✅ 1. Activity Types and Activity Dependency Determination

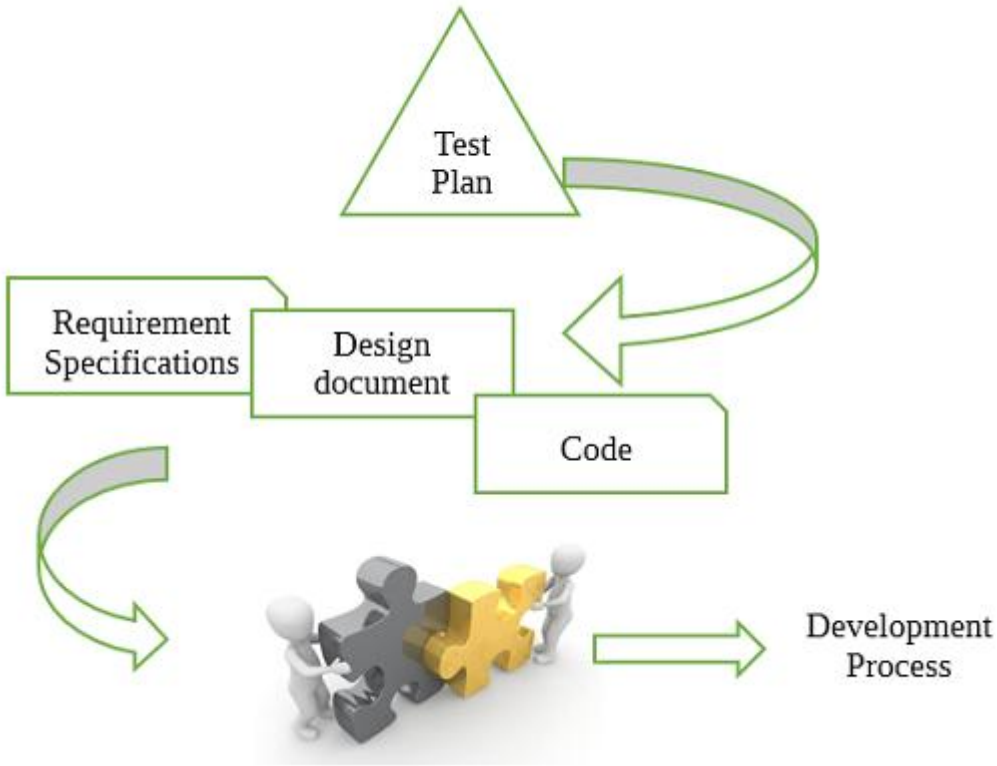
🛠️ Activity Types (Brief Explanation):

Activity types refer to the **different kinds of tasks or actions** that must be performed to complete a project. These are the **building blocks of a project schedule**.

Common activity types include:

- **Development**: Writing code, building software, or constructing components.
- **Testing**: Checking if the product works correctly (e.g., unit testing, system testing).
- **Documentation**: Writing user guides, manuals, or technical documentation.
- **Review**: Examining code, designs, or documents to ensure quality and accuracy.

Each activity has an estimated duration, resources, and may depend on other activities to start or



finish.

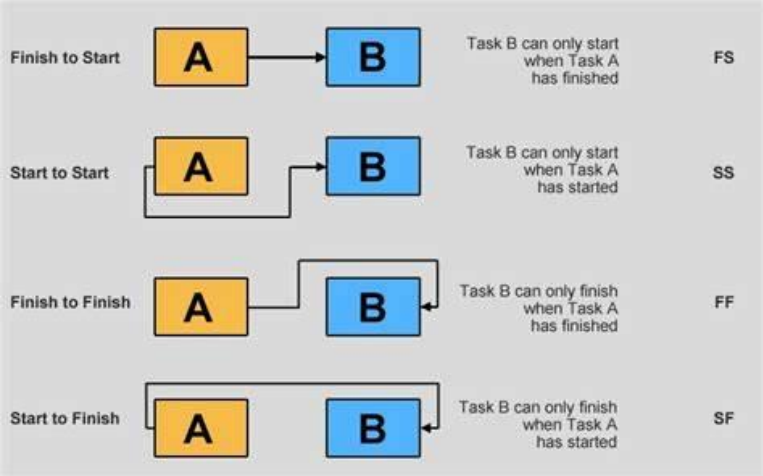
🔗 Activity Dependency Determination:

Activity dependencies show how **tasks are connected**—which task must be done **before, after, or at the same time** as another. Understanding these relationships helps build a logical and efficient schedule.

There are four main types of dependencies:

1. **Finish-to-Start (FS)**: Task B can't start until Task A finishes.
   Example: You can't start testing until development is complete.

