Module 1

Imagine you have a big school project, like building a model city. There are two ways you and your friends can work on it: the *Waterfall way* and the *Agile way*.

**The Waterfall Way**: In the Waterfall way, you finish one thing completely before moving on to the next. First, you might plan out the whole city, then you’d build all the houses, then you’d add roads, and finally, you’d put in trees and cars. You do each step one at a time, like water flowing down a waterfall, and you don’t go back to fix things after moving forward.

**The Agile Way**: In the Agile way, instead of doing everything one after the other, you build the city in small chunks, called *iterations*. Each iteration might focus on one small part of the city, like building one street with houses, roads, and trees all at once. You’d check each part after finishing it and see if anything needs fixing or changing. If there’s something you want to improve, you can do it in the next round of work. This means you’re always checking and updating, so your project can change as you go. Agile is all about being flexible and able to adapt if things change!

**Why People Like Agile**: Agile started in the 1990s when people were trying to make better software (like apps and games) more quickly. People realized that instead of making a big plan and sticking to it no matter what, it was often better to work in these smaller, flexible chunks. This way, teams could keep improving their projects and make changes based on feedback—like how people play with and test new apps.

Eventually, Agile became so popular that even industries outside of software, like healthcare, car manufacturing, and education, started using it. They saw that being flexible and open to changes helped them work better, too!

**The Agile Manifesto**: In 2001, a group of people who liked Agile came together and made the Agile Manifesto, which is a set of values and principles. It reminds everyone to:

1. Focus on people and teamwork.
2. Be flexible and open to change.
3. Make things work well for the user.
4. Collaborate and improve with each step.

So, to sum up:

* Waterfall = One thing at a time, no going back.
* Agile = Small steps, flexible, and making changes based on what works best.

**Waterfall vs. Agile: A Basic Comparison**

**Waterfall** is a step-by-step process that tries to predict exactly how a project will go from start to finish. You complete one phase fully before moving on to the next, and it’s hard to make changes once you start. Waterfall is best when you know exactly what the final result should look like.

**Agile**, on the other hand, is a more flexible approach. Agile assumes things will change as you go, so it allows teams to make adjustments based on new information or feedback. Instead of trying to plan everything from the start, Agile projects are built in small parts or *iterations*. Each iteration produces a small, working piece of the project, which can then be tested and improved based on feedback.

**Key Differences in Requirements, Documentation, and Deliverables**

1. **Requirements (What needs to be done)**
   * In Waterfall, you make a complete list of requirements (everything the project needs) at the beginning. This list doesn’t change easily, and you often need formal approval for any updates.
   * In Agile, requirements are flexible and can change based on feedback. The team focuses on the most important or valuable requirements first and adjusts the list as they go along, working closely with stakeholders.
2. **Documentation (Written Plans and Records)**
   * Waterfall requires a lot of formal documentation because each phase needs to be passed on from one team to another, and everything needs to be clear.
   * Agile, however, focuses on real-time conversations and uses only the essential documents needed to get the job done. Agile teams write down just enough information to be helpful without creating unnecessary paperwork.
3. **Deliverables (The End Products)**
   * In Waterfall, you don’t see the final deliverable until the end of the project. It’s like a big reveal, where the whole project is shown at once.
   * In Agile, the team releases smaller, working pieces of the project throughout. This way, the team can get feedback with each release and make changes based on that feedback, ensuring the final product meets everyone’s needs.

**Why Agile Works for Uncertain Projects**

Agile was created because teams needed a way to work in uncertain or changing environments. For example, sometimes customers don’t know exactly what they want until they see it. Agile lets teams show the customer parts of the project as they’re completed, making it easier to make changes along the way.

So, in summary:

* **Waterfall** is strict, with clear steps and big documents. It’s great when you know exactly what you need.
* **Agile** is flexible, with short cycles, quick feedback, and less paperwork. It works well when things might change, and you need to be able to adapt.

**The Four Agile Values**

1. **Individuals and Interactions Over Processes and Tools**
   * This value is about people working together and communicating directly instead of relying too much on specific processes or tools.
   * Imagine needing quick information from a teammate. Instead of endless emails, a short conversation could be more efficient. The idea is to keep communication simple and productive.
2. **Working Software Over Comprehensive Documentation**
   * Here, the focus is on creating something valuable rather than spending too much time on planning documents.
   * In Agile, it’s better to have a basic, working product (like a new app feature or design) than a perfect plan without results. This value can apply to any project; delivering what people want is more important than detailed documentation about the process.
3. **Customer Collaboration Over Contract Negotiation**
   * Agile emphasizes working closely with customers throughout the project rather than relying solely on formal contracts and agreements.
   * This means getting feedback from customers early and often, which helps teams deliver exactly what’s needed. Instead of sticking rigidly to contracts, Agile encourages collaboration so changes can be made easily when needed.
4. **Responding to Change Over Following a Plan**
   * Agile understands that things change, and sometimes even a well-made plan needs adjusting.
   * Rather than sticking to a strict plan, Agile teams adapt to changes, whether it’s a shift in customer needs or a new market trend. This flexibility helps projects stay valuable and relevant.

**Why These Values Matter**

Each of these values helps Agile teams focus on what’s most important: delivering something valuable, adapting to change, and working well with people. Agile teams try to find the right balance between both sides of each value, using these principles to create effective, adaptable projects.

**The Four Themes of Agile Principles**

1. **Value Delivery**
   * Agile teams focus on delivering results quickly, allowing customers and companies to see the value sooner rather than later.
   * This theme is about delivering only what’s necessary, avoiding unnecessary complexity, and focusing on features that users will appreciate.
   * Example: Build a basic prototype to get quick feedback and avoid wasting time on features that might not be useful.
2. **Business Collaboration**
   * Teams should work closely with "business people" (like sales, marketing, and customer support) to ensure they’re creating valuable products that meet business goals.
   * This collaboration enables quick adjustments based on new information and customer needs, rather than sticking rigidly to a plan.
   * Example: Hold regular meetings with business partners and customers to discuss feedback, priorities, and possible new ideas.
3. **Team Culture**
   * Agile projects succeed best when the team culture is positive, inclusive, and supportive.
   * This theme highlights the importance of keeping teams motivated, trusted, and equipped with the tools they need. Teams should have a sustainable workload and a say in how they work.
   * Example: Let the team decide on tools or templates that fit their needs, and provide them with the equipment they request for better performance.
4. **Retrospectives and Continuous Learning**
   * Agile teams should regularly reflect on their work and look for ways to improve.
   * After each project stage, they review what went well, what didn’t, and what could be optimized. The goal is to keep learning and make each iteration better than the last.
   * Example: In a regular “retrospective” meeting, ask questions like: Are customers satisfied? Are any processes slowing us down? Is there anything we need to change to make work easier?

**Agile Values**

1. **People first, not just rules and tools:** It’s more important to have people talking and working well together than just following strict rules or using fancy tools.
2. **Show what works, not just long instructions:** Instead of spending a lot of time on instructions, it’s better to have a small piece of working stuff to show.
3. **Team up with the customer, not just a contract:** It’s better to work together with the customer all the time than to just stick to what’s written in the contract.
4. **Be ready to change, don’t get stuck on a plan:** Sometimes plans change. In Agile, we’re okay with that and will adjust to keep things on track.

**Agile Principles**

1. **Make the customer happy by giving them things early and often:** Don’t wait until the end to show them everything—give them little pieces along the way!
2. **Say "Yes!" to changes, even if they come late:** We know things change, and we’re okay with making adjustments.
3. **Send out little parts of the work regularly:** By giving small parts often, we can check if things are on the right track.
4. **Business people and creators should work together every day:** When everyone talks regularly, we understand each other better and make better stuff.
5. **Let people who are working have the support they need:** Trust them to do a good job, and help them out when they need it.
6. **Talking face-to-face is best:** When people talk in person (or on video), they can understand each other better.
7. **We know we’re making progress when things actually work:** If it’s working, we’re on the right track—way better than just saying "we’re almost done."
8. **Don’t work too fast or too slow, find a nice rhythm:** Keep going at a steady speed so nobody gets too tired or bored.
9. **Always try to do things well:** Even when we’re working fast, we should still do things carefully and with good design.
10. **Keep things simple and only do what’s needed:** Don’t add extra stuff that isn’t important—it just gets in the way!
11. **Let teams make decisions on how they work best:** Teams should have the freedom to decide how they get things done.
12. **Take time to learn and improve:** After each big step, take a moment to think, “What can we do better next time?”

**Agile in Different Industries**

* **Agile** is great for any business or project where things can change quickly, like software, biotechnology, media, food, and fashion. Why? Because Agile helps teams react fast to new information, trends, or market changes.
* But even in industries that seem more stable, like farming, aerospace, or mining, change still happens. New rules, natural events, or global challenges can shake things up in these fields too. Agile helps these industries be ready to adapt when surprises come up.

**Introducing VUCA**

* **VUCA** is a way to understand and manage changes and surprises in the world. VUCA stands for **Volatility, Uncertainty, Complexity, and Ambiguity**—let’s go through each one:

1. **Volatility**: This is when things are always changing or disrupted. Imagine trying to make a plan, but new surprises keep coming up! That’s high volatility.
2. **Uncertainty**: In uncertain situations, it’s hard to predict what will happen next. You might make a lot of guesses, but things can still go differently than expected.
3. **Complexity**: Complexity means there are a lot of parts working together that influence each other. For example, a project with many suppliers worldwide is complex because a problem in one place could affect everything else.
4. **Ambiguity**: This happens when it’s hard to understand why something is happening or where a problem started. In ambiguous projects, it’s tough to find clear answers, so planning solutions becomes challenging.

**Why Use VUCA and Agile Together?**

* **VUCA** helps businesses understand what they’re dealing with, and **Agile** gives them a flexible way to adapt to these changes.
* By understanding VUCA, businesses can decide if Agile is the right approach for their projects—especially when facing change, unpredictability, or complexity.

### VUCA Reminder

* **VUCA** stands for **Volatility, Uncertainty, Complexity, and Ambiguity**—challenges that make projects hard to plan and manage.

### Why VUCA and Agile Matter for Projects

* When starting a project, it’s helpful to check the environment around it to see if VUCA factors are high. If they are, Agile can be a great approach to manage the project.
* While Agile doesn’t remove VUCA challenges, it gives teams the tools to work through them and reduce the risks.

### Office Green and Their New Project: Virtual Verde

* **Office Green LLC** is a landscaping company for offices and commercial spaces. But with more people working from home, they noticed a big opportunity in creating home office designs with indoor plants.
* Since the shift to remote work happened quickly, they didn’t have time to make detailed plans for this change. They needed to act fast to meet the new demand.

### The VUCA Environment for Office Green

1. **Volatility**: A major shift in demand disrupted their business model.
2. **Uncertainty**: It was hard to predict how long the demand for home office plants would last.
3. **Complexity**: Managing many factors, like suppliers, the economy, and new customer needs, made things complicated.
4. **Ambiguity**: They didn’t know what else might change or how to control it.

### Using Agile for Virtual Verde

* Office Green used an Agile approach to handle these high VUCA factors. By staying flexible, they could adapt to changes quickly and grab the new opportunity instead of losing business.

Module 2

**What is Scrum?**

* Imagine you’re playing rugby. The team huddles together, locks heads, and pushes forward as one big group to move the ball closer to the goal. Scrum in project management is kind of like that — everyone on the team works closely together to get things done.

**How Scrum Works:**

* In Scrum, the team works in **short bursts** of time (called **Sprints**) to finish tasks. These bursts usually last about **2 weeks**, and after each burst, the team shows what they’ve worked on and gets feedback.

**Important Words in Scrum:**

1. **Backlog**: It’s like a big list of everything the team needs to do. The team looks at this list and decides what’s most important to do first.
2. **Sprint**: A short period of time where the team works really hard to finish some tasks. It’s like saying, “Let’s work hard for the next 2 weeks!”
3. **Daily Scrum (Stand-up)**: Every day, the team quickly talks about what they did yesterday, what they’re doing today, and if anything is getting in the way.

**Who Does What in Scrum?**

1. **Scrum Master**: This person makes sure the team is following the rules and helps when there’s a problem.
2. **Product Owner**: This person decides what’s most important to work on and tells the team what to do first.
3. **Development Team**: These are the people who actually do the work and get things done.

**Why Scrum is Great:**

* Scrum is awesome because it has **clear roles** and **daily check-ins** that help everyone stay on track.
* It’s all about **working together** and **getting better** every day, so the team can keep improving.

**What Makes a Good Scrum Team?**

* A good Scrum team has about **3 to 9 people** who all know different things, so they can work on lots of different parts of the project.
* The team should be **flexible**, **ready to learn**, and willing to work together as one big group.

So, Scrum is just a fun and easy way for a team to work closely together to get things done in a smart, quick

### ****1. Kanban****

* **What is it?** Imagine a big board with three columns: "To Do," "Doing," and "Done." You stick tasks (like homework or chores) on the board, and as you work on them, you move them across the board.
* **Key idea**: You can only work on a few tasks at a time. When you finish one, you can start the next. This keeps things from getting too messy!

