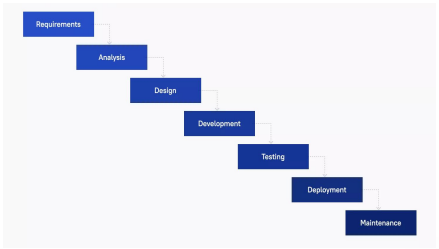
Vanier College
Continuing Education

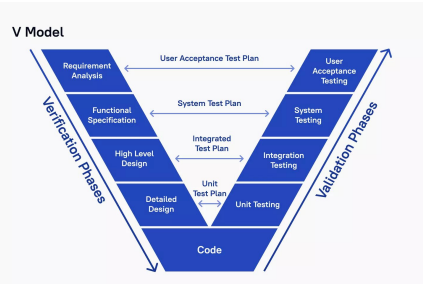
02 SDCL Models

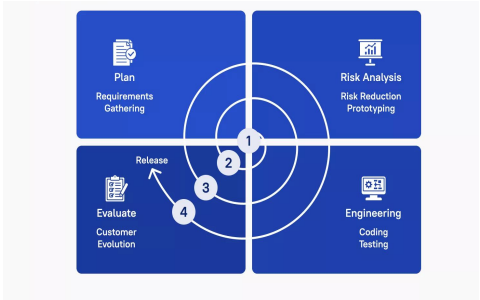
System Development

420-940-VA Sect 87414

Teacher: Jay Patel







What are SDLC Models?

- SDLC (Software Development Life Cycle) models are frameworks that define the process, structure, and sequence of phases in software development.
- These models outline how to move from initial planning to project completion, providing a systematic approach to creating high-quality software.

Purpose of SDLC Models

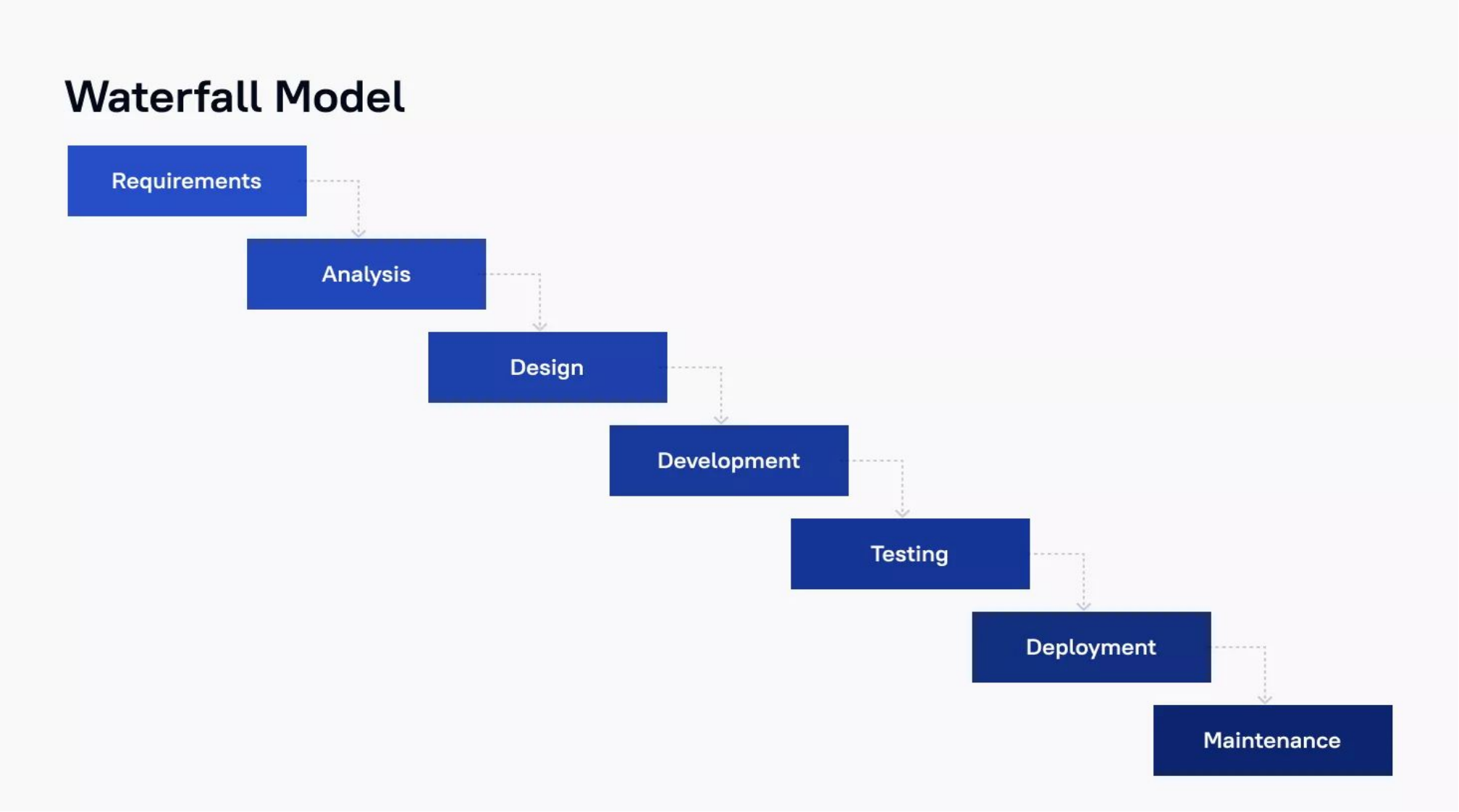
- **Guidance:** SDLC models guide development teams in organizing and managing tasks and phases in software projects.
- **Consistency:** By following a specific model, teams can maintain a consistent approach to developing software across different projects.
- **Predictability:** SDLC models help anticipate the progression of a project, enabling better planning of time, resources, and milestones.
- **Risk Management:** Many SDLC models are designed to manage risks by incorporating reviews, testing, and validation processes at various stages.
- **Quality Assurance:** Ensuring that software meets both functional and non-functional requirements through structured processes, thereby reducing errors and increasing efficiency.

Why Use SDLC Models?

1. **Structured Development Process:** Ensures that each step of software creation is well-planned and executed, reducing risks of project failure.
2. **Improved Communication:** By following a defined model, teams can communicate effectively across different departments (designers, developers, testers, etc.).
3. **Client and Stakeholder Involvement:** SDLC models provide frameworks for client reviews and feedback at regular intervals, enhancing satisfaction and alignment with goals.
4. **Enhanced Quality Control:** Each phase is typically reviewed or tested, ensuring that potential issues are identified and addressed early.

