DEVELOPMENT LIFECYCLE & AGILE TECHNIQUES

LESSON 10 - WORKSHOP

NANODEGREE → FOUNDATION MODULE

AGENDA

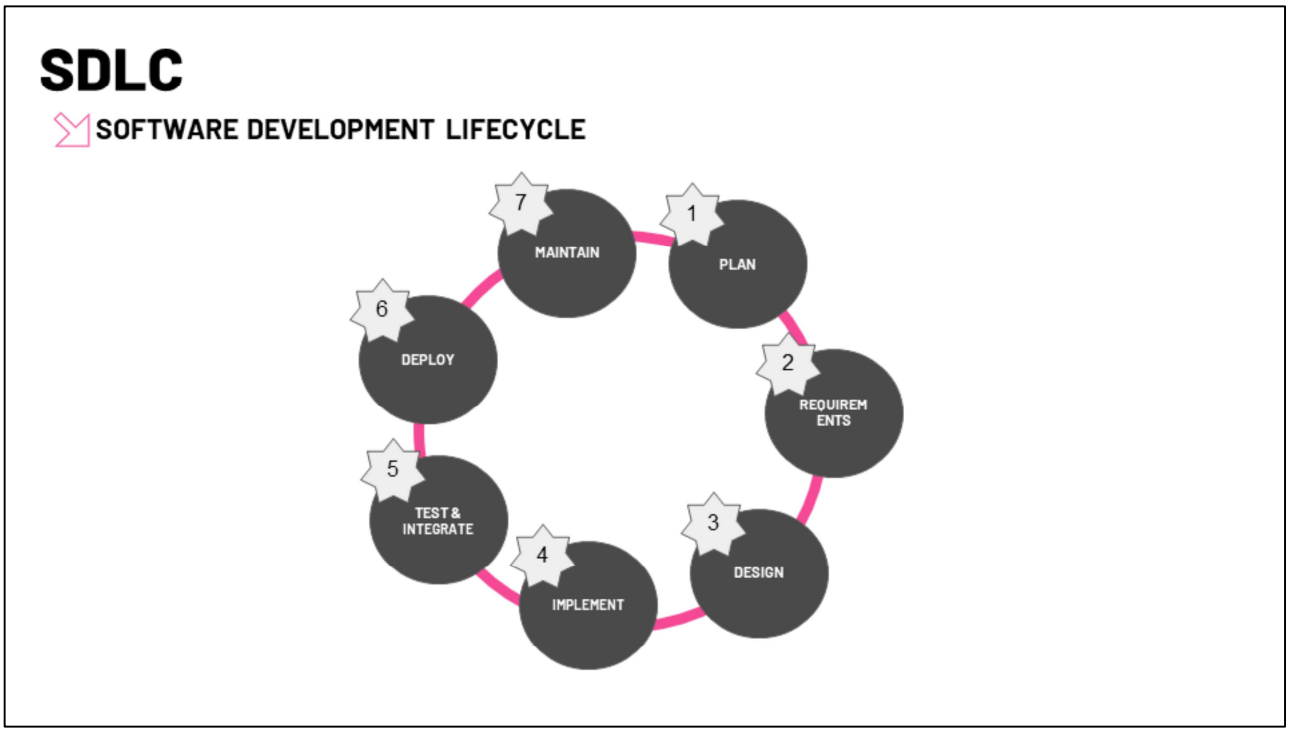
01 Introduction to SDLC

02 SDLC Methodology

03 Scrum

04 Scrum Board Demo

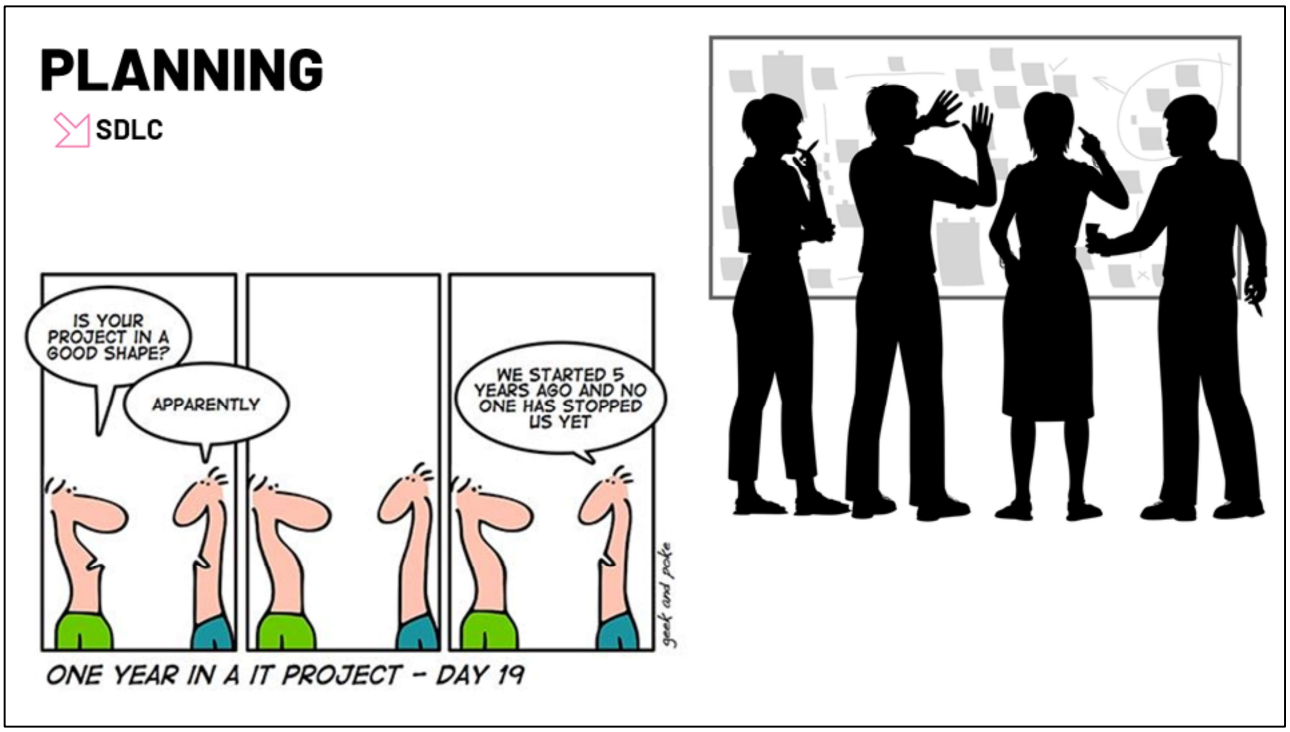
05 Scrum Retro group exercise



Software Development Life Cycle (SDLC) is a terminology used to explain how software is delivered to a customer in a series of steps.
These steps take software from the ideation phase to delivery. It is a process that produces software with the highest quality and lowest cost in the shortest time possible. SDLC provides a well-structured flow of phases that help an organization to quickly produce high-quality software which is well-tested and ready for production use.
The process of software development is a never-ending cycle. The first release of a software application is rarely “finished.” There are almost always additional features and bug fixes waiting to be designed, developed, and deployed.
These steps are (very) roughly the same from one methodology to another.