### ****2. Extreme Programming (XP)****

* **What is it?** XP is like doing your homework in small steps and checking each part to make sure it’s perfect before moving on. It’s all about doing things really well.
* **Key idea**: Work in small parts, check if it’s right, and then move on. You also work with a buddy (pair programming), so you help each other, and you test things all the time to make sure they work.

### ****3. Lean****

* **What is it?** Lean is about doing things faster and better by getting rid of stuff that doesn’t help. It's like cleaning your room: only keep what you really need!
* **Key idea**: Focus on what the customer really wants, make everything work smoothly, and keep improving. Always get better and faster.

### ****How They Work Together****

All these methods are ways to work smarter, not harder! They help teams get things done faster, better, and with less stress. And sometimes, teams mix them up to make their own way of working!

………………………

### ****1. Kanban****

* **What is it?** Kanban is like a visual to-do list that helps you see what tasks are being worked on, what needs to be done, and what’s already finished. Imagine sticky notes on a board: one column for tasks to do, one for tasks in progress, and one for completed tasks.
* **Key idea**: You can only have a certain number of tasks being worked on at the same time. This is called the **Work-in-Progress (WIP) Limit**. If you’ve finished one task, you can take on another, but the goal is to **keep things flowing** without overwhelming the team.

### ****2. Extreme Programming (XP)****

* **What is it?** XP is all about doing things "extremely well" by focusing on small, detailed tasks and getting feedback quickly. It started with software, but it can be used for other projects, too.
* **Key idea**: You write small, simple designs first (so you don’t waste time), test everything as you go, and keep improving. It also uses practices like:
  + **Pair programming**: Two people work together on one task to make sure it’s done right.
  + **Continuous testing and improvement**: Keep testing the product as you build it to find problems early.
  + **Listen to the customer**: Always make sure you’re building what the customer actually wants.

### ****3. Lean****

* **What is it?** Lean focuses on making everything run smoothly and efficiently by cutting out waste (things that don’t add value). It’s about creating better processes and improving continuously.
* **Key idea**: Lean is like a recipe for improvement. You focus on:
  1. **Value**: What does the customer want? Focus on that.
  2. **Flow**: Make sure work moves smoothly through the process.
  3. **Pull**: Let the customer guide when things should be done or added.
  4. **Perfection**: Always get better and improve what you’re doing.

### ****How They All Work Together****

Each of these methodologies (Scrum, Kanban, XP, and Lean) has its own focus, but they all share the core **Agile** values: being flexible, improving continuously, and focusing on delivering what the customer wants. Teams often **mix and match** these methods to fit their needs, so they can create their own special way of working.

### The Spotify Model Simplified

**What is it?**  
The Spotify model is a way of organizing teams at Spotify to encourage creativity and collaboration while keeping everyone independent.

**How it works:**

* **Squads:** Small, self-organizing teams that act like mini-startups. Each Squad has a mission (e.g., improving the app).
* **Tribes:** Groups of Squads working in the same area, usually under 100 people.
* **Chapters:** Small groups of people from different Squads who have similar skills.
* **Guilds:** Larger groups across the organization sharing knowledge and practices.

**Key Points:**

* Don’t copy the model exactly; adapt it to fit your team’s needs.
* Be flexible and willing to change as you grow.
* Always look for ways to improve; there’s no end to making things better.

**Takeaway:**  
Use the Spotify model for inspiration, but make it work for you!

WEEK 2

### Overview of Scrum in Agile Project Management

**What is Scrum?**  
Scrum is a widely-used framework within the Agile methodology that helps teams manage projects effectively. It's based on the Agile mindset, which is guided by the Agile Manifesto's values and principles.

**Key Sources:**

* The **Scrum Guide** is the main resource for understanding Scrum. It's available for free at [Scrumguides.org](https://scrumguides.org) and is essential for Scrum Teams.

**Agile vs. Scrum:**

* **Agile:** A philosophy and mindset for project management.
* **Scrum:** A specific framework that implements Agile principles. It predates the Agile Manifesto and inspired many Agile practices.

### What We’ll Cover in This Course

1. **Theory of Scrum:**
   * Explore the three pillars of Scrum theory.
   * Discuss the five Scrum values.
2. **Importance of Mission:**
   * Understand why teams need a clear mission.
3. **Roles in a Scrum Team:**
   * **Product Owner:** Defines the product vision and prioritizes the backlog.
   * **Scrum Master:** Facilitates the Scrum process and supports the team.
   * **Development Team:** Executes the work to deliver the product increment.
4. **Introduction of a Project:**
   * We'll use the **Office Green** project, specifically the **Virtual Verde** initiative, as a case study to apply Scrum concepts.
5. **Scrum Artifacts and Events:**
   * Learn about the key artifacts (e.g., Product Backlog, Sprint Backlog) and events (e.g., Sprint Planning, Daily Scrum) in Scrum.

### The Three Pillars of Scrum

Scrum is built on three foundational pillars that ensure its effectiveness as a framework for managing projects. Here’s a breakdown of each pillar:

1. **Transparency**
   * **Definition:** All significant aspects of the work must be visible to everyone involved in the project, including team members, stakeholders, and users.
   * **Purpose:** Transparency fosters trust and collaboration. It helps to prevent communication breakdowns and ensures that everyone is on the same page.
   * **Team Size:** Scrum Teams are intentionally small (3-9 members) to enhance communication and productivity.
2. **Inspection**
   * **Definition:** Regular checks on the progress and deliverables toward the Sprint goal are essential to identify any undesirable variances or issues.
   * **Purpose:** Frequent inspections allow teams to catch problems early and make necessary adjustments. Stakeholder reviews during these inspections are crucial for growth and improvement.
3. **Adaptation**
   * **Definition:** The process of making adjustments to the project, product, or processes to minimize further issues and improve outcomes.
   * **Purpose:** Embracing change is vital in Scrum. Adaptation ensures that teams continuously improve by addressing what doesn’t work and implementing solutions for future projects.

### Summary

These three pillars—**transparency, inspection, and adaptation**—form the foundation of Scrum, enabling teams to navigate uncertainty and deliver valuable products effectively. In the next session, we'll explore the five values that guide all Scrum Teams.

### The Five Values of Scrum

In addition to the three pillars of Scrum, there are five core values that guide Scrum Teams. These values help foster a productive and collaborative environment:

1. **Commitment**
   * **Definition:** Team members commit to achieving the goals of the Scrum Team.
   * **Example:** If a team member struggles with a task, others may step in to help, prioritizing collective success over individual workload.
2. **Courage**
   * **Definition:** Team members must have the courage to tackle difficult problems and speak up about challenges.
   * **Example:** A team member takes on a complex task that requires new skills or openly discusses obstacles they are facing, fostering resilience within the team.
3. **Focus**
   * **Definition:** Everyone concentrates on the work necessary to meet the Sprint and overall team goals.
   * **Example:** The Scrum Master facilitates daily activities, helping the team stay focused on their objectives and encouraging them to support one another.
4. **Openness**
   * **Definition:** Team members and stakeholders agree to be transparent about their work and challenges.
   * **Example:** If someone encounters an issue, they share it with the team, allowing for collaborative problem-solving and insight sharing.
5. **Respect**
   * **Definition:** Team members show respect for each other’s opinions, skills, and independence.
   * **Example:** Valuing each member's contributions encourages active listening and constructive feedback, essential for product success.

### The Relationship Between Values and Pillars

To effectively embody the three pillars of Scrum—**transparency, inspection, and adaptation**—teams must uphold these five values:

* **Transparency** requires **openness** and **respect** to share relevant information.
* **Inspection** relies on **courage** to give and receive feedback, and **respect** to listen to others' perspectives.
* **Adaptation** demands **commitment** to the team’s goals and the **courage** to implement changes based on inspection results.

### Understanding Scrum Roles and Mission

In Scrum, each team member plays a specific role, similar to players on a sports team. Here’s an overview of the Scrum Team structure and how their mission and product vision guide their work.

#### Mission and Product Vision

* **Mission:** A constant statement that defines why the team is doing the work. It motivates team members and keeps them aligned.
  + **Example:** For Office Green's Virtual Verde project, the mission is to improve users' health and happiness by bringing their home workspace to life.
* **Product Vision:** A clear picture of what the team aims to achieve and the outcomes they are responsible for.
  + **Example:** The product vision for Virtual Verde is to create a living marketplace that transforms the home office.

Together, the mission and product vision inspire the team to deliver a great user experience.

#### Scrum Roles

Every Scrum Team consists of three key roles:

1. **Product Owner**
   * **Responsibilities:** Defines what the team builds and ensures everyone understands the reasons behind it.
   * **Example:** For Virtual Verde, the Product Owner captures and promotes ideas for delivering plants to home offices.
2. **Development Team**
   * **Responsibilities:** Focuses on how the product will be delivered, including the technical aspects of building it.
   * **Example:** They might be tasked with creating a website for customers to order plants.
3. **Scrum Master**
   * **Responsibilities:** Facilitates the team’s processes and helps remove obstacles that might hinder progress. This role is similar to that of a project manager.
   * **Example:** The Scrum Master for Virtual Verde would help resolve issues like vendor delays or user feedback prioritization.

#### Team Dynamics

* **Collaboration:** While roles are clearly defined, all members collaborate to achieve the team’s goals. Development Team members may contribute to discussions about the "what" and "when," while the Product Owner may offer insights on the "how."
* **Cross-Functional Teams:** Scrum Teams consist of diverse skill sets (e.g., developers, marketers, quality assurance experts), which adds value and enhances project outcomes.
* **Self-Organizing:** Scrum Teams manage their own tasks without direct oversight from managers. This promotes flexibility and ownership, leading to better results.

#### Conclusion

Understanding these roles and how they interact is crucial for effective Scrum implementation. In the next videos, we'll delve deeper into the specific traits of an effective Scrum Master.

### The Role of the Scrum Master in Simple Terms

The Scrum Master is an important part of a Scrum Team. Here’s what they do and how they compare to a regular project manager.

#### What Does a Scrum Master Do?

1. **Support the Team**
   * The Scrum Master helps team members understand and use Scrum practices and values. They coach the team to work better together.
2. **Run Scrum Meetings**
   * They organize and lead important meetings, like Sprint Retrospectives, where the team discusses what went well and what can improve.
3. **Remove Obstacles**
   * If there are things blocking the team from doing their work (like missing information), the Scrum Master helps to fix those problems.
4. **Protect the Team**
   * They keep outside distractions and interruptions away from the team so everyone can focus on their tasks.

#### Key Traits of a Good Scrum Master

* **Organized:**  
  They keep everything in order and manage meetings well.
* **Supportive:**  
  They care about the team's needs and ask, "How can I help?" instead of bossing people around.
* **Good at Facilitating:**  
  They make sure everyone has a chance to speak and contribute in team discussions.
* **Great Coach:**  
  They encourage learning and discussion, rather than just giving answers.
* **Strong Communicator:**  
  They can talk to different people and understand their viewpoints.

#### Scrum Master vs. Project Manager

* **Differences:**
  + A Scrum Master focuses on helping the team work together and follow Scrum practices.
  + A project manager usually handles broader tasks, like budgets and schedules.
* **Similarities:**
  + Both roles need to be organized and good at communicating. Many people can do both jobs.

### Summary

The Scrum Master is essential for helping the team succeed by supporting them and guiding their work. While their job is different from a traditional project manager, they share many skills.

### Overview of the Technical Program Manager Role

Hi, I'm Pete, and I work as a Technical Program Manager at Google. My job involves overseeing the complete execution of programs I'm responsible for. Here’s a breakdown of my role and how we use Agile and Scrum in our work.

#### Key Responsibilities

* **Program Execution:**  
  I ensure that all aspects of our programs are running smoothly from start to finish.
* **Using Agile Framework:**  
  We adopt Agile methods to speed up our product delivery. This allows us to make adjustments and iterate on our processes effectively.
* **Scrum Framework:**  
  Scrum is one specific way to implement Agile. It helps us deliver value to customers quickly by working in short, focused cycles called Sprints.

#### Importance of Team Roles

* **Self-Organizing Teams:**  
  In Scrum, we bring together individuals with clear roles and responsibilities. This clarity helps the team work towards common goals.
* **Role of the Scrum Master:**  
  The Scrum Master acts as a leader who guides the team and ensures everyone is aligned with our objectives.

#### Qualities of an Effective Scrum Master

1. **Good Teacher and Communicator:**  
   A Scrum Master must effectively teach and communicate Scrum values to the team.
2. **Influential Leadership:**  
   Since Scrum Masters don’t manage team members directly, they lead through influence. This means encouraging effective teamwork and fostering a motivating environment.
3. **Direction and Focus:**  
   The Scrum Master provides direction, helping the team stay on track to meet the goals set for each Sprint. A Sprint is a focused effort to achieve specific objectives in a short timeframe.