2. **Start-to-Start (SS)**: Task B can't start until Task A starts.
   Example: Write documentation while development begins.
3. **Finish-to-Finish (FF)**: Task B can't finish until Task A finishes.
   Example: Final testing ends when final development ends.
4. **Start-to-Finish (SF)**: Task B can't finish until Task A starts (rare).
   Example: Old system can't shut down until new one starts.



📅 **Why They Are Important for Project Scheduling:**

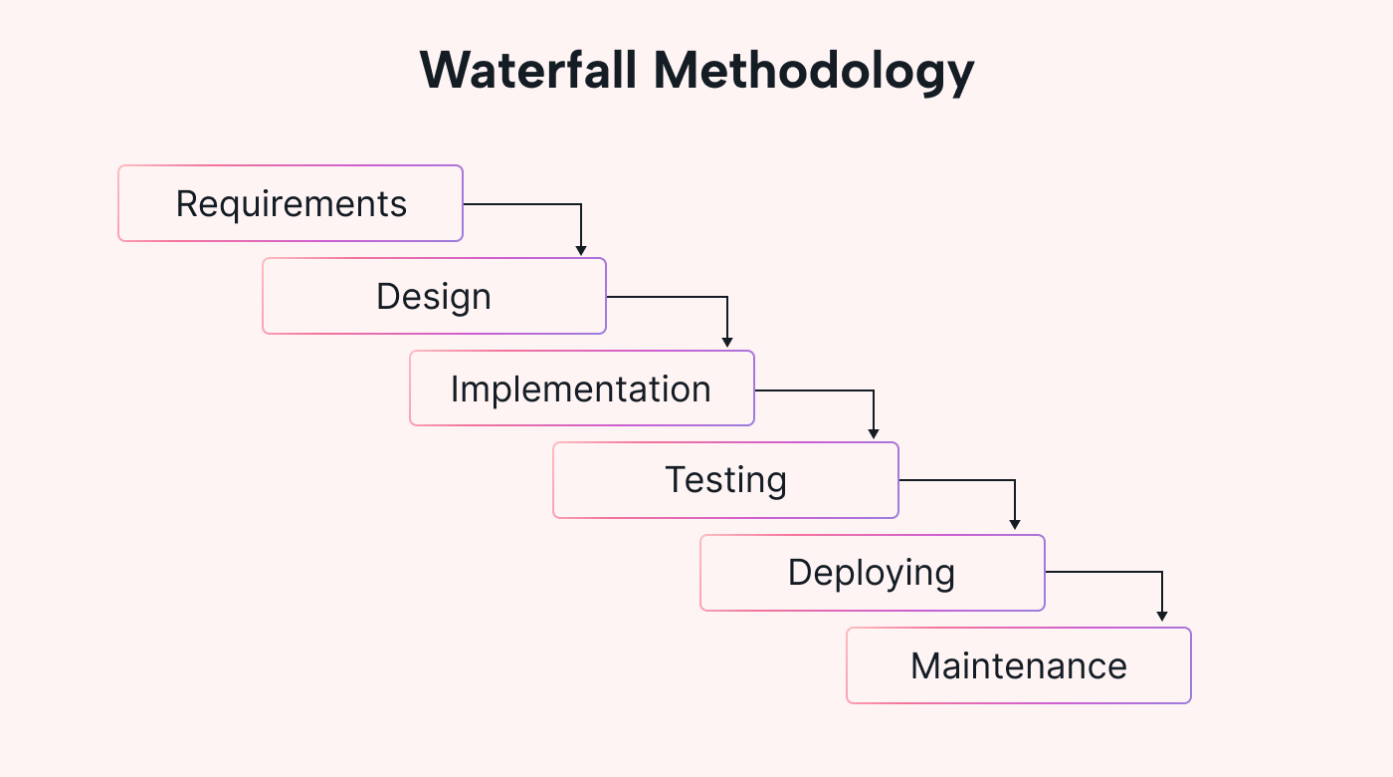
- 🔄 **Helps identify task order** (what comes first, next, last).
- ⏱️ **Improves time management** by preventing delays and overlaps.
- 🔍 **Highlights critical paths** – the sequence of key tasks that affect the project deadline.
- 👥 **Allocates resources better**, avoiding conflicts and overload.
- 📉 **Reduces project risk** by ensuring no activity is forgotten or started too early.

Here's a clear and simple explanation comparing **Waterfall (Predictive)** and **Agile (Adaptive)** project management life cycle methodologies:

✅

- **Linear and sequential**: Tasks are done **one after another**.
- Each phase (like planning, design, development, testing) must be **completed before moving on**.
- **Requirements are fixed** at the beginning.
- **Less flexible** – hard to make changes once development starts.

📝 **Example**:
Creating software for a **banking system**, where **security and compliance** require all requirements to be clearly defined and approved before starting.

