SDLC Introduction

SDLC stands for **S**oftware **D**evelopment **L**ife **C**ycle.

Simply put, this is the process by which software goes from a need or an idea to a finished fully functioning product.

There are two main software development methodologies and we're going to learn them.

The first is called **Waterfall** and the second is called **Agile**.

Note: There are other models such as Spiral, V-Shaped Model, Big Bang etc. which are not widely practised.

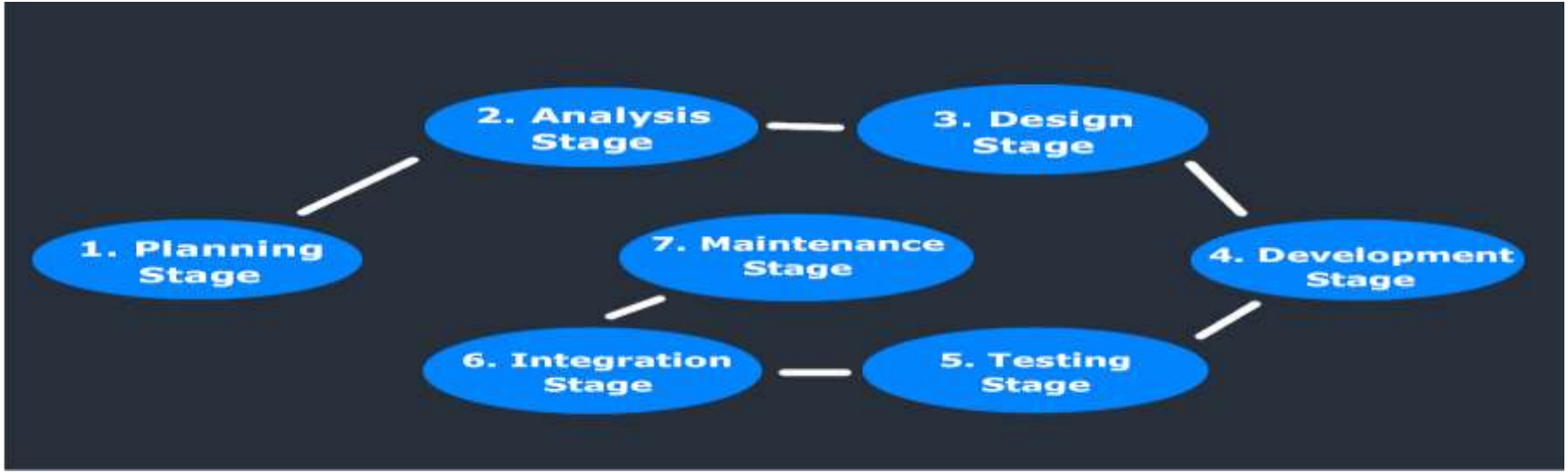
SDLC Introduction

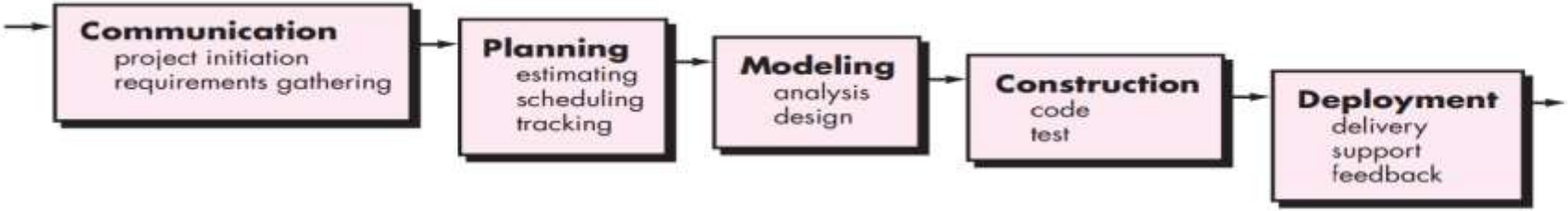
Before we look into these methodologies, there are few important questions that need to answered and need everyone involved to agree upon.

- What are we building?
- Who are we building it for?
- How should it behave?
- How do we know if it works?
- How do we know when we're done?
- How much time do we have?
- How much can we spend?

Having a structured, repeatable process helps ensure reliability and high quality.