#### Personal Satisfaction

I enjoy working with people towards a shared goal. While it can be challenging, it’s always rewarding to see the team come together to achieve success.

### The Role of the Product Owner

The Product Owner plays a crucial role in a Scrum Team. Here’s a summary of what they do:

* **Build the Right Product:**  
  The Product Owner ensures the team is creating a product that users actually want.
* **Maximize Value:**  
  They continuously work to deliver the most value by prioritizing features in the Product Backlog, which is the list of tasks the team needs to complete.
* **Voice of the Customer:**  
  The Product Owner represents customer needs and helps the team understand the importance of their work.

#### Key Traits of a Product Owner

1. **Customer-Focused:**  
   Understands customer needs and the industry.
2. **Decisive:**  
   Makes clear decisions and communicates them effectively.
3. **Flexible:**  
   Adapts priorities based on new information.
4. **Optimistic:**  
   Inspires the team with a positive attitude towards the product vision.
5. **Available:**  
   Regularly collaborates with the team to guide the project.
6. **Collaborative:**  
   Works closely with the Development Team and stakeholders to meet customer needs.

### Example in Action

For instance, if the Product Owner initially prioritizes certain features for a project (like flower arrangements), but the Development Team discovers a different feature (herb gardens) is actually easier to deliver, a good Product Owner will be flexible and adjust priorities accordingly.

### The Role of the Development Team in Scrum

In this video, we’ll look at the Development Team within a Scrum Team, which is essential for building the product. Here’s an overview of their responsibilities and characteristics.

#### Key Characteristics of the Development Team

1. **Team Size:**
   * The Development Team typically consists of **3 to 9 members**. This size is ideal for maintaining agility while ensuring enough skills and ideas are present.
2. **Cross-Functional:**
   * The team should have all the required skills to build the product internally, meaning members are capable of handling different aspects of the project.
3. **Self-Organizing:**
   * The Development Team manages its own processes and structures. They work together as a cohesive unit rather than as isolated individuals.
4. **Customer-Oriented:**
   * The best products come from teams that focus on the customer and consider user needs throughout the development process.
5. **Co-Located vs. Virtual Work:**
   * Many teams prefer to work in the same physical space (co-located) to enhance collaboration. However, virtual work is also common and can be effective with the right tools.

#### Example Scenario

Consider the Virtual Verde team facing an issue with a plant vendor unable to meet holiday demands due to quality problems. If the quality assurance specialist needs to travel to resolve the issue, being co-located allows the team to quickly brainstorm solutions together. If they aren't together, they would need to rely on phone calls or emails, which can slow down communication and lead to project delays.

#### Summary of Roles in a Scrum Team

* **Product Owner:** Ensures the team is building the right product for customers.
* **Development Team:** Responsible for building the product correctly.
* **Scrum Master:** Focuses on improving the team’s efficiency.

Each role is vital to the success of a Scrum project, and all members collaborate to create value for users and customers.

Module 3

Alright! Imagine you're making a list of everything you need to do to build something cool, like a treehouse. That list is called the **Product Backlog**. It’s like a big to-do list that tells you everything you need to work on to finish the project.

Here’s how it works:

1. **Always Changing**: Your list isn't fixed—it changes as you think of new things to do. Maybe you first just think of putting up the walls, then later remember you need to add windows or a ladder.
2. **The Boss of the List**: The **Product Owner** is the person who decides what should go on the list and in what order. They make sure you’re working on the most important things first.
3. **Order Matters**: You put the most important tasks at the top of the list and the least important ones at the bottom. For example, building the floor is probably more important than painting it, so you put "build floor" higher on the list.

Each item on the list has a few things written down:

* **What is it?**: Like, “I need to build a ladder.”
* **How important is it?**: Maybe building the ladder is super important, so it gets a high value rating (like 3 out of 4 stars).
* **How hard is it?**: The harder tasks take more time, so you give them an estimate of effort (like 3 hours or 5 hours).
* **Where does it go on the list?**: You put the tasks that are most important at the top, like building the floor first, then putting up walls, and so on.

In this video, we’re diving into **User Stories**, which are a way to capture what the user wants in simple terms. A **User Story** is a short, clear description of a feature, told from the user's perspective. It helps the team focus on the user's experience and needs.

A **User Story** has three main parts:

1. **User**: Who is the story about?
2. **Action**: What does the user want to do?
3. **Benefit**: Why does the user want to do this?

The typical format is:  
**"As a [user], I want [action] so that I can [benefit]."**  
For example: *"As a Virtual Verde client, I want to buy a bonsai tree so I can meditate while trimming its branches."*

**Creating Good User Stories**

Effective user stories follow the **INVEST** acronym:

* **I** = Independent: The story should stand alone and not depend on others.
* **N** = Negotiable: There’s room to discuss and adjust the story.
* **V** = Valuable: The story must deliver value to the user.
* **E** = Estimable: The team should be able to estimate how long it will take to complete.
* **S** = Small: The story should be small enough to fit within one Sprint. If it’s too big, break it down.
* **T** = Testable: The story should have clear criteria to test if it’s done.

**User Personas**

It’s helpful to create **user personas** to imagine the people who will use your product. For example, in the Virtual Verde project, we might have:

* **Leo**: A plant vendor who manages supply and deliveries.
* **Zach**: An amateur gardener who wants to use plants for cooking.
* **Nia**: A home office worker who wants plants for a professional backdrop.

These personas help the team understand who they're designing for.

**What is an Epic?**

An **Epic** is a large group of related user stories. For example, “Live Plant Delivery” could be an epic, and within that, you might have user stories like “Order Bonsai Tree” or “Track Plant Delivery.”

**Acceptance Criteria**

Each user story comes with **acceptance criteria**, a checklist that defines when the story is done. For the bonsai tree example, acceptance criteria might include:

* Browsing and comparing three types of bonsai trees.
* Offering care packages (fertilizers, trimming tools, etc.).
* Providing a troubleshooting FAQ page.

By having detailed user stories and acceptance criteria, the team knows exactly what needs to be done and can deliver a product that meets user expectations.

**User Stories**

A **User Story** is a short, simple description of a feature from the user’s perspective. It helps the team understand what the user needs and how to create a product that meets those needs. User stories are important because they keep the team focused on the user and the value they want to achieve.

When writing a user story, include these elements:

1. **User Persona**: Who is the user? What are their goals and needs?
2. **Definition of Done**: A list of criteria that must be met before the user story can be considered finished.
3. **Tasks**: What actions need to be done to complete the story?
4. **Feedback**: If you’re improving an existing product, include any feedback from past users.

Each user story should meet the **I.N.V.E.S.T.** criteria:

* **Independent**: The story should be able to be done on its own.
* **Negotiable**: There’s room to discuss and adjust the story.
* **Valuable**: The story must bring value to the user.
* **Estimable**: The team should be able to estimate the work needed to complete the story.
* **Small**: It should be small enough to finish in a single Sprint.
* **Testable**: It must have clear acceptance criteria that can be tested.

**Example of a User Story**

For a local library building a website, a user story might be:  
*"As an avid reader, I want to read reviews before I check out a book from my local branch, so I know I’m picking a book I’ll enjoy."*

This helps the team understand that the user wants to read reviews and what value they’ll get from that feature.

**Epics**

An **Epic** is a larger user story that is too big to be completed in one Sprint. It helps organize related user stories into bigger themes. Think of an epic as a “chapter” in the larger “book” of your project.

For example, for the library website:

* **Epic 1**: Website Creation (e.g., users can read reviews, add books to their cart)
* **Epic 2**: Organization of the Physical Space (e.g., users can easily find the non-fiction section in the library)

Each epic will contain several smaller user stories.

**Hierarchy of Stories and Epics**

Epics are the big picture, and user stories are the smaller, actionable items that help achieve the epic. The user stories fit within their corresponding epics to keep things organized.

**Who Writes User Stories and Epics?**

While the **Product Owner** is typically responsible for writing and managing user stories and epics, **Developers** can also write them. However, the Product Owner is accountable for ensuring that the Product Backlog is clear and prioritized.

**What is a Product Backlog?**

A **Product Backlog** is a list of everything the team needs to work on for the project. It includes features, tasks, and requirements, and is constantly updated as the project evolves. Tools like Asana make it easier to manage this list and keep track of the team's work.

**Using Asana to Create a Product Backlog**

1. **Create a New Project**: When setting up a new project in Asana, you can use templates, like the **Sprint Planning Template**, to structure your backlog.
   * **Board View** is often used for this, showing tasks (user stories) in columns that represent stages of work.
2. **Add User Stories**: Once your project is set up, add **User Story Titles** (tasks) to the **Backlog** column. For example, for a project like **Virtual Verde**:
   * “Low-maintenance options”
   * “Plant care tips”
   * “Watering reminders”
3. **Define User Stories**: Within each task (user story), add a **description** of the user’s needs. For example:
   * *“As a potential customer, I want to find out which plants are easiest to care for so that I can purchase low-maintenance options.”*
4. **Add Acceptance Criteria**: Under each user story, create **subtasks** for acceptance criteria (the conditions for the story to be considered "done"). For example, for the “Low-maintenance options” story, subtasks might include:
   * Ability to sort plants by difficulty level (beginner, intermediate, advanced)
   * Ability to search for plants with similar care needs
5. **Add Custom Fields for Epics**: You can also group user stories into **Epics**. For example, a "Website Features" epic might contain user stories like “Plant care tips” and “Watering reminders.”
   * Use **Custom Fields** in Asana to label each user story with an epic, making it easy to track which epic the story belongs to.
6. **Tracking and Managing Progress**: As you move through the sprint, you can update the status of each user story (task) in the Board View, moving them from the "Backlog" column to "In Progress" or "Completed."

**Why Use Work Management Tools for Product Backlogs?**

* **Efficiency**: These tools reduce time spent searching for information, managing documents, or switching between different apps.
* **Collaboration**: Everyone can see the same Product Backlog, and the team can easily update or adjust tasks as needed.
* **Automation**: Asana can automate some updates and help prioritize tasks according to the product owner’s needs.

**Key Takeaways:**

* Work management tools like **Asana**, **ClickUp**, or **Monday.com** can help you organize and track your Product Backlog effectively.
* Use templates, task descriptions, subtasks for acceptance criteria, and custom fields for epics to structure your backlog.
* Understanding how to use these tools will make you more efficient and allow your team to work in alignment, no matter which tool you use.

### Agile Estimation Techniques:

Agile teams use various methods to estimate the effort needed for tasks or user stories. Effective estimation helps the team plan better and ensures successful project delivery.

**Common Estimation Techniques:**

1. **Planning Poker™**:
   * A consensus-based method using cards with Fibonacci numbers.
   * Everyone privately chooses an estimate, then reveals and discusses.
   * Avoids bias by keeping estimates hidden initially.
2. **Dot Voting**:
   * Team members use colored dot stickers to estimate effort.
   * The color represents the size of the task (e.g., small, medium, large).
   * Good for small backlogs (less than 10 items).
3. **The Bucket System**:
   * Best for large backlogs (hundreds of items).
   * Team places items in "buckets" or categories based on complexity.
   * A fast, efficient way to estimate many items at once.
4. **Large/Uncertain/Small**:
   * Items are quickly categorized as "large," "uncertain," or "small" based on their complexity.
   * Useful for backlogs with similar items.
5. **Ordering Method**:
   * Team members order items from low to high effort.
   * Items are adjusted until everyone agrees on their placement.
   * Works well for smaller teams with many items.
6. **Affinity Mapping**:
   * Sticky notes are used to group items by theme or similarity.
   * Teams prioritize groups based on importance.
   * Great for larger backlogs (more than 20 items).

**Key Characteristics of Effective Estimation:**

1. **Avoids False Precision**:
   * Estimates are rough and relative. The goal is to save time, not to debate over small details.
2. **Prevents Anchoring Bias**:
   * By keeping initial estimates private (e.g., Planning Poker), team members avoid being influenced by others' opinions.
3. **Promotes Inclusivity**:
   * Everyone has a say in the estimation process, fostering trust and collaboration within the team.
4. **Uncovers Effort Discovery**:
   * Estimating can reveal hidden challenges or solutions that might not have been obvious at first.

**T-shirt Sizes**

T-shirt sizes (XS, S, M, L, XL, XXL) are a simple and intuitive way to estimate effort. It’s especially helpful for new teams or those just learning about relative estimation. Here's how it works:

1. **Choose a scale**: Decide on the T-shirt sizes to use (e.g., XS = very small, XXL = very large).
2. **Identify an anchor item**: Pick a backlog item to represent a “medium” or “average” size. Some teams may choose two anchors: one for small and one for large tasks.
3. **Assign sizes**: For each remaining backlog item, compare it to the anchor item and assign it a T-shirt size based on how it compares in effort.

**Benefits of T-shirt sizes**:

* Quick and easy to use.
* Good for teams new to Agile estimation.