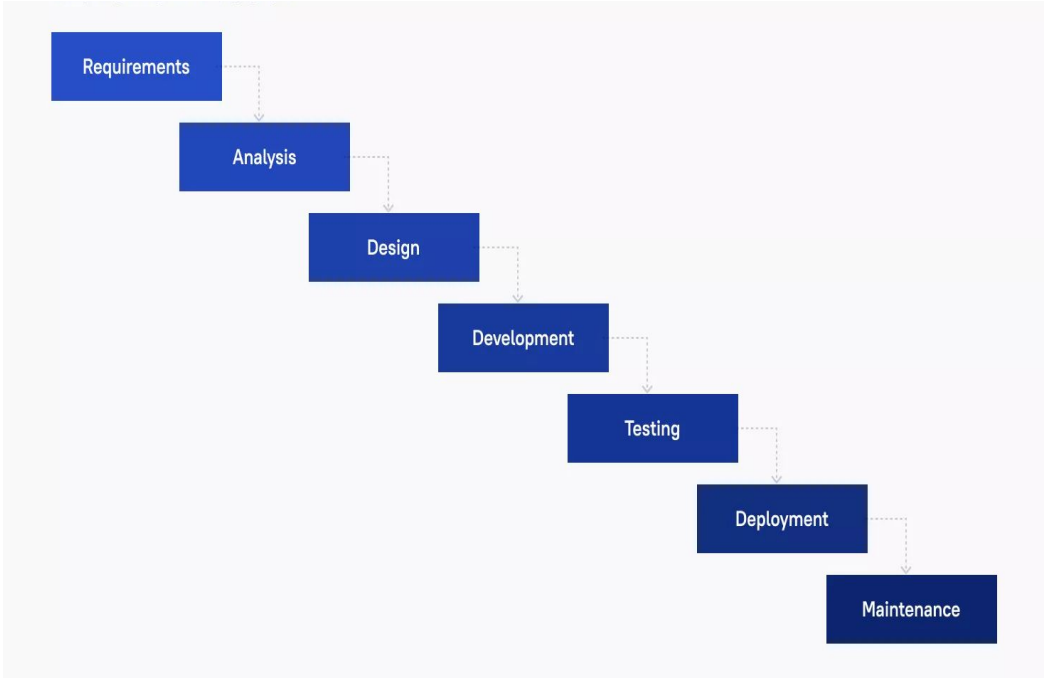
Waterfall Model

- The Waterfall Model is a linear and sequential SDLC model where each phase of the software development process must be completed before moving on to the next.
- It is one of the oldest and rarely used models in modern projects.



Phases in Waterfall Model

1. **Requirements Gathering and Analysis**: In this phase, all system requirements are collected and documented.
2. **System Design**: Based on the gathered requirements, the overall system architecture and software design are developed.
3. **Implementation**: The design is translated into code. Developers begin coding based on the system design.
4. **Integration and Testing**: After development, the software is tested for defects and bugs.
5. **Deployment of System**: Once testing is complete, the software is deployed to the production environment.
6. **Maintenance**: After deployment, the system enters the maintenance phase for any updates, bug fixes, or enhancements.



Waterfall Model

Advantages

- Simple and easy to understand.
- Clearly defined stages.
- Easy to manage due to the rigidity of the model.
- Best suited for smaller projects with well-defined requirements.

Disadvantages

- Not flexible for changes once the project has started.
- No working software is available until late in the development cycle.
- Difficult to respond to changing requirements or feedback.
- High risk and uncertainty due to lack of iteration.

Best Use Cases

- Projects with well-defined requirements that are unlikely to change.
- Systems that are not dependent on user feedback or ongoing changes (e.g., government or military projects).

