

PMG 622

Agile Project Management

**LECTURE 04: IMPLEMENTING AGILE:
DELIVERING IN AN AGILE ENVIRONMENT**

[Part 1 of 2]



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

Eng. Dr. Akram Mustafa Suliman Ali,
CEng CMarEng MIMarEST, PMP®, ASQ CSSBB

Eng. Dr. Akram Ali

Scrum roles, artifacts & ceremonies

- SCRUM ARTIFICATS

- Product Backlog
- Sprint Backlog
- Increment

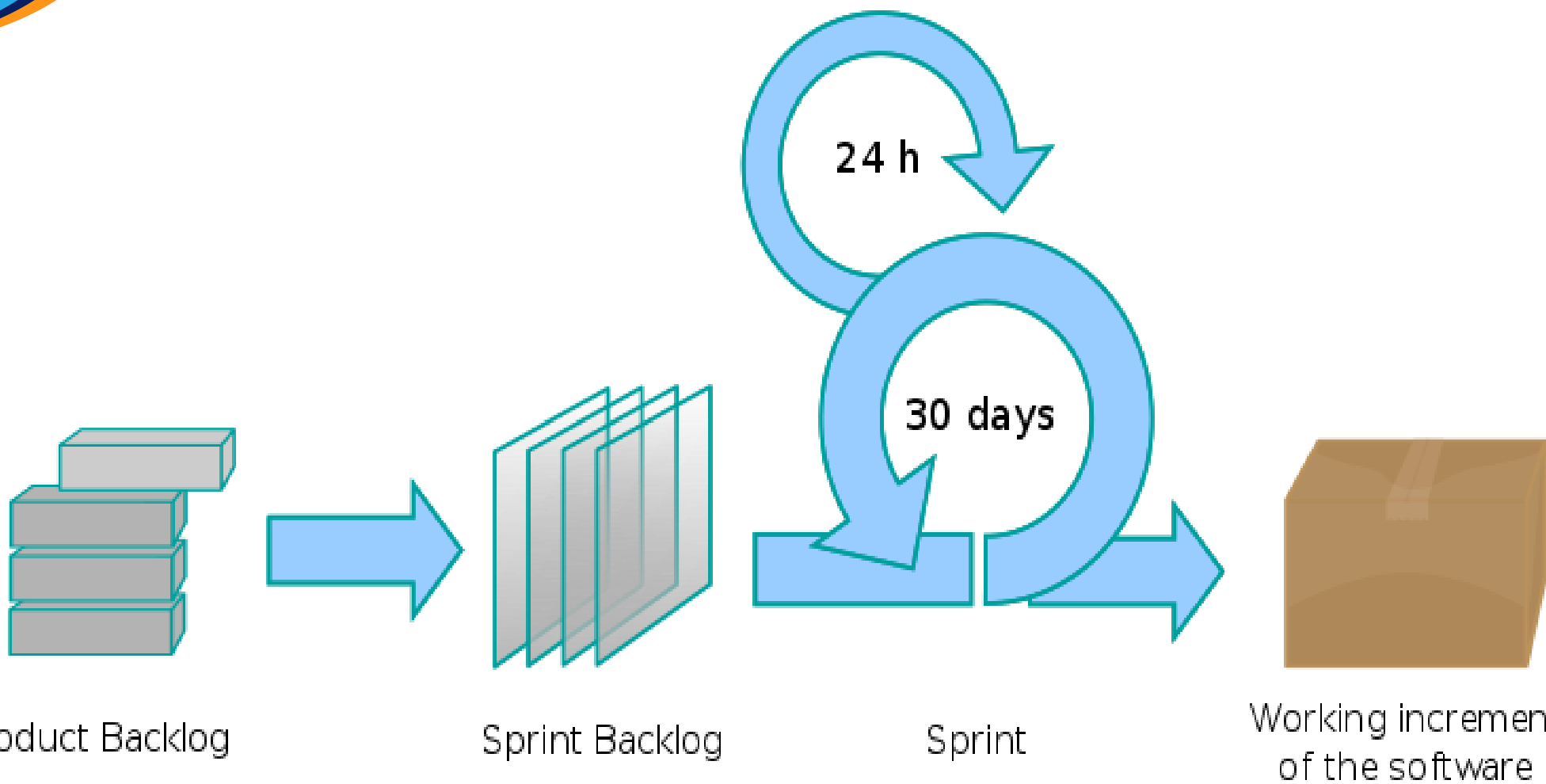
- SCRUM CERMONIES

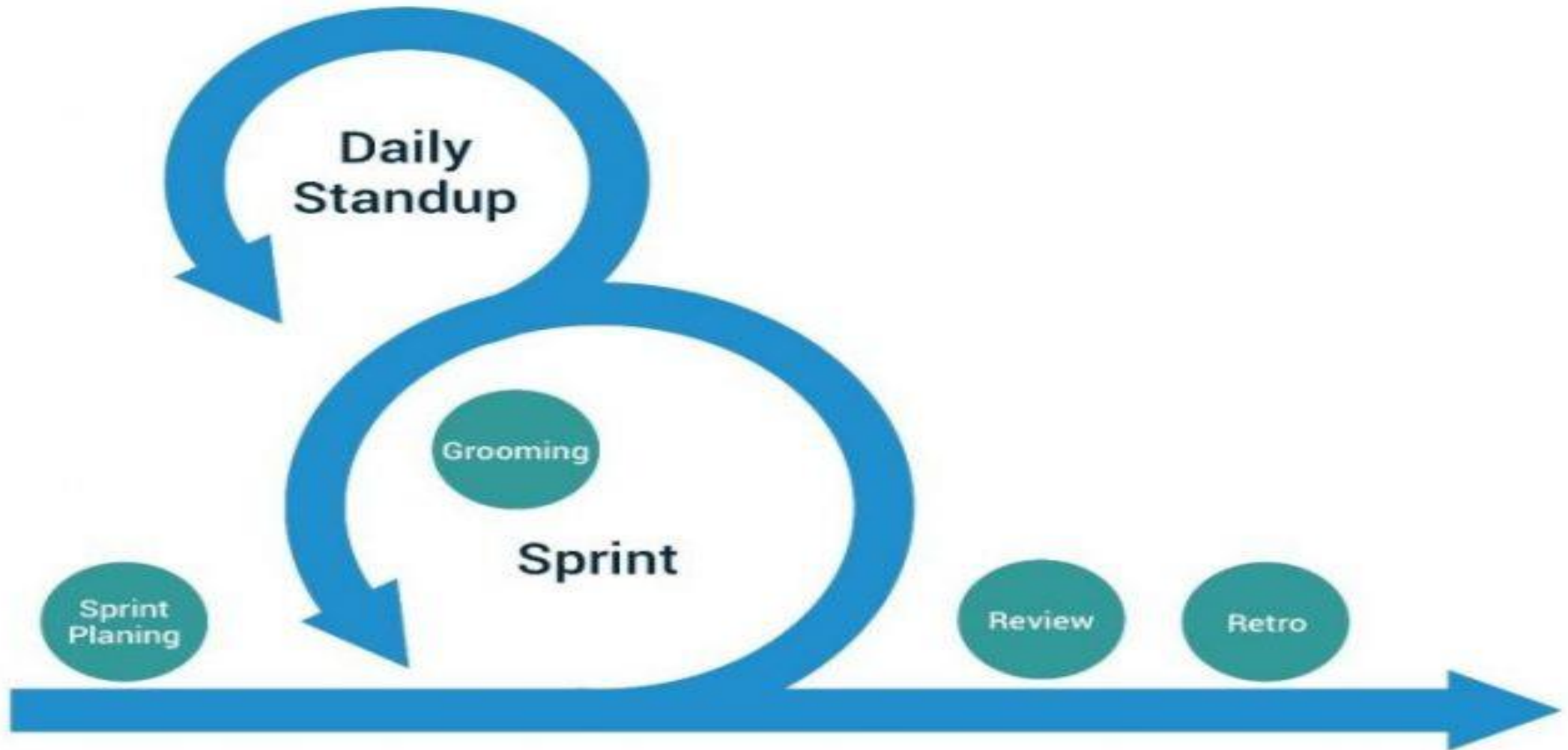
- Sprint Planning.
- Daily **Scrum**.
- Sprint Review.
- Sprint Retrospective.