**Story Points**

Story points are more advanced and provide more precision than T-shirt sizes. Teams use numbers from the **Fibonacci sequence** (1, 2, 3, 5, 8, 13, 21, etc.) to estimate effort. The numbers increase in a way that reflects growing complexity and uncertainty.

1. **Agree on point values**: Decide which Fibonacci numbers will be used (some teams cap the maximum, like 21 points, or jump to 100 for large tasks).
2. **Pick an anchor item**: Select a backlog item to represent a baseline estimate (e.g., 5 points). This item will serve as the reference for comparing all other tasks.
3. **Assign points**: Compare each backlog item to the anchor and assign it a story point value based on its complexity.

**Benefits of story points**:

* Provides more detail and specificity than T-shirt sizes.
* Good for experienced teams that are comfortable with relative estimation.

**Best Practices for Both Methods**

Regardless of whether you're using T-shirt sizes or story points, here are some best practices:

1. **Clarify with the Product Owner**: Before estimating, ensure you have enough details about each user story. Ask the Product Owner any questions that help define the scope.
2. **Discuss Divergent Estimates**: If estimates vary significantly, have a discussion. This helps the team align and ensures everyone understands the task.
3. **Agree on Final Estimates**: After discussion, settle on a final estimate and record it in the backlog management system.
4. **Break Down Larger Tasks**: If a task seems too large or uncertain, consider breaking it down into smaller, more manageable tasks before estimating.

**Summary**

* **T-shirt sizes** are simple, quick, and ideal for beginners.
* **Story points** are more precise and suitable for experienced teams.
* Both methods rely on **relative estimation**, meaning tasks are compared to one another to estimate effort rather than giving absolute time estimates.

### Look at the Examples:

1. **Finding Easy-to-Care Plants**
   * **Why it's valuable**: People want easy-to-care-for plants, so they don’t have to worry about killing them.
   * **How much effort**: It’s not too hard to make a list of easy plants, so the effort is rated **8** (on a scale of 1 to 20).
   * **Why**: It’s important, but not too hard to do because we’re not needing to buy a lot of stuff for it.
2. **Having Care Instructions for Plants**
   * **Why it's valuable**: People want to know how to take care of their plants to keep them alive.
   * **How much effort**: The effort is **8** because making care instructions isn’t hard, and the company already knows how to make them.
   * **Why**: It’s helpful, but we already have the knowledge, so it's not super hard to do.
3. **Having the Right Tools for Plants**
   * **Why it's valuable**: Having the right tools helps people take care of their plants.
   * **How much effort**: This is a little harder, with more work because we might need to get or make the tools. Effort is **13**.
   * **Why**: We might have to find or create the tools, which takes more work.
4. **Getting Watering Reminders**
   * **Why it's valuable**: Some people forget to water their plants, so they need a reminder.
   * **How much effort**: The effort is **5** because it’s simple, like putting up reminder stickers.
   * **Why**: It’s helpful but not too hard to set up.
5. **Getting Expert Help for Sick Plants**
   * **Why it's valuable**: People want to know how to help their plants if they get sick.
   * **How much effort**: This takes **8** effort because it’s about setting up customer support, but it’s not too hard.
   * **Why**: People can search online for help, so the value is not super high, but it's still useful.
6. **Making Returns Easy**
   * **Why it's valuable**: If someone doesn't like their plant, they want to return it easily.
   * **How much effort**: This takes **5** effort because it’s easy to set up a return system and the company already knows how to do it.
   * **Why**: It’s important for customers, but it's not too hard to make returns easy.

**What Went Well:**

* I did a good job thinking about how much work each task would take and how valuable it would be for customers.

**Where I Can Improve:**

* I can make sure I think about **all** the possible things needed for each task, like hidden challenges, so I can get the effort estimate even more accurate.

**How to Add Effort Estimates Using Work Management Tools (like Asana)**

Work management tools, like Asana, can help you plan, track, and organize your project tasks more easily. One important feature is **effort estimation**, which helps you figure out how much work a task will take. This can help make sure you're realistic about how long tasks will take, so you don’t run into problems with deadlines or budgeting.

**What is Effort Estimation?**

Effort estimation is about figuring out how much work a task will take, but **not necessarily** how much time it will take. This is because tasks often take longer than we expect! To avoid underestimating, you can use a method called **relative estimation**, where you compare the task to another one that’s similar.

For example:

* **Task A** might be easy (like updating a website).
* **Task B** is harder (like creating a new feature for the website).

Instead of saying "Task A will take 2 hours" and "Task B will take 10 hours," you could say "Task A is a 5, and Task B is a 10." This way, you compare them in terms of effort, not exact time.

**Steps to Add Effort Estimates in Asana (or Similar Tools)**

1. **Create a New Project**:
   * Open Asana (or another tool like ClickUp or Monday).
   * Create a new project called **Virtual Verde Backlog**.
   * Instead of using a template, select **Import Spreadsheet** to upload a CSV file with your tasks and information.
2. **Preview Your Project**:
   * Once you upload the CSV file, Asana will show you a preview of your new project to make sure everything looks good.
3. **Add Custom Fields (like Estimates)**:
   * Asana automatically adds some fields for your project, like **Epic**, **Value**, and **Estimate (Story Points)**.
   * You will add your effort estimates (called "Story Points") for each task in the **Estimate** column. For example:

| **User Story** | **Estimate (Story Points)** |
| --- | --- |
| Find low-maintenance plants | 8 |
| Access care instructions for plants | 8 |
| Have the right tools for plant care | 13 |
| Get watering reminders | 5 |
| Get expert help for sick plants | 8 |
| Easy return policy | 5 |

1. **Sort Your Backlog by Value**:
   * To make it easier to prioritize tasks, you can sort your backlog by the **Value** column. This way, tasks that are more important show up first.
2. **Use Board View**:
   * After sorting, switch to **Board View** to see your tasks like cards on a Kanban board. This is a visual way to track your tasks.
   * You can organize tasks in columns, like **To Do**, **In Progress**, and **Done**.
3. **Save Your Layout**:
   * After sorting your tasks and adjusting your view, you can save your layout as the default. This will keep everything organized in the way that works best for you and your team.

### What is a Sprint?

A **Sprint** is like a **mini-project** that a team works on for a set amount of time. It's a way to get things done step by step, and it helps teams stay focused and finish things faster.

Each Sprint usually lasts between **1 to 4 weeks**. After each Sprint, the team looks at what they finished and checks how they can improve for the next Sprint.

**What Happens in a Sprint?**

1. **Sprint Planning**: The team meets at the start of the Sprint and decides what work they’ll do.
2. **Daily Scrum**: Every day, the team has a quick meeting to talk about what’s been done and if there are any problems.
3. **Sprint Review**: At the end of the Sprint, the team shows what they’ve worked on and gets feedback from others.
4. **Sprint Retrospective**: After the review, the team talks about what went well and what could be better for the next Sprint.

**How Do You Choose How Long a Sprint Should Be?**

There are three things to think about when choosing how long a Sprint should be:

1. **How often do things change?**
   * If the project is changing a lot, you might want shorter Sprints (like **1 week**) so you can adjust quickly.
   * If things are stable, longer Sprints (like **2 weeks**) might be better.
2. **How much focus does the team need?**
   * If the work takes a lot of time, like creating something big or complicated, a **2-week Sprint** gives more time to get it done without rushing.
3. **How much work is needed to finish?**
   * If the team has to do a lot of reviews or testing (like checking if the work is good enough), you might need a longer Sprint (like **3 or 4 weeks**) to fit that in.

**Can You Change the Sprint Length?**

Yes! You can always change how long the Sprint is if you think it’s too long or too short.

For example:

* If you start with **1-week Sprints** but find the work is too much for just one week, you can try **2-week Sprints** next time.

**Why are Sprints Good?**

1. **Quick Feedback**: You get to see if the work is going well sooner, so you can fix problems quickly.
2. **Teamwork**: Everyone knows what they need to do and works together to get it done.
3. **Focus**: The team only works on a few things at a time, so they can do a better job.
4. **Regular Rhythm**: Having a set time for each Sprint helps the team know what to expect and when things will get finished.

**Your Team’s Example**

Your team is currently doing **1-week Sprints** because you expect a lot of changes every week. But sometimes the work takes longer than a week to finish, so you're thinking about trying **2-week Sprints** instead.

**In Simple Terms**

* A **Sprint** is a short time (1-4 weeks) where the team works together to finish a small chunk of the project.
* You can choose how long your Sprint is depending on how often things change, how much focus the team needs, and how long it takes to finish the work.
* **Sprints help the team stay focused, get feedback quickly, and keep improving!**

**How to Plan and Manage Sprints Using Tools (like Asana)**

Sprints are like mini-projects that help teams stay focused, collaborate, and get things done faster. You can use work management tools, like **Asana**, to plan and track your Sprints in one place. Here’s how to do it:

**Key Steps to Plan a Sprint:**

1. **Create a Custom Field for Sprints**:
   * In Asana, you can **add a Sprint field** to track which tasks are assigned to which Sprint.
   * You can create fields for the **current Sprint** and **next Sprint** to stay ahead.
2. **Add Tasks to the Sprint Backlog**:
   * Once the Sprint field is ready, you can **assign tasks** to the Sprint backlog for **this Sprint** or the **next Sprint**.
   * Use a **drop-down menu** in the Sprint field to select which Sprint each task belongs to.
3. **Set Due Dates**:
   * For the **current Sprint**, you can add **due dates** to keep track of when each task should be finished.
4. **View and Manage Tasks**:
   * Switch to **List View** in Asana to see all tasks and track which ones are assigned to the current or next Sprint.
5. **Other Features**:
   * **Automate steps**: Use **rules** to automate tasks.
   * **Visualize work**: Use **Timeline View** to see your Sprint’s schedule.
   * **Create templates**: Make a **template** to easily set up future Sprint plans.

**Main Points to Remember:**

* **Custom Fields** help track tasks for specific Sprints.
* **Assign tasks** to either the **Current Sprint** or **Next Sprint**.
* **Due dates** keep the team on track to finish tasks on time.
* **Use tools like Timeline View** to visualize the work.
* **Automate and create templates** to make Sprint planning easier next time.

**Key Points to Remember about Daily Scrum and Sprint Review**

Here’s a simple breakdown of two important Scrum events: the **Daily Scrum** (sometimes called the "Daily Stand-Up") and the **Sprint Review**.

**Daily Scrum (Stand-Up)**

* **What it is**: A quick 15-minute meeting every day where the **Development Team** syncs up and checks in.
* **When**: Happens at the same time and place every day, usually standing up to keep it short.
* **What team members share**: Each person answers 3 questions:
  1. **What did I do yesterday** to help the team meet the Sprint goal?
  2. **What will I do today** to help the team meet the Sprint goal?
  3. **Do I see any obstacles** (impediments) that might stop me or the team from meeting our goals?
* **Goal**: The Scrum Master helps solve any problems quickly, and the team stays focused on the **Sprint Goal** and **Backlog**.
* **Timebox**: Should only take **15 minutes**.

**Sprint Review**

* **What it is**: A meeting at the end of the Sprint where the team **shows off** what they’ve built. The team gets feedback on what’s finished and what needs more work.
* **Who’s involved**: The **Scrum Team** (including the **Product Owner** and **Development Team**) and sometimes other people like stakeholders.
* **What happens**:
  1. The team **demonstrates** the product (or a part of it) they built during the Sprint.
  2. Everyone gives **feedback** on what’s working and what needs improvement.
  3. The team talks about what’s done in the **Product Backlog** and what’s ready for the next Sprint.
* **Goal**: To inspect the work done, and use feedback to **adapt** and improve the product. The team also practices **openness** and **respect** during feedback.
* **Timebox**: Shouldn’t last more than **4 hours**.
* **Example**: If the team made a new marketing email, they might show it during the Sprint Review and ask for feedback, like:
  1. "Can we make the opening line stronger?"
  2. "Can we make this part of the email clearer or shorter?"
* **Benefits**: Immediate feedback, everyone has a say, and team members learn more about each other’s work. It helps build **trust** and **shared ownership**.

**Important Concepts**

* **Product Increment**: The final product or feature that’s finished and ready to be released after a Sprint. It’s what the team shows during the **Sprint Review**.
* **Definition of Done**: Only work that meets the “Definition of Done” is considered finished and part of the Product Increment. If it's not done, it goes back into the **Product Backlog**.