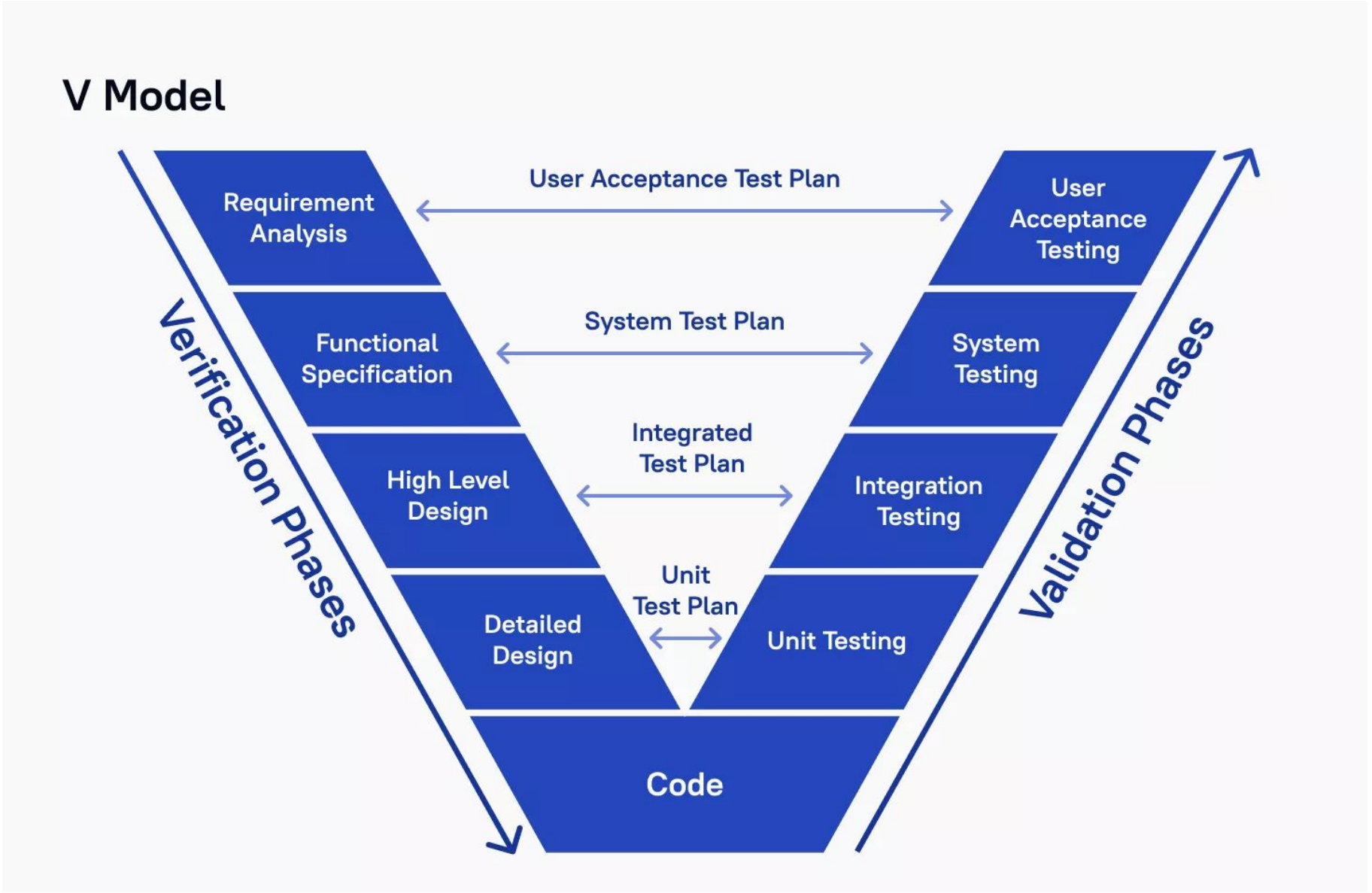
Waterfall Model: Use Case

Building a physical product, such as hardware, is an excellent example of where the Waterfall Model is applicable. In hardware development:

- All the design and specification must be completed and finalized before production begins.

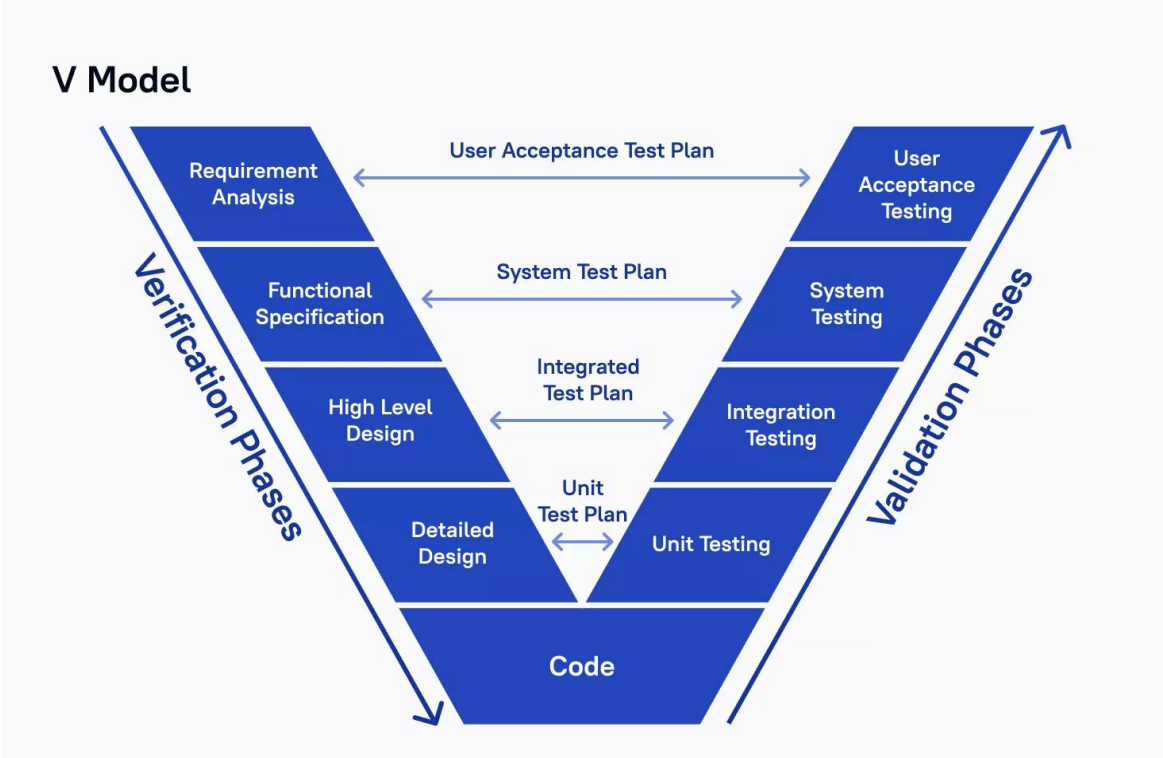
V-Model

- The V-Model, or the Verification and Validation model, is a variant of the Waterfall model.
- It emphasizes the parallel testing and development phases, with each development phase having a corresponding testing phase.



V-Model: Overview

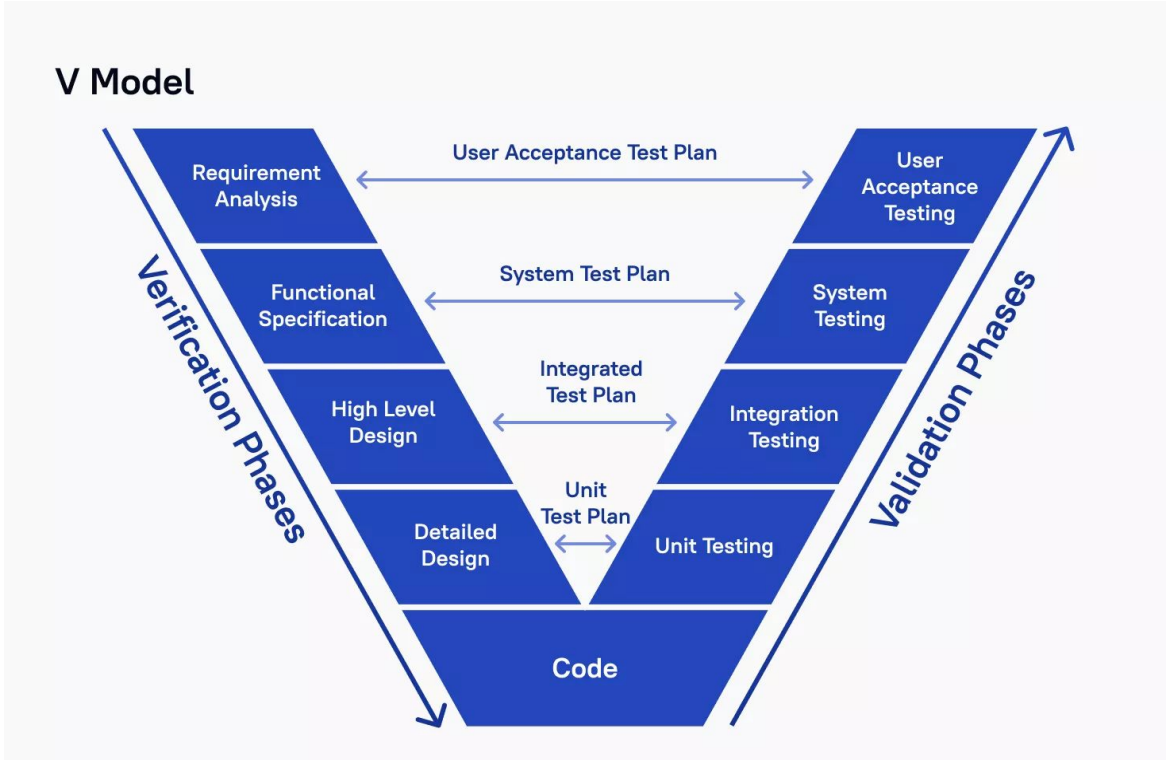
- The V-Model is shaped like the letter “V,” with the left side representing the Verification Phases and the right side representing the Validation Phases.
- This model is particularly well-suited for systems where errors are costly to fix once the system is deployed, such as embedded systems or critical infrastructure systems.



