DEVELOPMENT FUNDAMENTALS

SDLC, AGILE, GIT (VERSION CONTROL)

SDLC

Software Development Life Cycle

SDLC is a systematic process used for planning, creating, testing, deploying,

It defines a series of steps or phases that guide software development from the initial idea to its final implementation and maintenance

ensures high-quality software that meets customer expectations.

KEY PHASES OF SDLC

- Planning and Requirement Analysis
- System Design
- Implementation (Coding/Development)
- Testing
- Deployment
- Maintenance and Support

REQUIREMENT ANALYSIS

- Planning and Requirement Analysis.
- What to do here?
  - Understand the business requirements
  - define the project's scope.
- How to do?
  - Gather and analyze requirements from stakeholders.
  - Conduct feasibility studies (technical, economic, operational).
  - Prepare project plans and resource allocation
- Result:
  - Business Requirement Document - BRD

SYSTEM DESIGN

- What to do here?
  - Transform requirements into system architecture and design
- How to do?
  - Design system architecture, database, and application components.
  - Create wireframes and user interface design.
  - Define data flow diagrams and ER diagrams.
- Result:
  - System Design Document
    - High-Level Design – HLD
    - Low-Level Design - LLD

IMPLEMENTATION

- **Implementation (Coding/Development)**
- What to do here?
  - Convert design into executable code.
- How to do?
  - Write code in the chosen programming language.
  - Follow coding standards and development guidelines.
  - Conduct unit testing by developers.
- Result:
  - Working source code

TESTING

- What to do here?
  - Ensure that the software is bug-free
  - Meeting all requirements.
- How to do?
  - Conduct different types of testing (unit, integration, system, regression, user acceptance testing).
  - Log and fix bugs..
- Result:
  - Test Report and Bug Resolution Document

DEPLOYMENT

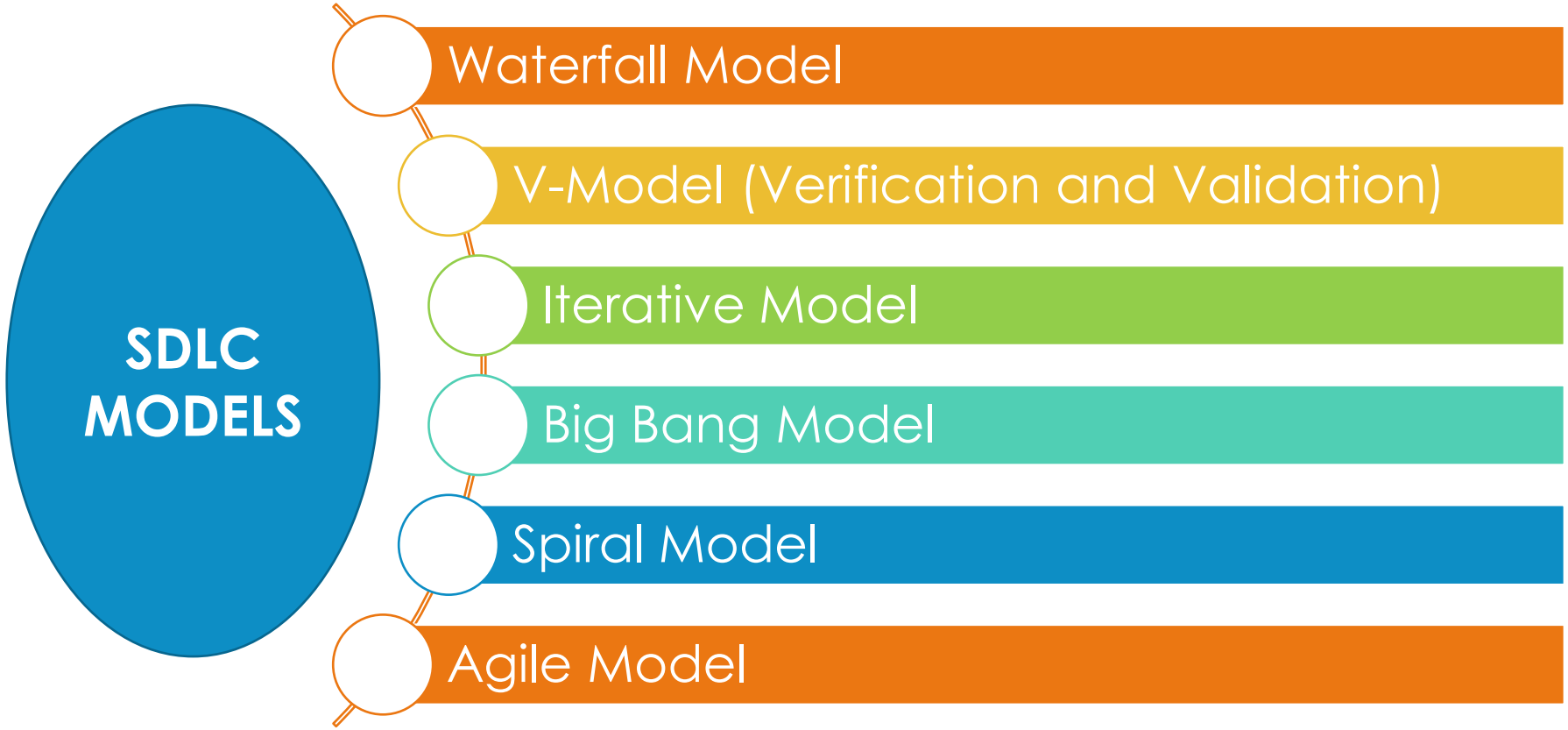
- What to do here?
  - Release the software to the production environment.
- How to do?
  - Deploy the system to staging and production environments.
  - Perform final system tests in production.
  - Roll out updates or patches if needed.
- Result:
  - Deployed software system

MAINTENANCE AND SUPPORT

- What to do here?
  - Ensure smooth software operation and handle updates or bug fixes.
- How to do?
  - Monitor system performance.
  - Provide technical support and bug resolution.
  - Implement enhancements or updates.
- Result:
  - Maintenance Logs and Update Patches