- SCRUM ROLES

- Product Owner
- Development Team Members
- Scrum Master

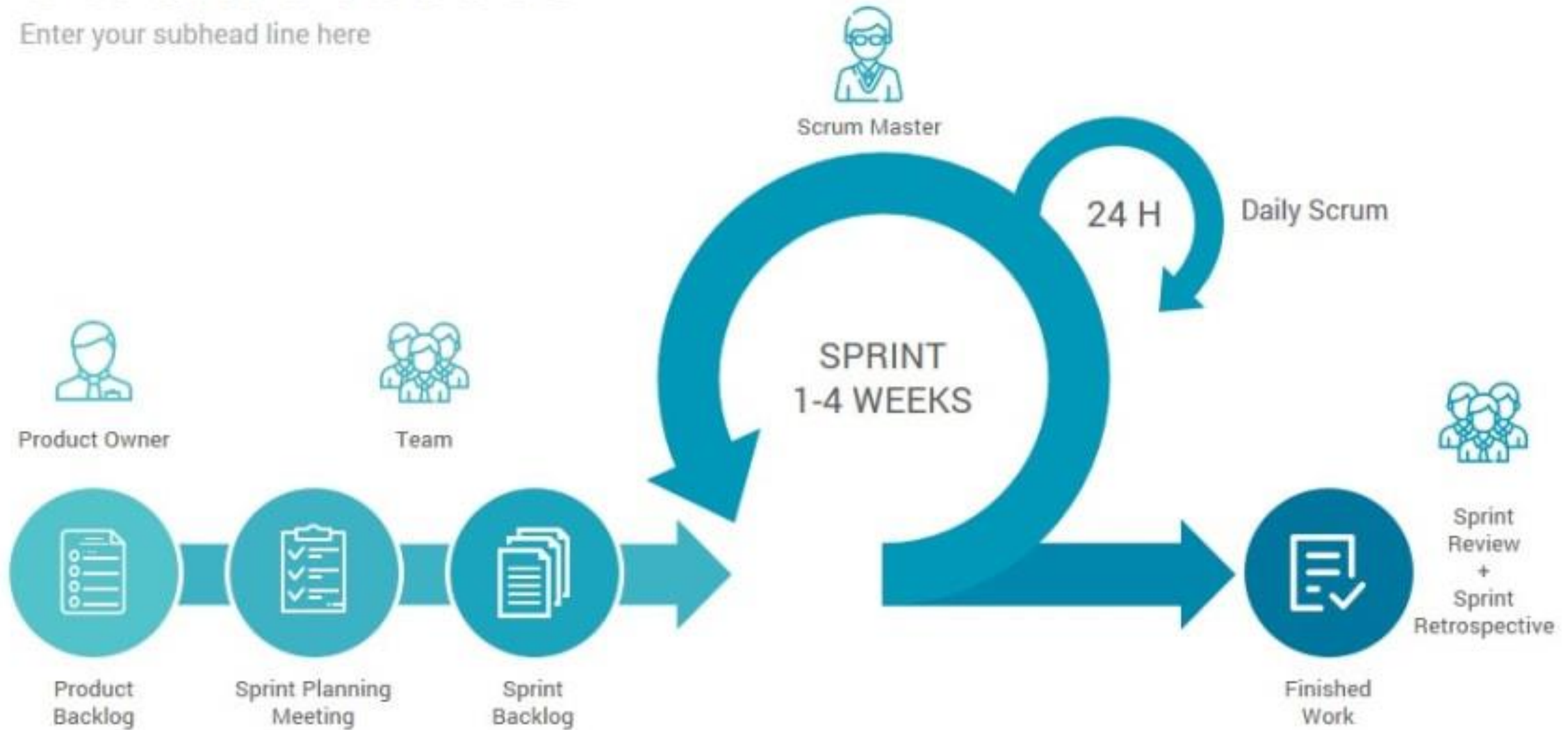




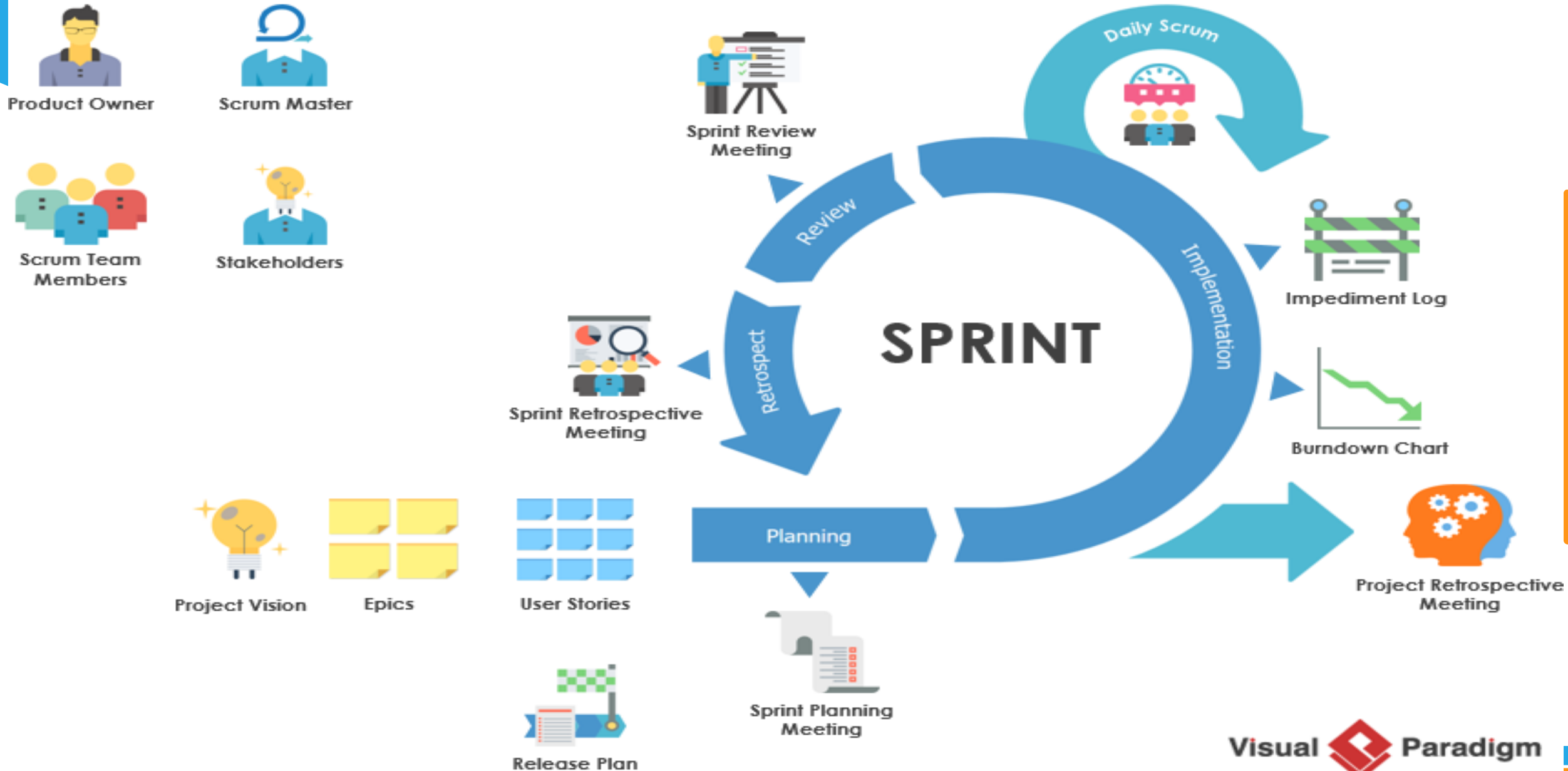


Scrum Process

Enter your subhead line here



The Agile – Scrum Framework



DELIVERING IN AN AGILE ENVIRONMENT

Current and last lecture overview

- **IMPLEMENTING AGILE: CREATING AN AGILE ENVIRONMENT**
- **IMPLEMENTING AGILE: DELIVERING IN AN AGILE ENVIRONMENT**



CHARTER THE PROJECT AND THE TEAM

- Project Charter
- Team Charter



PROJECT CHARTER

- Why this project matters **The WHY**
- Where the team is headed **The HOW**
- What the project objective is **The WHAT**
- Why are we doing this project? This is the project vision.
- Who benefits and how? This may be part of the project vision and/or project purpose.
- What does done mean for the project? These are the project's release criteria.
- How are we going to work together? This explains the intended flow of work.

TEAM CHARTER

- Team norms
- Understanding of how to work together
- Teams do not need a formal process for chartering as long as the teams understand how to work together.
- Some teams benefit from a team chartering process.
 - ☐ Team values, such as sustainable pace and core hours;
 - ☐ Working agreements, such as what “**ready**” means so the team can take in work; what “**done**” means so the team can judge completeness consistently; respecting the **time-box**; or the use of **work-in-process** limits;
 - ☐ Ground rules, such as one person talking in a meeting; and
 - ☐ Group norms, such as how the team treats meeting times.



COMMON AGILE PRACTICES

RETROSPECTIVES [1/2]

- The single most important practice is the retrospective because it allows the team to learn about, improve, and adapt its process
- Retrospectives help the team learn from its previous work on the product and its process
- One of the principles behind the **Agile Manifesto** is: “At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.”
- Teams benefit from allocating enough time to learn, either from an **interim retrospective** or an **end-of-the-project retrospective**



RETROSPECTIVES [2/2]

- Teams need to learn about their **work product** and **work process**.
- First and foremost, a retrospective is not about **blame**;
- The retrospective is about looking at the **qualitative (people's feelings)** and
- **quantitative (measurements) data**, then using that data to find root causes, designing countermeasures, and developing action plans.
- The project team may end up with many action items to remove **impediments**
- A facilitator from the team leads them through an activity to rank the importance of each improvement item.
- Once the improvement items are ranked by the team, the team chooses the appropriate number to work on for the next iteration (or adds work to the flow if flow-based).



Team members may decide to retrospect at these key times:

- Many teams use iterations—especially 2-week iterations—because the iteration prompts a demonstration and a retrospective at the end
 - When the team completes a release or ships something. It does not have to be a monumental increment. It can be any release, no matter how small.
 - When more than a few weeks have passed since the previous retrospective.
 - When the team appears to be stuck and completed work is not flowing through the team.
 - When the team reaches any other milestone.