V-Model: Phases

Verification Phases (Left side of the V)

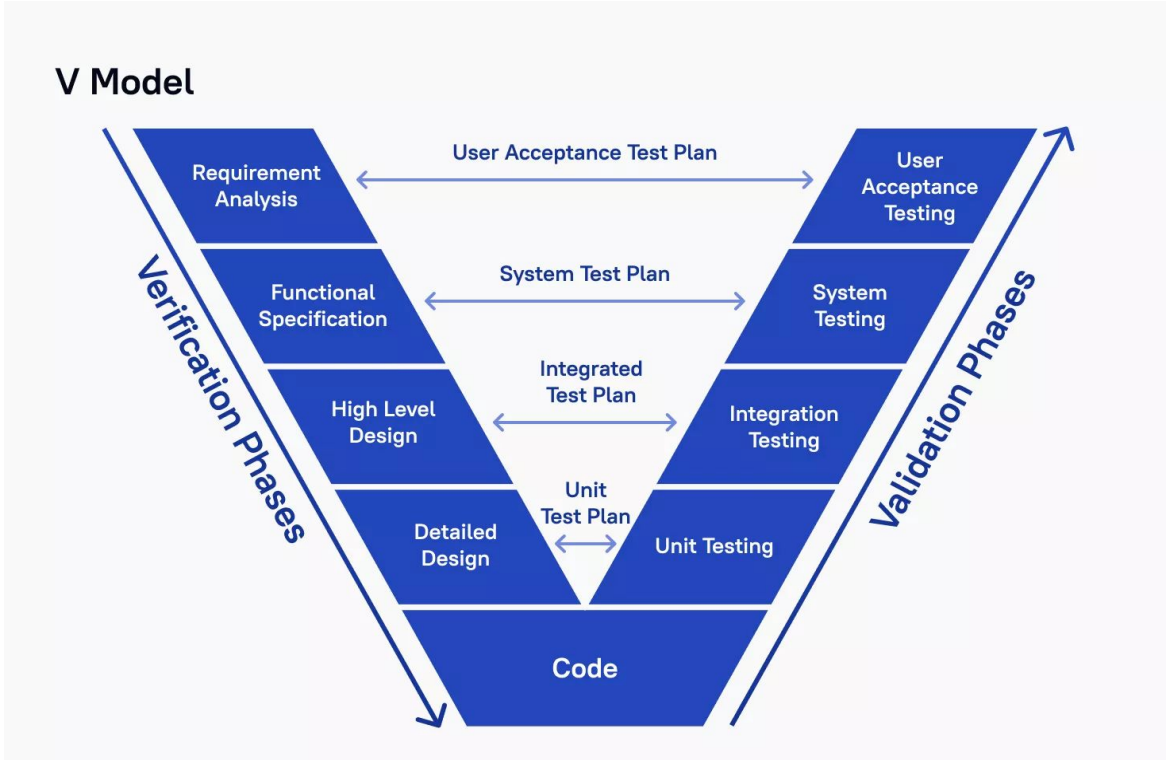
1. **Requirement Analysis**: This initial phase involves gathering and analyzing the requirements from the customer to understand what needs to be developed.
2. **Functional Specification**: The next step is to define the functionality that the software must perform. This is documented in a functional specification.
3. **High Level Design**: Based on the functional specification, a high-level design is created, outlining the system architecture and an overview of the software structure.
4. **Detailed Design**: The design is further refined into detailed specifications for each component of the software. This includes detailed workflows and data models.
5. **Code**: The base of the V, where the actual coding takes place. The detailed design guides developers in coding the software components.



V-Model: Phases

Validation Phases (Right side of the V)

1. **Unit Testing**: After coding, each component (or unit) of the software is tested individually to ensure it meets the detailed design specifications. This is the first step in testing the software.
2. **Integration Testing**: As units are tested and validated, they are integrated into larger groups. Integration testing ensures that these units work together as expected.
3. **System Testing**: After all components are integrated, the entire system is tested to verify that it meets the original functional specifications. This is often performed in a staging environment that mimics the production environment.
4. **User Acceptance Testing**: This is the final phase of testing, where the software is tested by the end-users to ensure it meets their needs and the business requirements initially gathered. This testing is often done in the production-like environment to ensure all user workflows are tested accurately.



V-Model: Feedback Loops

- Each validation phase is directly linked to its corresponding verification phase. This means that the results from unit testing are used to improve the detailed design, integration testing results can lead to changes in the high-level design, and so on.
- This linkage ensures that any discrepancies found during testing can directly impact and improve the design documents and requirements, thereby minimizing the risk of major issues when the software is deployed.
- At every stage, test plans and test cases are created to verify and validate the product according to the requirement of that stage. For example, in requirement gathering stage the test team prepares all the test cases in correspondence to the requirements. Later, when the product is developed and is ready for testing, test cases of this stage verify the software against its validity towards requirements at this stage.
- This makes both verification and validation go in parallel.

V Model

Advantages

- Early detection of defects due to testing at every phase.
- More structured and disciplined approach than Waterfall.
- Good for projects with clearly defined requirements.

Disadvantages

- Like the Waterfall model, the V-Model is quite rigid, making it difficult to adapt to changes midway through the project.
- It requires a clear and complete definition of requirements at the beginning of the project, which can be challenging for projects where requirements are likely to change or evolve.