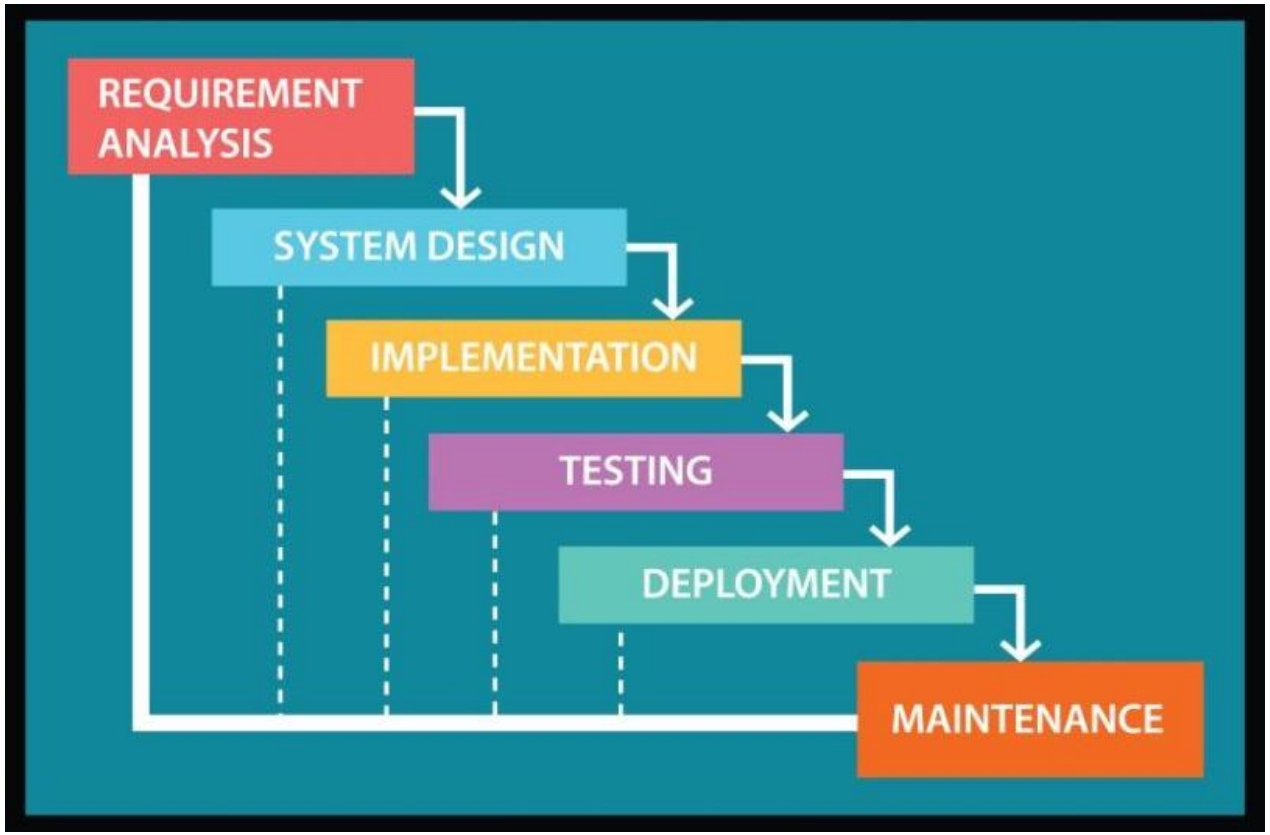
BENEFITS OF SDLC

SDLC MODELS


SDLC MODELS
Waterfall Model
V-Model (Verification and Validation)
Iterative Model
Big Bang Model
Spiral Model
Agile Model

WATERFALL MODEL

- The **Waterfall Model** is a linear and sequential software development approach.
- Each phase must be completed before moving to the next, and there is no overlap between phases.



Advantages:

- Flexible and adaptive to changing requirements.
- Continuous delivery of working software.
- Encourages stakeholder collaboration and feedback.

Disadvantages:

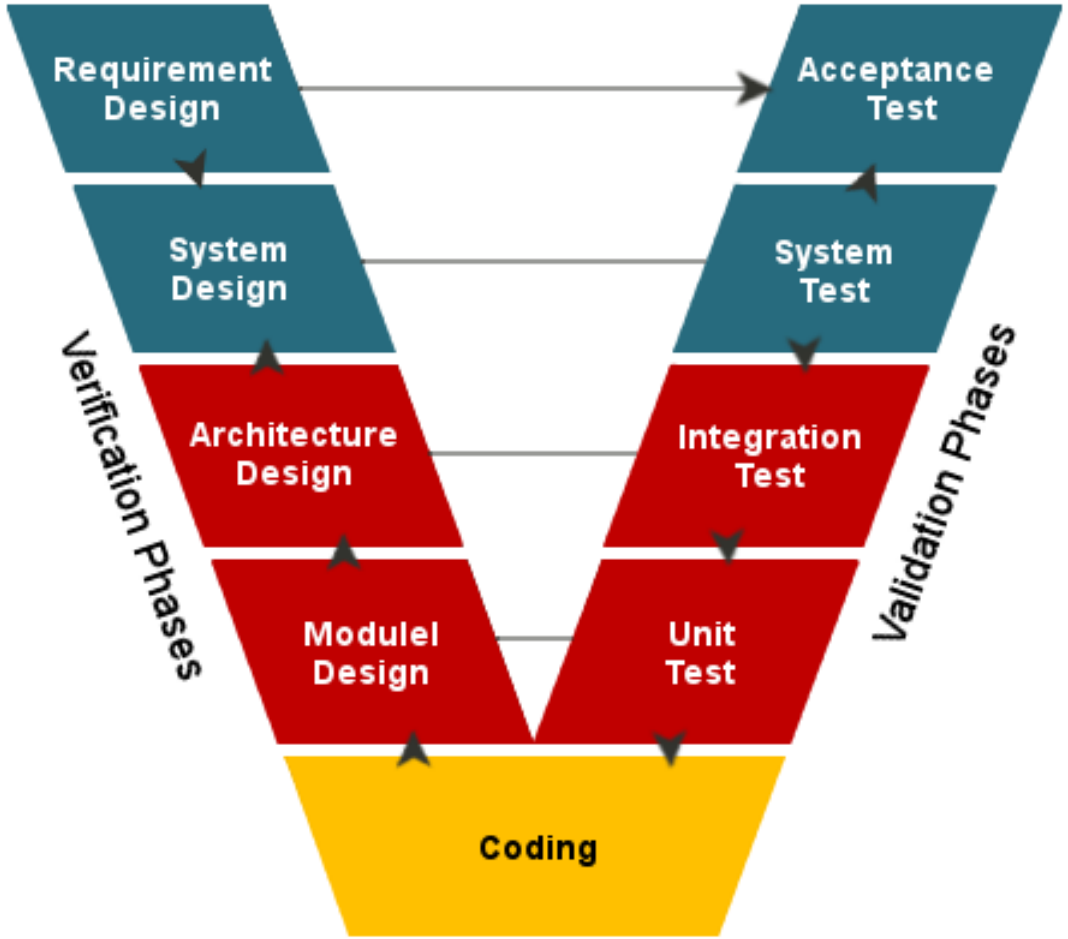
- Requires active client involvement.
- Difficult to predict the final cost and timeline.
- May lead to scope creep if not well-managed.

