Software Development Lifecycle

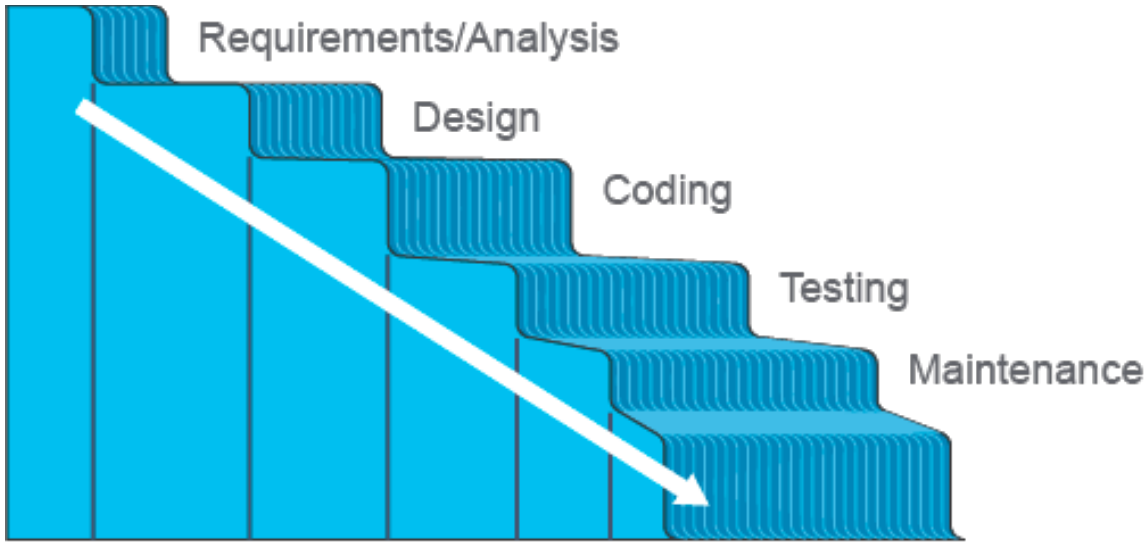
Software Development Life Cycle

- The Software Development Life Cycle (SDLC) process is used by software creators to design, develop, and test high-quality software products and applications. In addition to creating high-quality software, the SDLC aims to meet and exceed customer expectations on budget and on time.

Software development methodology

- **Waterfall**
- **Lean**
- **Agile**

Waterfall

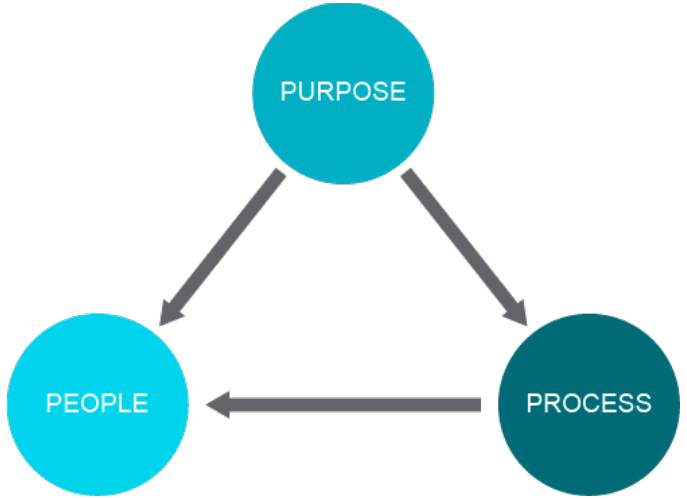


- The Waterfall methodology is based on a linear and sequential process.
- A good choice when all requirements are very well known before a project begins; therefore, it works well for small, well-defined projects.
- The Waterfall model assumes that once you move to a certain phase of the life cycle, you can go only "downhill" to the next phase.