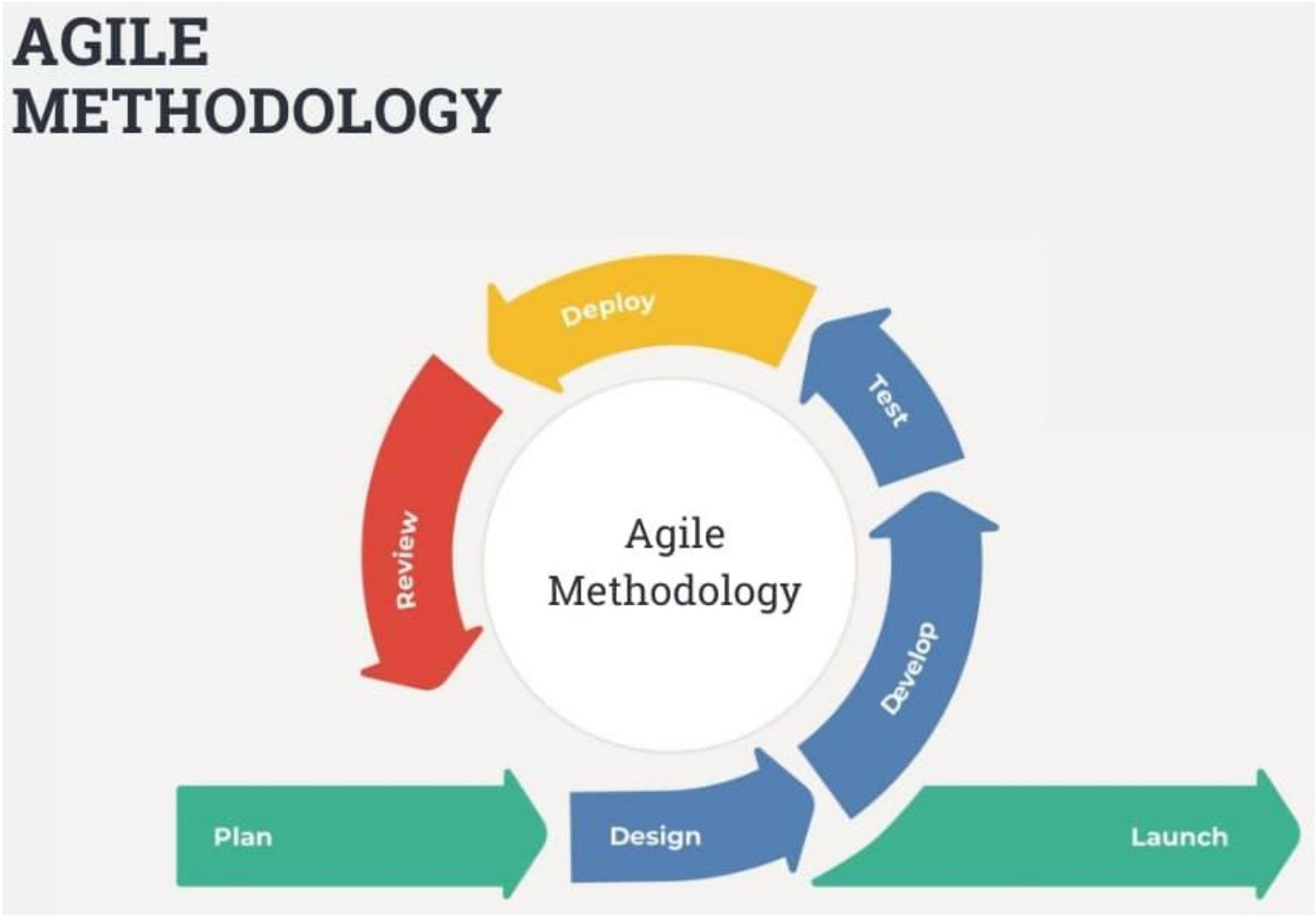


✅ **Agile (Adaptive) Methodology:**

- **Iterative and flexible**: Work is done in **short sprints or cycles**.
- Regular **client feedback** and **changes** are welcomed at any stage.
- **Working software** is delivered frequently.
- Focuses on **collaboration** and **continuous improvement**.

📝 **Example**:
Developing a **mobile app**, where the **client's needs evolve** with user feedback and market trends,

requiring frequent updates.



AGILE VS WATERFALL: A DETAILED COMPARISON

	WATERFALL	AGILE
FLEXIBILITY	No Flexibility	Much Flexibility
EASE OF SETTING AND KEEPING DEADLINES	Clear and Set Deadlines	Hard to Set Deadlines
TESTING	Only at the End	Testing in Every Iteration
PREDICTABILITY	Much Predictability	Poor Predictability
DOCUMENTATION	Significant Amount of Documentation	Little Documentation
QUALITY OF DEVELOPMENT	Average Quality	Better Quality of Development
CLIENT'S AND USER'S ENGAGEMENT	Low Level of Involvement	High Level of Involvement

CYBER CRAFT

LEARN MORE AT CYBERCRAFTINC.COM

✅ 3. What are the Competencies Required for a Project Manager?