They tend to occur in this order, though they can also be mixed together, such that several steps occur in parallel.

As we’ll discuss later, Agile methods tend to “wind together” all of these steps into a tight, rapidly-repeating cycle. Outputs from one become inputs to the following step.



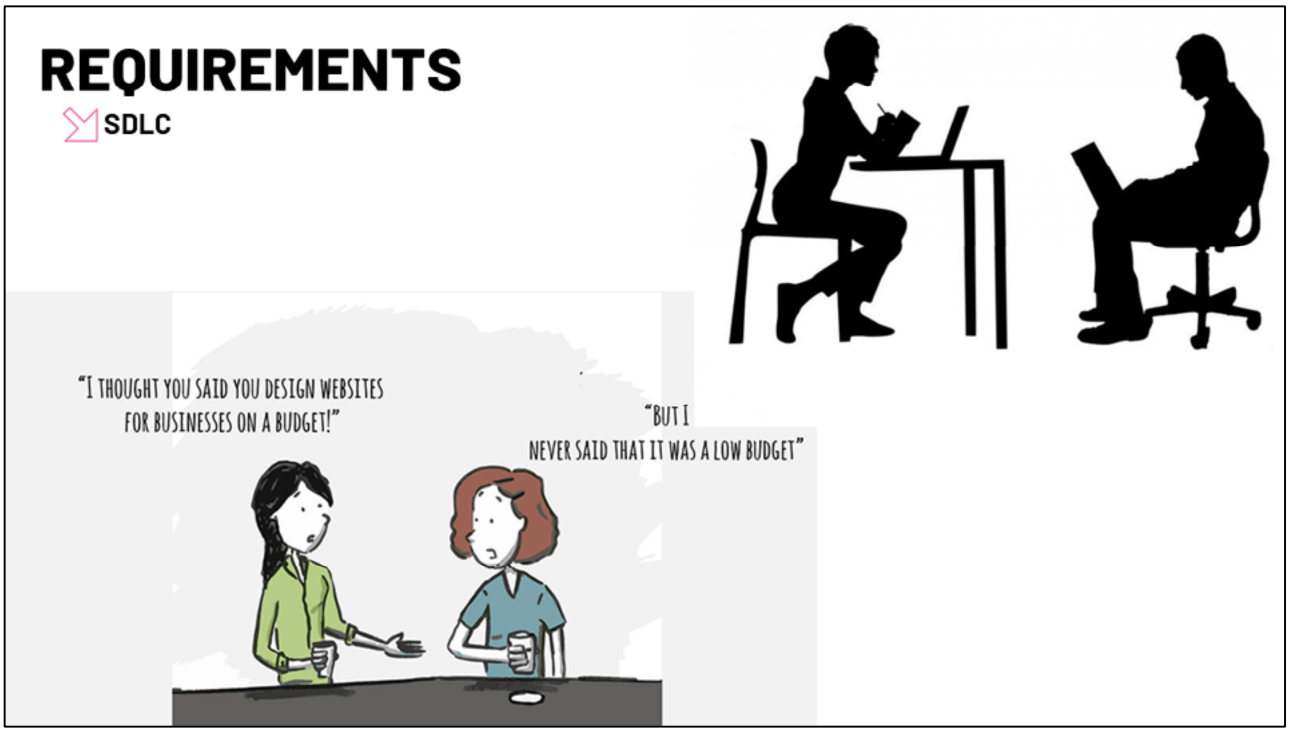
1. Planning

In this stage of the SDLC, the team determines the cost and resources required for implementing the analyzed requirements. It also details the risks involved and provides sub-plans for softening those risks. In other words, the team should determine the feasibility of the project and how they can implement the project successfully with the lowest risk in mind.

The planning phase involves aspects of project and product management. This may include:

- Resource allocation (both human and materials)
- Capacity planning
- Project scheduling
- Cost estimation
- Provisioning

The outputs of the planning phase include: project plans, schedules, cost estimations, and procurement requirements. Ideally, Project Managers and Development staff collaborate with Operations and Security teams to ensure all perspectives are represented.



1. Requirements

The business must communicate with IT teams to convey their requirements for new development and enhancement. The requirements phase gathers these requirements from business stakeholders and Subject Matter Experts (SMEs.)

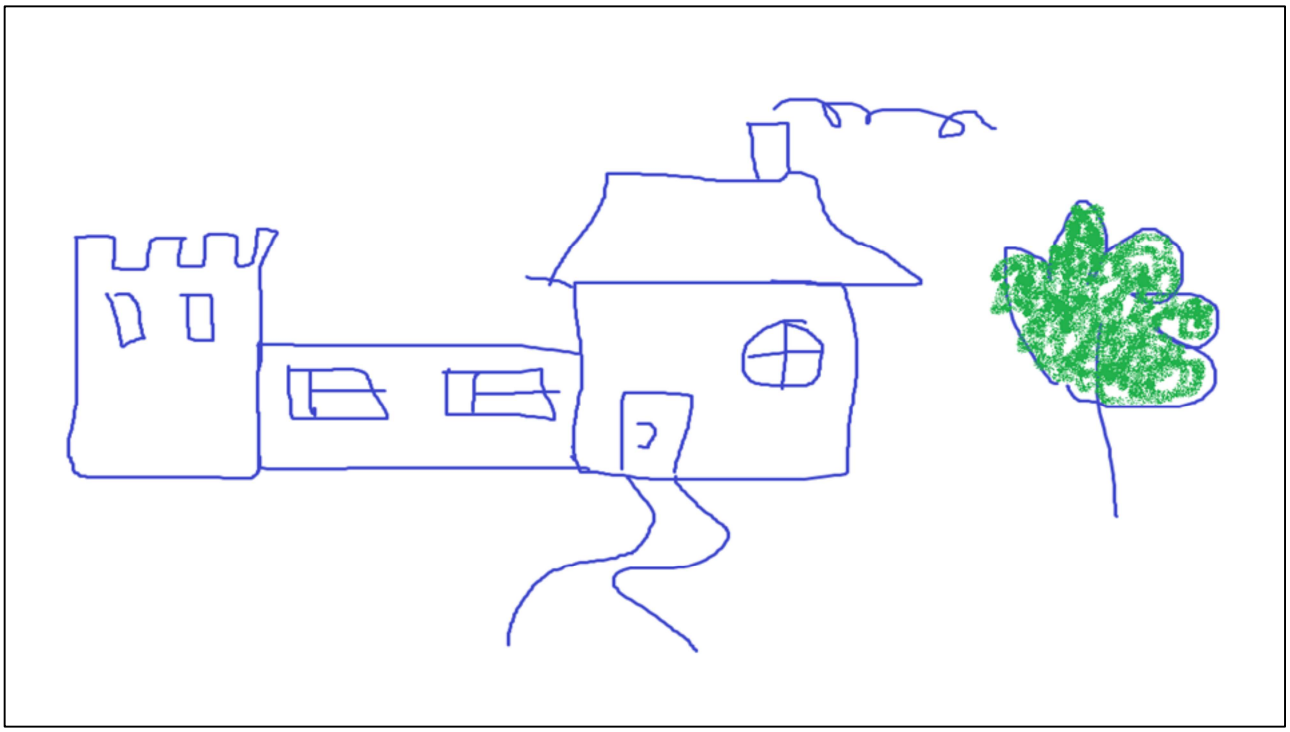
Architects, Development teams, and Product Managers work with the SMEs to document the business processes that need to be automated through software.