Waterfall – pro & con

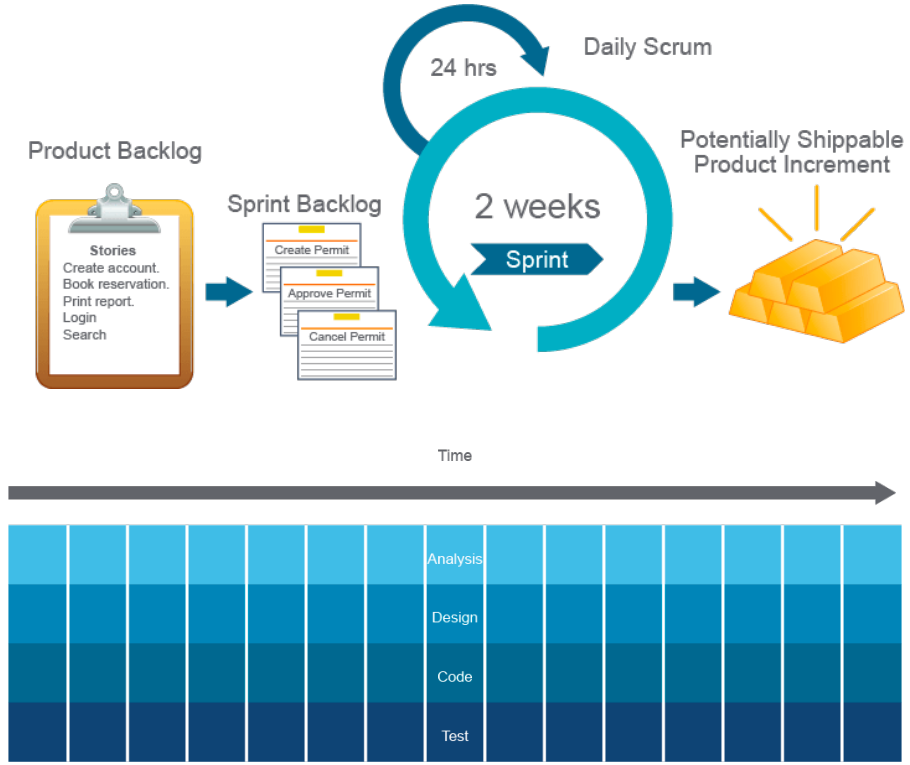
- **Advantages:**
  - Design errors are highlighted before any code is written, saving time during the implementation phase.
  - Good documentation is mandatory for this kind of methodology. This effort is useful for engineers in the later stages of the process.
  - Because the process is rigid and structured, it is easy to measure progress and set milestones.
- **Disadvantages:**
  - In the very early stages of a project, it can be difficult to gather all the possible requirements. Clients can only fully appreciate what is needed when the application is delivered.
  - It becomes very difficult (and expensive) to re-engineer the application, making this approach very inflexible.
  - The product will only be shipped at the very end of the chain, without any middle phases to test. Therefore, it can be a very lengthy process.

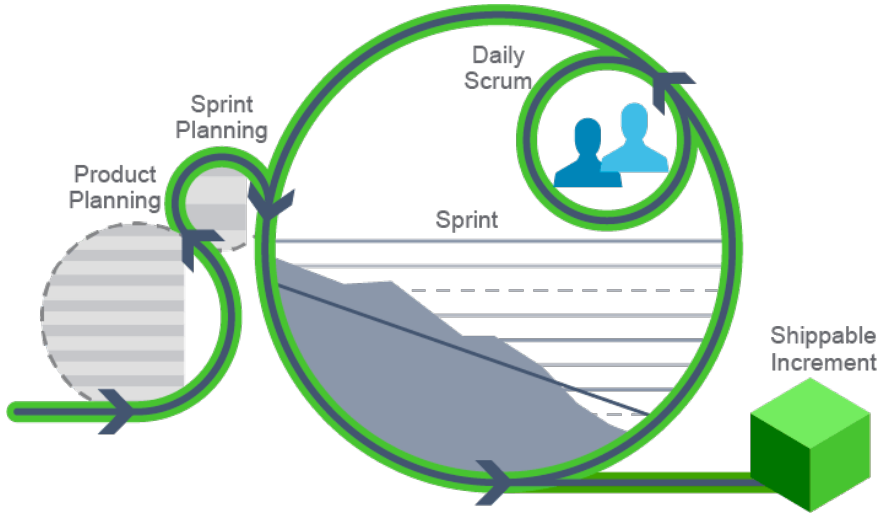
Lean Principles for software development



TRAINOCATE

Agile





- Agile is a means of implementing the Lean philosophy in the software development industry.
- It is primarily based on the concept of short sprints, seeking to do as much as possible in a relatively short time, and without losing a focus on value. Agile software development includes customers in the software development life cycle by delivering software in very early stages, to gain valuable feedback from actual consumers of the software. This procedure is a popular way to test software and learn about issues that can be addressed in future releases.