Best Use Cases:

- Projects with dynamic requirements.
- Startups and software product development.
- Web and mobile applications.

V-MODEL

- The **V-Model** is an extension of the Waterfall Model that integrates testing at every development phase.
- It emphasizes validation (building the right product)
- verification (building the product right).



Advantages:

- Early detection of defects due to testing at every phase.
- Easy to manage due to clearly defined processes.
- High quality and reliability of the final product.

Disadvantages:

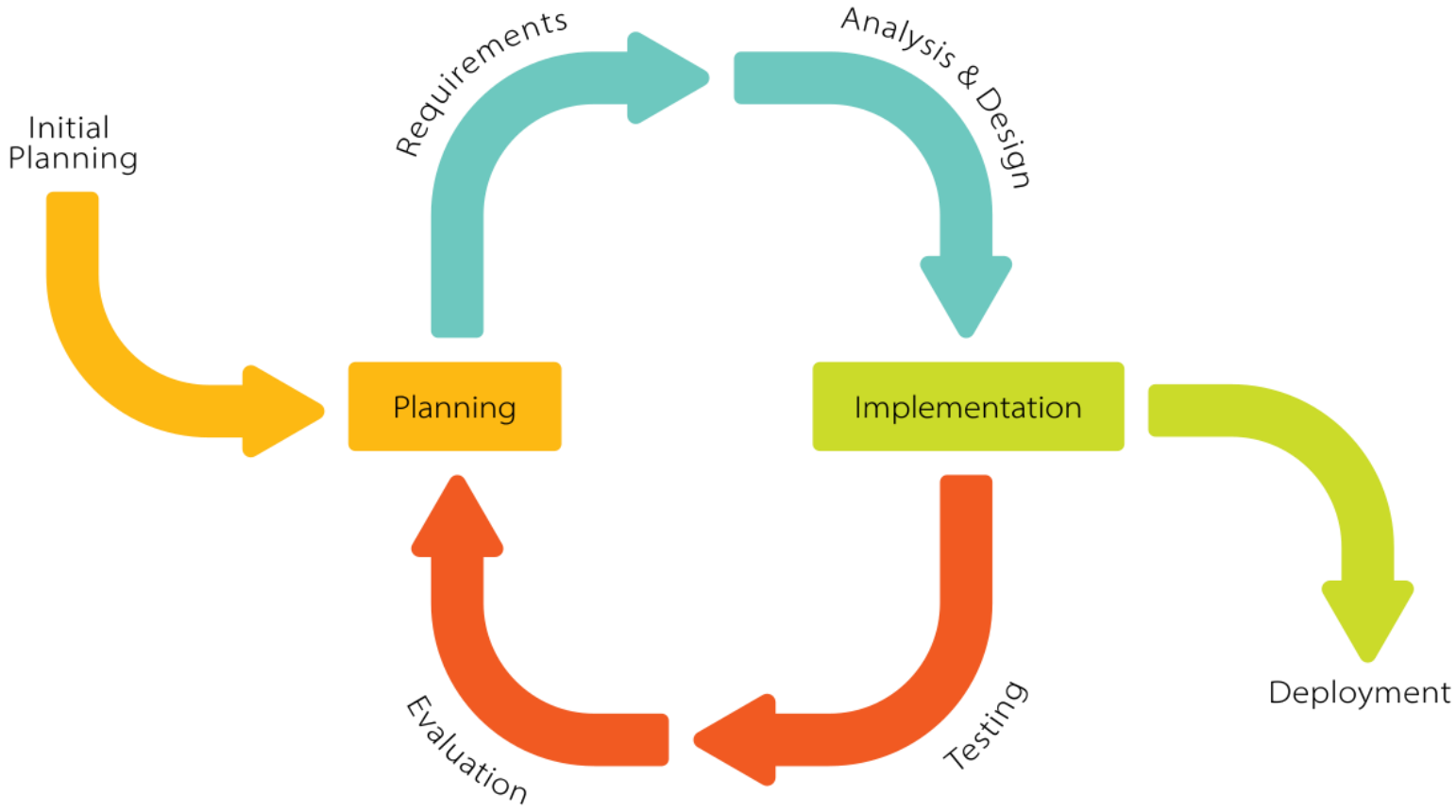
- Rigid and inflexible model.
- High costs due to extensive testing.
- Not suitable for dynamic or complex projects.

Best Use Cases:

- Embedded systems and safety-critical software.
- Projects with stable requirements.

ITERATIVE MODEL

- The **Iterative Model** focuses on building the core functionalities first
- and enhancing them through repeated cycles (iterations) until the complete product is delivered.



Advantages:

- Early delivery of partial solutions.
- Easy to identify and fix issues in early stages.
- More user involvement during development.