Best Use Cases

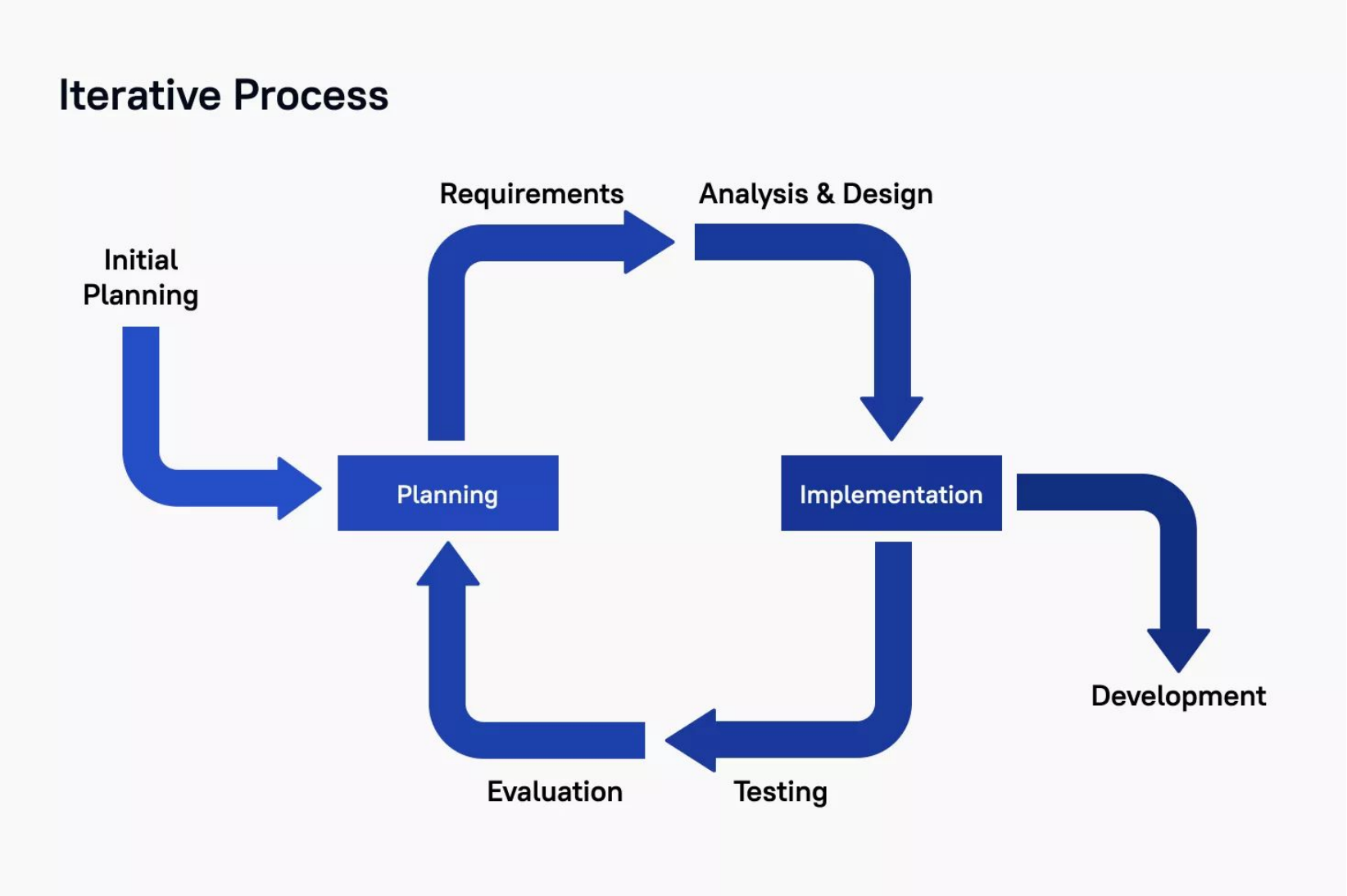
- The V-Model is particularly well-suited for scenarios where precision, reliability, and safety are paramount. The most effective use cases involve environments where changes are minimal once development begins and the cost of failure is high.

V-Model :Use Case

The development of medical software systems where each function must be rigorously tested due to strict regulatory standards.

Iterative and Incremental Model

The Iterative and Incremental model focuses on building small parts of the system and improving upon them in iterations. The system is developed piece by piece rather than all at once.

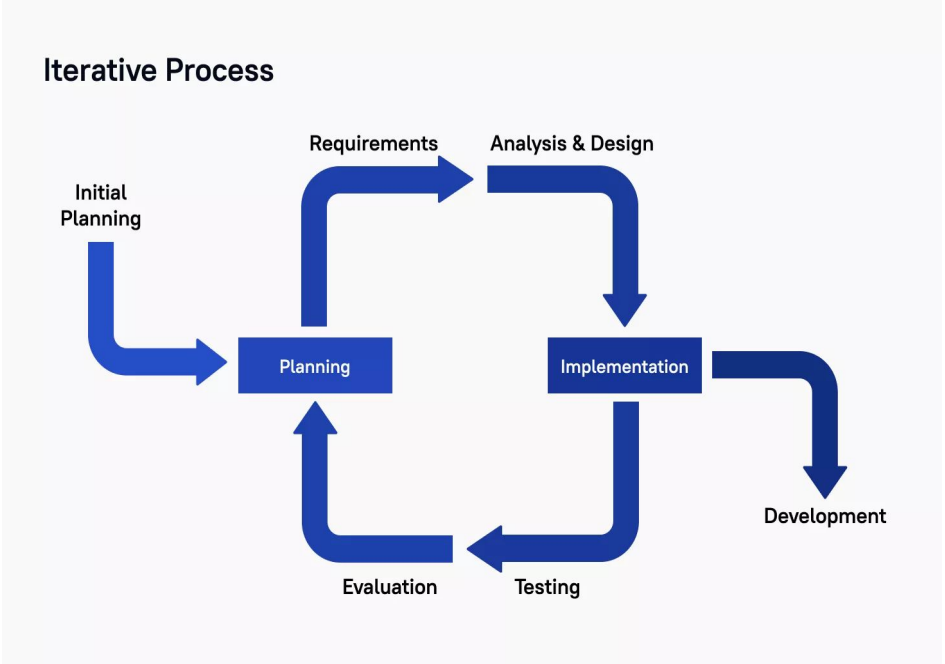


Iterative and Incremental Model: Overview

Incremental Improvements: Rather than redesigning or developing the entire software from scratch with each iteration, this model builds on the existing software base. Each iteration enhances or extends the software's functionality.

Synchronous Development: Iterations can be planned according to a schedule, but they can also occur as required based on technical needs or business priorities, making the process adaptable.

Flexibility and Risk Management: Because the software is developed in increments, it's easier to manage risk by resolving issues as they arise. This approach is highly adaptive to changes in user requirements or



Iterative Models

Advantages

- Flexibility to make changes during development.
- Continuous feedback and improvements.
- Risk is reduced as the product is developed and tested in increments.

Disadvantages

- Can be difficult to manage due to the need for continuous iteration.
- Requires constant feedback and re-evaluation.