### Key Points to Keep in Mind for Successful Standups:

1. **Keep Standups Short (15-20 minutes)**:
   * The goal is to keep it quick, focused, and to-the-point. It's not a retrospective or a town hall, so no long discussions. Just clear updates.
   * **Suggested format**:
     + **What did I do yesterday?**
     + **What am I working on today?**
     + **Do I have any blockers?**
2. **Focus on Visibility and Teamwork**:
   * Even though team members might not always work directly with each other (e.g., data analyst and marketer), it's helpful to bring everyone into the same space to share what they’re doing.
   * It builds a sense of **camaraderie**, and **everyone has context** about what's happening across different workstreams.
3. **Blockers Are Key**:
   * Standups aren’t just for updates; they’re also a place to quickly surface **blockers** or issues that might be preventing progress.
   * If a team member mentions a blocker, schedule a separate time to work through it in more detail rather than letting it take over the standup.
4. **Respect Different Time Zones**:
   * When working with a global team, time zone management is crucial. It’s helpful to **survey** the team to find a time that works for most people and then **stick to that time** consistently.
   * This shows respect for everyone’s time and avoids friction over scheduling.
5. **Make Meetings Optional (Not Mandatory Every Time)**:
   * If your team is working on tight deadlines and is really focused on certain components of the project, not everyone needs to attend every single daily standup.
   * Make it clear that if someone has nothing to update or if their part of the project isn’t moving forward that day, they don’t have to attend every meeting.
6. **Agenda and Timeboxing**:
   * Before each standup, it’s a great idea to **set an agenda** with allocated time slots for each update. This helps keep everyone on track and prevents the meeting from running long.
   * If a topic comes up that requires more time, you can **put a pin in it** and handle it either later in the meeting or outside the standup.
7. **Be Flexible and Respectful**:
   * If a discussion runs over time, don’t abruptly cut people off. It’s better to **find a natural break** and say something like, “We need to move on, but let’s pick this up later if we have time or in our next meeting.”
   * This shows respect for everyone’s time while also maintaining the flow of the meeting

**Releasable Increment vs. Minimum Viable Product (MVP)**

Both a **Releasable Increment** and a **Minimum Viable Product (MVP)** are related concepts but serve different purposes in Agile product development. Let's break down the differences in simpler terms!

**Releasable Product Increment**

* A **Releasable Product Increment** refers to a **completed, working piece** of the product after each Sprint. It is fully tested, functional, and ready to be released, though it doesn’t necessarily mean it will be shipped to users immediately.
* **Goal:** The goal of each Sprint is to create something that can be released **at any time** if the Product Owner decides it’s valuable. Even if the product isn’t shipped right away, it is "potentially releasable," meaning it could be put into the market if needed.
* **Definition of Done:** The **Definition of Done** is the team’s checklist for what needs to be completed for an Increment to be considered done. This includes things like testing, documentation, and any approvals required.

**Example:**

* + If you're building a pet adoption app, a potentially releasable increment might be the **feature that shows available pets**. This is a working, tested, and finished feature that can be deployed when needed, though other features are still in progress.

**Minimum Viable Product (MVP)**

* An **MVP** is a **simple version of the product** that has just enough features to **satisfy early customers** and gather feedback. It doesn’t have all the features you’d eventually want, but it has the core functionality to learn from users.
* **Goal:** The purpose of an MVP is to **launch quickly** and **learn** from real users to validate ideas before building out the full product. The MVP is about testing the concept with the least amount of work.
* **Definition:** Eric Ries, the creator of the MVP concept, describes it as "that version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort." Essentially, you’re testing core features to see if users find value in them.

**Example:**

* + In the pet adoption app, the MVP might be a simple page that shows only **cats available for adoption**. This allows the team to gather feedback from real users who want to adopt cats and improve the product before expanding it to other pets (like dogs or rabbits).

**Comparing Releasable Increment to MVP**

* **Releasable Increment** is all about producing **a complete, usable piece** of the product that can be shipped at any time. It’s about having something **fully developed** and ready for the next step.
* **MVP**, on the other hand, is a **simplified version** of the product with just enough features to **validate the product idea** and get user feedback. It’s not about having everything done, but rather learning quickly from users.

**Can a Releasable Increment be an MVP?**

Yes, a **Releasable Increment can be an MVP**, but not always. The key difference is that an Increment focuses on **completing a small, fully working part of the product**, while an MVP focuses on **gathering feedback with minimal features**.

**Example:**

* If the pet adoption app team releases a feature that shows only cats available for adoption (and everything about that feature is fully finished), this would be both a **Releasable Increment** (it’s ready to ship) and an **MVP** (it’s the minimal set of features to get feedback on).

### ****What is the Sprint Retrospective?****

* The **Sprint Retrospective** is a meeting that happens **at the end of every Sprint** (up to 3 hours for a 4-week Sprint) where the Scrum Team gets together to **reflect on the last Sprint**.
* The goal is to evaluate **what went well**, **what didn’t go well**, and **how the team can improve** in the next Sprint.

### ****What Happens in a Sprint Retrospective?****

* **Reflect on Teamwork:** The team discusses how they worked together—**the people, the processes, and the tools**.
  + What’s working? What isn’t working?
  + What improvements should be made in the next Sprint?
  + Did the improvements from the last Retrospective help?
* **Safe Space for Feedback:** It’s important to **create a safe space** where everyone feels comfortable sharing their thoughts. **Respect** is key—everyone should feel safe giving feedback without fear of negative consequences.
  + If necessary, offer **anonymous feedback options** for more candid responses.

### ****Tips for a Successful Sprint Retrospective****

1. **Create a Safe, Blameless Environment:**
   * The goal is to **improve the team’s process**, not to place blame. Make sure no one feels targeted when things go wrong.
   * Encourage honesty and openness by assuring the team there will be no negative consequences for sharing their thoughts.
2. **Get Everyone Involved:**
   * **Participation is key.** If people aren’t volunteering their feedback, actively engage them by asking:
     + What’s one thing we could try in the next Sprint?
     + What slowed us down?
     + What unexpected events happened?
3. **Balance Positive and Negative Feedback:**
   * **Celebrate successes** as well as identifying areas for improvement.
   * Ask questions like: “Where did we notice success?” or “What worked well?”
   * Recognize the team’s achievements to help boost morale and motivation.
4. **Make Actionable Improvements:**
   * **Act on the feedback.** If you talk about changes but don’t follow through, the team may get discouraged.
   * Turn good ideas into **team habits** or **norms** that improve the workflow and help everyone work better together.
5. **Focus on the Process, Not Just the People:**
   * The Retrospective isn’t just about individual performance but **how the team as a whole can improve** its processes, tools, and interactions.

### ****Example of a Sprint Retrospective****

* **Issue Found:** The team realized that **dependencies on external stakeholders** were slowing them down.
* **Solution:** In the retrospective, the team decided to improve **communication with stakeholders** to make priorities clearer. This new approach was put in place for the next Sprint.

### Sprint Retrospectives: Pitfalls to Avoid & Best Practices

The **Sprint Retrospective** is a key Scrum event where the team reflects on the past Sprint and identifies ways to improve for the next one. While retrospectives are a great opportunity for continuous improvement, they can become ineffective if not done well. In this reading, we’ll cover **common pitfalls to avoid** and **best practices** to make your retrospectives more productive and engaging.

### ****Pitfalls to Avoid****

1. **Too Many Gimmicks or Games**
   * Fun exercises and games can make retrospectives interesting, but not every team enjoys them. Use games sparingly or only when the team is looking for something new.
   * Overusing gimmicks can make retrospectives feel like a distraction instead of a meaningful reflection.
2. **Focusing Only on the Negative**
   * It’s important to discuss what went wrong, but you must also recognize the team’s successes. Focusing only on problems can lead to frustration and disengagement.
   * Be sure to **celebrate successes** and **repeat what worked well**.
3. **Changing Processes Too Quickly**
   * After every retrospective, it can be tempting to introduce a new process or change. However, it's important to give a new process **time to take effect** before deciding whether to keep it.
   * Change **too often** can overwhelm the team and create confusion. Allow a new process to run for a few Sprints before revisiting it.

### ****Best Practices for Effective Retrospectives****

1. **Ask Open-Ended Questions**
   * Open-ended questions spark deeper discussions and reflection. Avoid yes/no questions, and ask questions that require thoughtful responses.
     + For example, instead of asking **"Did we achieve the Sprint Goal?"**, ask **"How could we have better achieved our Sprint Goal?"**
   * This kind of questioning encourages critical thinking and actionable feedback.
2. **Support Diverse Communication Styles**
   * Not everyone feels comfortable speaking up in a large group. Find ways to include everyone, such as:
     + **Silent reflection**: Allow team members time to jot down their thoughts individually before the discussion starts.
     + **Pair work**: Break the team into smaller pairs or groups to discuss topics before coming together for a larger conversation.
   * This ensures all voices are heard, especially from quieter team members.
3. **Cover All Aspects of the Sprint**
   * A retrospective should cover more than just “what went wrong.” Be sure to address:
     + **Team productivity and efficiency**: How well did the team work together? What slowed us down?
     + **Scope and definition of “done”**: Were the expectations clear? Did everyone understand the definition of done?
     + **Communication within the team**: How did we communicate with each other? What could we improve?
     + **Stakeholder communication**: Were external communications effective? Did we involve stakeholders in the right way?
     + **Progress toward long-term goals**: Are we getting closer to the larger goals and release plans?
4. **Reflect on Scrum Theory and Values**
   * Periodically, take time to ask the team how they are applying Scrum principles in their work. This encourages alignment with the Scrum framework and helps the team improve their Scrum practices.
     + Example questions:
       - **“How could the team become more transparent?”**
       - **“How did we live our Scrum values (courage, commitment, focus, openness, respect, and openness) in this Sprint?”**
   * This reflection deepens the team's connection to Scrum and helps them stay focused on continuous improvement.

### ****Burndown Chart****

* A **burndown chart** shows how much work is left in the Sprint over time. It tracks the progress of tasks (user stories) being completed.
* **Goal**: Keep the team on track to meet Sprint goals.
* **How it works**:
  + Every day, the chart shows how many points (work) are finished and how many are still left.
  + If the team is behind, the Scrum Master helps unblock them.

### ****Velocity****

* **Velocity** is how many points a team can finish in one Sprint on average.
* The team uses their **average velocity** to plan how much work they can take on in future Sprints.
* **Important**:
  + There’s no "good" or "bad" velocity. It’s just how much work a team gets done in their usual time.
  + Every team has different velocity based on their size, skills, and work system.

### ****How Burndown and Velocity Help****

* With a stable **velocity**, the team can predict:
  + **How long** it will take to finish all the work in the Product Backlog.
  + **How much work** will be done by a specific date.
* Example: If a team finishes 200 points every Sprint and has 1500 points left, they know it will take about **7-8 Sprints** to finish the work.

### ****Why These Tools Matter****

* **Burndown charts** and **velocity** give teams **predictability**—they help everyone see if the team will meet deadlines and make project decisions.

### Interpreting Velocity: Dos and Don'ts

#### **What is Velocity?**

* **Velocity** is the measure of how many story points (effort estimates) a Scrum team can complete in a Sprint.
* Teams use **story points** or **T-shirt sizes** to estimate effort and calculate velocity over time.

#### **Calculating Velocity**

* When your team starts, velocity will be unknown and will only become clear after a few Sprints.
* After several Sprints, you'll know your team's average velocity, which helps in Sprint planning.

#### **Dos: Best Practices**

* **Do share velocity carefully**: When sharing velocity with external stakeholders, **provide context** (e.g., trends, date ranges). This helps avoid misinterpretation, as each team’s velocity is unique.
* **Do use velocity for insights**: Velocity trends can show how well the team is progressing over time, helping to improve future Sprints.

#### **Don’ts: Things to Avoid**

* **Don’t use velocity as a performance metric**: Using velocity to measure individual or team performance can **harm team morale**. Teams may feel pressured, leading to unhealthy work tactics.
* **Don’t compare teams based on velocity**: Each team’s velocity is subjective because story points may vary between teams. Comparing velocities is unfair and misleading.
* **Don’t use velocity to set strict delivery dates**: Velocity is **not a promise**. Estimating project delivery dates based on velocity can create **false expectations** and pressure. It takes a few Sprints to gauge velocity accurately, and lots of factors can change.

### Kanban and Scrum Boards: Key Points

#### **Purpose of the Board**

* **Visualization**: The board helps teams track progress visually by showing all work items in one place.
* **Work-in-Progress (WIP) Limits**: The board helps the team manage how many tasks they are actively working on at once, promoting focus and reducing inefficiency.
* **Flow of Work**: Teams can track the movement of tasks through stages (e.g., **To Do**, **Doing**, **Done**), making it easy to see where work is getting stuck or progressing.

#### **Why Use a Kanban or Scrum Board?**

* **Visualizing Progress**: Teams can quickly see where work is and what's next. It's especially useful in **Daily Scrums** for team discussions.
* **Managing WIP Limits**: WIP limits ensure the team doesn’t take on too much work, helping to maintain focus and avoid multitasking.
* **Tracking Work Flow**: Teams can move tasks from one stage to another (e.g., from **Doing** to **Done**) during Daily Scrums, providing a clear view of progress.

#### **Real-World Example**

* In the **Virtual Verde team**, a team member like Leo might move tasks from **Doing** to **Done** after completing them, and if he's done, he checks with teammates to see if he can assist them.