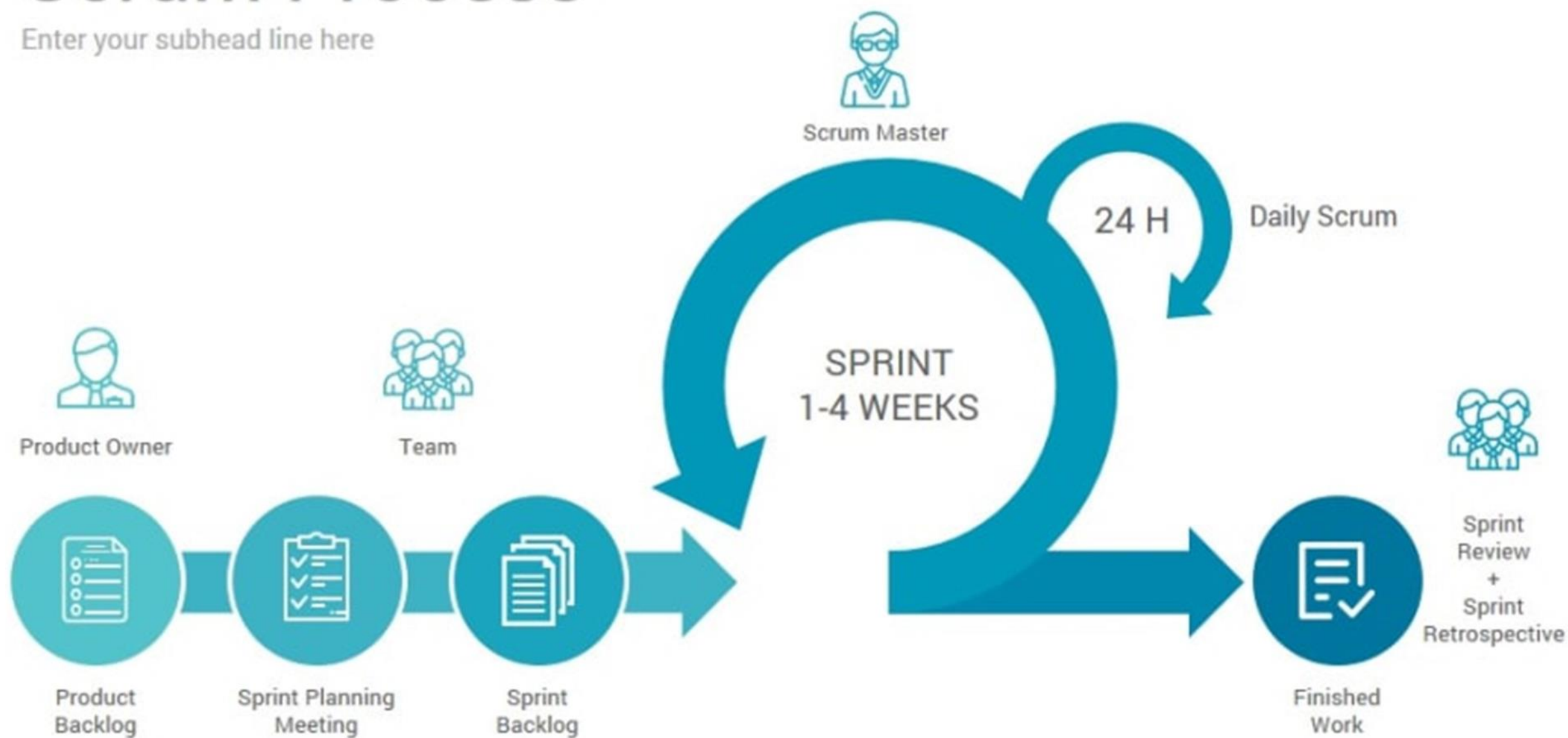
BACKLOG PREPARATION

- The backlog is the **ordered list** of **all the work**, presented in **story** form, for a team.
- There is no need to create all of the stories for the entire project before the work starts—only enough to understand the first release in broad brushstrokes and then sufficient items for the next iteration.
- **Product owners** might produce a **product roadmap** to show the anticipated sequence of deliverables over time. The product owner re-plans the roadmap based on what the team produces.