Best Use Cases

- Complex projects where the requirements are expected to evolve over time (e.g., website or software with changing user needs).

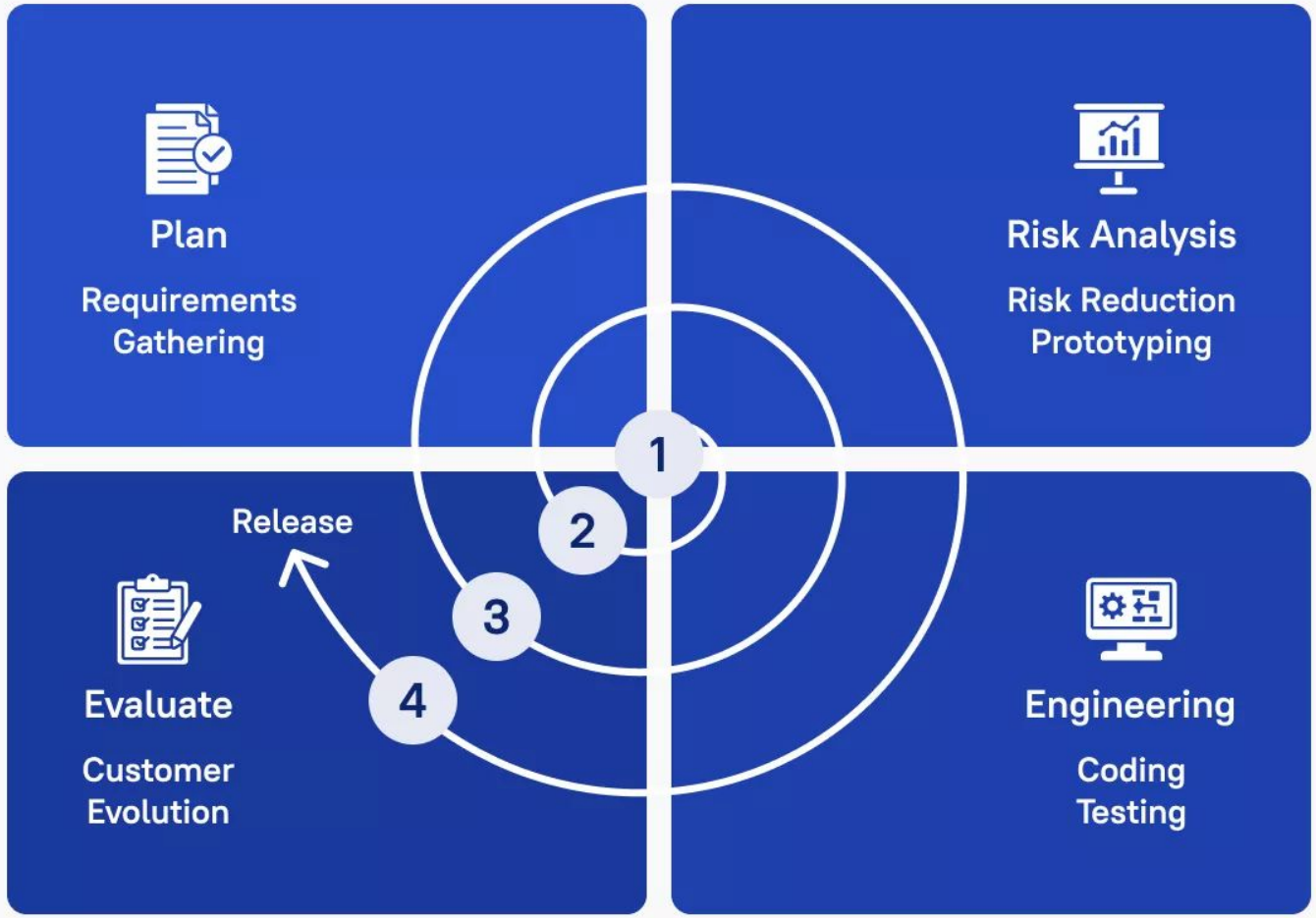
Iterative Models: Use Case

Projects that start with ambiguous goals or incomplete specifications where clarity and direction might evolve as the project progresses.

Spiral Model

The Spiral Model combines the iterative nature of Agile with a risk assessment component. It is suited for large, high-risk projects.

Spiral Model



Spiral Model: Overview

Working of the Spiral Model:

1. **Planning:**
   - Establish project objectives and gather requirements.
2. **Risk Analysis:**
   - Identify and mitigate potential risks through prototyping.
3. **Engineering:**
   - Develop and test software components based on refined requirements.
4. **Evaluation:**
   - Obtain stakeholder feedback and refine the product for the next iteration.

Iteration Process:

- **Iterate and Refine:** Repeat the cycle, adjusting based on feedback and risk analysis to improve the product incrementally.

Spiral Models

Advantages

- Risk management is built into the process.
- Flexible and allows for changes throughout the project.
- Focuses on continuous improvement and refinement.

Disadvantages

- Can be more complex to manage.
- High cost due to continuous iterations and risk analysis.

Best Use Cases

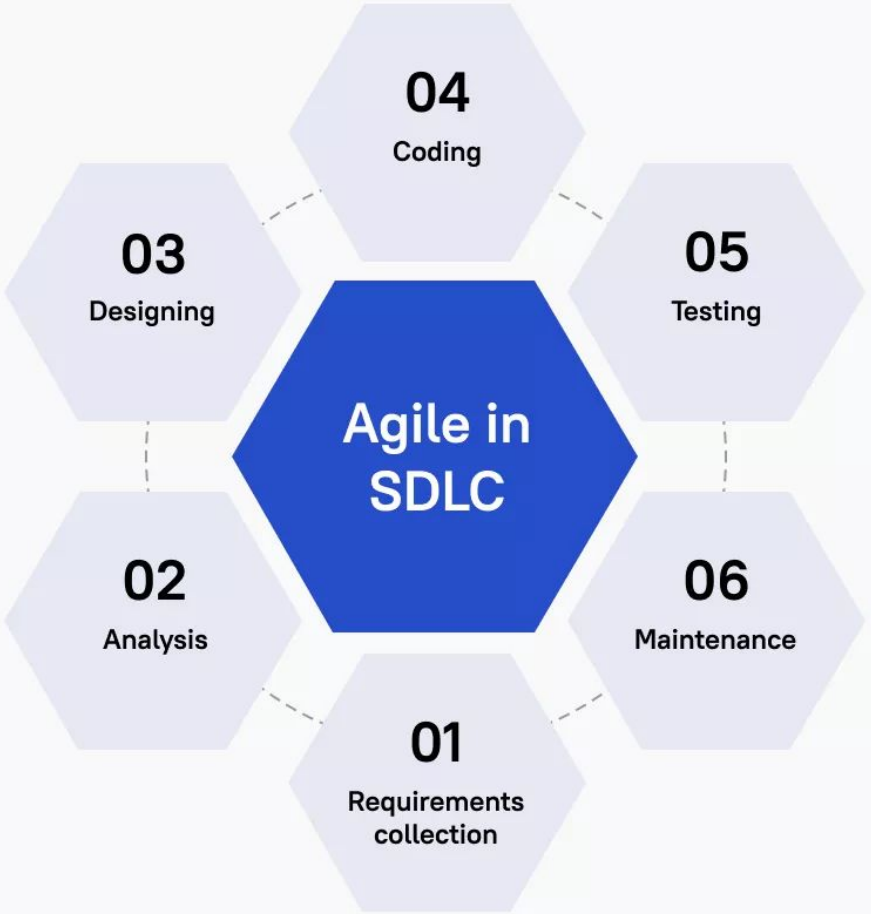
- Projects with high uncertainty, risk or complex requirements (e.g., large enterprise applications).

