

Software Development Life Cycle

Where coding turns into development

Table of Contents

01

...

SDLC

The software development lifecycle.

02

...

Methodologies

The ways we implement SDLC

03

...

Agile Frameworks

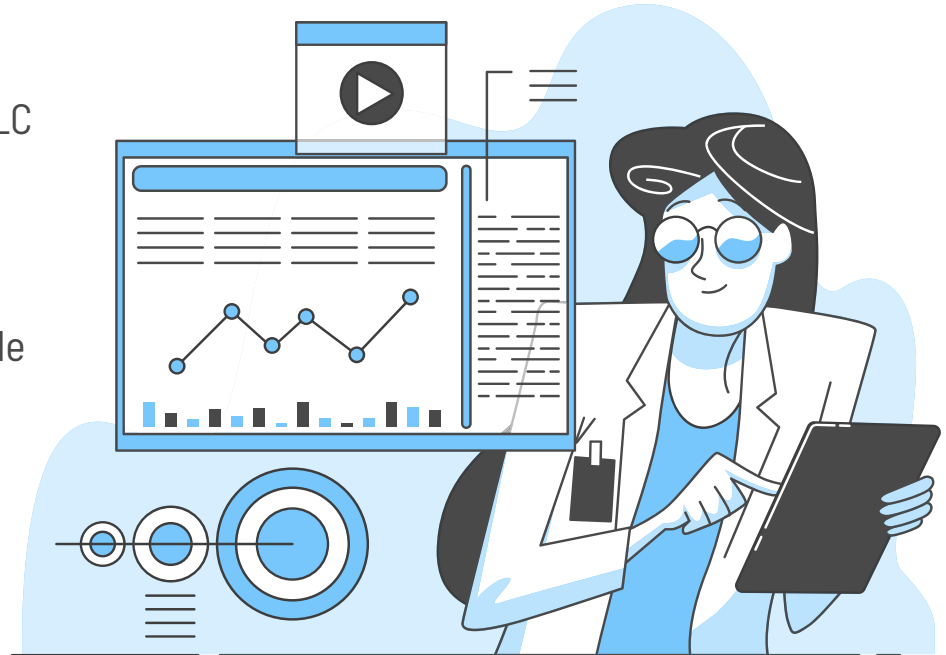
The ways we implement Agile

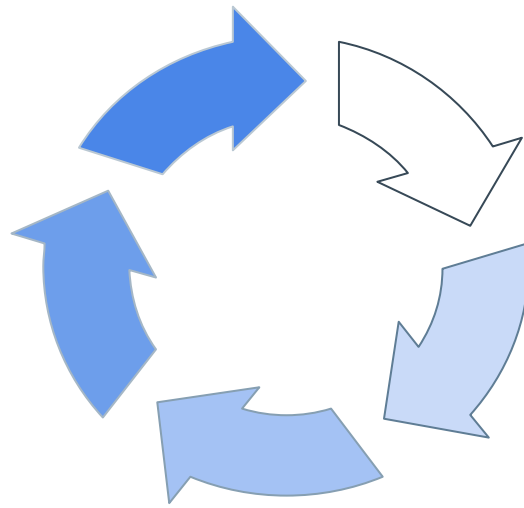
04

...

Story Points & Velocity

How we help ourselves plan





SDLC

The Software Development Life Cycle is the general process followed by software development. There are different methodologies to achieve this, including Waterfall and Agile.

...

1) Gather Requirements

2) Analysis

3) Design

4) Development

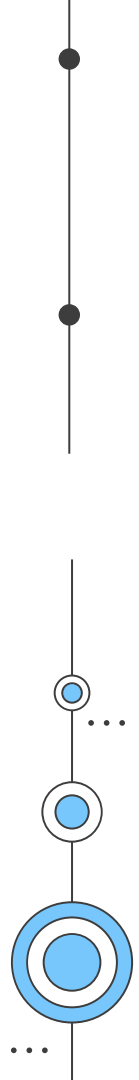
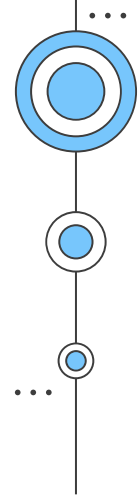
5) Testing

6) User Acceptance Testing

7) Release

8) Maintain

SDLC





SDLC Methodologies



01

Big Bang

Where most
developers begin.