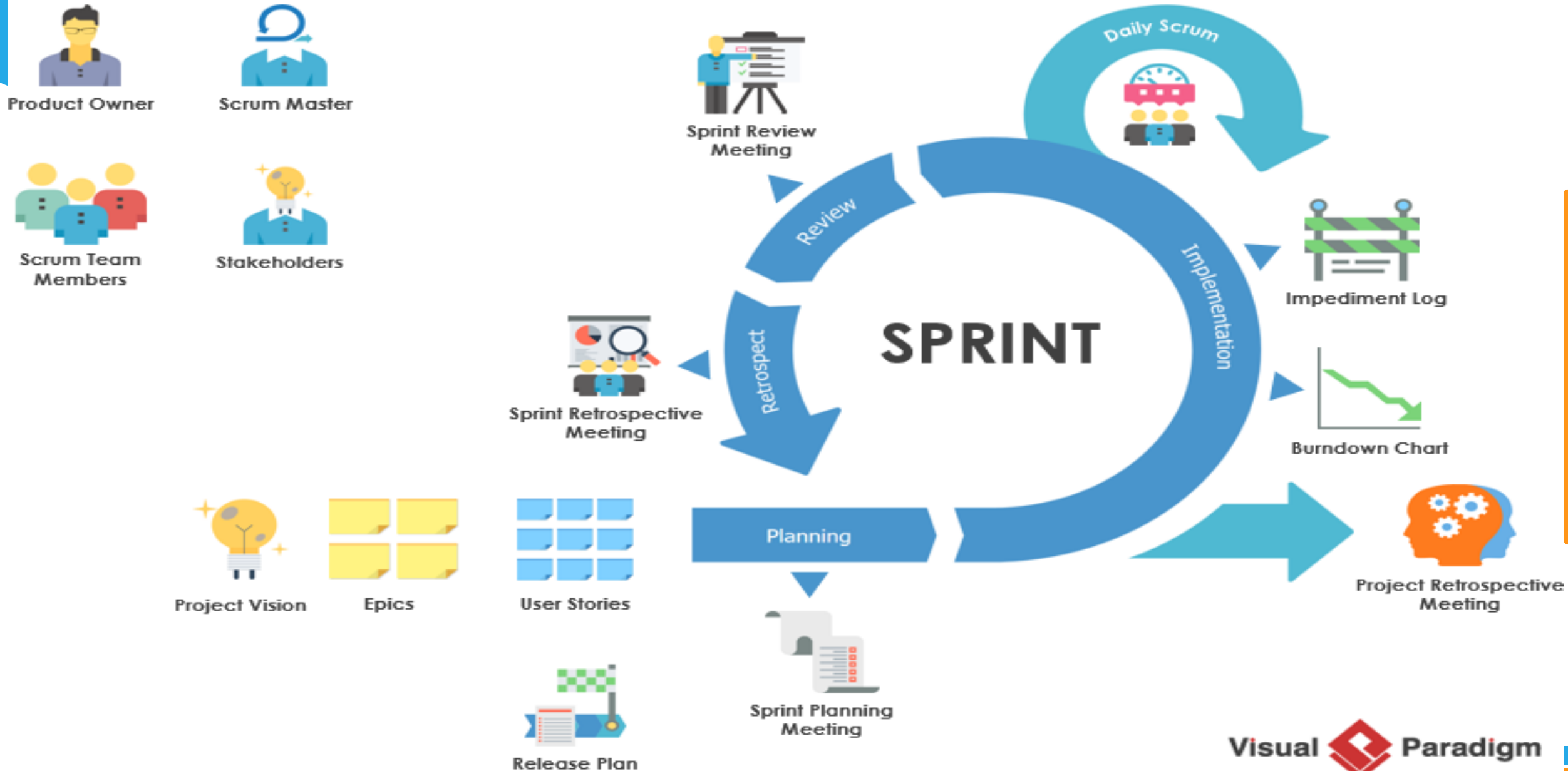


Scrum Process

Enter your subhead line here



The Agile – Scrum Framework



BACKLOG REFINEMENT

- In **iteration-based agile**, the **product owner** often works with the team to prepare some **stories** for the upcoming iteration during one or more sessions in the middle of the iteration.
- The **purpose** of these meetings is to **refine** enough **stories** so the **team** understands **what** the stories are and **how large** the stories are in relation to each other.
- Allow the **product owner** to **present story ideas** to the **team** and for the team to learn about the potential challenges or problems in the stories.
- If the **product owner** is **unsure** of the **dependencies**, the product owner can request the **team** to **spike** the feature in order to understand the risks.