#### **Benefits**

* **Easier Tracking**: You can track progress visually, spot bottlenecks, and make adjustments easily.
* **Clearer Focus**: By limiting WIP, the team can concentrate on completing tasks rather than juggling too many at once.
* **Improved Flow**: You see how work is progressing from one phase to the next, helping to identify any issues early on.

#### 1. **Scheduling & Work Management Tools**

* **Jira** (by Atlassian): A popular tool for managing **Product Backlogs**, **Sprint Backlogs**, **velocity**, **burndown charts**, and more. It is customizable and centralizes all Scrum-related information.
* **Trello**: A simpler tool great for visual task management using **Kanban boards**, useful for personal or small team projects.
* **Asana**: Ideal for **Sprint Planning**, **Backlog management**, and task coordination. It helps teams plan, track, and communicate work with ease.

#### 2. **Documentation Tools**

* **Google Docs** & **Microsoft Word**: For capturing project details and documentation in long format.
* **Google Sheets** & **Microsoft Excel**: Useful for creating **Backlogs**, tracking data, or other project information in a flexible, spreadsheet format.
* **Google Slides** & **Microsoft PowerPoint**: For creating presentations to share project updates, findings, or plans with the team or stakeholders.

#### 3. **Collaboration & Communication Tools**

* **Video Conferencing**: Tools like **Zoom**, **Microsoft Teams**, or **Google Meet** help facilitate team meetings, Daily Scrums, and retrospectives.
* **Online Chat**: Tools like **Slack** or **Microsoft Teams** allow for quick team communication, answering questions, and unblocking work between meetings.
* **Email**: For formal communication or sharing important project updates and documents.

#### 4. **Transparency & Team Collaboration**

* Scrum teams decide together which tools work best for their needs, ensuring everyone stays informed and aligned.
* These tools enable **transparency** by keeping everyone on the same page and ensuring that information is easily accessible for all team members.

### Wrap-Up

Using the right tools helps Scrum Teams stay organized, communicate effectively, and maintain transparency. They support work management, documentation, collaboration, and communication, all critical to a Scrum Team’s success.

### Congratulations on Completing This Section!

You've covered a lot of ground in this module, and it's exciting to see how much you've learned. Here's a quick recap of the key topics we explored:

1. **The Product Backlog**:
   * We learned about refining and managing the **Product Backlog**.
   * We also covered **relative effort estimation** methods like **T-shirt sizes** and **story points** to help teams understand the work ahead.
2. **The Five Scrum Events**:
   * **Sprint**: The time-boxed iteration for delivering work.
   * **Sprint Planning**: The event where teams plan their work for the upcoming Sprint.
   * **Daily Scrum**: A short daily check-in to ensure everyone’s aligned and can unblock work.
   * **Sprint Review**: A meeting to review the work completed during the Sprint and adjust priorities.
   * **Sprint Retrospective**: A time for the team to reflect and improve their processes.
3. **Visualization Tools**:
   * **Burndown Charts**: Visualizing work remaining versus time, to track Sprint progress.
   * **Kanban Boards**: A tool to visualize the flow of tasks through different stages, like "To Do," "Doing," and "Done."
4. **Tools for Scrum Teams**:
   * **Jira**, **Trello**, and **Asana**: Tools to manage Backlogs, Sprints, and track progress in a transparent way.
   * **Collaboration tools**: For team communication, such as **Zoom**, **Slack**, and **Google Docs**, keeping everyone on the same page.

Module 4

### ****What is Value in Project Management?****

* **Value** refers to what the user or customer gains from the final product. This could be **financial benefits**, **user engagement**, or **compliance**.
* Value isn’t just about delivering a product—it's about ensuring the product truly **satisfies** the customer’s needs. Agile's number one principle is to **satisfy the customer by delivering valuable software** (or product/solution for non-software projects).

### ****Maximizing Value Delivery****

To deliver value efficiently, Scrum Teams need to focus on these three areas:

1. **Build the Right Thing**:  
   Understand **what** the customer really wants.
   * **Example**: Instead of just asking for a website, ask your customer about their **goals**—do they want more customers, greater brand recognition, etc.? This helps the team focus on solving the customer’s **real problem**.
2. **Build the Thing Right**:  
   Focus on delivering only the features that add value.
   * Avoid overbuilding—additional features may complicate the product and **delay value delivery**.
   * Keeping features relevant prevents **wasted effort** and **potential bugs**.
3. **Run It Right**:  
   Ensure the product continues to provide value after launch.
   * Think about **long-term support** and **sustainability**: how do users get help, and how do you keep improving the product?
   * **Example**: Offer **follow-up surveys** for feedback and continuously improve the product based on **real user experiences**.

### ****Virtual Verde Example****: Applying the 3 Strategies

For the **Virtual Verde** team, delivering value could look like this:

1. **Build the Right Thing**:
   * Conduct a survey to learn what types of plants customers want.
   * Use this feedback to refine the **Product Backlog** and prioritize features based on customer demand.
2. **Build the Thing Right**:
   * Secure reliable plant vendors and design products that align with customer preferences.
   * Work with marketing to ensure the right products are prominently featured.
3. **Run It Right**:
   * Send out **customer satisfaction surveys** to gather feedback after delivery.
   * Based on insights, offer value-added services like **automatic plant health systems** or **gardening tips** to support customers and enhance their experience.

### ****Case Study: Value-Driven Delivery at Penta****

#### **Background and Context**

Penta is a construction software company that had been experiencing growth and an increasing need for streamlined, efficient project delivery. As the company expanded, it faced challenges in meeting deadlines, managing resources effectively, and delivering products that consistently met customer needs. The company recognized that they needed to embrace new methodologies to enhance their value delivery and better respond to customer demands in a fast-paced industry.

#### **The Problem**

The problem at Penta was that they were not delivering software or services in a way that truly maximized value for their customers. The team was facing inefficiencies, such as:

* **Slow response to changes**: Traditional project management methods were slowing down their ability to adapt quickly to customer feedback.
* **Lack of collaboration**: There were silos within the organization, and different departments (engineering, marketing, product) were not collaborating in a cohesive way, which led to misaligned priorities and confusion.
* **Product features were not always aligned with customer needs**: Even though the company was delivering software, it wasn’t always clear whether the product features added enough value to their users.

The company needed a way to focus on delivering the **right features**, with greater **efficiency** and better **alignment** to customer value.

#### **The Goal**

The goal of Penta’s shift to Agile was to improve their ability to **deliver value-driven software** faster and more effectively, while enhancing collaboration across the organization. They aimed to:

* **Respond more quickly to customer needs**.
* **Ensure alignment** between different teams (e.g., engineering, product, marketing).
* **Increase transparency and communication** across the organization to avoid silos and ensure everyone was working toward the same objectives.
* **Build only the features that truly added value**, avoiding the temptation to overbuild or create unnecessary features.

#### **The Agile Task Force: Solution and Implementation**

To address these challenges, Penta formed an **Agile task force**—a team made up of members from different departments (engineering, product, sales, marketing). This task force was tasked with driving the Agile transformation and ensuring that every project focused on delivering value.

The process they followed can be broken down into several key steps:

1. **Cross-functional Collaboration**:  
   The task force brought together different departments, fostering better communication and understanding across functions. This was critical in ensuring that everyone understood the value each department brought to the product and customer experience.
2. **Scrum Teams for Specific Tasks**:  
   Penta adopted the **Scrum framework** and created **Scrum teams** around specific initiatives. Each team was responsible for delivering a feature or product increment that contributed directly to the customer’s needs. This allowed the company to focus on delivering small, iterative improvements that added immediate value to the customer.
3. **Regular Feedback Loops**:  
   The task force implemented **feedback loops** to ensure that customer feedback was being consistently integrated into the product development process. This included frequent **Sprint Reviews** and **Sprint Retrospectives**, where teams would review what had been built, how well it met customer needs, and how processes could be improved.
4. **Value Prioritization**:  
   One of the most crucial steps was ensuring that the team focused on building only the **right features**. This meant working with customers to truly understand their needs and prioritizing the most valuable features to the user. The Product Backlog was continuously updated based on customer feedback, ensuring that only features that contributed the most value were being worked on.
5. **Continuous Improvement and Iteration**:  
   By working in shorter cycles and using **Sprint Retrospectives**, the company was able to regularly evaluate and improve both the product and their internal processes. This continuous improvement allowed Penta to make adjustments quickly and increase their overall productivity.

#### **Key Results and Benefits**

As Penta moved toward Agile and Scrum, the company saw several key benefits:

1. **Improved Efficiency**:  
   Scrum's iterative approach allowed Penta to break down large projects into smaller, manageable chunks, which led to **faster delivery** of software features. Teams were able to deliver value to customers more quickly than with traditional waterfall processes.
2. **Increased Customer Satisfaction**:  
   By focusing on the features that customers actually wanted, rather than building based on assumptions or internal desires, Penta was able to increase **customer satisfaction**. The regular feedback from customers ensured the product was aligned with user needs and expectations.
3. **Better Team Collaboration**:  
   The cross-functional Scrum teams led to better communication and **collaboration** across departments. No longer were teams siloed; everyone had a shared understanding of the goals and progress, which improved overall team morale and productivity.
4. **Enhanced Transparency**:  
   Through the use of **Scrum tools** like the **Product Backlog**, **Sprint Reviews**, and **Burndown Charts**, Penta gained **increased transparency** in both the product development process and their overall business operations. Everyone in the organization was able to see where the team was in terms of delivering value and could contribute or adjust priorities as needed.
5. **Continuous Value Delivery**:  
   The focus on value-driven delivery allowed Penta to prioritize **incremental value delivery**. By breaking down features into smaller increments and delivering them in Sprints, they were able to **adapt to changing customer needs** and deliver the highest-value items first.

#### **Analyzing the Case Study**

**What is the issue?**

* The company struggled with inefficiencies, lack of collaboration, and misalignment between departments and customer needs.

**What is the goal of the analysis?**

* To understand how Penta’s Agile transformation, specifically through Scrum, helped the company focus on **value delivery** and improve operational efficiency.

**What is the context of the problem?**

* Penta was growing quickly and facing increasing competition in the construction software industry. They needed to improve the speed and relevance of their product delivery to stay ahead.

**What key facts should be considered?**

* Penta implemented Scrum, formed cross-functional teams, and prioritized customer feedback. They improved collaboration, efficiency, and transparency through these steps.

**What alternatives are available to the decision-maker?**

* While Scrum and Agile worked well for Penta, other methods like **Kanban** or **Lean** could also be explored as alternatives for streamlining processes and improving value delivery.

**What would you recommend, and why?**

* I would recommend continuing to build on the Scrum framework and incorporating more advanced **Agile practices**, such as **continuous integration** and **test-driven development**, to further enhance value delivery and maintain competitive advantage.

### ****Conclusion: Scrum and Agile’s Value at Penta****

Penta's journey demonstrates how Scrum and Agile principles can bring **tangible value** to an organization. By focusing on value-driven delivery, prioritizing collaboration, and continuously improving processes, Penta was able to deliver higher-quality products faster and meet customer needs more effectively. The Agile task force played a crucial role in this transformation, ensuring that the company remained focused on its ultimate goal: delivering **value** to users.

### ****1. Flexibility of Agile****

* **Agile as a Framework and Mindset**: Camron emphasizes that Agile is not just a set of processes or methodologies, but a **mindset** that adapts as new information and insights are gathered. This is crucial when making decisions early in a project when the team knows the least, as it's expected that some decisions will need to change as the project progresses.

### ****2. Incremental Delivery****

* **Delivering in Chunks Over Time**: One of the fundamental principles of Agile is **incremental delivery**, where the product is developed and delivered in small, manageable chunks (or increments) over a long period. This contrasts with Waterfall, where everything is delivered at the end. In Agile, the team delivers value early and continuously, allowing customers to get useful features and feedback at each stage of development.

### ****3. Waterfall vs. Agile****

* **Waterfall (All-at-once Delivery)**: Camron compares Agile with the traditional **Waterfall** approach, where the entire product is delivered at once, typically after long periods of development. This creates a **delay in value extraction** since customers must wait until the end to experience the product’s benefits. For some types of projects, like building a house or a car, this approach can make sense, as partial deliveries might not be functional or useful.
* **Agile’s Incremental Value**: In Agile, the idea is that **you can ship 90% of the project** and deliver value to the customer immediately, while the remaining 10% is still being refined or worked on. This ensures that the customer can start using the product and gaining value from it while the team works on the final components.

### ****4. Managing Change****

* **Change and Cost**: Camron mentions that **change is never free**. When changes are made to the project plan or scope, it impacts the existing work in progress and could involve revising estimates, shifting resources, or adjusting timelines. However, Agile provides flexibility to manage these changes within the **project’s threshold or budget**. It’s not about perfection, but about making the product **good enough** at each stage, enabling a balance between quality and agility.