The output of this phase in a Waterfall project is usually a document that lists these requirements. Agile methods, by contrast, may produce a backlog of tasks to be performed.

REQUIREMENTS

Grab a pen and some paper

GROUP EXERCISE



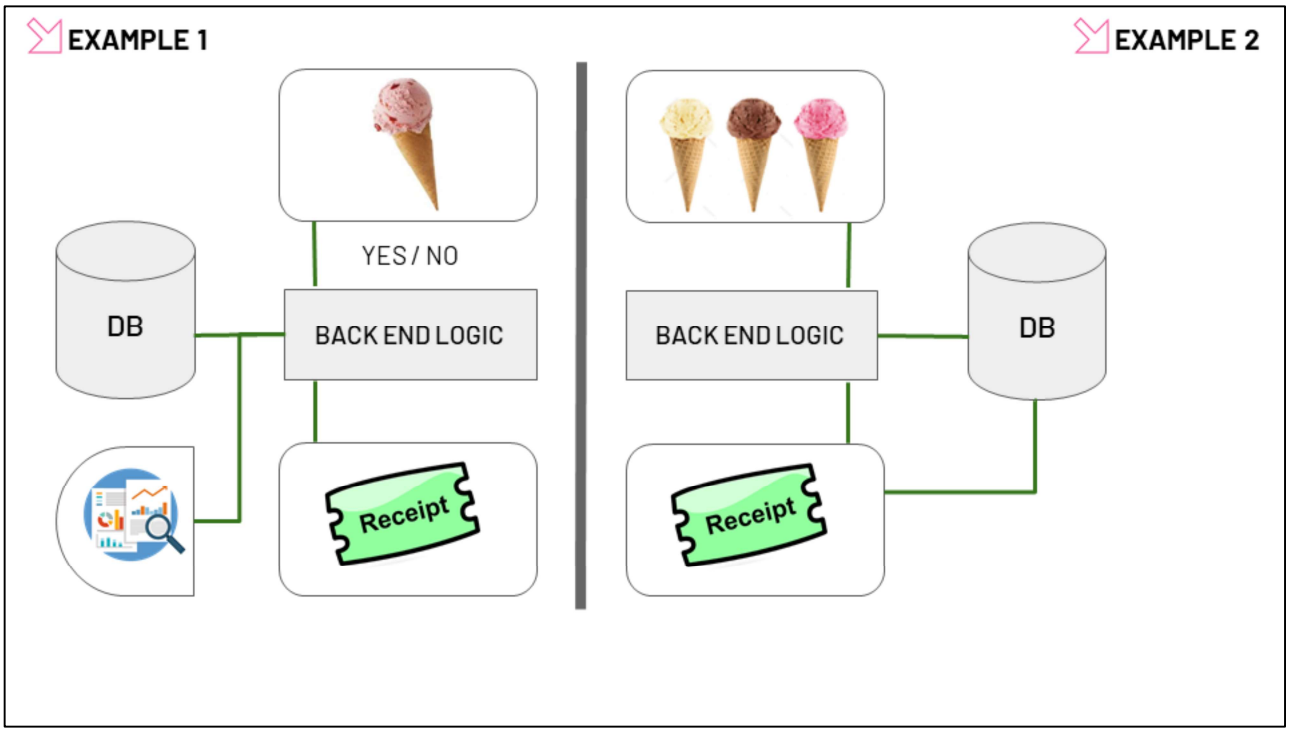
3. Design

This phase of the SDLC starts by turning the software specifications into a design plan called the Design Specification. All stakeholders then review this plan and offer feedback and suggestions. It's crucial to have a plan for collecting and incorporating stakeholder input into this document. Failure at this stage will almost certainly result in cost overruns at best and the total collapse of the project at worst.

Once the requirements are understood, software architects and developers can begin to design the software. The design process uses established patterns for application architecture and software development.

Developers use proven **Design Patterns** to solve algorithmic problems in a consistent way. This phase may also include some rapid prototyping, also known as a **spike**, to compare solutions to find the best fit. Teams can run **whiteboard** sessions to outline system flow or architecture together. The output of this phase includes:

- Design documents that list the patterns and components selected for the project
- Code produced by spikes, used as a starting point for development



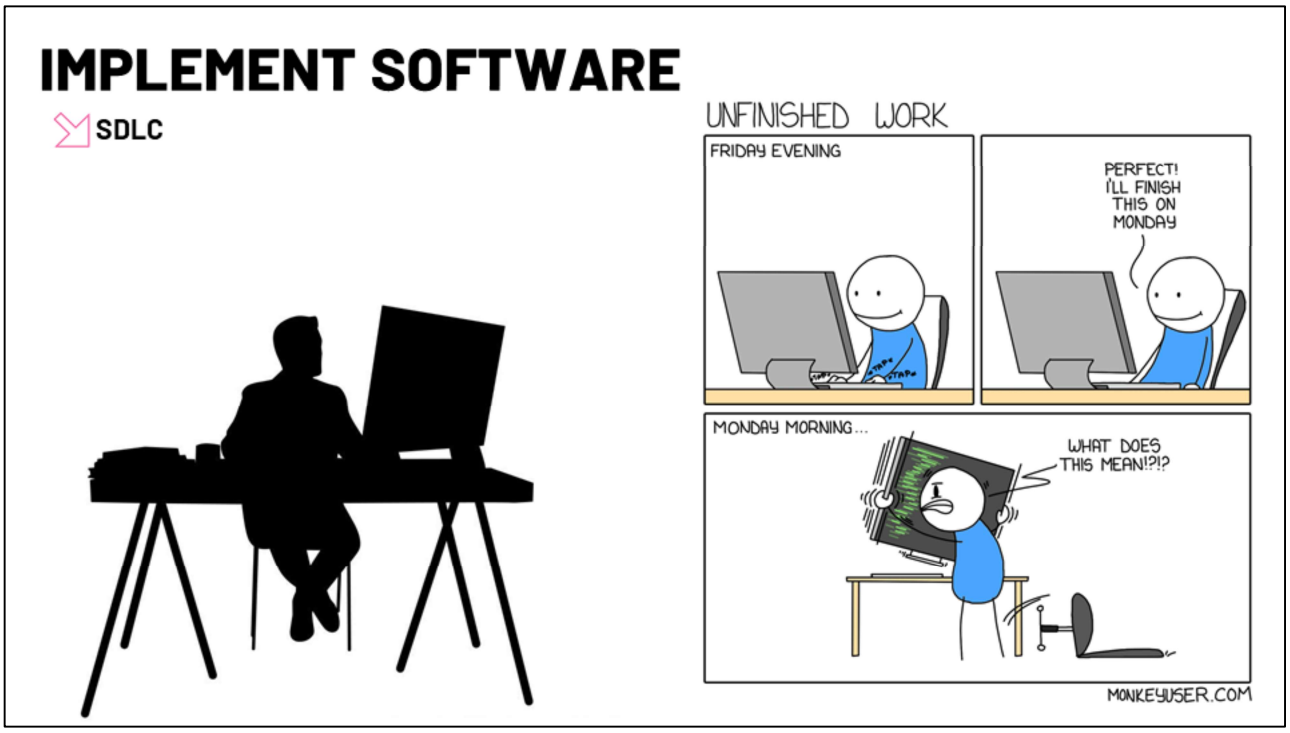
Let's imagine that we need to design an ice cream order process. We have a UI to accept customer order, backend and a database.