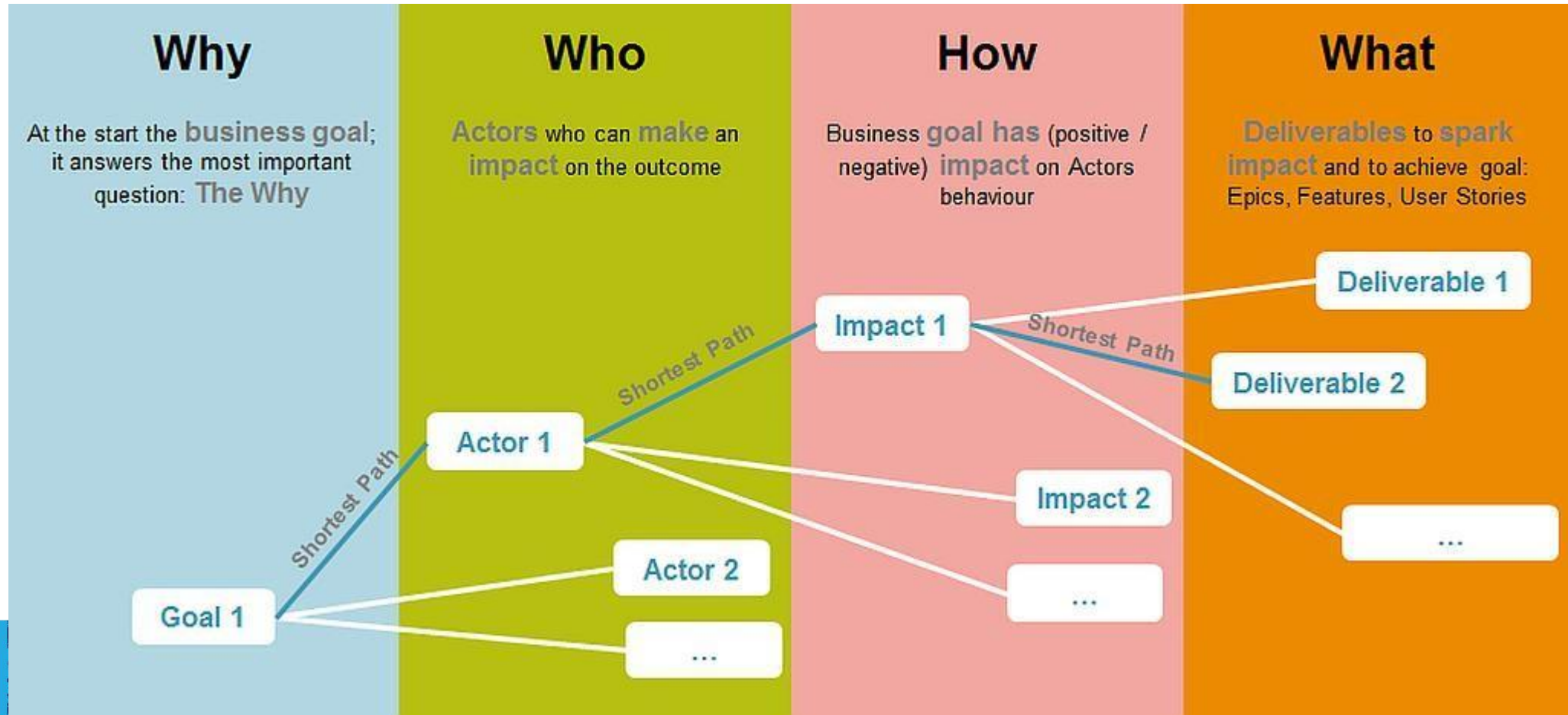


tip

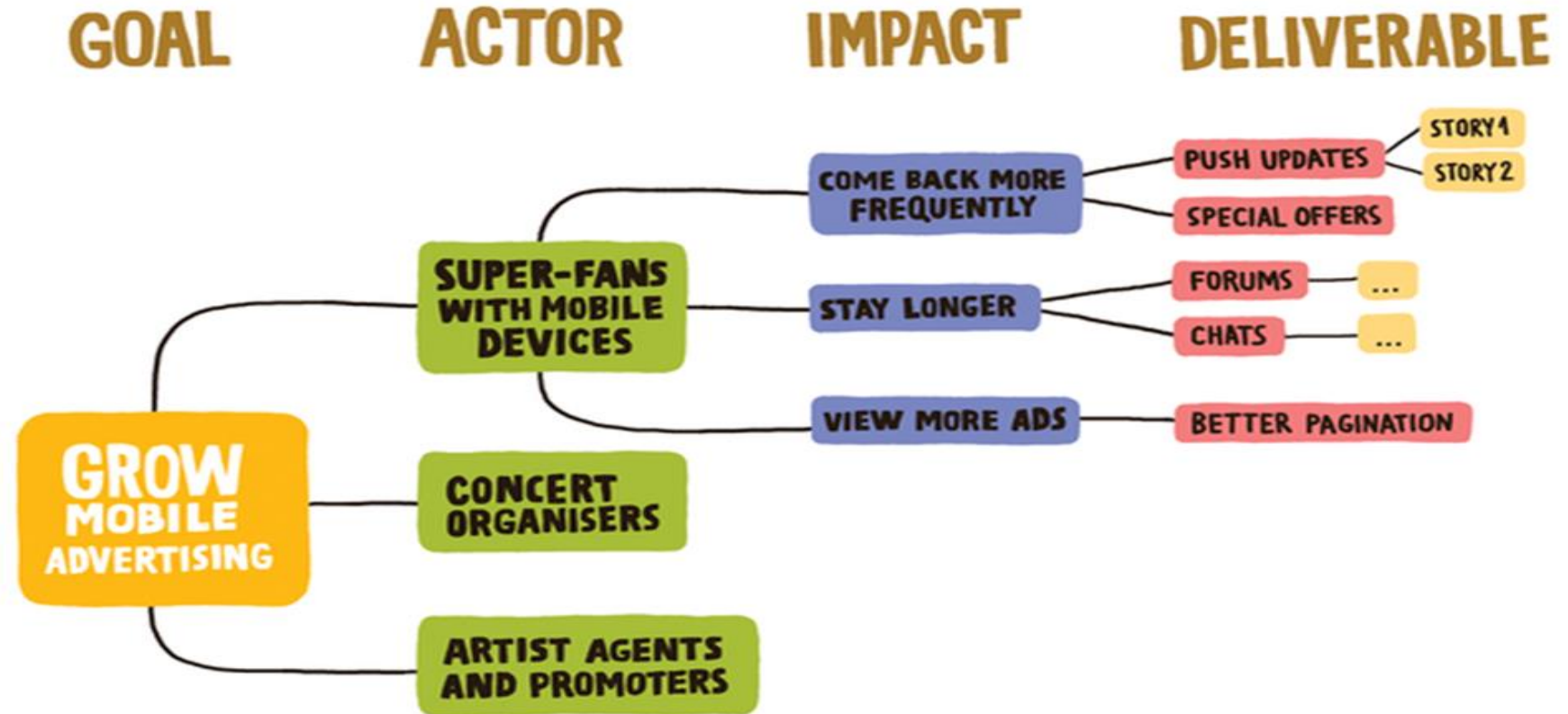
- Consider using **impact mapping** to see how the product fits together.
- Under normal circumstances, the **product owner** leads this work. A **servant leader** can **facilitate** any necessary meetings as a way of serving the project.