A successful project manager needs a mix of **technical knowledge** and **soft skills** to lead the team, manage the process, and deliver results. Below are the key competencies:

💼 1. Leadership

- **Inspires, motivates, and guides** the team toward the project goals.
- Sets a **vision** and ensures team members are aligned and productive.

🔹 *Example:* Encouraging team collaboration to meet a tight deadline.

🗣️ 2. Communication

- Shares ideas, updates, and decisions clearly with the team and stakeholders.
- Uses **active listening**, regular meetings, and clear documentation.

🔹 *Example:* Explaining project progress to clients and resolving misunderstandings.

⏳ 3. Time Management

- Plans and manages schedules, ensuring tasks are completed on time.
- Uses tools like **Gantt charts or calendars** to keep the project on track.

🔹 *Example:* Setting weekly goals and checking task completion status.

⚠️ 4. Risk Management

- Identifies possible problems in advance and finds ways to reduce or avoid them.
- Creates **risk response plans** to handle issues if they arise.

🔹 *Example:* Preparing a backup plan if a key resource becomes unavailable.

💰 5. Budgeting & Cost Control

- Manages the project budget to avoid overspending.
- Tracks expenses and makes sure resources are used wisely.

🔹 *Example:* Choosing cost-effective tools or solutions to stay within budget.

🧠 6. Problem Solving

- Quickly understands and finds solutions to unexpected problems.
- Keeps the project moving despite challenges.

🔹 *Example:* Finding an alternative vendor when the original one fails.

🤝 7. Negotiation

- Resolves conflicts, negotiates with stakeholders, and secures resources.
- Balances the needs of different parties to keep everyone satisfied.

🔹 *Example:* Getting extra support from another team during high workload.

✅ 8. Decision Making

- Makes fast and effective choices based on available data and team input.
- Takes responsibility for the results of those decisions.

🔹 *Example:* Choosing between two vendors based on quality, cost, and deadline.

🖥️ **9. Technical Knowledge**

- Understands project tools, techniques, and the subject matter.
- Helps in making realistic plans and assessing team progress.

🔹 *Example:* Knowing how to use project management software like Jira or Trello.

Here is a **simple and clear explanation** of the **12 Principles of Agile (Agile Manifesto):**

✅ **12 Principles of Agile Methodology – Explained Briefly**

1. **Customer Satisfaction by Early and Continuous Delivery**
    - Deliver working software **early and often** to keep customers happy.

2. **Welcome Changing Requirements**
    - **Even late in the project**, changes are accepted to meet customer needs better.
3. **Deliver Working Software Frequently**
    - Release software **in short cycles** (like every 2–4 weeks) to get quick feedback.
4. **Business and Developers Must Work Together Daily**
    - **Close collaboration** between clients and the development team leads to better results.
5. **Build Projects Around Motivated Individuals**
    - Trust the team, give them support, and they will do great work.
6. **Face-to-Face Conversation Is Best**
    - **Direct communication** (in person or video) is clearer and faster than emails or documents.
7. **Working Software Is the Main Measure of Progress**
    - Focus on delivering **real, usable features**, not just plans or reports.
8. **Maintain Sustainable Development Pace**
    - Work at a pace that the team can keep up **consistently**, avoiding burnout.
9. **Continuous Attention to Technical Excellence**
    - Keep improving **code quality and design** for better performance and maintainability.
10. **Simplicity Is Essential**

- Do only what's needed—**avoid unnecessary work** or complexity.

11. **Self-Organizing Teams Deliver the Best Results**

- Let teams **manage themselves** and decide how best to get the work done.

12. **Regularly Reflect and Improve**

- At regular intervals, **review what's working and what's not**, then make adjustments.

✅ **4 Values of Agile**

1. **Individuals and interactions over processes and tools**
    - People and teamwork are more important than strict rules or tools.
    - 👉 *A good team can solve problems better than just relying on software or rules.*

---

2. **Working software over comprehensive documentation**
    - It's better to have a **working product** than lots of written plans or reports.
    - 👉 *Customers care about results, not paperwork.*