Example 1

1. A customer asks for a specific ice cream (vanilla, choc, etc.)
2. The system makes a call to DB to check if we have it available
3. If we do, then the order gets processed and finally we display a receipt, plus purchase summary in UI.
4. We also have an additional analytics component add-on (3rd party system) that analyses all our orders/sold items/times they were purchased and so on.

Example 2

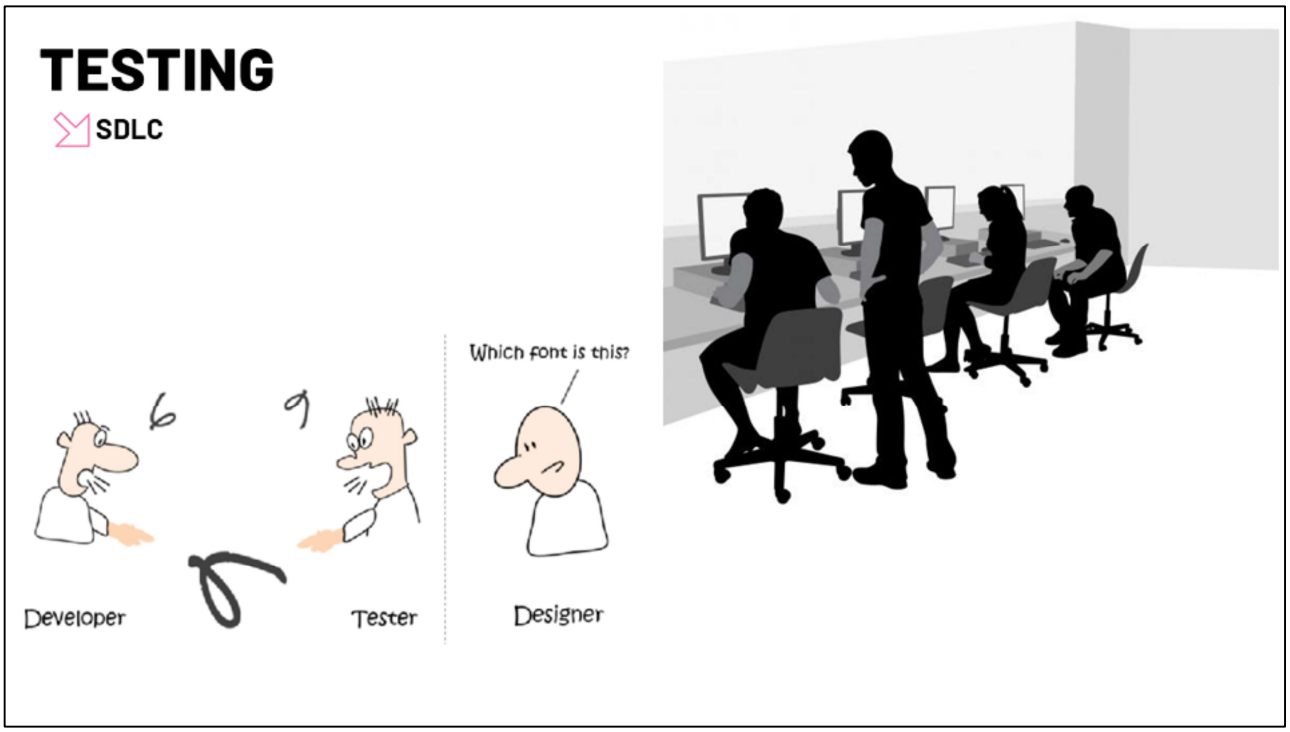
1. All types of available ice cream are displayed to the customer straight away
2. Accept the order and make a call to DB to update the 'stock' table, process the order
3. Then the order gets processed and finally we display a receipt, plus purchase summary in UI.
4. We do not have a dedicated analytics component, BUT we save all order processing details (including timestamps and other metadata) to a table in the DB.
5. Later we can run various queries to perform analytics in the DB.



Software Development

This phase produces the software under development. Depending on the methodology, this phase may be conducted in time-boxed “sprints,” (Agile) or may proceed as a single block of effort (Waterfall.) Regardless of methodology, development teams should produce working software as quickly as possible. Business stakeholders should be engaged regularly, to ensure that their expectations are being met. The output of this phase is testable, functional software.

It’s important that every developer sticks to the agreed blueprint. Also, make sure you have proper guidelines in place about the code style and practices. For example, define a nomenclature for files or define a variable naming style such as camelCase. This will help your team to produce organized and consistent code that is easier to understand but also to test during the next phase.