Disadvantages:

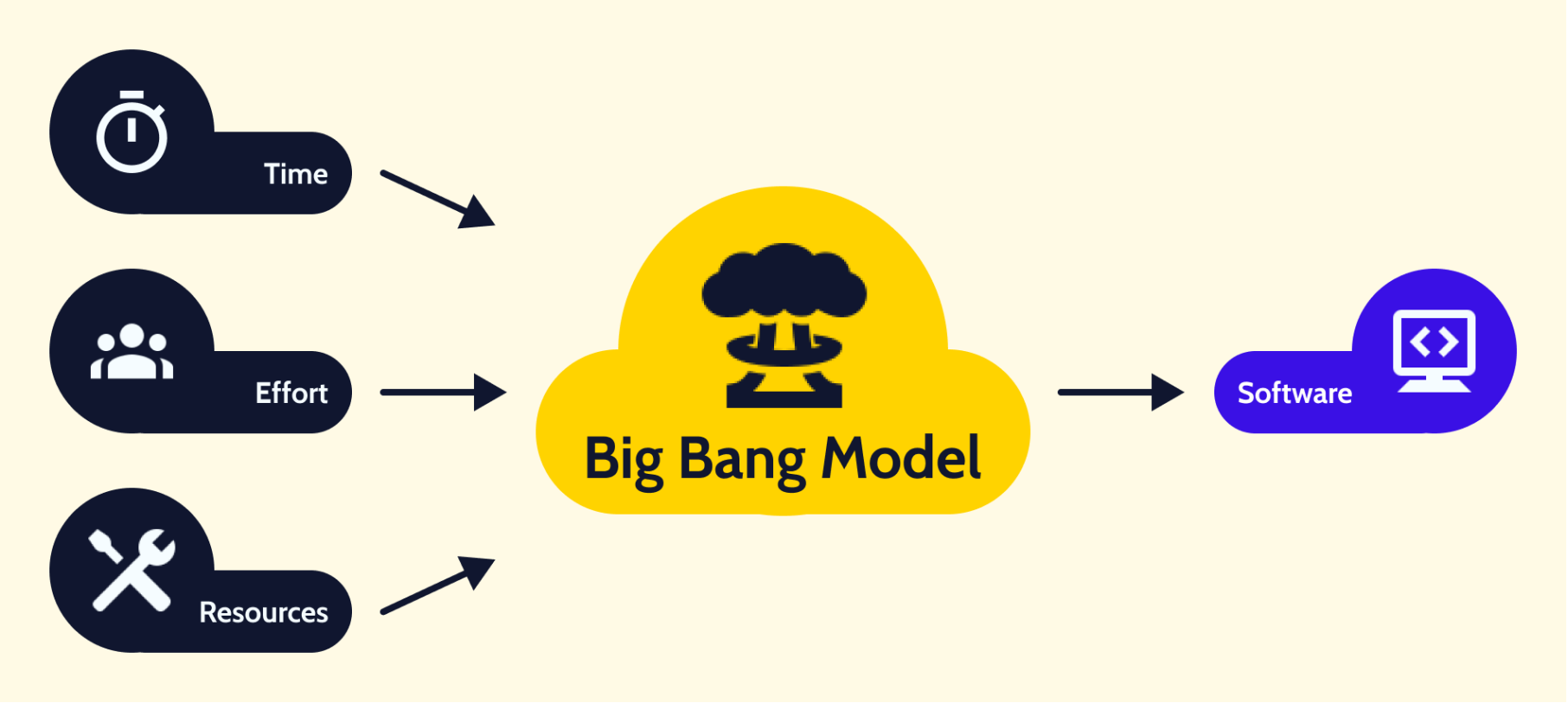
- Requires more resources.
- Poor planning can lead to scope creep.
- System architecture must be well-defined early on.

Best Use Cases:

- Large projects where early feedback is essential.
- Projects involving evolving technologies.

BIG BANG MODEL

- The **Big Bang Model** is an unstructured approach where development begins without much planning or analysis.
- Developers simply code and fix issues as they arise.



Advantages:

- Simple and requires minimal planning.
- Suitable for small projects with limited requirements.

Disadvantages:

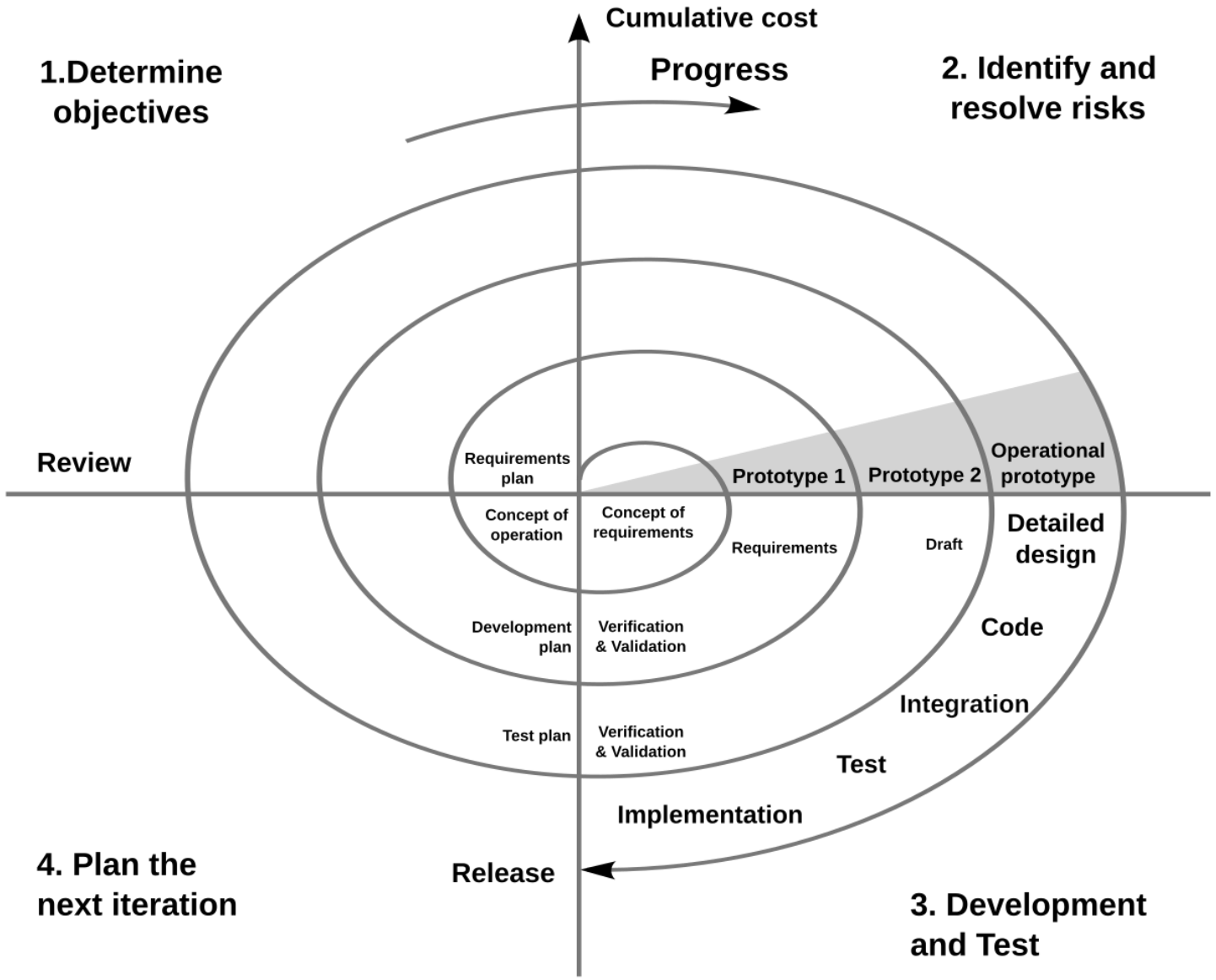
- High risk and uncertainty.
- No proper documentation.
- Difficult to manage and scale.