02

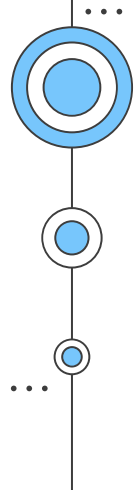
Waterfall

The classic method.

03

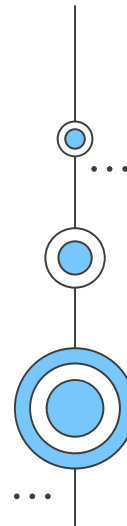
Agile

The modern method.



01 Big Bang

The Explosive Methodology




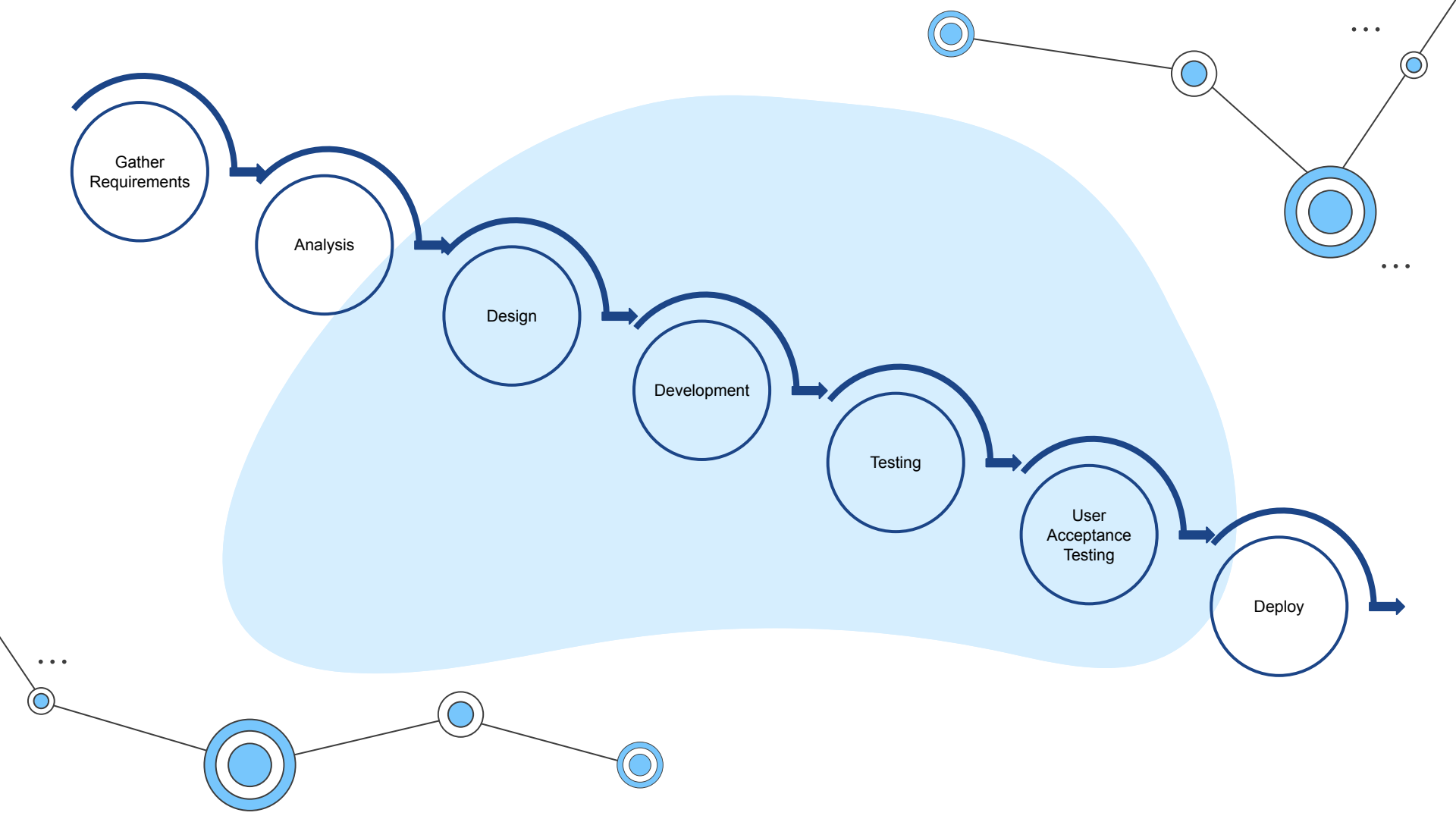


02

Waterfall

The Classic Methodology





Waterfall

Pros

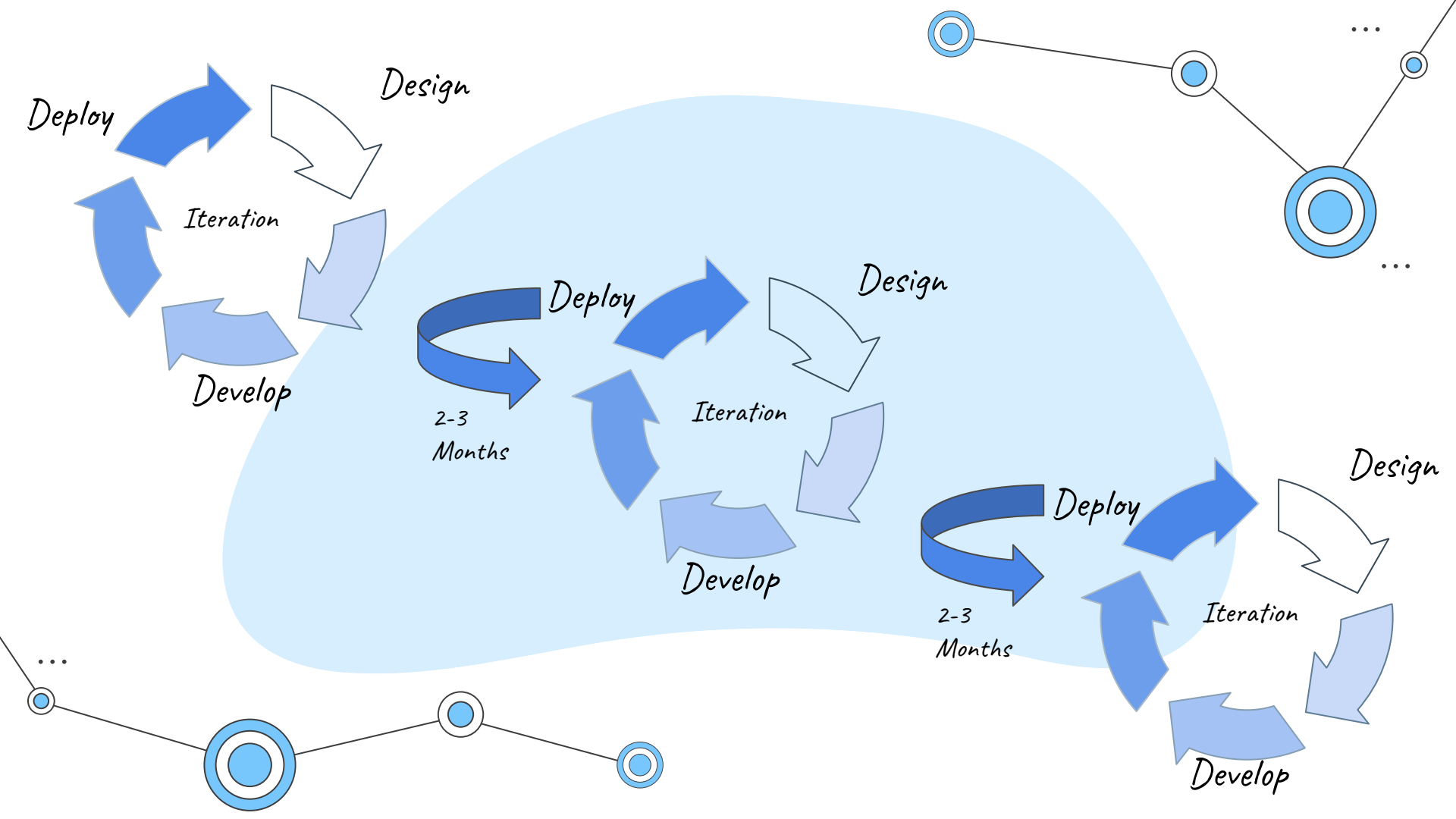
- Waterfall features clear, well defined steps.
- Waterfall is intuitive and doesn't require specialized knowledge.
- Waterfall has a clear and concrete end goal early on.
- Waterfall allows for clean transfer of information from step to step.

Cons

- Waterfall makes changes difficult as it always moves forward.
- Waterfall excludes the client beyond the initial gather requirements phase.
- Waterfall delays testing until completion, leaving potential problems unnoticed until the latter half of a project.