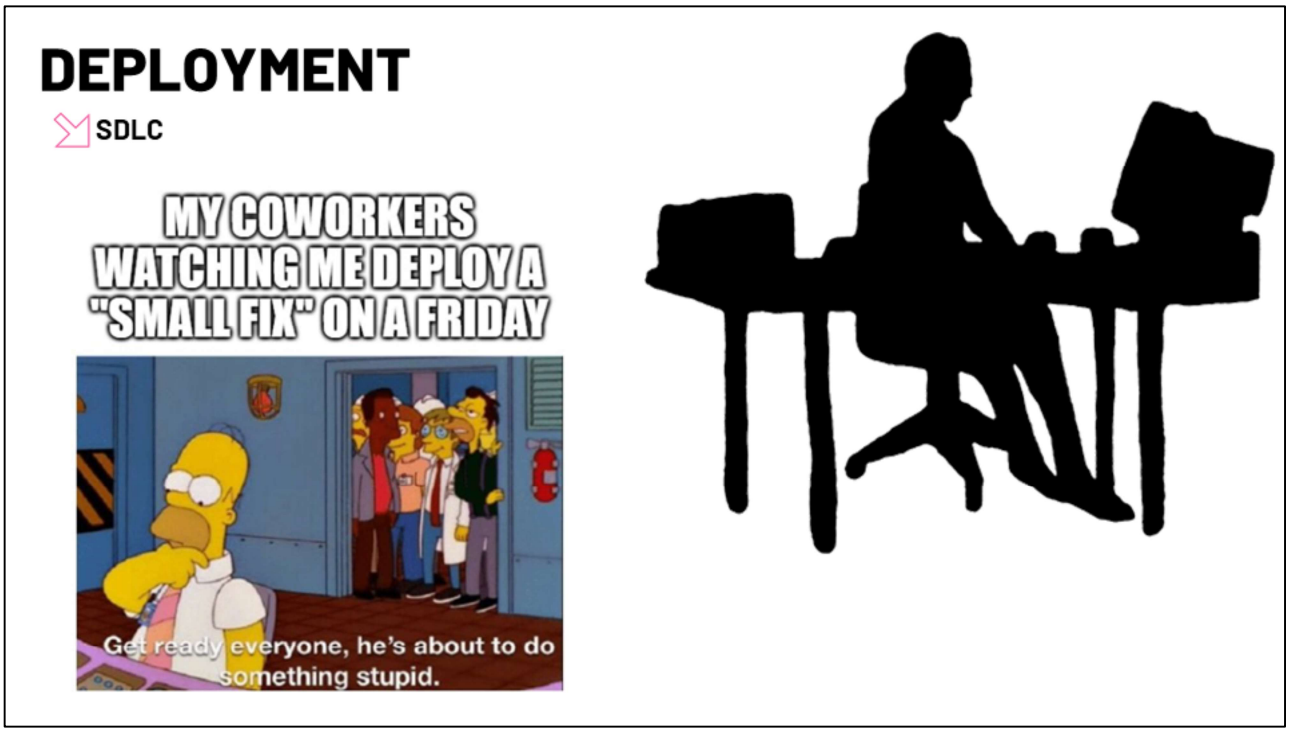


5. Testing

The testing phase of the SDLC is arguably one of the most important. It is impossible to deliver quality software without testing. There is a wide variety of testing necessary to measure quality:

- Code quality
- Unit testing (functional tests)
- Integration testing
- Performance testing
- Security testing

The best way to ensure that tests are run regularly, and never skipped for expediency, is to *automate* them. The output of the testing phase is functional software, ready for deployment to a production environment.



6. Deployment

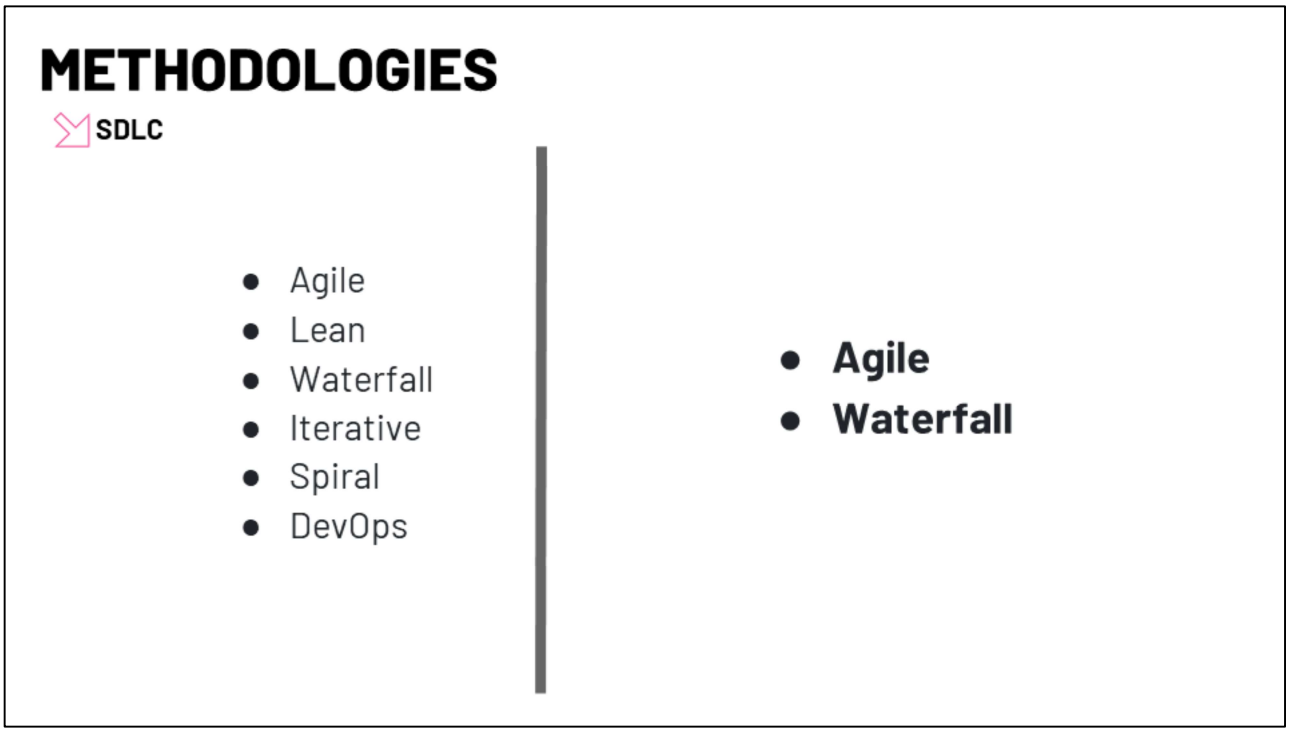
The deployment phase is, ideally, a highly automated phase. In high-maturity enterprises, this phase is almost invisible; software is deployed the instant it is ready. Enterprises with lower maturity, or in some highly regulated industries, the process involves some manual approvals. However, even in those cases it is best for the deployment itself to be fully automated in a continuous deployment model.

Application Release Automation (ARA) tools are used in medium and large-size enterprises to automate the deployment of applications to Production environments. ARA systems are usually integrated with Continuous Integration tools. The output of this phase is the release to Production of working software.