Best Use Cases:

- Small projects with one or two developers.
- Experimental or proof-of-concept projects.

SPIRAL MODEL

- The **Spiral Model** combines iterative development with risk assessment.
- It involves multiple cycles (spirals), with each cycle involving planning, risk evaluation, development, and validation.



Advantages:

- Effective risk management.
- Flexibility in accommodating changes.
- Continuous customer feedback.

Disadvantages:

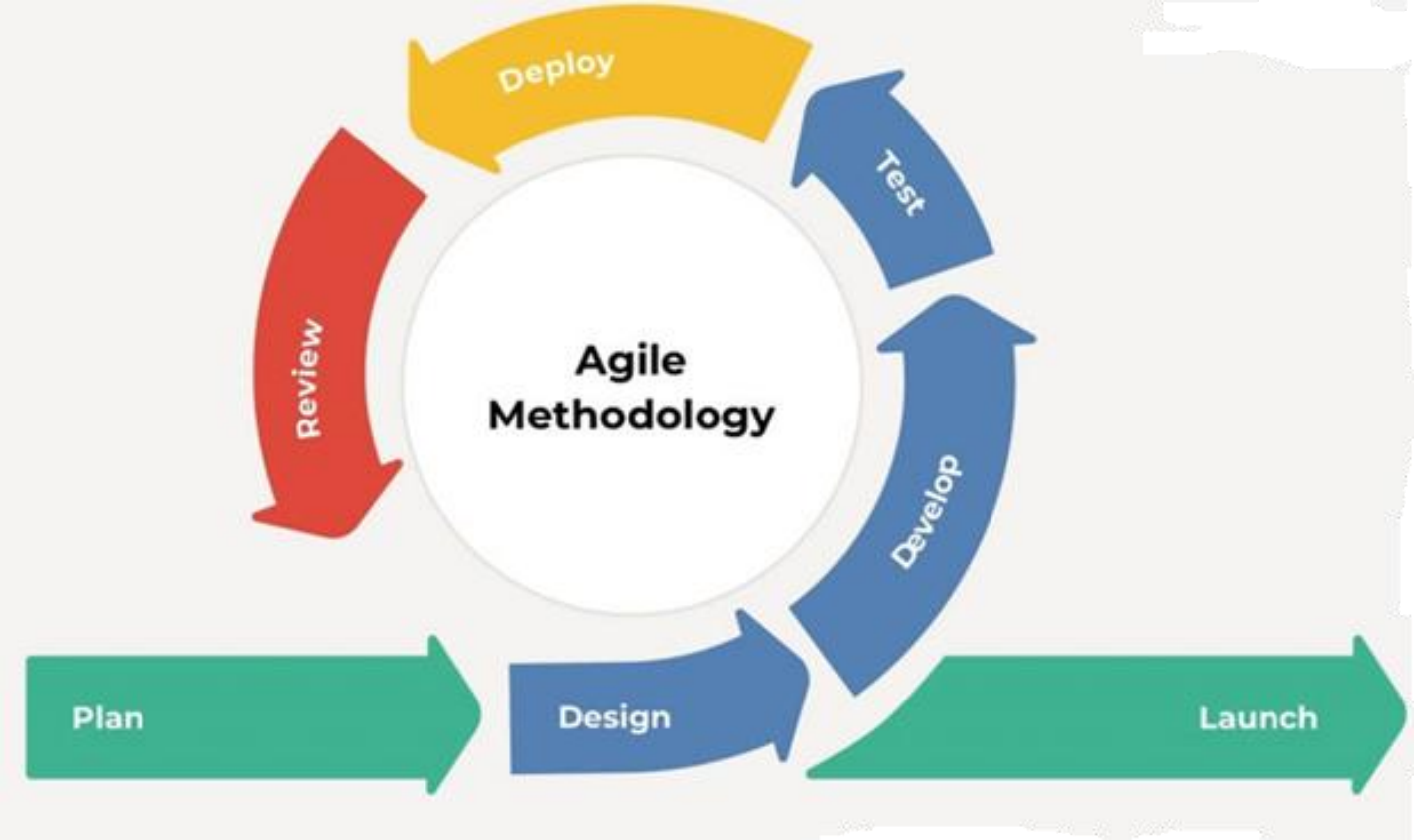
- Complex and costly to implement.
- Requires skilled risk assessment professionals.
- Not suitable for small projects.

Best Use Cases:

- Large and high-risk projects.
- Projects requiring frequent evaluations.

AGILE MODEL

- The **Agile Model** focuses on iterative and incremental development with constant feedback from stakeholders.
- It breaks down the project into small iterations (sprints).



Advantages:

- Flexible and adaptive to changing requirements.
- Continuous delivery of working software.
- Encourages stakeholder collaboration and feedback.

Disadvantages:

- Requires active client involvement.
- Difficult to predict the final cost and timeline.
- May lead to scope creep if not well-managed.

Best Use Cases:

- Projects with dynamic requirements.
- Startups and software product development.
- Web and mobile applications.

ACTIVITY

Which of the following is a major drawback of the Waterfall model?

A) High flexibility in requirement changes

B) Sequential execution of phases

C) Easy testing and debugging in early phases

D) Allows parallel development of multiple components

ACTIVITY

The Spiral model is best suited for which type of projects?