### ****5. Delivering Value Early and Often****

* The core message is that **Agile enables the extraction of value earlier** in the development process. Instead of waiting for the final product, customers can begin using and benefiting from the product incrementally. This makes Agile particularly suited for projects where early feedback is crucial and where products can evolve with real-world usage and insights.

### ****What is a Value Roadmap?****

A **value roadmap** is essentially a guide that helps Agile teams stay aligned with the product vision and strategic goals, ensuring the team delivers value at each stage of the project. It outlines the **where, how, and what** of the development process. This roadmap helps the team navigate through the project, gather customer and stakeholder input, and apply those insights to create a product that meets customer needs.

### ****Three Key Components of a Value Roadmap****

1. **Product Vision**:
   * **Purpose**: The product vision is the foundational statement that guides the entire project. It’s derived from **user interviews**, **market analysis**, and the team’s understanding of customer needs.
   * **What it defines**: The product vision answers three key questions:
     + What is the product?
     + How does it support the customer’s business strategy?
     + Who will use it?
   * The product vision becomes the **team’s north star**, guiding decisions and aligning everyone on the same goal.
2. **Product Roadmap**:
   * **Ownership**: The **Product Owner** is responsible for creating and maintaining the product roadmap.
   * **Purpose**: The product roadmap provides a **high-level view** of the expected product. It outlines:
     + The **requirements** (what the product must do)
     + An **estimated schedule** for reaching important milestones.
   * **Goal**: It ensures the team is building **the right thing**, with a focus on meeting the product vision.
3. **Release Plans**:
   * **Ownership**: The **Product Owner** and **Project Manager** work together on the release plan.
   * **Purpose**: A release plan defines the **milestones for delivering features** and includes:
     + The **release goal** (what business value is expected to be delivered)
     + A **list of backlog items** (such as epics, user stories, or features) required for the release
     + The **approximate release date** and any other important dates (e.g., holidays, conventions) that could impact the timeline.
   * **Flexibility**: The first release date is set in stone, but the rest of the release plan is based on early estimates and can change as the project progresses.

### ****Working with the Value Roadmap****

* The roadmap is a **living document** that evolves over the course of the project. The team should work collaboratively with stakeholders, and **regular feedback** is crucial to ensuring the project stays aligned with the product vision and the customers' needs.
* **Iterative Process**: The roadmap guides the team through multiple iterations, and each iteration brings the product closer to delivering value to customers.

### ****Why Use a Value Roadmap?****

* **Focus on Value**: The roadmap ensures the team remains focused on the product’s strategic goals, delivering value to customers in increments.
* **Alignment**: By having the product vision, roadmap, and release plan clearly defined, all team members and stakeholders remain aligned with the project’s goals and progress.
* **Agile Principles**: A value roadmap supports key **Agile principles**, such as flexibility, customer collaboration, and delivering working products early and often.

### Product Roadmaps: Benefits, Pitfalls, and Best Practices

#### **Benefits**:

* **Clarifies deliverables**: Outlines the sequence and timeline of project tasks.
* **Aligns efforts with vision**: Helps teams understand how their work supports the overall goal.
* **Shows incremental value**: Demonstrates progress over time, rather than delivering everything at once.
* **Helps stakeholder planning**: Gives stakeholders a rough layout of work and milestones.

#### **Pitfalls to Avoid**:

* **Treating roadmaps as fixed**: It can restrict flexibility and hinder adaptation to new information.
* **Over-focusing on dates**: Avoid spending too much time perfecting delivery dates; keep them flexible and refine as the project progresses.
* **Focusing too much on planning**: Don’t let roadmap creation replace actual deliverable work.

#### **Best Practices**:

* **Keep it visible**: Regularly refer to the roadmap to keep the team aligned.
* **Highlight priorities**: Make sure the most important and high-value items stand out.
* **Share with stakeholders**: Make it accessible to all relevant parties for planning.
* **Regular reviews**: Continuously update and review the roadmap with the team and stakeholders to stay on track.

### Tips for Creating an Effective Value Roadmap

1. **Product Roadmap**:
   * Provides a high-level view of the product, its requirements, and estimated timelines.
   * **Tip**: Keep product release dates as rough estimates. Agile embraces change, so overly specific dates could lead to unrealistic expectations and pressure.
2. **Release Plans**:
   * Developed by the Product Owner and Project Manager/Scrum Master, connecting the product roadmap with the team's capacity and velocity.
   * **Tip**: Release plans should be realistic, based on the team’s actual pace (velocity). If there's a fixed deadline (like an event or convention), communicate this clearly to prioritize must-have features.
   * **Tip**: Treat the release plan as a living document that can change based on new information (e.g., team velocity, product scope changes, or better understanding of effort needed for features).
3. **Review Before Sprint Planning**:
   * Always review the release plan before each Sprint Planning session.
   * **Tip**: If the team is off-track, the Scrum Master should have a transparent conversation with the Product Owner and business stakeholders to adjust the plan.

### Responding to Change Over Following a Plan

Agile values "responding to change over following a plan," and this principle guides how teams handle changes in the project. Changes are inevitable, and Agile emphasizes the need to adjust plans to deliver the best value. The process of implementing a change can be broken down into three stages:

#### **1. Identifying a Needed Change**

First, determine when a change is needed by evaluating three key areas:

* **Scope**: The content of the product, features in the Product Backlog, deliverables, or intended users.
* **Time**: Changes in the project timeline, release dates, or Sprint duration.
* **Costs/Resources**: Changes in the team size, roles, or available equipment.

**Sources of Change**:

* **Scope**: Customer feedback or feature adjustments.
* **Time**: Shifting critical dependencies or deliverable dates.
* **Resources**: Identified gaps in staffing or team capacity (from Sprint Retrospectives).

#### **2. Deciding to Make the Change**

Once the need for a change is identified, it’s time to decide whether to implement it.

* **Decider**: Typically the **Product Owner** or a senior stakeholder should have the final say.
* **Considerations**: Gather supporting data, discuss the benefits and costs of the change, and identify areas of uncertainty.
* **Documentation**: Ensure the decision-making process and the decision itself are documented.

#### **3. Implementing the Change**

Once approved, follow these steps to implement the change:

* **Document the Change**: Include meeting notes, pros/cons, assumptions, and any data used in the decision.
* **Update Affected Artifacts**: Revise documents like the Product Backlog, roadmaps, staffing plans, and integration dates. Add version labels (e.g., "Version 1.2") for clarity.
* **Communicate with Stakeholders**: Share the change with all relevant stakeholders through meetings, documentation, or email.
* **Monitor the Change**: Track how the change affects the project, and ensure the team is aligned and aware of the new direction.

### Leading Agile Change Made Simple

If you're a project manager in a company that wants to switch to Agile, or you're helping a team start using it, here’s a quick guide on how to make that change smooth and successful!

#### **1. Understand the Culture**

Every company has its own way of working. Agile isn’t just about new processes—it’s about changing how people think and work together. If you don’t consider the company’s culture (how people usually act), the change can be harder.

#### **2. Take Your Time**

Changing a company’s way of working doesn’t happen overnight. It might take months or even years! But, even small steps forward are important and lead to bigger changes over time.

#### **3. Create Ownership and Urgency**

To make change happen, the team has to feel like it’s their job to make it work. Help them understand why the change is important. Ask questions like:

* What’s stopping us from doing our best work?
* How can we be faster and better than our competitors?

These questions get people thinking about the need for change and make them excited to improve.

#### **4. Link Agile to Company Goals**

Make sure everyone understands how Agile helps the company’s bigger goals. If the company wants to be faster or better at customer service, explain how Agile helps make that happen by delivering work more quickly and getting feedback faster.

#### **5. Show Progress and Keep Improving**

Once you start, show the team how things are getting better. Use tools like feedback sessions to make sure things are improving and adjust as needed. This keeps people motivated!

#### **6. Be Ready for Challenges**

Change can be hard, especially if the team is unsure about Agile. But, with time, they’ll see how it makes their work easier and better.

#### **Example: Virtual Verde**

When Virtual Verde started their Agile journey, they used the growing trend of home office decor to create a sense of urgency. They showed how Agile would help them seize this opportunity, and their past experience gave them the confidence to move forward.

#### **Final Thoughts**

Agile transformation takes time, but with patience, clear communication, and helping people feel part of the process, you can make it work. Keep encouraging your team, and soon they’ll see the benefits of Agile for themselves!

### The Influencer Change Framework - Made Simple

In this guide, we'll learn how to **influence** change in a team or organization, especially when using Agile practices. The key is **influence**, not persuasion. Influence creates **lasting change**, while persuasion is usually short-term.

#### What is an Influencer?

An **influencer** is someone who can inspire others to **change their behavior** and **adopt new ways of doing things**—whether it's in their work habits, mindset, or goals. As a project manager, you'll often need to influence people to try new tools, processes, or even change the way they work entirely. The goal is to make change **sustainable**—not just a temporary shift.

#### **Three Keys to Influence**

The **Influencer Change Framework** has three main steps to help you lead successful change:

1. **Clarify Measurable Results**
   * First, define **exactly what you want** to change and why. Make your goals **specific** and **measurable** (SMART goals).
   * Ask questions like:
     + What do we want to achieve?
     + How will we know if we succeed?
     + When should we see results?
   * Keep the results **visible** to the team so they can track progress and stay focused.
2. **Find Vital Behaviors**
   * A **vital behavior** is a key action or decision that can create big changes.
   * For example, if you want more Product Owner involvement in the project, a developer might **send an email to the Product Owner** after finishing a feature, asking for feedback.
   * **Tracking behaviors** is important. If you want change, look for the small, pivotal actions that need to happen to drive it.
3. **Use the Six Sources of Influence** The researchers behind this framework found **six key factors** that help people adopt new behaviors. You can use all of them to increase the chance of successful change:
   * **Personal Motivation**: Are people internally motivated? Help them get excited about the change.
     + Example: Make sure the Product Owner is timely and effective in their feedback.
   * **Personal Ability**: Do people have the skills and knowledge they need to change?
     + Example: Teach developers how to quickly send feature demos to the Product Owner.
   * **Social Motivation**: Are there social networks (team members, peers) encouraging the new behavior?
     + Example: Encourage team members to remind each other during Daily Scrums to contact the Product Owner.
   * **Social Ability**: Does the team have resources or tools that help them perform the new behavior?
     + Example: Give the team a tool to track all demos sent to the Product Owner.
   * **Structural Motivation**: Are there rewards or incentives for performing the new behavior?
     + Example: Offer a small reward, like a coffee gift card, for the Product Owner to give after each Sprint.
   * **Structural Ability**: Are there systems or rules in place to make the right behavior easier?
     + Example: Set up the content management system to **automatically include the Product Owner’s name** when sending out reviews.

#### **Final Thoughts**

By applying these steps, you’ll influence your team or organization in a **positive, lasting way**. You’ll not only change how people behave, but also **help them see the benefits** of those changes, making the shift to Agile much smoother!

### Your Role as an Agile Coach – Simplified

As a **project manager** or **Scrum Master**, your job isn’t just to manage tasks, it’s to **coach** the team to improve continuously. Think of yourself as a **sports coach** who helps the team get better, win, and learn from mistakes. Here’s how you can approach coaching in three simple steps:

#### 1. **Design the Plays with the Team**

* Just like a sports coach creates the game plan, the Scrum Master helps the team create their **"playbook"**—a plan for how the team will work together.
* The playbook includes things like how to run Sprint Reviews, how the team works daily, and how to communicate plans to stakeholders.
* Always involve the team in the process. When it’s time to update or improve how the team works, make sure you get **input from everyone**. Work through the changes together and ensure the team understands how it all flows.
* **Example**: I once led a meeting where the team used sticky notes to brainstorm what wasn’t working in our process. We then prioritized the changes together and put them into action.

#### 2. **Provide Feedback**

* As a coach gives feedback during the game, the Scrum Master gives **feedback throughout the project**—both big and small.
* **Daily feedback**: Give quick, in-the-moment guidance to help the team stay on track.
* **Big picture feedback**: Step back and review how the team is doing, just like watching game footage to find patterns. Recognize what’s working well and encourage the team to keep it up.
* Feedback isn't just about fixing things that are wrong. It’s also about **celebrating successes** and reinforcing what’s going right.

#### 3. **Celebrate and Learn**

* Whether the team wins or loses, it’s important to acknowledge the outcome. **Celebrate the wins** (successful launches, happy customers) and **learn from the losses** (mistakes or unmet goals).
* **Failures** are just **data** that help the team improve. Remember, Thomas Edison said, “I have not failed—I’ve just found 10,000 ways that won’t work.”
* As the Scrum Master, your job is to help the team stay **positive** even in tough moments, learning from each experience.