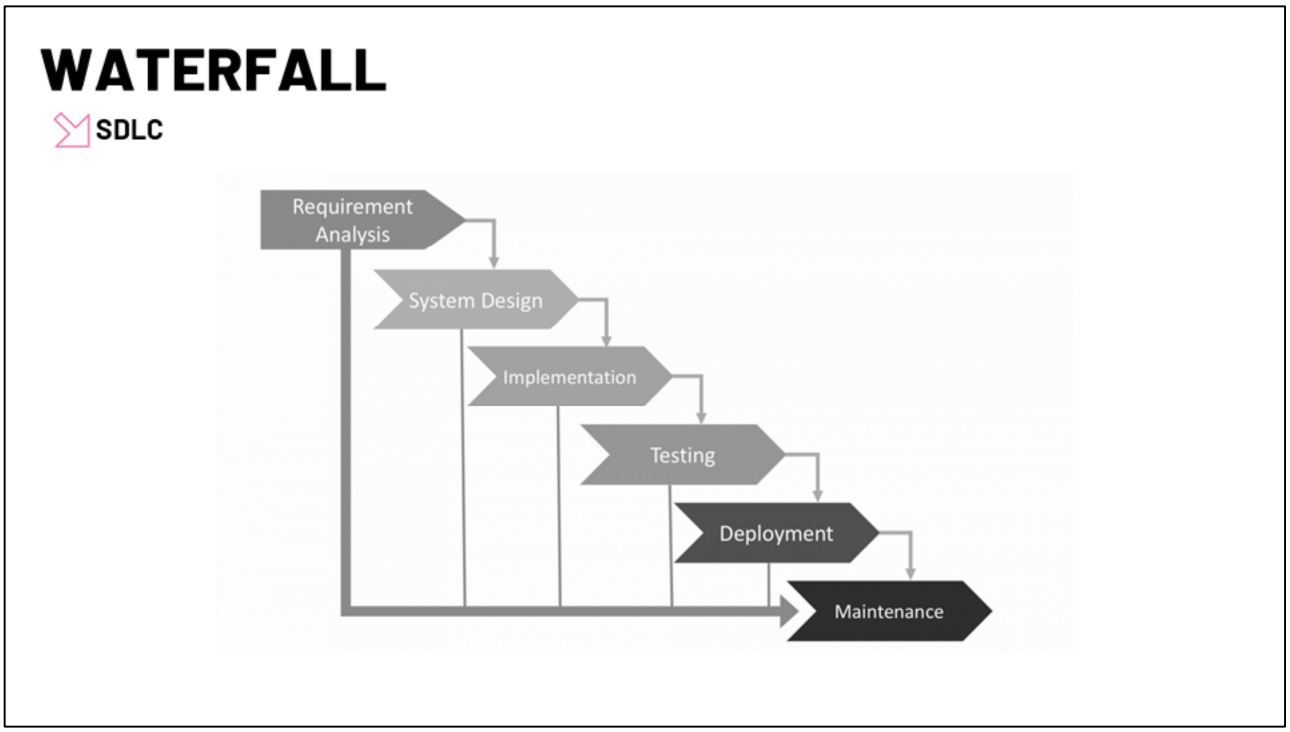
7. Maintenance

The operations and maintenance phase is the “end of the beginning,” so to speak.
The Software Development *Life Cycle* doesn’t end here. Software must be monitored constantly to ensure proper operation.

Bugs and defects discovered in Production must be reported and responded to, which often feeds work back into the process. Bug fixes may not flow through the entire cycle, however, at least an abbreviated process is necessary to ensure that the fix does not introduce other problems (known as **a regression.)**



We focus on 2 commonly used. Please research the other methodologies in their own time.



Waterfall – is a cascade SDLC model, in which development process looks like the flow, moving step by step through the phases of analysis, projecting, realization, testing, implementation, and support. This SDLC model is the oldest and most straightforward. With this methodology, we finish one phase and then start the next. Each phase has its own mini-plan and each phase “waterfalls” into the next. The biggest drawback of this model is that small details left incomplete can hold up the entire process.

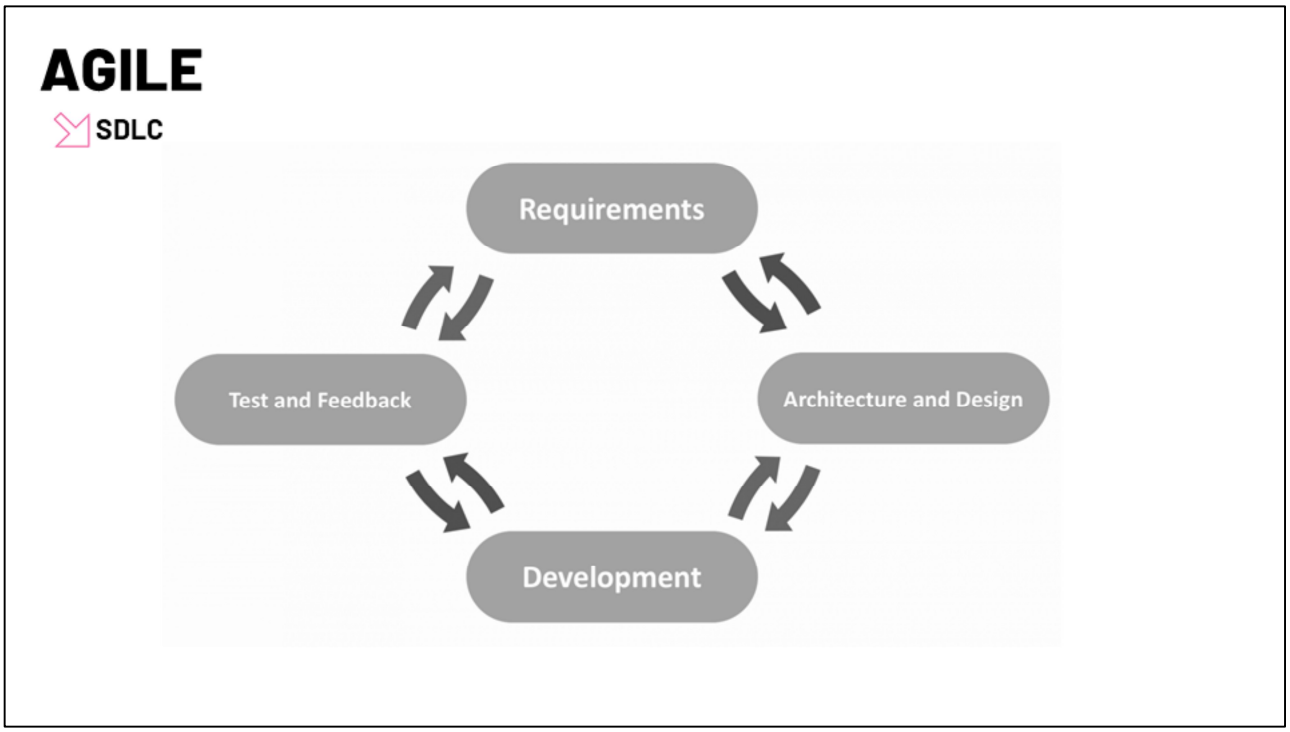
This SDLC model includes gradual execution of every stage completely. This process is strictly documented and predefined with features expected to every phase of this software development life cycle model.

Use cases for the Waterfall SDLC model:

- The requirements are precisely documented
- Product definition is stable
- The technologies stack is predefined which makes it not

dynamic

- No ambiguous requirements
- The project is short



In the agile methodology after every development iteration, the customer is able to see the result and understand if he is satisfied with it or he is not. This is one of the advantages of the agile software development life cycle model.

One of its disadvantages is that with the absence of defined requirements it is difficult to estimate the resources and development cost. Extreme programming is one of the practical use of the agile model. The basis of such model consists of short weekly meetings – Sprints which are the part of the Scrum approach.

The Agile SDLC model separates the product into cycles and delivers a working product very quickly. This methodology produces a succession of releases. Testing of each release feeds back info that's incorporated into the next version. The drawback of this model is that the heavy emphasis on customer interaction can lead the project in the wrong direction in some cases.