SDLC Stages/Phases





SDLC Phases & Activities

Stage 1: Project Planning

The planning stage (also called the feasibility stage) is exactly what it sounds like: the phase in which developers will plan for the upcoming project.

It helps to define the problem and scope of any existing systems, as well as determine the objectives for their new systems.

By developing an effective outline for the upcoming development cycle, they'll theoretically catch problems before they affect development and help to secure the funding and resources they need to make their plan happen.

Perhaps most importantly, the planning stage sets the project schedule, which can be of key importance if development is for a commercial product that must be sent to market by a certain time.

SDLC Phases & Activities

Stage 2: Gathering Requirements & Analysis

The analysis stage includes gathering all the specific details required for a new system as well as determining the first ideas for prototypes.

Developers may:

- Define any prototype system requirements
- Evaluate alternatives to existing prototypes
- Perform research and analysis to determine the needs of end-users

Furthermore, developers will often create a software requirement specification or SRS document.

This includes all the specifications for software, hardware, and network requirements for the system they plan to build. This will prevent them from overdrawing funding or resources when working at the same place as other development teams.

SDLC Phases & Activities

Stage 3: Design

The design stage is a necessary precursor to the main developer stage.

Developers will first outline the details for the overall application, alongside specific aspects, such as its:

- User interfaces
- System interfaces
- Network and network requirements
- Databases

They'll typically turn the SRS document they created into a more logical structure that can later be implemented in a programming language. Operation, training, and maintenance plans will all be drawn up so that developers know what they need to do throughout every stage of the cycle moving forward.