TRAINOCATE

Alige – pro and con

Advantages:

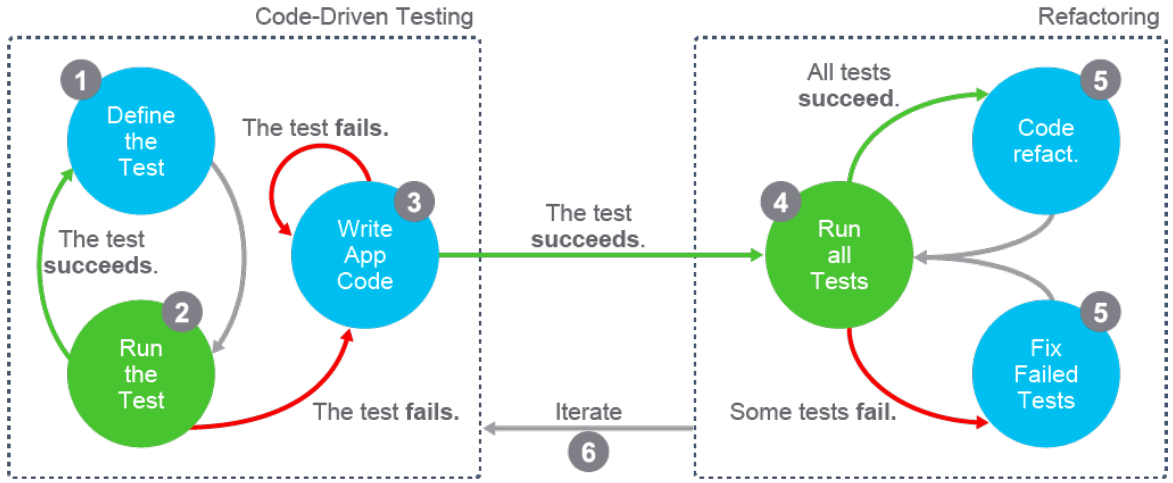
- The rapid and continuous delivery of software releases helps customers to better understand what the final piece will look like. This capability also provides a fully working code improved week after week, leading to customer satisfaction.
- Thanks to Scrum, people interaction is emphasized.
- Late project changes are more welcome.

Disadvantages:

- This kind of approach lacks emphasis on good documentation, because the main goal is to release new code quickly and frequently.
- Without a complete requirement-gathering phase at the beginning of the process, customer expectations may be unclear, and there is a good chance for scope creep.
- Rapid development increases the chance for major design changes in the upcoming sprints, because not all the details are known.

Test-Driven Development

Test-Driven Development



- Test-driven development (TDD) is a software-development methodology where you write the test code before the actual production code.
- Development is done in iterations, where you do the following:
  - Write tests.
  - Run these tests; they must fail (code may not even compile).
  - Write source code.
  - Run all tests; they must pass.
  - Refactor the code where necessary.

TDD benefits

- Writing these tests first has immediate benefits for development:
  - Gives you a clear goal—make the tests pass
  - Shows specification omissions and ambiguities before writing code, avoiding potentially costly rewrites
  - Uncovers edge cases for you to address from the start
  - Makes debugging easier and faster, because you can simply run the tests
  - Passing each test is a small victory that drives you forward.

Refactoring the code

- **Better code structure:** You might find out that a certain part of the code will be used multiple times, so you move it to a separate function.
- **Better code readability:** You might want to split or combine different parts of the code to be more readable for other developers who are not familiar with the task.
- **Better design:** You might find that refactoring the code in some way will simplify the design or even improve the performance of the software.

The TDD iteration is finished when code refactoring is done and all tests pass. Then, you can proceed to the next iteration.


TRAINOCATE