---

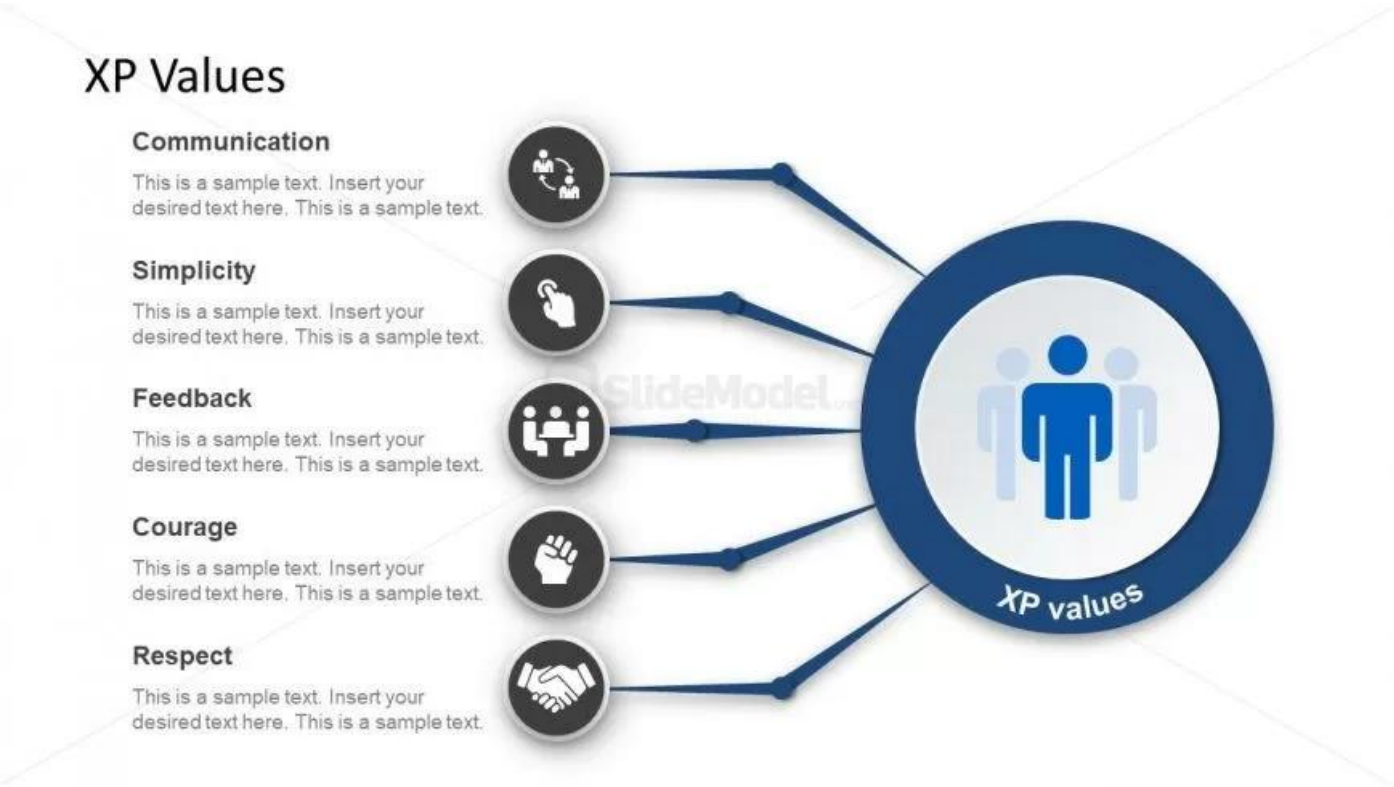
3. **Customer collaboration over contract negotiation**
    - Work closely with the customer instead of just sticking to a signed contract.
    - 👉 *Better communication = better product.*

---

4. **Responding to change over following a plan**
    - Be **flexible** and ready to adjust as things change, rather than blindly following the original plan.
    - 👉 *Change is normal—Agile embraces it.*

✅ **Extreme Programming (XP): Main 5 Elements Explained**

Extreme Programming (XP) is an Agile software development method that emphasizes **rapid development**, **continuous feedback**, and **close teamwork**. It helps teams build high-quality software efficiently.



1. Communication

- Team members **constantly share ideas, issues, and solutions**.
- Everyone (developers, testers, customers) works together **closely**.
- Frequent stand-up meetings and **pair programming** are used to boost collaboration.
- 👉 *Good communication helps avoid misunderstandings and keeps the project on track.*

---

2. Simplicity

- Developers **build only what is needed now**, not extra features for the future.
- Code is kept **simple and clean** to make it easy to understand and maintain.
- 👉 *Simple code means fewer bugs and faster updates.*

---

3. Feedback

- XP relies on **continuous feedback** from testing and from customers.
- Tests are written **before** the code (Test-Driven Development - TDD).
- Frequent releases help customers **see and review progress**.
- 👉 *Early feedback allows quick fixes and better satisfaction.*