Once complete, development managers will prepare a design document to be referenced throughout the next phases of the SDLC.

SDLC Phases & Activities

Stage 4: **Development Stage**

The development stage is the part where developers actually write code and build the application according to the earlier design documents and outlined specifications.

Product program code is built per the design document specifications. In theory, all of the prior planning and outlined should make the actual development phase relatively straightforward.

Developers will follow any coding guidelines as defined by the organization and utilize different tools such as compilers, debuggers, and interpreters.

Environments : Dev, Test, UAT, Prod

https://www.unosquare.com/blog/what-is-a-uat-environment-why-it-matters/

SDLC Phases & Activities

Stage 5: Testing

Once the developers build the software, then it is deployed in the testing environment. Then the testing team tests the functionality of the entire system. In this fifth phase of SDLC, the testing is done to ensure that the entire application works according to the customer requirements.

After testing, the QA and testing team might find some bugs or defects and communicate the same with the developers. The development team then fixes the bugs and send it to QA for a re-test.

SDLC Phases & Activities

Stage 6: Deployment

After testing, the overall design for the software will come together. Different modules or designs will be integrated into the primary source code through developer efforts, usually by leveraging training environments to detect further errors or defects.

The information system will be integrated into its environment and eventually installed. After passing this stage, the software is theoretically ready for market and may be provided to any end-users.

SDLC Phases & Activities

Stage 7: Maintenance

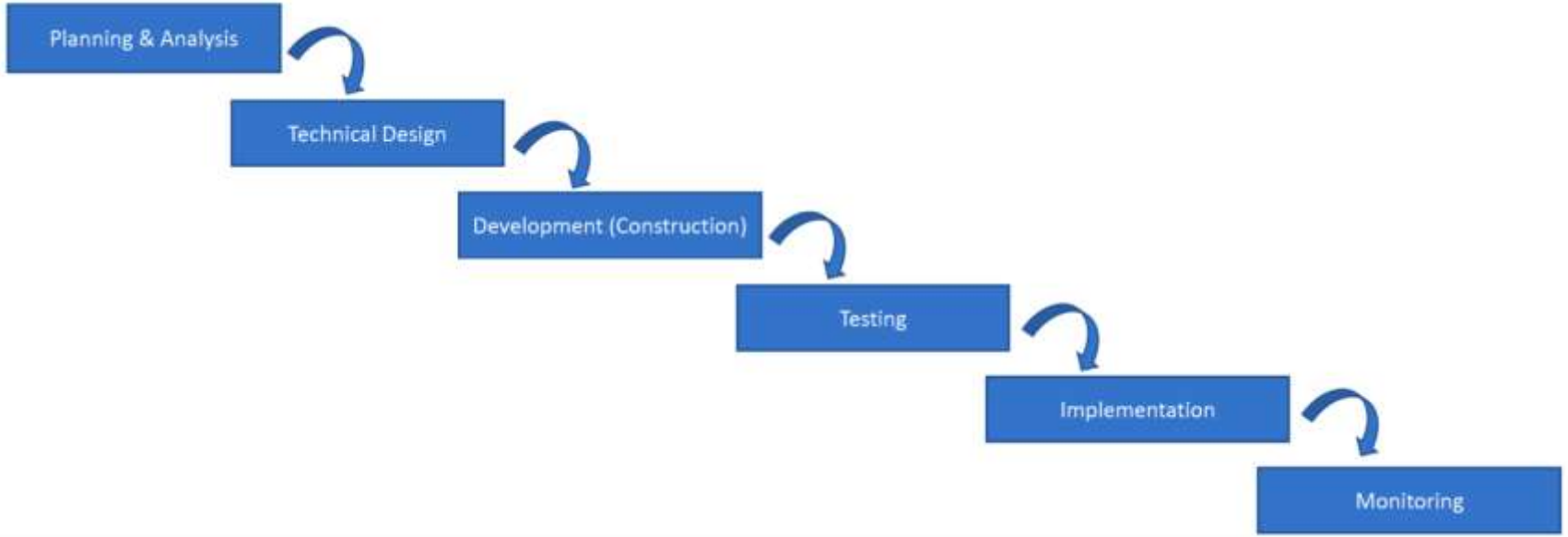
The SDLC doesn’t end when software reaches the market. Developers must now move into a maintenance mode and begin practicing any activities required to handle issues reported by end-users.