03 Agile

The Modern Methodology



Agile

Pros

- Agile allows for a team to quickly respond to change.
- Agile allows for uncertainty at the beginning.
- Agile has faster review cycles compared to Waterfall.
- Agile has greater flexibility in releases.

Cons

- Agile's increased flexibility can lead to bad behaviours.
- Agile requires more knowledge to use properly.
- Agile has a lack of predictability.

Agile Frameworks



Scrum



Kanban

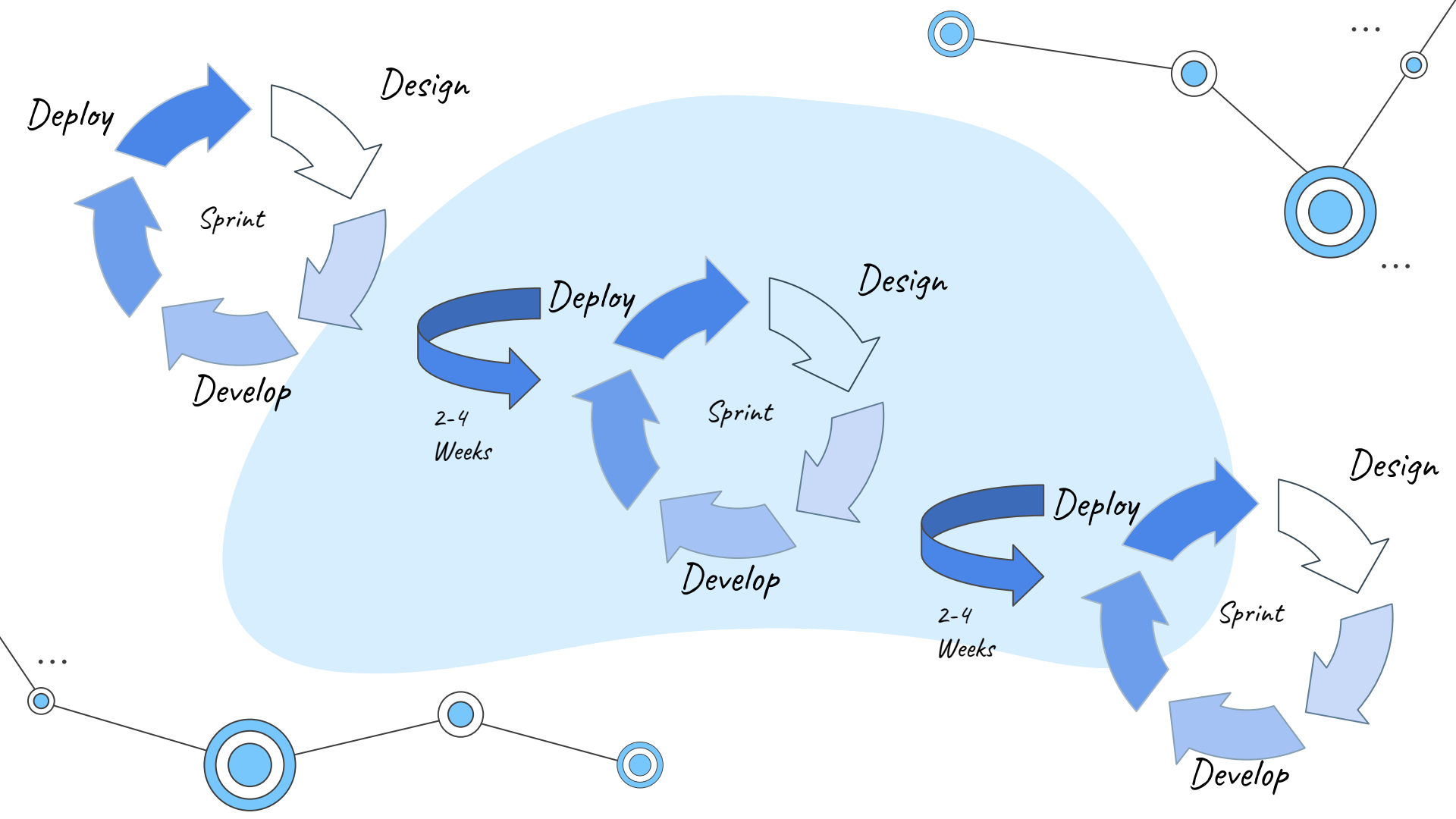


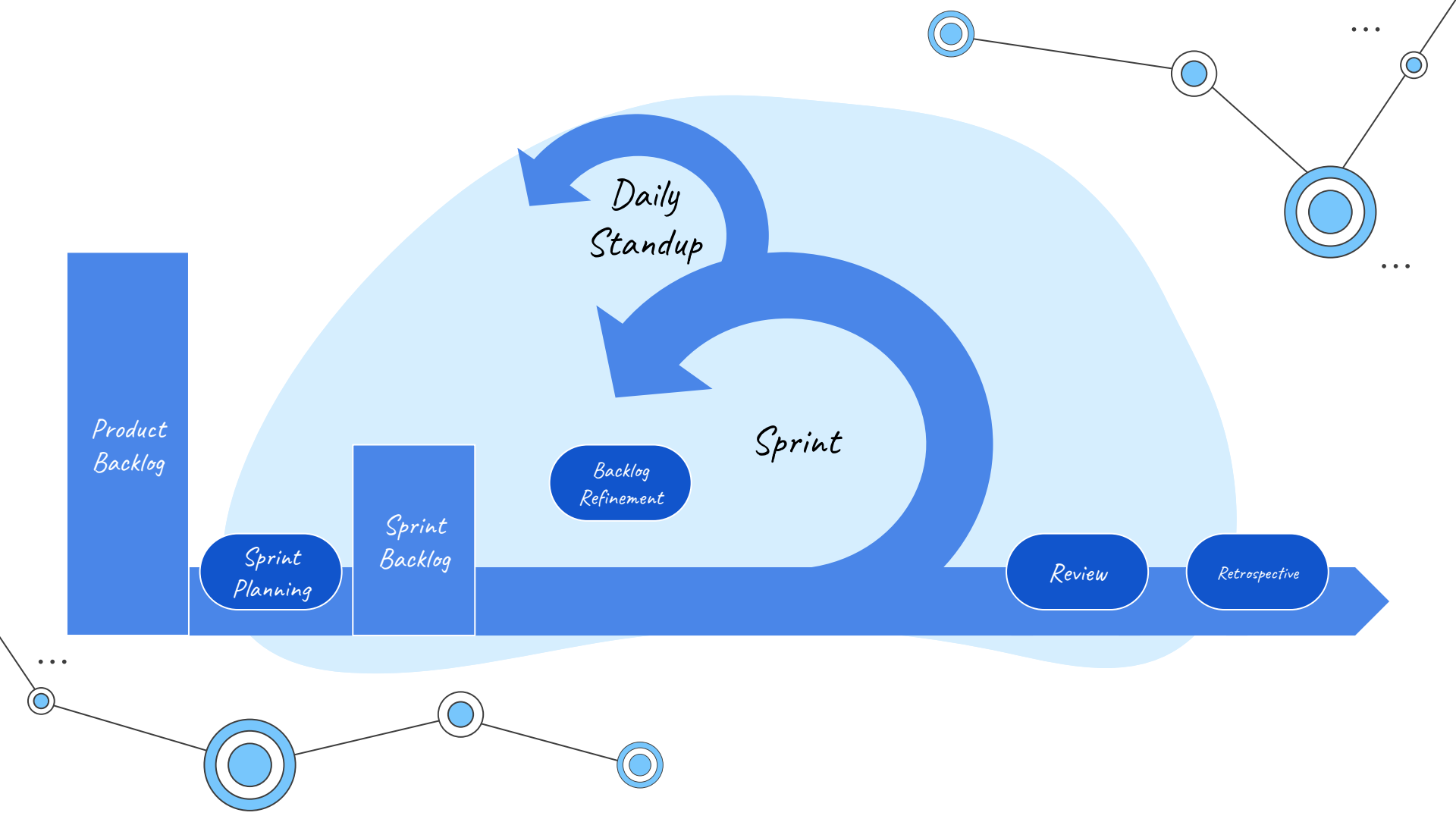
eXtreme
Programming

Scrum

Agile Framework







Values

Teams following scrum are expected to learn and explore the following values

Commitment

Team members personally commit to achieving team goals

Courage

Team members do the right thing and work on tough problems.