A) Small, well-defined projects

B) Projects with fixed requirements

C) Projects with high risk and evolving requirements 

D) Projects requiring minimal documentation

ACTIVITY

What is the primary advantage of the Iterative model?

A) Reduced time for final product delivery

B) Requirements must be fully defined upfront

C) Incremental improvements after each iteration 

D) No feedback loop during development

ACTIVITY

In the V-Model, the corresponding phase to the coding phase on the testing side is:

A) Unit Testing 

B) System Testing

C) Integration Testing

D) Acceptance Testing

ACTIVITY

Which of the following is a characteristic of the Big Bang model?

A) Heavy planning and documentation

B) High customer involvement throughout development

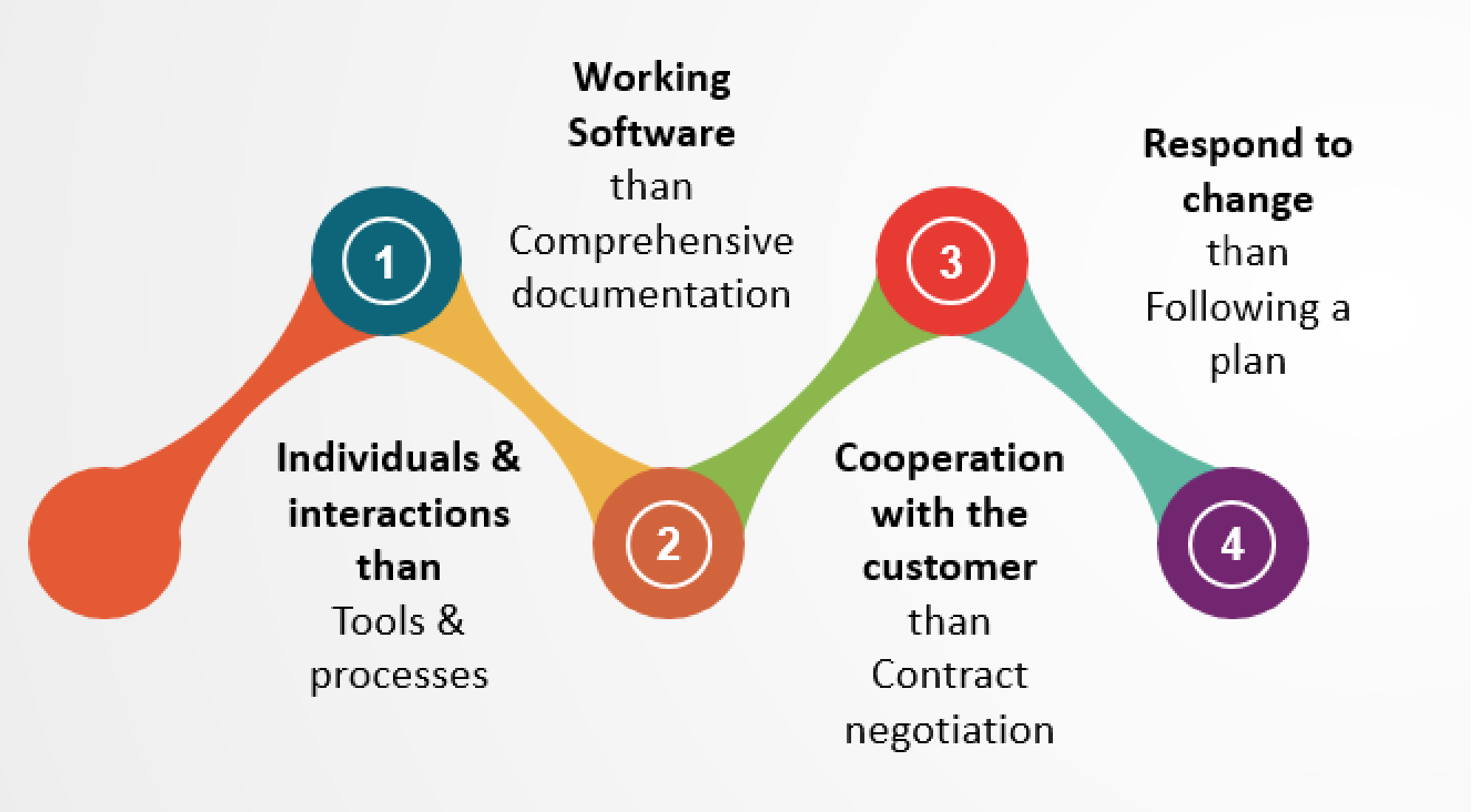
C) No defined process or structure 

D) Incremental product releases

AGILE MANIFESTO

- Agile is a **project management methodology** that emphasizes
- **iterative development**
- **continuous feedback**
- **collaboration** between cross-functional teams to deliver high-quality software.

AGILE 4 VALUES



KEY PRINCIPLES

- **Customer Collaboration:**
  - Continuous involvement and feedback from customers.
- **Iterative Development:**
  - Frequent delivery of working software in smaller increments.
- **Value-driven Delivery:**
  - Focus on delivering features that offer maximum value first.
- **Adaptability:**
  - Responding to changing requirements throughout the project.
- **Continuous Improvement:**
  - Regular reflection on processes to make improvements.

AGILE FRAMEWORKS

SCRUM: Time-boxed sprints with defined roles and ceremonies.

Kanban: Visual board for task management with continuous workflow.