Furthermore, developers are responsible for implementing any changes that the software might need after deployment.

This can include handling residual bugs that were not able to be patched before launch or resolving new issues that crop up due to user reports. Larger systems may require longer maintenance stages compared to smaller systems

Waterfall Methodology

The Waterfall methodology — also known as the Waterfall model — is a sequential development process that flows like a waterfall through all phases of a project (analysis, design, development, and testing, for example), with each phase completely wrapping up before the next phase begins.

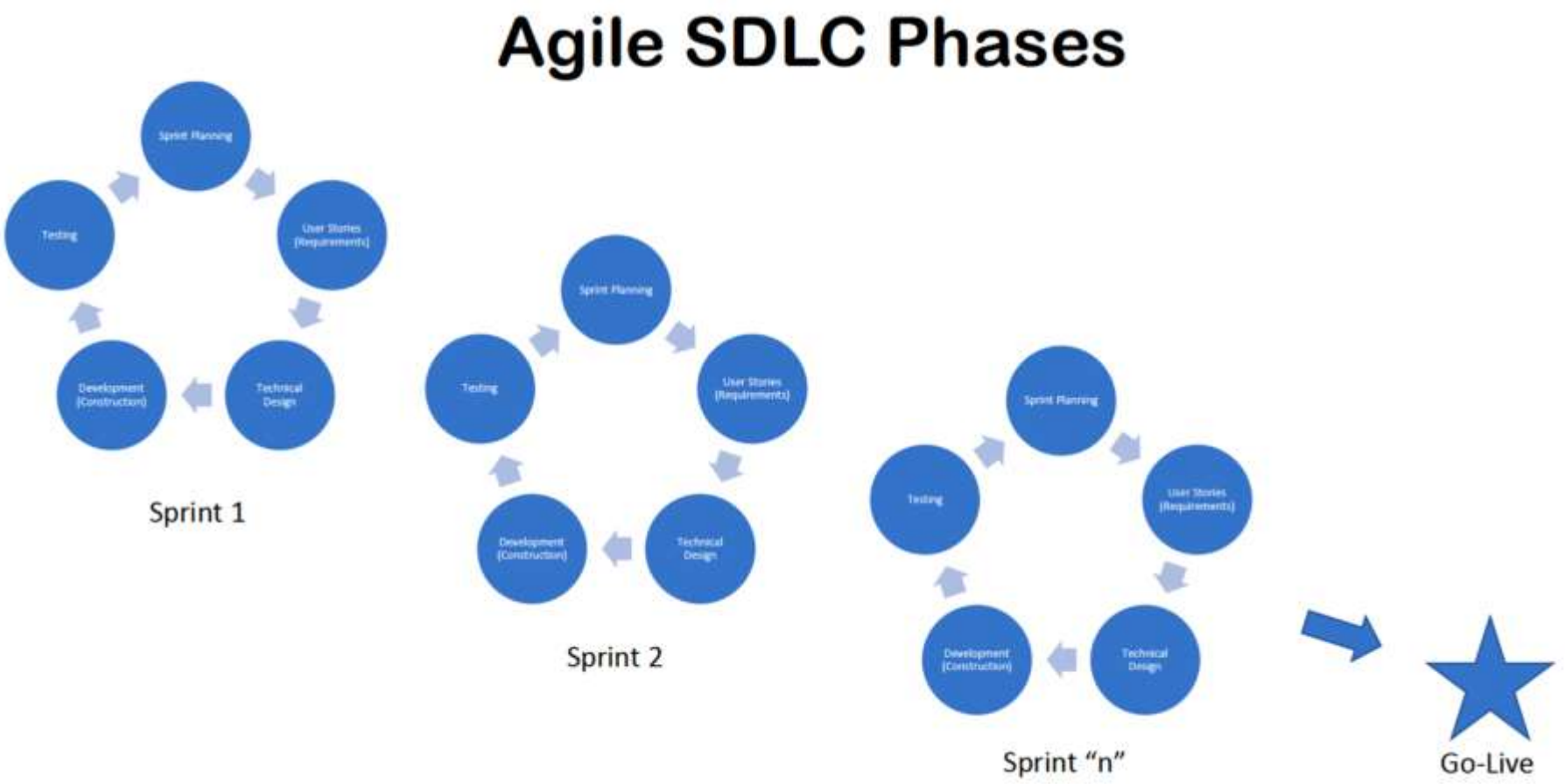


Waterfall Methodology

The waterfall model is the oldest of all SDLC methodologies. It’s linear and straightforward and requires development teams to finish one phase of the project completely before moving on to the next.

Each stage has a separate project plan and takes information from the previous stage to avoid similar issues (if encountered). However, it is vulnerable to early delays and can lead to big problems arising for development teams later down the road.

Agile Methodology



Agile Methodology

Agile Model is a combination of the Iterative and incremental model. This model focuses more on flexibility while developing a product rather than on the requirement.

In Agile, a product is broken into small incremental builds. It is not developed as a complete product in one go. Each build increments in terms of features. The next build is built on previous functionality.

In agile iterations are termed as sprints. Each sprint lasts for2-4 weeks. At the end of each sprint, the product owner verifies the product and after his approval, it is delivered to the customer.

Customer feedback is taken for improvement and his suggestions and enhancement are worked on in the next sprint. Testing is done in each sprint to minimize the risk of any failures.

Kanban and Scrum

https://asana.com/resources/waterfall-agile-kanban-scrum?utm_campaign=&utm_source=&utm_medium=&gclid=Cj0KCQjw2qKmBhCfARIsAFy8buKESNDSNAHZWs9XM_JW9XolGk25esAKSAApDTrLDmRCe5PKxpSUqrkaAqEjEALw_wcB&gclsrc=aw.ds