Focus

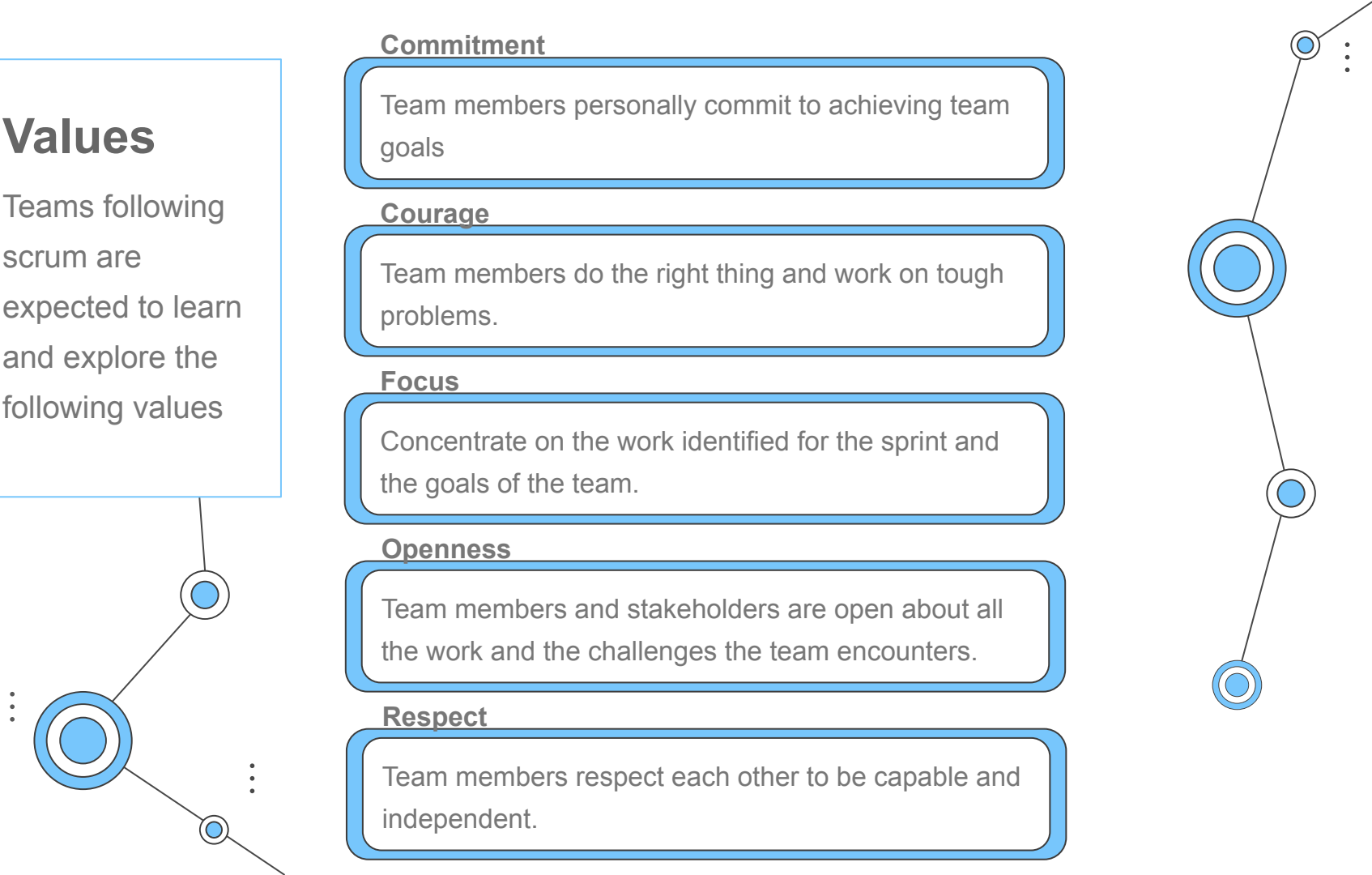
Concentrate on the work identified for the sprint and the goals of the team.

Openness

Team members and stakeholders are open about all the work and the challenges the team encounters.

Respect

Team members respect each other to be capable and independent.