#### **Summary**

You, as the **Scrum Master** or **Agile project manager**, are crucial to the team’s success. Your three main roles are:

1. **Design plays** (plan how the team works together).
2. **Give feedback** (both in real-time and from a bigger picture).
3. **Celebrate and learn** (help the team stay positive, even after setbacks)

### Coaching vs. Managing in Agile – Simple Breakdown

In Agile, **coaching** and **managing** are two important roles for a project manager or Scrum Master. They are similar but have key differences, especially in how you communicate with your team. Here’s a simplified explanation:

#### 1. **Managing**: Giving Direction

* **Managing** is about **giving direction** and making decisions to keep things on track. As a manager, you are responsible for organizing the team, delegating tasks, and monitoring progress.
* In **traditional project management**, a manager oversees the work and makes decisions on behalf of the team.
* In **Agile**, the team is more self-managing, meaning they have the freedom to figure out how to complete tasks themselves. But sometimes, a manager’s action is needed when:
  + There’s an emergency
  + You are behind a deadline
  + You need to make important decisions quickly

In these situations, **managing** becomes crucial.

#### 2. **Coaching**: Teaching and Empowering

* **Coaching** is about **teaching** and helping the team grow. Instead of just giving directions, a coach helps team members learn to solve problems themselves by asking questions and providing feedback.
* A coach trusts the team to make decisions and encourages them to develop their own skills.
* **Key aspects of coaching**:
  + **Motivate**: Encourage team members to see the value in their work.
  + **Support**: Be available when team members need help or guidance.
  + **Encourage and appreciate**: Acknowledge the team’s efforts and celebrate their successes, even after setbacks.

### Common Challenges in Agile and How to Overcome Them

As a **Scrum Master** or **Project Manager**, you're there to help your team improve and work through challenges as they adopt **Agile** practices. In this video, we explore some common challenges Agile teams face in areas like **value delivery**, **business collaboration**, and **team dynamics/culture**. Here’s a simplified breakdown of what you might encounter and how to handle it:

### 1. ****Value Delivery Challenges****

**What is it?**  
Value delivery is about ensuring your team delivers high-quality work **frequently**. When teams struggle with value delivery, here are some signs to look for:

* Missing deadlines or taking too long to complete tasks
* Burnout, long working hours, and exhaustion
* Too many tasks in progress (which delays completion)

**How to fix it?**

* **Demos**: Run more demos to check if the team is on track and delivering value.
* **Retrospectives**: Ask the team what might be slowing them down (e.g., dependencies or communication problems).
* **Clarify "Done"**: Make sure everyone understands what “done” means for each task.
* **Limit Tasks per Sprint**: Focus on completing fewer tasks in one Sprint, instead of spreading the team too thin.

**Example**:  
My team once struggled with taking on too much in each Sprint. The solution was to focus on fewer tasks, complete them well, and then move on to the next ones. This improved the flow and reduced stress.

### 2. ****Business Collaboration Challenges****

**What is it?**  
Business collaboration is all about ensuring developers are working closely with business people (like the Product Owner) to build the right product. If you're facing issues, signs might include:

* Overwhelming feedback or constant change requests from business people
* The team avoiding feedback because it leads to more work or frustration
* An "us vs. them" attitude between developers and business stakeholders

**How to fix it?**

* **More Demos**: Regularly demo the work to get feedback early and often, so it's not overwhelming later.
* **Solution Design Sprint**: Consider running an entire Sprint focused on **designing** the solution, with both the team and business people collaborating.
* **Control Backlog Changes**: Limit changes to the **Backlog** between Sprints. This reduces distractions and helps the team maintain focus.

**Example**:  
At one point, a director kept interrupting the team for urgent requests, disrupting their workflow. We solved this by having the director approach the **Scrum Master** first, ensuring requests were planned and didn’t interfere with ongoing work.

### 3. ****Team Dynamics and Culture Challenges****

**What is it?**  
Team dynamics and culture issues often arise from differences in motivation and communication styles. Here are a few signs your team might be struggling:

* **Low morale**: Team members seem unhappy or irritable.
* **Frequent conflict**: Arguments or unresolved issues within the team.
* **Lack of conflict**: No one is disagreeing, but it’s because they feel uncomfortable speaking up.

**How to fix it?**

* **Team Brainstorming**: Run sessions to identify and improve team working practices. Get everyone to share their experiences (both positive and negative).
* **Change the Workflow**: Pair team members on tough tasks or change how meetings are run to shake things up.
* **Training**: Take a class together or watch videos on **team dynamics** and discuss it as a group.
* **Retrospectives**: Use fun and creative techniques like **Six Hats Thinking** during retrospectives to get everyone to see different perspectives and ensure no issue is overlooked.

**Example**:  
I once worked with a team where no one spoke up in meetings. We used the **Six Hats Thinking** method to encourage different views and get the team comfortable with disagreeing in a healthy way.

### Key Takeaways:

1. **Value Delivery**: Ensure the team delivers high-quality work on time by limiting tasks and checking in regularly through demos and retrospectives.
2. **Business Collaboration**: Maintain a steady flow of feedback between the team and business stakeholders to avoid overwhelming change requests and conflicts.
3. **Team Dynamics & Culture**: Watch for signs of low morale, unresolved conflicts, or too much harmony (which might indicate fear of conflict), and use brainstorming, training, and creative retrospectives to improve team communication and collaboration.

### Common Coaching Challenges for Agile Teams

As an **Agile Project Manager** or **Scrum Master**, you help your team improve and work better together. But sometimes, you’ll face challenges. Here are **three common problems** and how to fix them:

### 1. ****Managing a Stable Product Roadmap****

**What’s the problem?**  
In Agile, the plan can change a lot. But if things change **too much**, it can confuse the team. This can happen because the **Product Owner** is too ambitious (setting goals that are too big) or because the team is making too many guesses (assumptions) about what the customer really wants.

#### **Solution**

* **Set clear rules**: Agree on when new ideas should be reviewed and planned.
* **Do regular reviews**: Check the roadmap with the team often to keep everyone on the same page.
* **Share knowledge**: Help the Product Owner understand how much work is involved, so they set realistic goals.

### 2. ****Incomplete Implementation of Scrum****

**What’s the problem?**  
Sometimes teams try **Scrum** but only use parts of it, or they don’t fully understand how it works. This can cause confusion, like people not knowing their roles or missing key meetings.

#### **Solution**

* **Do Scrum fully**: Make sure the team understands and follows all Scrum practices.
* **Define roles clearly**: Ensure everyone knows their job (Product Owner, Scrum Master, Developers).
* **Give coaching**: Explain the "why" behind Scrum and help the team use it properly.

### 3. ****Lack of Stability within the Team****

**What’s the problem?**  
When people leave or join the team a lot, it makes things unstable. This can slow down progress because new members need to catch up and the team’s workflow gets disrupted.

#### **Solution**

* **Quick onboarding**: Help new team members get up to speed fast.
* **Pair programming**: Pair new members with others to learn quickly.
* **Shorter sprints**: If team members leave often, use shorter sprints so work can wrap up more quickly.

### To Sum Up

The three main challenges are:

1. Managing a stable product roadmap with **realistic goals**.
2. Implementing **Scrum properly** with full support.
3. Dealing with **team changes** by helping new members join smoothly.

### The Evolution of Agile: What's Next?

Agile has grown incredibly popular since its creation in 2001, and today, **85% of organizations** use Agile methods, especially in product-centric models. One report even says that **30%** of these companies mix Agile with other methods to create a **hybrid approach**. Knowing how to blend different methods is a useful skill for your career in project management.

### Why Agile is So Popular

The world today is full of **volatility, uncertainty, complexity, and ambiguity** (VUCA). Businesses face many challenges, and Agile helps them respond quickly and adapt. While the **Agile Manifesto** (the core values of Agile) hasn’t changed much, the **frameworks** inspired by it have evolved to keep up with new challenges.

### Emerging Agile Frameworks

#### **DevOps**

One framework that’s evolving is **DevOps**, which combines **software development** and **IT operations**. DevOps aims to **deliver software quickly** and reliably, especially for large-scale systems used by millions of people. It’s all about creating teams that can build, update, and manage software systems quickly, securely, and with high quality.

* DevOps helps companies release new products and features faster, which gives them a competitive edge.
* As a project manager in **DevOps**, you’d be working with teams that are building **large systems** that change the world.

#### **Business Agility**

Another growing area is **Business Agility**, where **Agile principles** are applied across an entire organization. This means using Agile to rethink things like **financial planning, hiring, governance**, and more—helping businesses stay **flexible** and **adapt** in a rapidly changing world.

* Large companies may use frameworks like **Scrum of Scrums** or **SAFe** (Scaled Agile Framework) to scale Agile across different teams.

### Agile in Other Industries

Agile isn’t just for **tech** anymore—it's spreading to other industries too:

* **Sales**: I trained the Google sales team in **Latin America**, and they loved how Agile helped them handle market changes and reduce risks. They found that Agile’s **early feedback** and **team collaboration** helped them deliver better results.
* **Construction**: The construction industry is using Agile to tackle delays and budget overruns. They’ve adapted the **Agile Manifesto** to fit their world, focusing on **cooperation** and **minimizing silos** in projects.

### Agile in Your Personal Life

Agile can even be applied to **personal projects**. For example, I used a **Kanban board** to plan my recent move. Whether it's organizing a family event or cleaning your garage, you can use **Agile tools** to stay organized and focused.

### The Future of Agile

From **DevOps** to **business agility**, Agile continues to evolve. Whether in technology, construction, or even personal tasks, Agile helps teams adapt quickly and deliver better results. As an Agile project manager, you can be part of **shaping the future** of Agile by contributing to these innovations.

**Scaling Agile for Large Teams and Projects**

When your team grows beyond the typical **Scrum size** (9 members) or the product requires multiple teams, you’ll need a **scaled Agile framework**. Here are five popular frameworks for handling larger initiatives:

**1. Scaled Agile Framework (SAFe)**

SAFe is a **Lean-Agile** framework used to scale Agile across large organizations. It focuses on delivering value quickly and efficiently, organizing work into **Agile Release Trains** (ARTs) based on value streams (e.g., sales). Key values include **alignment**, **built-in quality**, **transparency**, **program execution**, and **leadership**. SAFe provides structured guidance but requires flexibility to maintain agility.

**2. Scrum of Scrums**

This technique helps coordinate multiple Scrum teams working on the same product. **Scrum of Scrums** meetings (daily, weekly, or bi-weekly) are used to sync progress across teams. Each team sends a representative to these meetings, and a **Scrum of Scrums Master** oversees coordination. There’s no official framework; teams design their own way to manage integration and communication.

**3. Large Scale Scrum (LeSS)**

LeSS expands Scrum to larger teams while staying true to Scrum principles. **LeSS** promotes **transparency**, **customer focus**, and **continuous improvement**. It offers two frameworks: **Basic LeSS** (up to 50 people) and **LeSS Huge** (50–6000+ people). The goal is to scale Scrum with minimal overhead while focusing on value delivery.

**4. Disciplined Agile Delivery (DAD)**

DAD is a **hybrid** approach that combines strategies from **Scrum, Lean, Kanban**, and others. It helps organizations make decisions about scaling based on their needs and context. DAD includes **four layers**: Foundations, Disciplined DevOps, Value Streams, and Disciplined Agile Enterprise, ensuring a tailored approach to scaling Agile in complex environments.

**5. The Spotify Model**

Not a true framework, but an **organizational model** based on **Spotify’s culture** of autonomy and trust. It organizes teams into **Squads** (small, autonomous teams), **Tribes** (groups of Squads), **Chapters** (specialist groups), and **Guilds** (interest-based communities). While it’s proven effective at Spotify, it’s not a one-size-fits-all solution and relies on strong organizational culture.

### Tips for Landing an Agile Project Management Role

**Agile Project Manager Jobs**  
Roles like **Agile Project Manager, Scrum Master, IT Agile Project Manager**, or **DevOps Project Manager** are in demand. When searching for a job, look for a position that suits your experience, industry knowledge, and growth potential. Cultural fit is also key—ensure the company supports your personal and professional growth.

**Interview Insights**  
As a hiring manager at Google, I often start by asking candidates, "What’s the difference between Agile and Waterfall?" This helps gauge their understanding of Agile beyond just Scrum. I also ask about their experience with choosing between Agile and Waterfall for different projects, and how they handle team resistance to Agile practices. It’s important to hire someone who works with the team, not forces practices onto them.

**Questions You Should Ask**  
When interviewing, ask questions like:

* How supportive is the management of blending project management approaches?
* What’s the culture like here?
* How often will I hear from users or customers?
* What would a typical day look like in this role?

### Bringing Agile to Your Team

**Start Small**  
Introduce Agile gradually—try things like a **Kanban board** or a **retrospective** after milestones.

**Listen to Feedback**  
Get the team's input on what’s working and adjust accordingly.

**Be Strategic**  
Address current team challenges with targeted Agile practices. For example, use **relative estimation** to improve effort prediction or assign a **Product Owner** to streamline decision-making.

**Find Allies**  
Seek out others who support Agile in your organization for guidance and support.

By following these tips, you’ll be well on your way to landing an Agile project management role or successfully implementing Agile within your team.