---

4. Courage

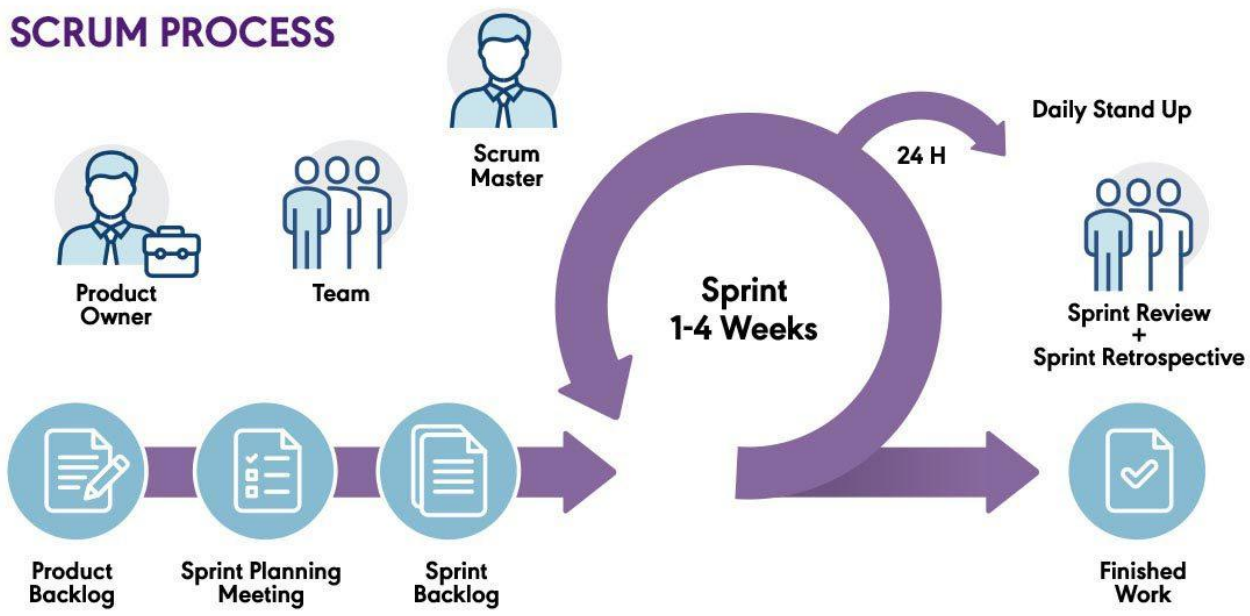
- Developers should have the **confidence to make changes** when needed.
- Be brave to **refactor code**, suggest improvements, or even discard bad ideas.
- 👉 *Courage leads to better decisions and cleaner software.*

---

5. Respect

- All team members **respect each other's work and ideas**.
- Everyone contributes equally and supports one another.
- 👉 *Respect builds trust and leads to a stronger, happier team.*

Here's a **detailed and easy-to-understand explanation** of the main **Agile/Scrum events** with examples:



✅ Agile / Scrum Events Explained

---

a. Sprints

- A **Sprint** is a fixed time period (usually **1 to 4 weeks**) during which the team works to complete specific tasks and deliver a **working product**.
- Each Sprint should result in a **usable piece of software**.

Example:

- Sprint 1: Team builds and tests a **login feature** for the app.
- Sprint 2: They add a **user profile page**.

---

b. Sprint Planning Meeting

- Held **before the Sprint begins**.
- The team, Scrum Master, and Product Owner **plan what work** will be done.
- Tasks are selected from the **Product Backlog** (the full list of desired features).

Example:

- The team chooses to work on the **shopping cart feature** in Sprint 3.

---

c. Daily Scrum (Stand-up)

- A **short (15-minute) daily meeting** where team members quickly share:
    1. What they did **yesterday**
    2. What they plan to do **today**
    3. Any **blockers** or problems

Example:

- A developer says, “Yesterday I fixed the login bug. Today I’ll start on the profile page. I’m waiting for the API from the backend team.”

---

d. Sprint Review

- Happens **at the end of a Sprint**.
- The team **shows** what they built to stakeholders (e.g., clients or product owner).
- Stakeholders give **feedback**, which may lead to new items in the backlog.

Example:

- The team demos the checkout system to the client. The client asks for a new payment method to be added next Sprint.

---

e. Sprint Retrospective

- Held **after the Sprint Review**.
- The team reflects on the Sprint:
    - What went well?
    - What didn't go well?
    - How can we improve?

Example:

- The team agrees to improve **communication** and reduce **last-minute testing** delays in the next Sprint.

---

f. Scrum Artifacts

1. Product Backlog

- A full list of **all desired features**, created by the **Product Owner**.
- Example: "User registration," "Search bar," "Dark mode."

2. Sprint Backlog

- A **subset of tasks** taken from the Product Backlog for the **current Sprint**.
- Example: "Design homepage layout," "Code login page," "Test user flow."

3. Increment

- The **working software** delivered at the **end of a Sprint**.
- Example: A functional login + sign-up page ready for users.