Use cases for the Agile model:

- The users' needs change dynamically
- Less price for the changes implemented because of the many iterations
- Unlike the Waterfall model, it requires only initial planning to start the project

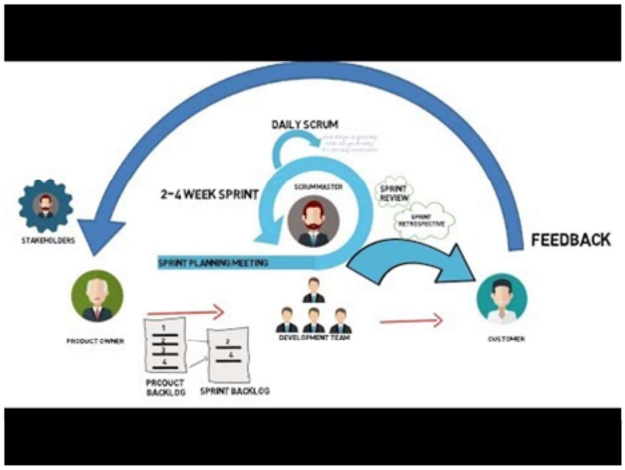
Criticisms of Agile Project Management

- Focuses on the team rather than the customer.
- Focuses on the process not the product.
- Not very efficient in large organizations and programs
- Requires leadership vs. command
- Promotes implied knowledge vs. explicit
- Teams need to self-organize and adapt
- Communication tends to be informal
- Mainly for technology-driven projects

- Basically… Requires mature teams which communicate well

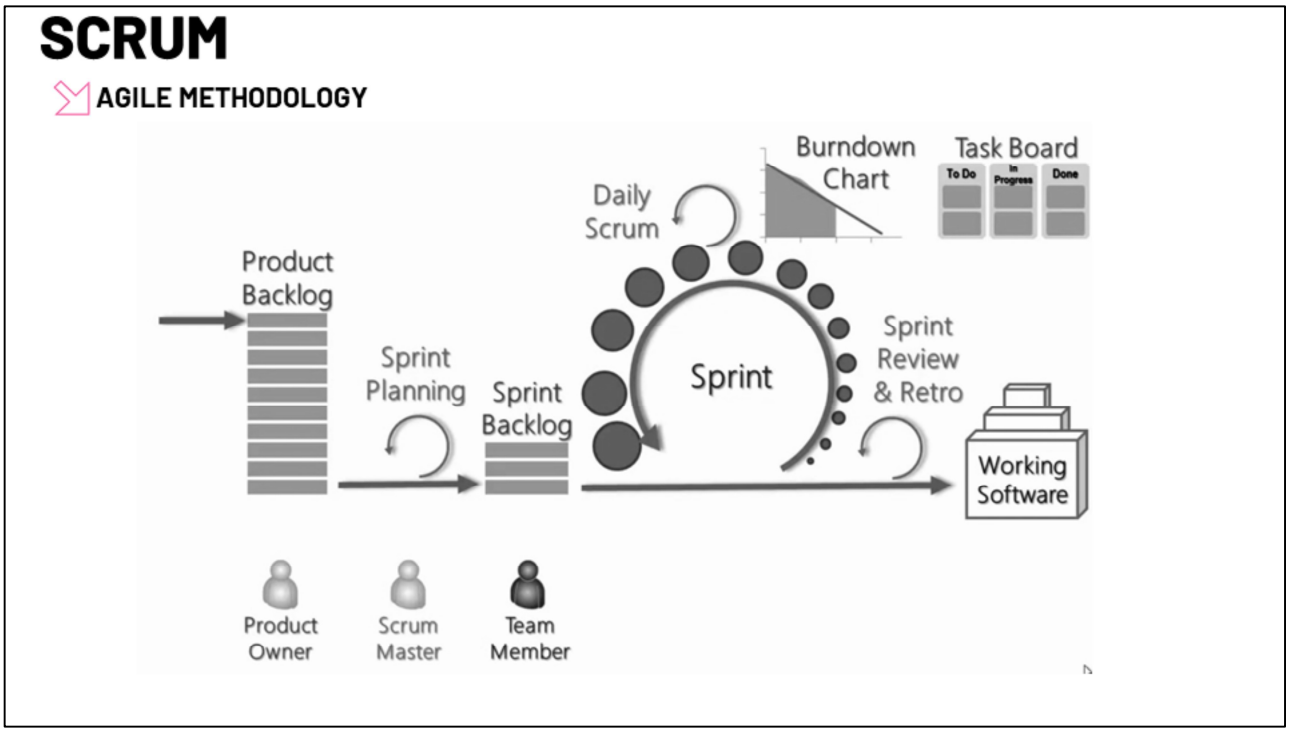
SCRUM

VIDEO



Scrum is a framework that helps teams work together. Much like a rugby team (where it gets its name) training for the big game, scrum encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve.

While the scrum I'm talking about is most frequently used by software development teams, its principles and lessons can be applied to all kinds of teamwork. This is one of the reasons scrum is so popular. Often thought of as an agile project management framework, scrum describes a set of meetings, tools, and roles that work in concert to help teams structure and manage their work.



This is a HUGE subject, so we won't be able to do a deep dive, but our objective is to understand the main principles:

Scrum process, roles and meetings.

Below is some terminology to help you walk through the diagram:

Scrum Team – Product Owner, Scrum Master and Development Team

Product Owner – The person responsible for maintaining the Product Backlog by representing the interests of the stakeholders, and ensuring the value of the work the Development Team does.

Scrum Master – The person responsible for the Scrum framework, making sure it is used correctly and maximizing its benefits.

Development Team – A cross-functional group of people responsible for delivering potentially shippable increments of product at the end of every Sprint.

Sprint burn-down chart – Daily progress for a Sprint over the Sprint's length.

Release burn-down chart – Feature level progress of completed Product Backlog Items in the Product Backlog.

Product Backlog (PBL) list – A prioritized list of high-level requirements.

Sprint Backlog (SBL) list – A prioritized list of tasks to complete during the Sprint.

Sprint – A time period (typically 1–4 weeks) in which development occurs on a set of Product Backlog Items that the team has committed to—commonly referred to as a time-box or iteration

Definition of Done (DoD) – The exit-criteria to determine whether a Product Backlog Item is complete. In many cases the DoD requires that all regression tests should be successful. The definition of "done" may vary from one Scrum Team to another, but must be consistent within one team.

Velocity – The total effort a team is capable of in a Sprint. The number is derived by evaluating the work (typically in user story points) completed in the last Sprint. The collection of historical velocity data is a guideline for assisting the team in understanding how much work they can likely achieve in a future Sprint.

Impediment – Anything that prevents a team member from performing work as efficiently as possible.

Summary of meetings

1.Sprint Planning

Attended by the scrum master, product owner, and development team. Outside

stakeholders may attend by invitation (rare). The two defined artifacts coming out of this meeting are the sprint goal and sprint backlog. During this meeting, the product owner describes the highest priority items in the product backlog.