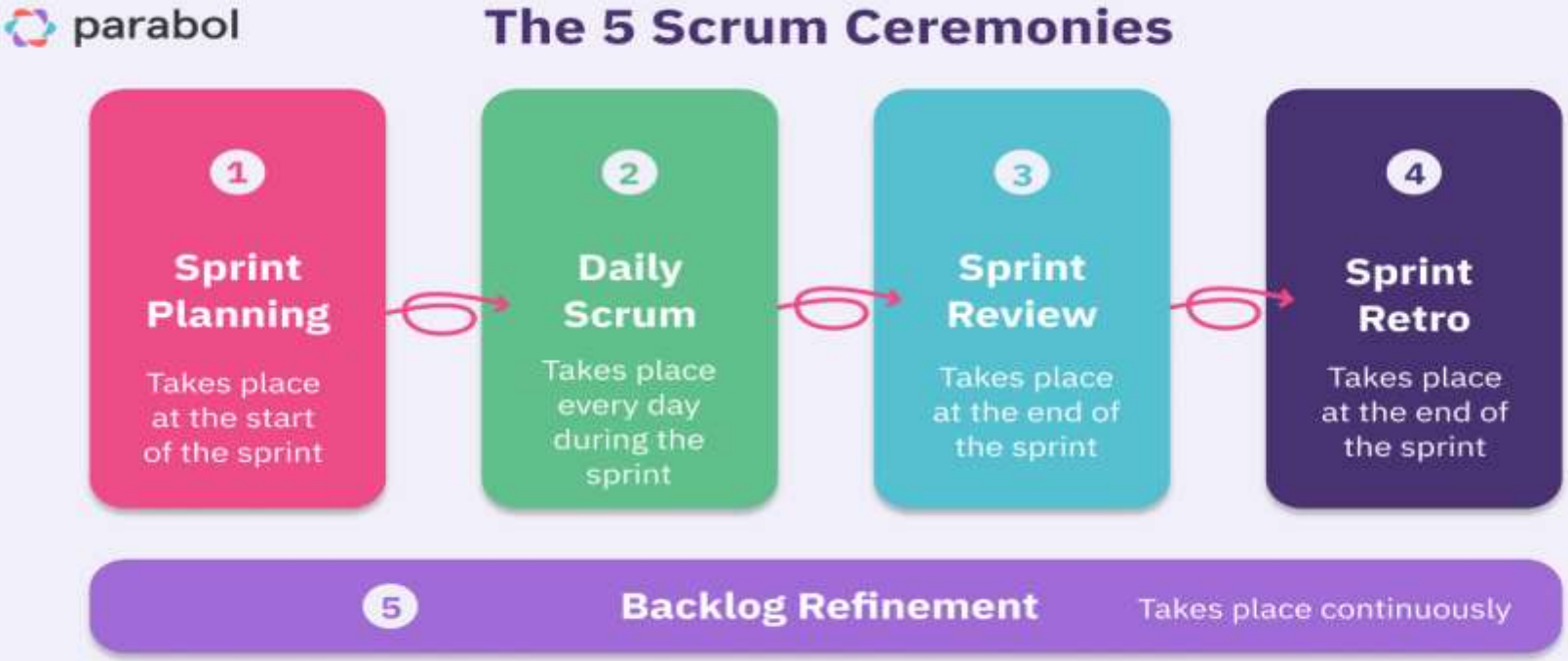
Scrum Ceremonies

The scrum ceremonies are

•**Sprint Planning**, **Daily Stand-Up**, **Sprint Review** and **Sprint Retrospective**

•Ceremonies in agile facilitate visibility, transparency, and collaboration.
•Each ceremony has a clear structure and objective.
•Clear communication, flexibility, and cultural alignment are the keys to successful ceremonies.

Agile Ceremonies



https://asana.com/resources/agile-scrum-ceremonies

Scrum artifacts

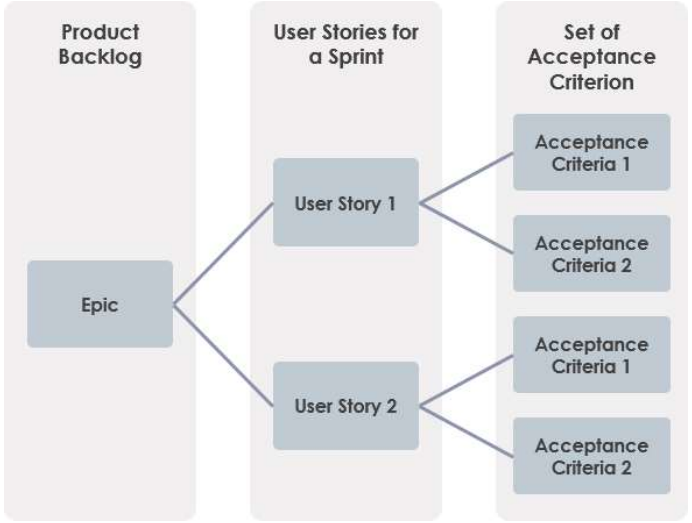
Project Management Tools

1. **JIRA**
2. ClickUp
3. Shortcut
4. Zoho
5. YouTrack
6. Asana …

DoD and AC

Definition of Done (DoD) is a list of requirements that a user story must adhere to for the team to call it complete.

While the **Acceptance Criteria** of a User Story consist of set of Test Scenarios that are to be met to confirm that the software is working as expected.



DoD and AC

Definition of Done (DoD) is a list of requirements that a user story must adhere to for the team to call it complete.

While the **Acceptance Criteria** of a User Story consist of set of Test Scenarios that are to be met to confirm that the software is working as expected.

Which Methodology

If you have a small application that only has 1 or 2 developers who are familiar with it, that's probably a good candidate for waterfall.

But a larger, more complicated application that's made up of multiple discrete modules might be a good candidate for agile development.


thank
you