🧠 **Summary:**

Event	Purpose	Example
Sprint	Develop working software in a fixed time	Build login feature in 2 weeks
Sprint Planning	Choose tasks to complete in Sprint	Pick user profile tasks
Daily Scrum	Quick status update	Share progress & blockers
Sprint Review	Demo completed work	Show checkout to stakeholders

Event	Purpose	Example
Retrospective	Team reflection and improvement	Decide to improve testing early
		Backlogs + Delivered Product
Artifacts	Track all work and progress	

Here is a **simple explanation** of the **trends and emerging practices in project risk management** with clear points:

✅ **Trends and Emerging Practices in Project Risk Management**

As project environments become more complex, traditional risk management is evolving. The following are **new trends and practices** being used today:

---

a. Non-Event Risks

- These are risks that **do not have a single, identifiable event** that triggers them.
- They happen **gradually over time**.

- Example:
    - **Inflation** slowly increasing costs
    - **Market condition changes** reducing demand for your product

🔹 *Why it's important*: These risks are harder to notice but can have a big long-term impact.

---

b. Variability Risk

- Refers to **uncertainty in estimates**, such as:
    - **How long a task will take**
    - **How much it will cost**
    - **How complex it will be**
- Example:
    - Task A might take **3 to 5 days**, depending on the developer's experience.

🔹 *Why it's important*: Helps project managers prepare for a range of outcomes instead of one fixed value.

---

c. Ambiguity Risk

- Happens when things are **unclear, incomplete, or confusing**.
- Often caused by:
    - **Vague requirements**
    - **Uncertain technologies or markets**
- Example:
    - A client says, "Make it user-friendly" — but doesn't define what that means.

🔹 *Why it's important*: Can lead to misunderstandings, rework, and delays if not clarified early.

---

d. Project Resilience

- Means the project has the **ability to absorb shocks and recover quickly**.
- Includes:
    - Flexible planning
    - Backup resources
    - Strong communication
- Example:

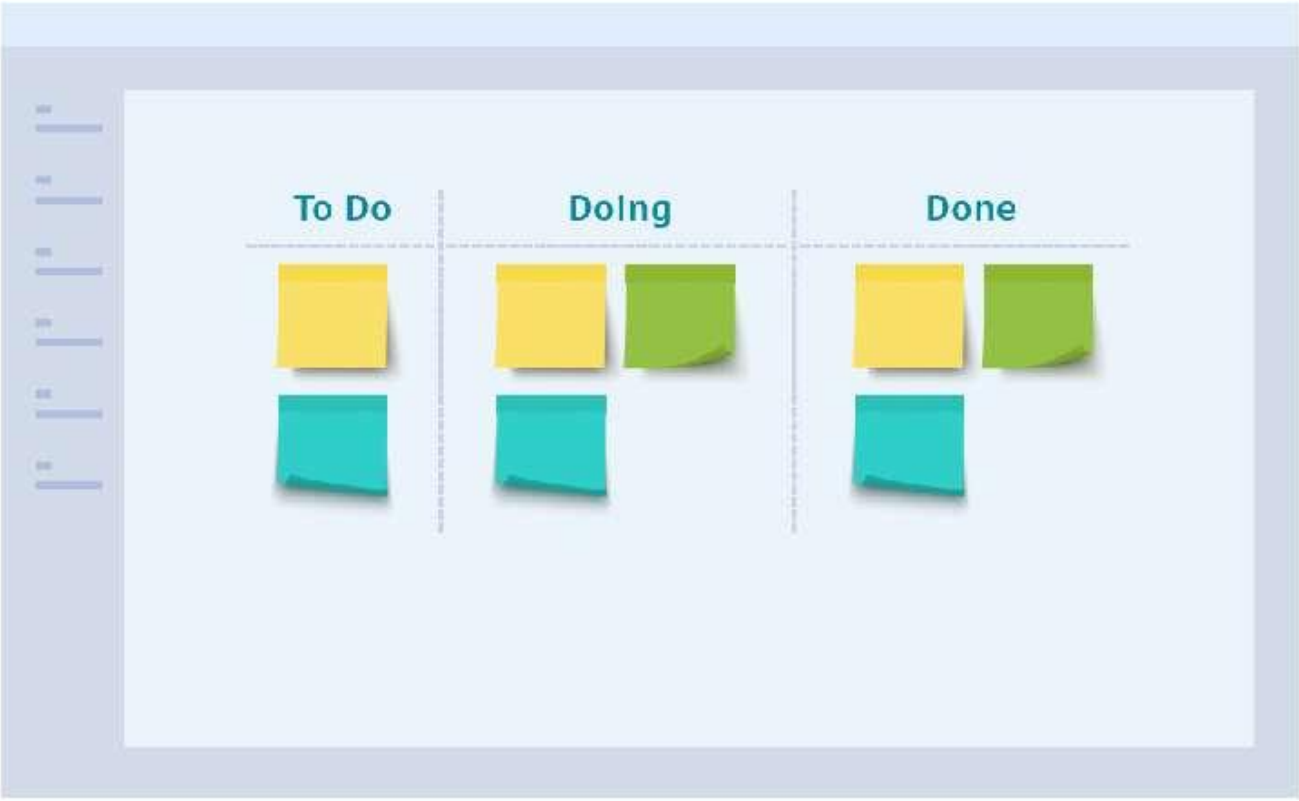
- A team member suddenly leaves, but the team quickly adjusts roles to stay on track.