Principles

The following principles underpin the empirical nature of scrum

Transparency

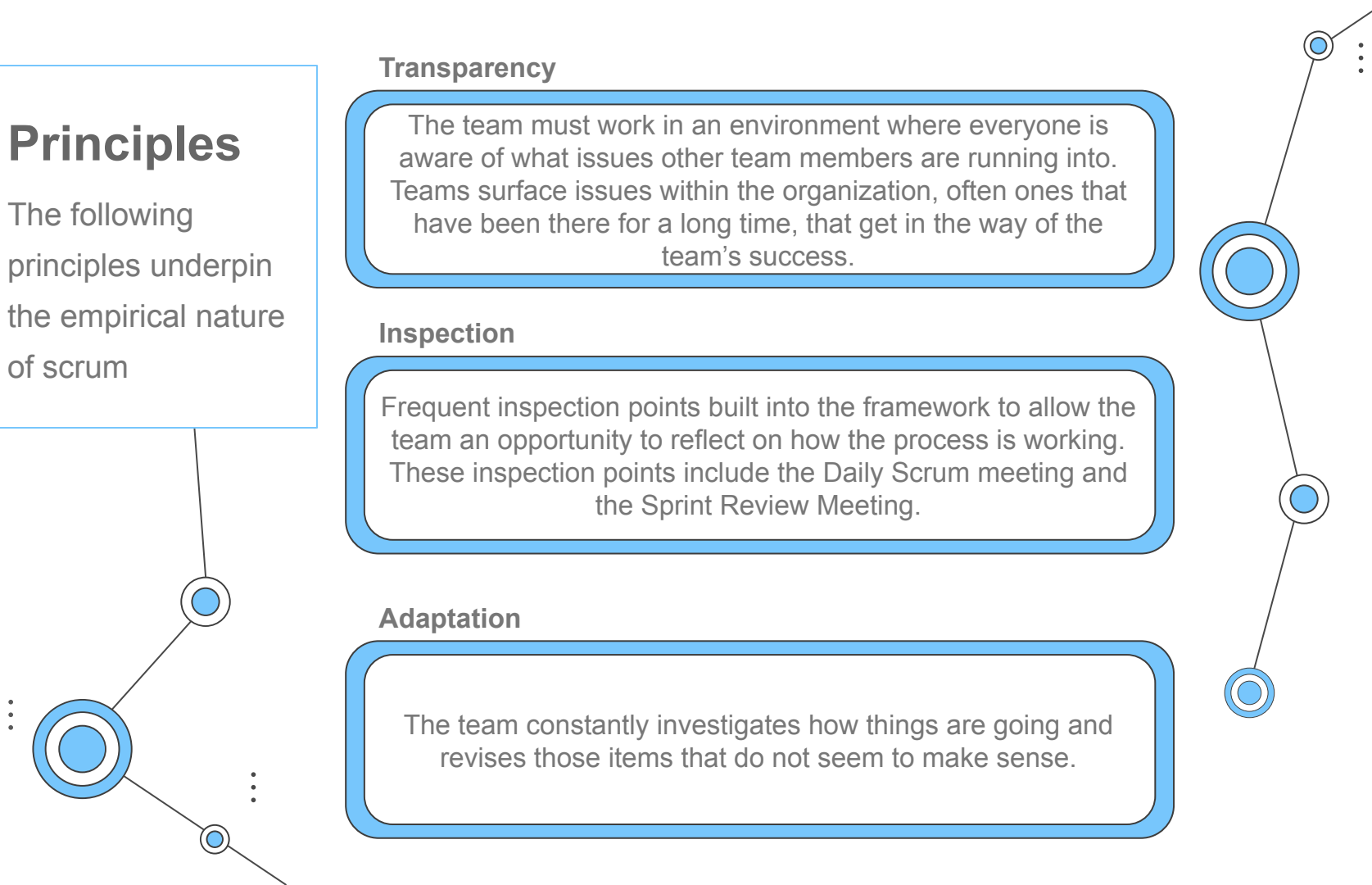
The team must work in an environment where everyone is aware of what issues other team members are running into. Teams surface issues within the organization, often ones that have been there for a long time, that get in the way of the team's success.

Inspection

Frequent inspection points built into the framework to allow the team an opportunity to reflect on how the process is working. These inspection points include the Daily Scrum meeting and the Sprint Review Meeting.

Adaptation

The team constantly investigates how things are going and revises those items that do not seem to make sense.