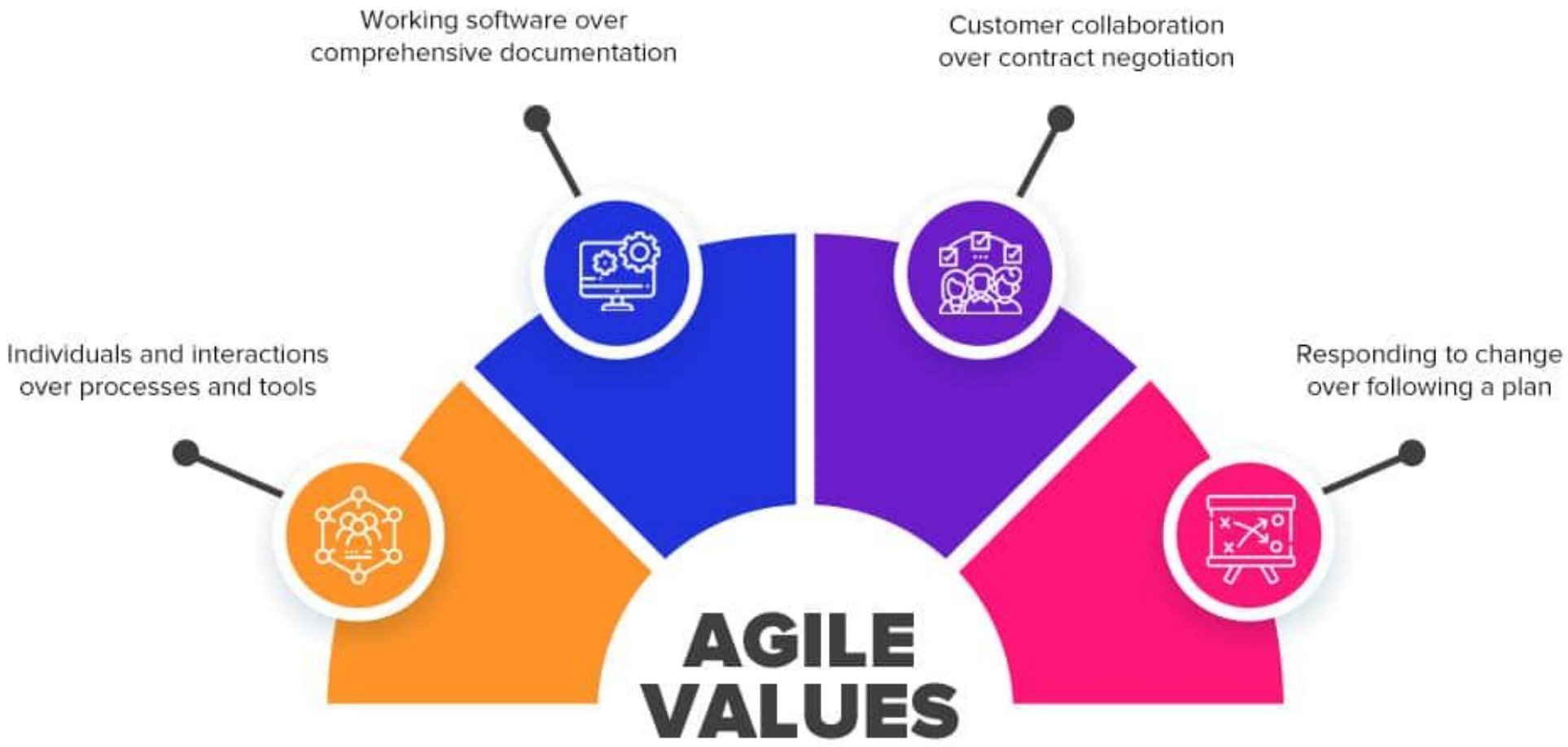
Spiral Model: Use Case

Large-scale financial systems where requirements and regulations may change, necessitating a high focus on risk analysis.

Agile Model

- The Agile Model is an iterative and incremental approach to software development.
- It emphasizes flexibility, customer collaboration, and the ability to adapt to changing requirements throughout the development process.

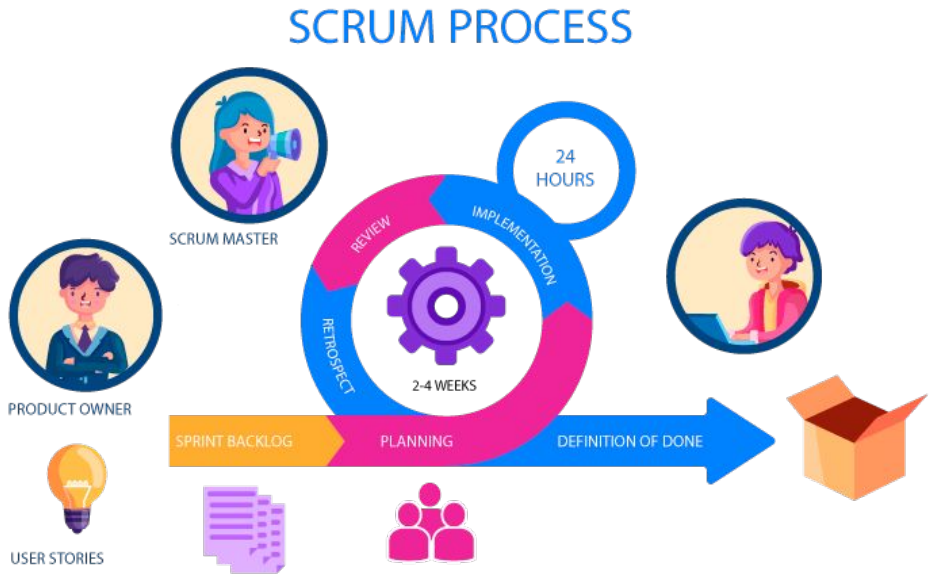




Agile Frameworks

Scrum:Its a process that focuses on short "sprints" (usually 2-4 weeks) where a set amount of work is completed.