2. Daily Scrum

Think of this as your morning coffee…you have to have it at the start of every day to be effective. The Daily Scrum is a brief daily meeting where the entire team meets to discuss what they worked on during the previous day, what they plan to do today and any obstacles that are hindering them or the team from meeting the Sprint goal. This meeting should last no longer than 15 minutes.

3. Sprint Review

Think of this as the “Show it off” meeting. This meeting is open to anyone: the team, Product Owner, Scrum Master, stakeholders, management, customers, etc. The team demonstrates to the attendees a working product of what they were able to complete during the Sprint. The team must only demonstrate items that meet the “definition of done”. Typically this meeting is intentionally kept informal. Did the achieve the overall “goal of the sprint”?

4. Sprint Retrospective

The sprint retrospective meeting follows the sprint review meeting. The team will review the parts of the scrum process that went right, wrong, and what could have gone better. It is a chance for the team to learn from both their mistakes and successes in hopes of using that information to improve future sprints and projects.

5. Backlog Refinement Meeting

Think of the backlog as the roadmap of the project. As the team collaborates to create a list of everything that needs to be built or done for project completion, this list can be modified and added to throughout the project to ensure that all of the necessary needs of the project are met.

There is no defined answer, it would all depend on requirements, budget, timelines etc.
However, real life examples could be:

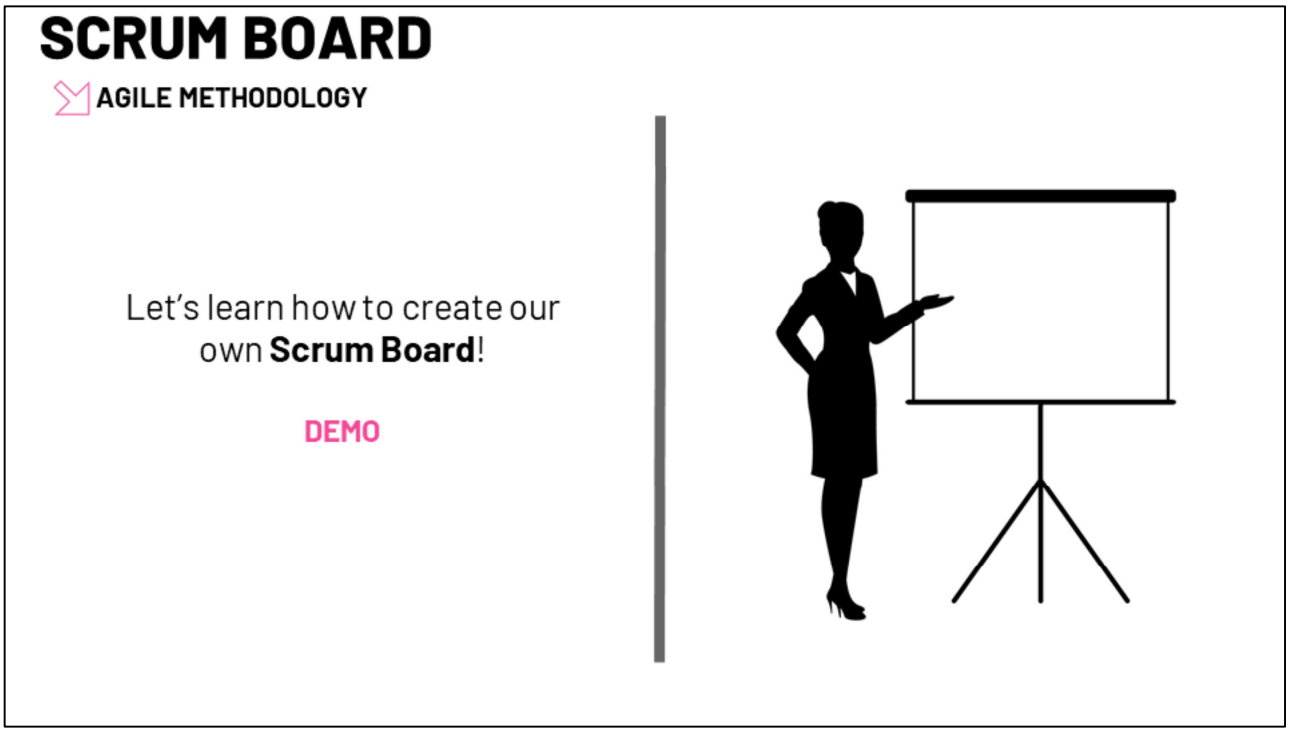
> Waterfall -- implementing regulatory software (e.g.in trading MIFID II), well established known requirements, e.g. election / voting system that accepts votes from a public etc.
>
> Agile -- Checkout and Payment system for an online shop We start with an MVP and then add more and more features.

IMPORTANT

INTERVIEW QUESTIONS

SCRUM QUESTIONS OFTEN GET FEATURED IN INTERVIEWS

- **Know what Scrum meetings are for**
- **Know Scrum roles**
- **Know the benefits of Agile.**



Design your own SCRUM BOARDS (it is up to you which one to use):

1. Use http://scrumblr.ca/ or example 1
2. Use https://trello.com/ for example 2

Also known as a sprint board, a scrum board is a visual representation of **the work to be done in a single sprint.**
The Scrum board allows the Scrum team to:

- Identify the tasks that need to be completed
- Ensure that everyone is working on a project task
- Keep track of the progress of an active sprint

Teams hold their daily Scrum or stand-up meetings in front of a Scrum board to discuss how their day went.
A basic Scrum board is a digital or physical board that is divided into four vertical columns. Here are the key elements of Scrum task boards:

- **Stories**: the user stories in the current sprint backlog that need to be divided into tasks
- **To do**: the tasks and subtasks which have not been started yet. These are visualized as cards or sticky notes and include details like due dates and

owners (the team members working on the Scrum task).

- **Work in progress**: the tasks your team is currently working on
- **Done**: the tasks which have been completed

Based on your Scrum process, you can also create additional columns for added functionality.

- **In Review:** includes tasks that have been completed, but need to be tested or quality checked
- **On Hold**: includes tasks that you won't work on in this Scrum sprint, but might be added during future sprint planning sessions

Design your own Retro board

1. Use https://reetro.io/

For in-person teams, find a whiteboard or large paper, and set out Post-It notes and markers in a meeting room.

What we did well (15 MIN)

- Using either a digital whiteboard or a physical one, have each team member write down what the team did well, one idea per note. Post the notes, and group similar or duplicate ideas together. Discuss each one briefly as a team.

What we can do better (10 MIN)

- Have everyone write down what they think can be improved, one idea per note. Post the notes, and group similar or duplicate ideas together. Discuss each theme as

a team.

Actions (10 MIN)

- Have everyone brainstorm actions that can be taken to improve problem areas, one idea per note.

- Post the notes and group similar or duplicate ideas. Discuss each idea as a team, and assign owners to these actions and due dates as necessary.


CODE
FIRST
GIRLS
THANK YOU!