🔹 *Why it's important*: In today's fast-changing environment, resilience helps projects continue smoothly despite unexpected issues.

🧠 **Summary Table**

Risk Trend	Meaning	Example
Non-Event Risk	Risk without a clear trigger	Inflation slowly raising costs
Variability Risk	Uncertainty in time, cost, or scope estimates	Task may take 3–5 days
Ambiguity Risk	Confusing or unclear requirements	Vague user needs or expectations
Project Resilience	Project's ability to recover from unknown issues	Team adapts to unexpected staff changes

Sure! Below is a **brief explanation of common Agile templates** along with simple diagrams to help you understand how they work.



✅ 9. Agile Templates Explained (with Diagrams)

Agile methodology uses visual tools to help teams plan, track, and deliver work efficiently. Here are four popular Agile templates:

---

1. Product Backlog

- A **to-do list** of all features, enhancements, and fixes for the product.
- Owned by the **Product Owner**.
- Items are prioritized based on business value.

📝 **Example:**

Priority	User Story	Status
High	As a user, I want to sign up via email	To Do
Medium	As a user, I want to reset my password	In Progress
Low	As admin, I want to view usage stats	Done

---

2. Sprint Backlog

- A **subset of the product backlog** selected for the current sprint.
- Managed by the development team.
- Includes tasks broken down from user stories.

📋 **Example:**

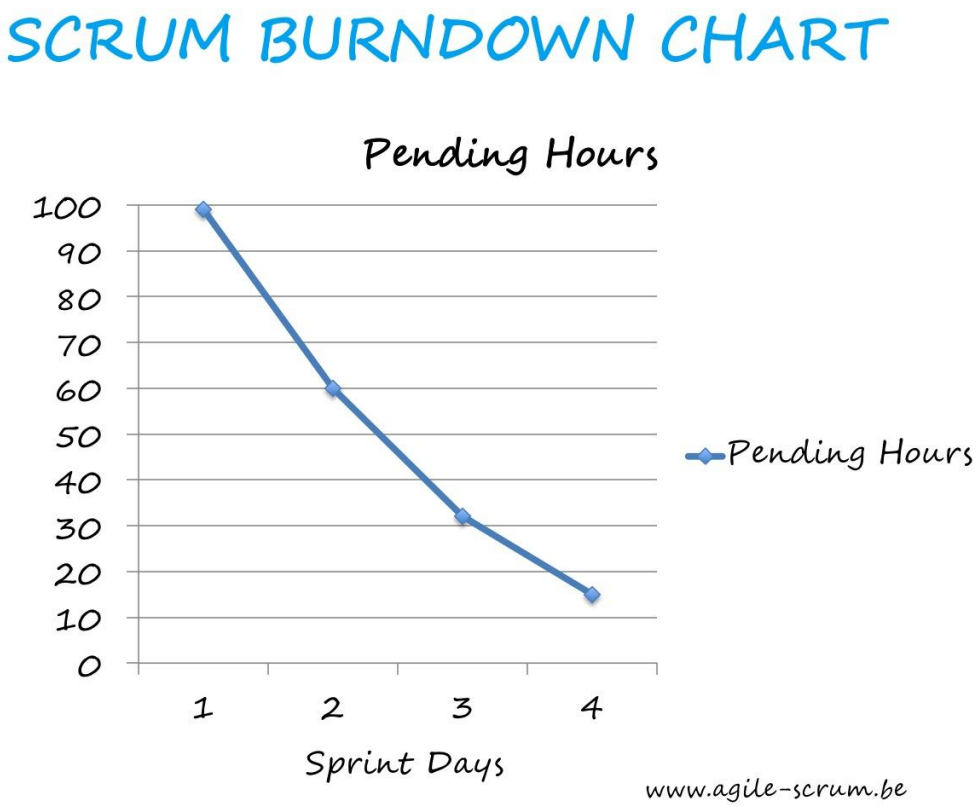
Task	Assigned To	Status
Design login page	UI Team	Done
Implement email signup backend	Dev Team	In Progress
Write test cases	QA Team	To Do

---

3. Burndown Chart

- A line chart that shows the **amount of work remaining** in a sprint.
- Helps track if the team is on pace to complete tasks.

📉 **Diagram:**



---

4. Kanban Board

- A visual board divided into columns: **To Do**, **In Progress**, and **Done**.
- Helps teams see the status of each task at a glance.

🧩 **Diagram:**

🧠 Summary Table

Template	Purpose	Managed By
Product Backlog	Master list of all work	Product Owner
Sprint Backlog	Tasks for the current sprint	Development Team
Burndown Chart	Track sprint progress	Scrum Master/Team
Kanban Board	Visual task tracking	Whole Team