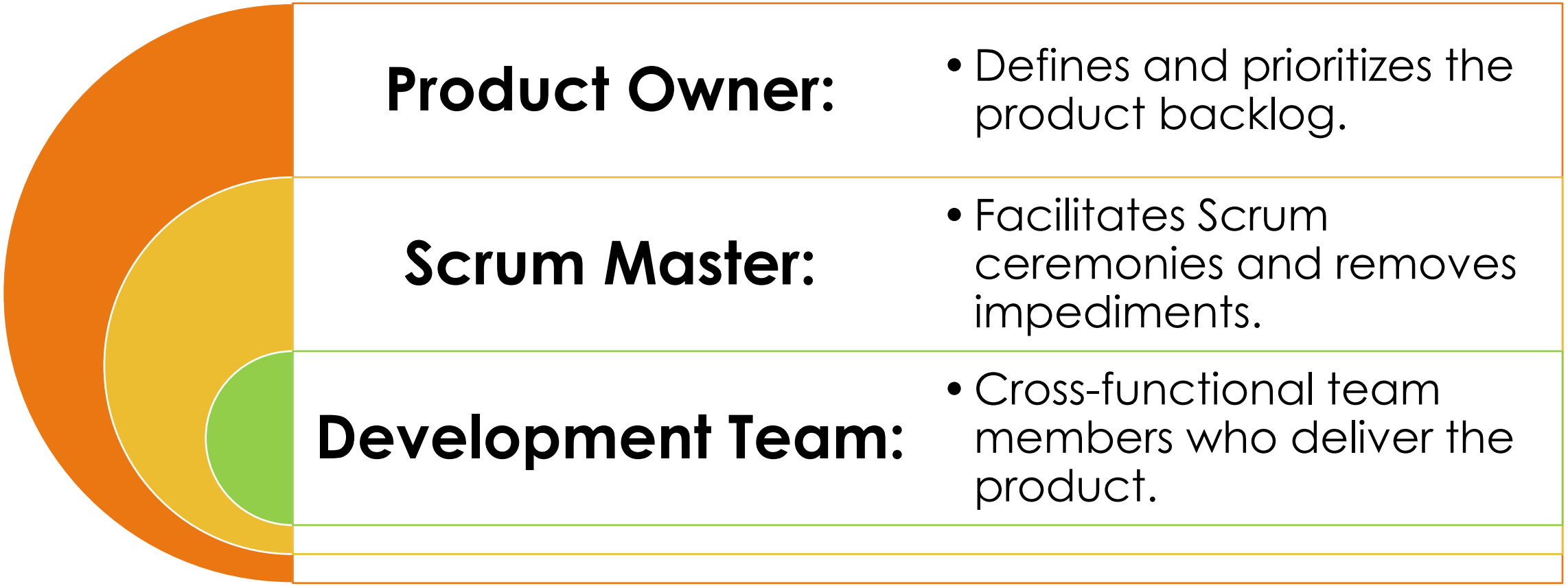
Extreme Programming (XP): Focus on frequent releases with high customer involvement.

Agile practices for large entreprises.

WHAT IS SCRUM?

- **Scrum** is an Agile framework used to develop, deliver, and sustain complex products.
- It emphasizes team collaboration, iterative development, and time-boxed iterations called **sprints**.

SCRUM ROLES


Product Owner:
• Defines and prioritizes the product backlog.
Scrum Master:
• Facilitates Scrum ceremonies and removes impediments.
Development Team:
• Cross-functional team members who deliver the product.

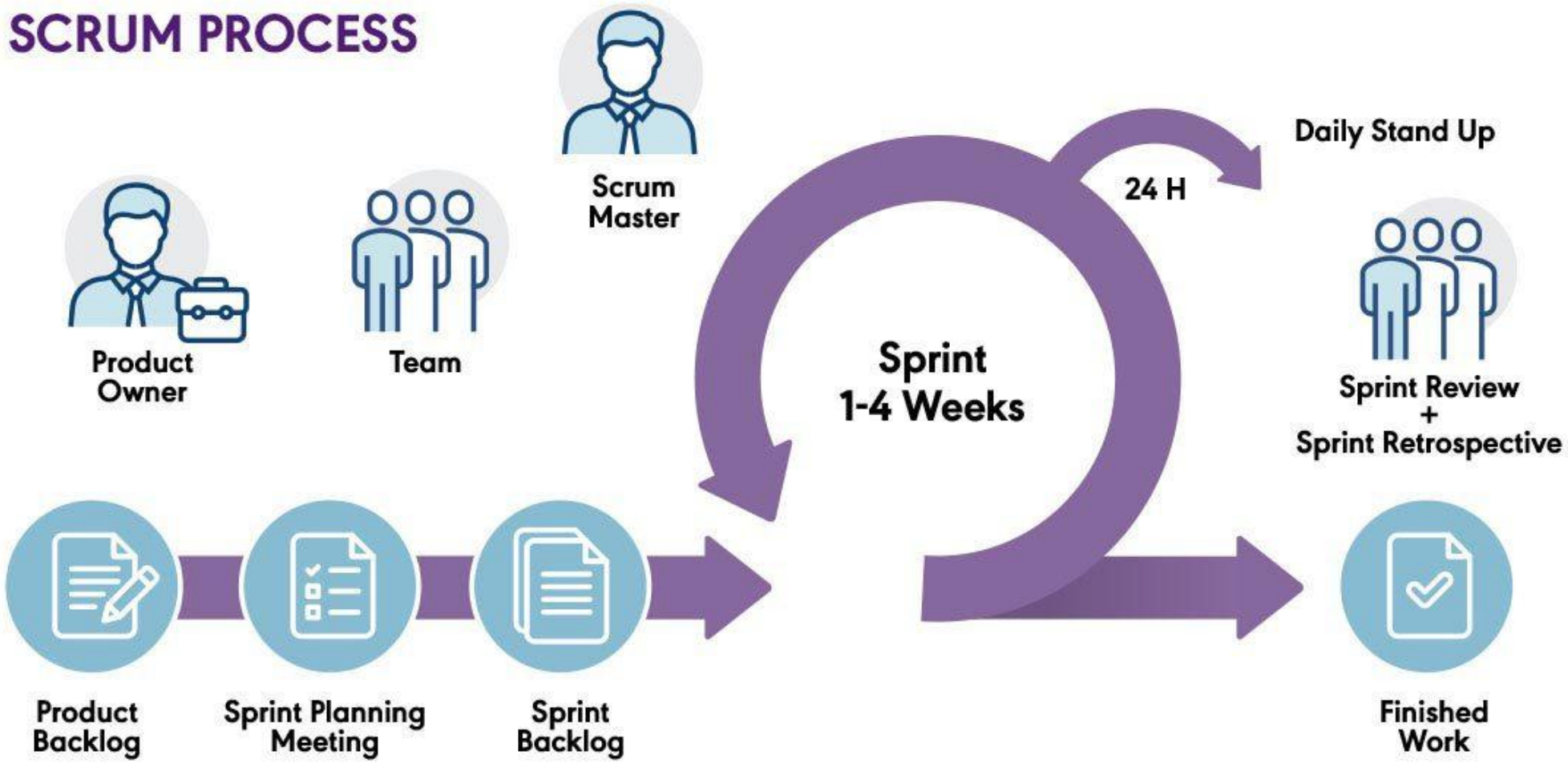
SCRUM CEREMONIES

Sprint Planning:	• Decide the sprint goal and tasks.
Daily Stand-Up (Daily Scrum):	• 15-minute meeting to discuss progress and obstacles.
Sprint Review:	• Present the completed work to stakeholders.
Sprint Retrospective:	• Reflect on the sprint and identify improvements.