Kanban: Framework that focuses on visualizing workflow and limiting work in progress to increase efficiency.



Agile Model

Advantages

- High customer satisfaction due to continuous feedback.
- Flexibility to respond to changes quickly.
- Regular delivery of working software in short iterations.
- Encourages strong collaboration and communication within teams.

Disadvantages

- Less predictability with time and budget since scope can change frequently.
- Requires continuous customer involvement, which can be challenging.
- Can become chaotic if not properly managed, especially in larger teams.

Best Use Cases

- Projects where requirements are likely to change frequently.
- Projects that benefit from continuous customer feedback (e.g., app development, SaaS).
- Startups and fast-moving industries that need rapid iterations and feedback.

Agile Model: Use Case

Software development for mobile applications (e.g., iOS/Android apps) where frequent updates, user feedback, and new features are continually added. Agile allows development teams to adapt to the changing market and user demands.

Waterfall Model:

- A linear and sequential approach where each phase must be completed before the next begins. Ideal for projects with clear, unchanging requirements.

V-Model:

- An extension of the Waterfall model that incorporates rigorous validation and verification steps at each stage. Suitable for projects requiring high reliability and thorough testing.

Iterative Model:

- Focuses on cyclically refining and enhancing the software in successive versions. Allows for gradual improvement and incorporation of user feedback throughout the development process.

Spiral Model:

- Combines iterative development with a strong emphasis on risk assessment at each cycle. Best for large, complex projects with significant risk factors.

Agile Model:

- Highly adaptive and iterative, focusing on continuous feedback and iterative progress through small, fast increments. Ideal for projects where requirements are expected to change or evolve frequently.

Comparisons

Agile vs Iterative Model

1. Flexibility in Requirements:

- **Agile:** Highly flexible. Agile methodologies like Scrum and Kanban allow for frequent reassessment of requirements and encourage changes even late in the development process.
- **Iterative:** Moderately flexible. Changes are possible between iterations, but each iteration typically follows a mini-Waterfall process with less emphasis on changing requirements mid-way.

2. Customer Involvement:

- **Agile:** Continuous and intense. Stakeholder and customer feedback is integrated after each sprint (a short, time-boxed period), and customer collaboration is a core value.
- **Iterative:** Periodic. While the iterative model allows for learning and adaptation, the customer feedback loops are not as frequent or as critical to the process as in Agile.

3. Project Scope and Delivery:

- **Agile:** Delivers working software frequently, with releases often completed in weeks.
- **Iterative:** Focuses on improving and refining the project over time, with releases that might take longer depending on the iteration length.

Comparisons

1. Risk Management:

- **Agile:** Generally does not have a formal risk management process; instead, risks are managed informally as they arise during each sprint.
- **Spiral:** Explicit risk analysis and mitigation are built into each phase of the development cycle, making it ideal for large, complex projects with significant risks.

2. Structured Design:

- **Agile:** Less structured, with teams encouraged to self-organize and adapt the workflow as needed. The design evolves throughout the project.
- **Spiral:** More structured in terms of risk and development phases, though it is flexible in handling changes. Each loop through the spiral involves specific objectives, including risk assessment.

3. Scale of Implementation:

- **Agile:** Best suited for smaller to medium-sized teams and projects where quick, incremental delivery is feasible and beneficial.
- **Spiral:** Can be scaled for very large projects that require thorough risk assessments, such as in industries like aerospace and defense.

Comparisons

Model	Characteristics	Advantages	Disadvantages	Best For
Waterfall	Linear and sequential	Simple and easy to understand and manage	Inflexible; poor adaptability to changes	Projects with well-defined, stable requirements
V-Model	Waterfall with rigorous testing at each step	Emphasizes quality and reliability	Rigid like Waterfall; costly changes	Projects where precise, error-free results are critical
Iterative	Cycles through planning, implementation, and evaluation	Allows for gradual improvements and adaptation	Can become repetitive and inefficient over time	Projects with evolving requirements but known objectives
Spiral	Risk-driven, combines iterative and Waterfall aspects	Strong focus on risk management; highly customizable	Complex and potentially costly to implement	Large, complex projects with significant risks
Agile	Iterative and incremental with a focus on collaboration and customer feedback	High adaptability; frequent delivery of products	Requires constant stakeholder engagement; less predictable	Projects with uncertain or rapidly changing requirements

Summary

- **Waterfall and V-Model** are more traditional and structured, suitable for projects with clear requirements and where changes are unlikely or costly.
- **Iterative** provides a balance between structure and flexibility, allowing for adjustments based on learning from previous iterations.
- **Spiral** is especially effective in managing risks and is ideal for projects where potential risks could have serious repercussions.
- **Agile** focuses on quick iterations and constant feedback, adapting quickly to changing needs and ensuring the product meets user demands efficiently.

Match the Following:

Scenarios:

1. A government defense project that requires meticulous documentation and risk management.
2. A startup developing a new app where features may change based on early user feedback.
3. A bank updating its legacy system, where requirements are known and stability is paramount.
4. A software project that needs frequent testing and validation every step of the way due to strict regulatory standards.
5. A small software firm updating a client's existing project where incremental improvements are planned over several months.

Models:

A. Waterfall Model B. V-Model C. Iterative Model D. Spiral Model E. Agile Model

Match the Following: Answers

Matches:

1. **D. Spiral Model** - Best for projects where risk management is crucial, such as in defense projects.
2. **E. Agile Model** - Ideal for startups needing to adapt quickly based on user input and market conditions.
3. **A. Waterfall Model** - Suitable for projects with clear, well-defined requirements where changes are not expected.
4. **B. V-Model** - Perfect for projects needing stringent testing and validation due to regulatory demands.
5. **C. Iterative Model** - Fits well with projects planning for gradual enhancements, allowing for feedback and refinements.

References

- Azal University. (2020). *Software Development Life Cycle* [PDF file]. Retrieved from https://www.auhd.edu.ye/upfiles/elibrary/Azal2020-01-22-12-35-12-90529.pdf
- "Images sourced from: 'Understanding the Software Development Life Cycle', stfalcon.com. Available at: https://stfalcon.com/en/blog/post/SDLC-meaning"
- Comparison Content: ChatGPT-4o by Open AI

Thank You!