Impact Map Example



Impact Map Example



how long the refinement should be

- Many **iteration-based agile** teams use a **time-boxed** 1-hour discussion midway through a 2-week iteration.
- **Just-in-time** refinement for **flow-based agile**. The team takes the next card off the to-do column and discusses it.
- **Multiple refinement discussions** for **iteration-based agile teams**. Teams can use this when they are **new to the product**, the **product area**, or the **problem domain**.



ways to conduct backlog preparation and refinement meetings

There are many ways for the **product owner** to conduct **backlog preparation** and **refinement** meetings, including for example:

- Encourage the team to work as **triads** of developer, tester, business analyst/product owner to discuss and write the story.
- Present the **overall story concept** to the team. The team discusses and **refines** it **into** as many **stories** as required.
- Work with the team to find various ways to explore and write the stories together, making sure all of the **stories are small enough** so the team can produce a **steady flow of completed work**. Consider becoming able to complete a story at least once a day.



DAILY STANDUPS

- Teams use standups to **micro-commit** to each other, **uncover problems**, and **ensure the work flows smoothly** through the team.
- **Time-box** the standup to no longer than **15 minutes**. The team “walks” the **Kanban** or **task board** in some way, and **anyone** from the team can **facilitate** the standup.



iteration-based agile daily stand-up questions

- What did I complete since the last standup?
- What am I planning to complete between now and the next standup?
- What are my impediments (or risks or problems)?



flow-based agile daily stand-up questions

- What do we need to do to advance this piece of work?
- Is anyone working on anything that is not on the board?
- What do we need to finish as a team?
- Are there any bottlenecks or blockers to the flow of work?



Standup anti-patterns

- Standups become status meetings. Teams who have traditionally worked in a predictive environment may tend to fall into this anti-pattern since they are used to providing a status.
- The team begins to solve problems as they become apparent. Standups are for realizing there are problems —not for solving them. Add the issues to a parking lot, and then create another meeting, which might be right after the standup, and solve problems there.
- Standups can be very useful, provided the nature of the team's work requires intense collaboration.



DEMONSTRATIONS/REVIEWS

- As the team completes the features usually in the form of user stories, the team periodically demonstrates the working product.
- The product owner sees the demonstration and accepts or declines stories.
- In iteration-based agile, the team demonstrates all completed work items at the end of the iteration.
- In flow-based agile, the team demonstrates completed work when it is time to do so, usually when enough features have accumulated into a set that is coherent.



guidelines

- A fundamental part of what makes a project agile is the frequent delivery of a working product.
- Teams, including the product owner, need feedback to decide how early to ask for product feedback
- Demonstrate whatever the team has as a working product at least once every 2 weeks



PLANNING FOR ITERATION-BASED AGILE

- Teams consider their story size so they do not try to commit to more stories than there is team capacity to complete within one iteration.
- When teams have a reduced capacity, they will only plan for work that meets that capacity.
- Teams estimate what they can complete, which is a measure of capacity
- Agile teams do not plan just once in one single chunk. Instead, agile teams plan a little, deliver, learn, and then replan a little more in an ongoing cycle.



EXECUTION PRACTICES THAT HELP TEAMS DELIVER VALUE

- If the team does not pay attention to quality, it will soon become impossible to release anything rapidly.
- The following technical practices, many of which come from eXtreme Programming, may help the team to deliver at their maximum speed:
 - **Continuous integration**
 - **Test at all levels**
 - **Acceptance Test-Driven Development (ATDD)**
 - **Test-Driven Development (TDD) and Behavior-Driven Development (BDD)**
 - **Spikes (timeboxed research or experiments)**



Continuous integration

- Perform frequent incorporation of work into the whole, no matter the product, and then retest to determine that the entire product still works as intended.



Test at all levels

- Employ system-level testing for end-to-end information and unit testing for the building blocks.
- In between, understand if there is a need for integration testing and where.
- Teams find smoke testing helpful as a first look at whether the work product is any good.
- Teams have found that deciding when to run regression tests and which ones helps them maintain product quality with good build performance.
- Agile teams have a strong preference for automated tests so they can build and maintain a momentum of delivery.



Acceptance Test-Driven Development (ATDD)

- In ATDD, the entire team gets together and discusses the acceptance criteria for a work product.
- Then the team creates the tests, which allows the team to write just enough code and automated tests to meet the criteria.
- For non-software projects, consider how to test the work as the team completes chunks of value.



Test-Driven Development (TDD) and Behavior-Driven Development (BDD)

- Writing automated tests before writing/creating the product actually helps people design and mistake-proof the product.
- For non-software projects, consider how to “test-drive” the team's designs.
- Hardware and mechanical projects often use simulations for interim tests of their designs.



Spikes (timeboxed research or Experiments)

- Spikes are useful for learning and may be used in circumstances such as: estimation, acceptance criteria definition, and understanding the flow of a user's action through the product.
- Spikes are helpful when the team needs to learn some critical technical or functional element.



HOW ITERATIONS AND INCREMENTS HELP DELIVER WORKING PRODUCT

- Iterations help a team create a cadence for delivery and many kinds of feedback.
- Teams produce increments of value for delivery and feedback.
- The first part of this delivery is a demonstration.
- Teams receive feedback about how the product looks and operates through a demo.
- Team members retrospect to see how they can inspect and adapt their process to succeed.
- Demonstrations or reviews are a necessary part of the agile project flow.
- Schedule the demonstration as appropriate for the team's delivery cadence.

