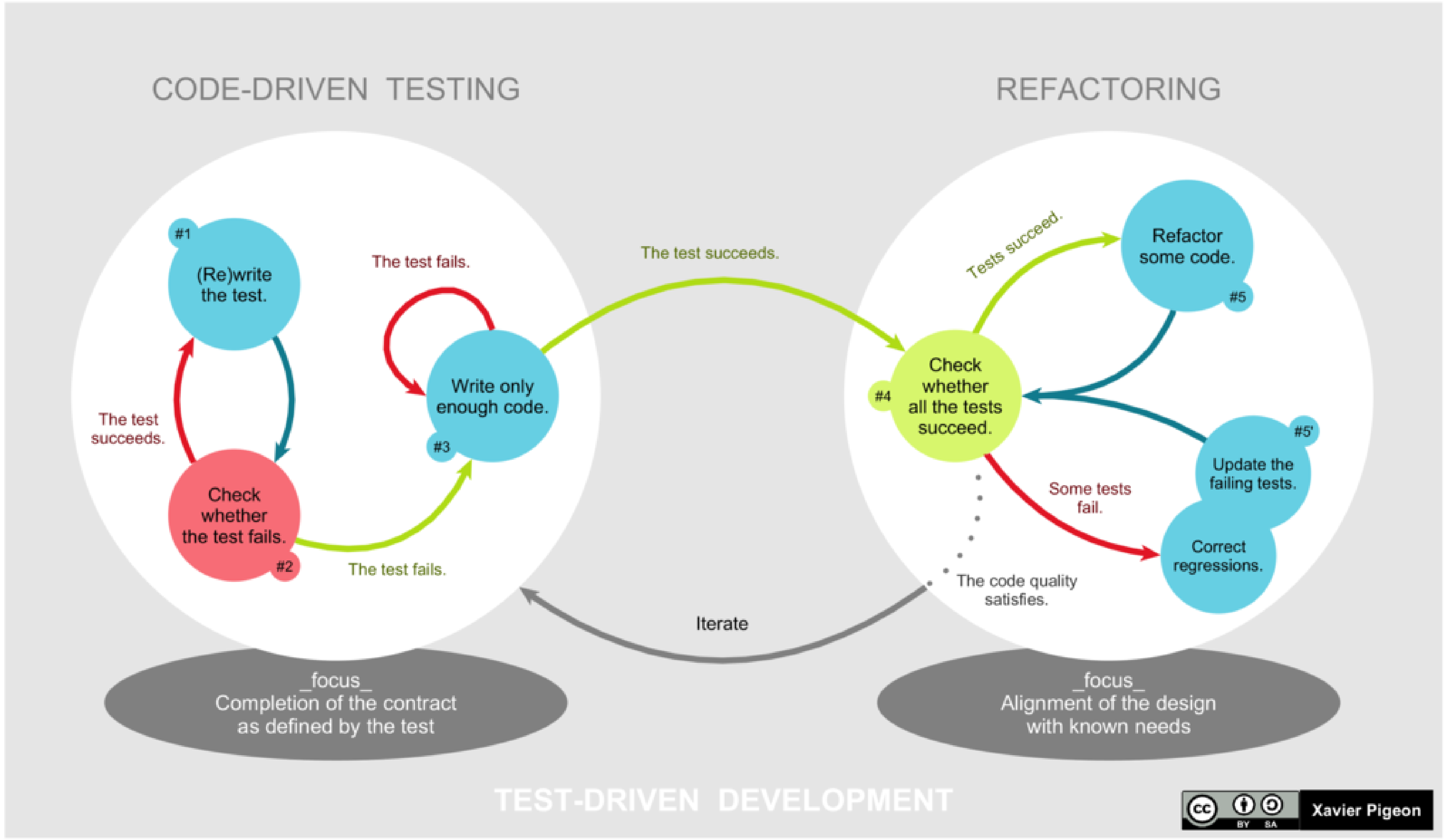

Inputs from End-Users, Customers, Teams and Other Stakeholders
Product Owner
Team
Features
1.
2.
3.
4.
5.
6.
7.
8.
Product Backlog
Plans how much work to commit for the Sprint
Sprint Planning Meeting
Tasks
1.
2.
3.
4.
5.
Sprint Backlog
Scrum Master
Scrum Meeting
Every 24 Hours
Product Backlog Refinement
Sprint
2-4 Week
No changes in duration or scope during the Sprint cycle
Review
Potentially Shippable Product Increment
Retrospective

https://pkpmp.files.wordpress.com/2014/11/scrum2.png

Test Driven Development (TDD)

- TDD is a way to develop software that revolves around writing test cases.
- The basic concept is to write the unit tests needed to be passed for a feature to be considered working. You then code to the unit tests –writing the minimum amount for the tests to succeed.
- Once working, you review and refactor. Then move on to the next set of tests.

Test Driven Development (TDD)



https://commons.wikimedia.org/wiki/File:TDD_Global_Lifecycle.png

Feature Driven Development (FDD)

- 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:
  1. Develop overall model
  2. Build feature list
  3. Plan by feature
  4. Design by feature
  5. Build by feature