ARTIFACTS

Product Backlog:	List of features, requirements, and tasks.
Sprint Backlog:	Subset of the product backlog planned for the sprint.
Increment:	A potentially shippable product at the end of each sprint.

SCRUM


SCRUM PROCESS
Scrum Master
Product Owner
Team
Daily Stand Up
24 H
Sprint 1-4 Weeks
Sprint Review + Sprint Retrospective
Product Backlog
Sprint Planning Meeting
Sprint Backlog
Finished Work

REAL TIME EXAMPLE

- **Scenario:** Developing an e-commerce website for a client.
- **Sprint 1:** Develop user authentication and home page UI.
- **Sprint 2:** Implement product search, category filters, and cart functionality.
- **Sprint 3:** Develop payment gateway integration and order history.

LET'S DO SPRINT1

- Sprint 1: User Authentication and Home Page UI (2 Weeks)
- **Tasks**
- **User Authentication:**
  - Set up backend authentication using JWT (or session-based login) for user registration and login.
  - Create APIs for user registration, login, and logout.
  - Implement password encryption using bcrypt.
- **Frontend Authentication Integration:**
  - Create a registration and login UI with form validation using React or Angular.
  - Implement conditional rendering based on authentication state.
- **Home Page UI:**
  - Design and implement the homepage layout, including the product showcase section and navigation bar.
  - Add a promotional banner component and category navigation section.

- **Development**:
  - **Backend Setup:**
  - Initialize the backend project with Node.js and Express.
  - Set up user schema in MongoDB or a relational database like MySQL.
  - **Frontend Setup:**
  - Create the React or Angular project.
  - Use Bootstrap or Material UI for UI components.
  - **API Integration:**
  - Connect the frontend to the backend for authentication features.
  - Implement route guards for protected pages.
- **Testing**
  - Test user registration, login, and authentication token flow.
  - Verify form validation and error handling.

SPRINT REVIEW: DEMONSTRATE CART AND PAYMENT FEATURES (1 WEEK)

- **Preparation**
  - Deploy the latest application build on a staging server.
  - Prepare a presentation and walkthrough of cart and payment functionalities.
  - Highlight key features and improvements from the previous sprint.
- **Demonstration**
  - Walk the client through the following scenarios:
    - Product search and category filters.
    - Adding and managing cart items.
    - Completing a payment using the payment gateway.
    - Viewing the order confirmation and order history.
- **Client Feedback**
  - Document client suggestions and pain points.
  - Identify priority tasks for the next sprint based on feedback.
- **Task Adjustments**
  - Create a backlog with updated features and bug fixes.
  - Prioritize tasks for future sprints based on client requirements.

ACTIVITY

- Scenario: Developing a Student Management Portal
- A college administration has approached your team to develop a **Student Management Portal** to help manage student records, view courses, and track student attendance.
- The solution must have secure login for administrators and provide a user-friendly interface for managing the data.
- Divide into 3 sprints
- For each spring you can write objective, tasks, success criteria

SOLUTION

- Sprint 1: User Authentication & Dashboard UI (2 Weeks)
- Sprint 2: Student and Course Management (3 Weeks)
- Sprint 3: Attendance Tracking and Reporting (4 Weeks)

KANBAN FRAMEWORK

- **Kanban** focuses on visualizing workflows and limiting work in progress (WIP) to ensure a continuous and efficient flow of tasks.
- **Key Components of Kanban:**
  - **Kanban Board:**
  - A visual board with columns representing stages of the workflow (To Do, In Progress, Testing, Done).
  - **Work in Progress (WIP) Limits:**
  - Set limits on the number of tasks per stage to avoid bottlenecks.
  - **Continuous Delivery:**
  - No fixed sprints; tasks are completed and deployed continuously.

EXAMPLE

- Task Management for a Marketing Campaign:
- Your team has been assigned to run a marketing campaign for a new product launch.
- The campaign includes tasks like
  - content creation
  - social media promotions
  - designing banners
  - email marketing.

KANBAN BOARD COLUMNS:

- **To Do:** Tasks that need to be started.
- **In Progress:** Tasks currently being worked on.
- **Review:** Tasks that are completed but need review.
- **Done:** Tasks that are completed and approved.

Let's DO the same

TO DO

- Write social media posts.
- Design product banners.
- Schedule email marketing.
- Create campaign analytics dashboard.
- Approve and publish website content.

START TASK EXECUTION:

- Team members pick tasks from the **To Do** column based on priority and move them to the **In Progress**
  - The content writer moves the "Write social media posts" task to **In Progress**.
  - The designer picks "Design product banners" and moves it to **In Progress**
- **Monitor Progress:**
  - During daily check-ins, the Kanban board is reviewed to visualize progress and identify bottlenecks.
  - If a task is stuck (e.g., awaiting approval or resources), it remains in the **In Progress** column, and the issue is discussed during the check-in.

REVIEW

- Once tasks are completed, they are moved to the **Review** column for peer review or approval by a supervisor.

- The completed "Design product banners" task is moved to **Review** for quality and brand consistency checks.
- The supervisor reviews "Write social media posts" and requests changes if needed.

COMPLETION

- After successful review and approval, tasks are moved to the **Done** column. Example:
- The task "Design product banners" moves to **Done** after the manager's approval.
- "Schedule email marketing" moves to **Done** once the emails are successfully scheduled.

Kanban	Scrum
Roles are fluid. Project manager optional.	Roles are predefined. Scrum master required.
Tasks are shared by everyone.	Tasks have assigned owners.
Timelines evolve on an as-needed basis.	Timelines are timeboxed into sprints.
Changes can be made mid-stream, allowing for iterations before completion of a project.	Changes can only be made upon completion of a sprint.
Productivity is measured by the cycle time of the complete project.	Productivity is measured by the number of story points completed in each sprint.