Roles

Product Owner

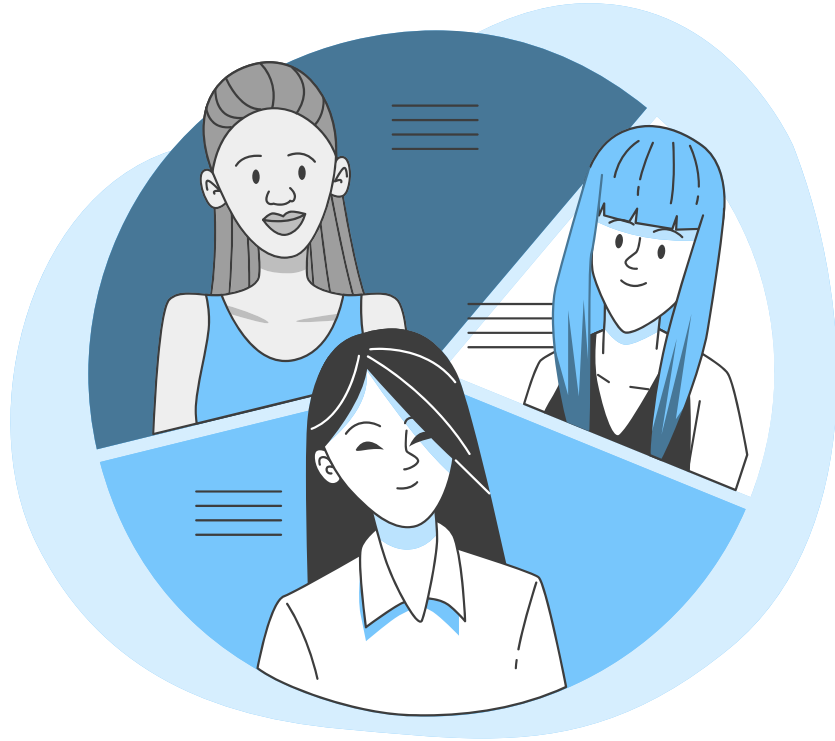
Manages the Backlog

Scrum Master

Ensures ceremonies are followed

Development Team

Delivers the product



Scrum

Pros

- Scrum makes it easier to find and rectify mistakes.
- Scrum enforces daily standups which allows you to identify problems quickly.
- Scrum allows for constant feedback with short sprints, letting you to adapt to changing client needs and input.
- Scrum involves clients throughout, leading to more desired and accurate outcomes.

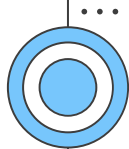
Cons

- Scrum lacks firm final deadlines which can result in feature creep.
- Scrum requires a strong commitment from the team members, and has added pressure on each member.
- Scrum lacks a firm time limit and cost valuations, which can cause uncertainty.

Kanban

Agile Framework





...



...



...

Kanban

Pros

- Kanban is event-driven instead of timeboxed.
- Kanban board can be added to whenever there is capacity.
- Kanban allows for specialists, whereas cross-functional teams are optional.
- Kanban board is persistent.

Cons

- Kanban boards are easy to become overcomplicated.
- Kanban boards if allowed to become outdated cause issues in development.
- Kanban lacks a timing element, causing potential issues with project timelines.

Scrumban

Agile Framework



eXtreme Programming

Agile Framework



User Epics

Identify unique sets of user stories, and combine them into epics.

If multiple user stories/use cases relate to a 'Manager' user, then they should be collected into the manager epic.



Paired Programming



Paired programming is the process of two developers working on a specific piece of code or feature together.

Pilot

The pilot is focused on the immediate needs of the code and is the one who writes it.

Navigator

The navigator should be focused on the bigger picture and how it will fit in the greater project.

Story Points & Velocity

Agile Toolset



Story Points & Hours



Story points are used to help give us an estimate, determine difficulty, and prioritization of our use cases.

Different groups might use different methods, the important part is that the group collaborates and everyone understands the method used.

Velocity

Velocity is used hand in hand with story points. Velocity is the number of story points completed in a sprint. By understanding our velocity we can make better long term planning for our project.



eXtreme Programming

Pros

- XP is fast, it is a very fast paced environment with CI/CD.
- XP is open, communication and work environment is geared towards helping make everyone aware and making progress.
- XP reduces costs, with a incredibly small feedback loop and rapid ability to change it keeps things in line with customer expectations.
- XP teamwork plays a large role, especially with paired programming coming into use. Which tends to increase employee commitment and satisfaction.

eXtreme Programming

Cons

- XP is code over design, incremental design focuses on just getting the code working in the simplest way first rather than focusing on the design aspect.
- XP can be difficult to implement, it is highly specific, and requires the client to be embedded in the team. Both of those can be tough hurdles.
- XP has poor documentation. Because of the rapid changes it is exceedingly difficult to keep things documented, and that documentation up to date.
- XP is stressful, there are tight deadlines and it is very fast paced, which can make things difficult.

Thanks!

Do you have any questions?

CREDITS: This presentation template was created by [Slidesgo](#), including icons by [Flaticon](#), infographics & images by [Freepik](#) and illustrations by [Stories](#)

Please keep this slide for attribution

