

SOFTWARE ENGINEERING

MODULE 1

Module 1

- ▶ Introduction
- ▶ Software engineering ethics
- ▶ **Process models**
- ▶ **Agile Software Development**
- ▶ Case studies

Software and Software Engineering

- ▶ **Software** is a program or set of programs containing instructions that provide desired functionality.
- ▶ **Software Engineering** is the process of designing, developing, testing, and maintaining software. It is a systematic and disciplined approach to software development that aims to create high-quality, reliable, and maintainable software. Software engineering includes a variety of techniques, tools, and methodologies.

Software Framework Activities:

- ▶ **Communication:** By communication, customer requirement gathering is done. Communication with consumers and stakeholders to determine the system's objectives and the software's requirements.
- ▶ **Planning:** Establish engineering work plan, describes technical risk, lists resources requirements, work produced and defines work schedule.
- ▶ **Modeling:** Architectural models and design to better understand the problem and for work towards the best solution. The software model is prepared by:
 - o Analysis of requirements
 - o Design
- ▶ **Construction:** Creating code, testing the system, fixing bugs, and confirming that all criteria are met. The software design is mapped into a code by:
 - o Code generation
 - o Testing
- ▶ **Deployment:** In this activity, a complete or non-complete product or software is represented to the customers to evaluate and give feedback. On the basis of their feedback, we modify the product for the supply of better products.

Software engineering ethics and professionalism

- ▶ Software engineering ethics refers to the principles, guidelines, and moral considerations that govern the behavior and actions of software engineers in their professional practice.
- 1. Privacy and Data Security:** Software engineers must consider the privacy and security of user data. They should design systems that protect user information and handle data responsibly, especially in the face of growing concerns about data breaches and unauthorized access.
 - 2. Transparency and Accountability:** Engineers should be transparent about how their software works and its potential implications. This includes being honest about limitations, potential biases, and risks associated with the software they develop.
 - 3. Intellectual Property and Licensing:** Software engineers should respect intellectual property rights, including copyrights, patents, and open-source licenses. They should also consider the implications of their work on the larger software ecosystem.

4. Social Impact: Engineers should consider the broader societal impact of their work. This includes thinking about how software might affect employment, social interactions, and other aspects of society.

5. Professional Development: Engineers should stay updated with the latest developments in software engineering and ethical considerations. This includes continuing education and professional growth to maintain a high standard of practice.

6. Whistleblowing: Engineers should have mechanisms in place to report unethical behavior or violations within their organization without fear of retaliation.

7. Balancing Stakeholder Interests: Engineers often work within organizations with various stakeholders, including customers, shareholders, and the public. Balancing these interests while upholding ethical standards can be challenging but is important.

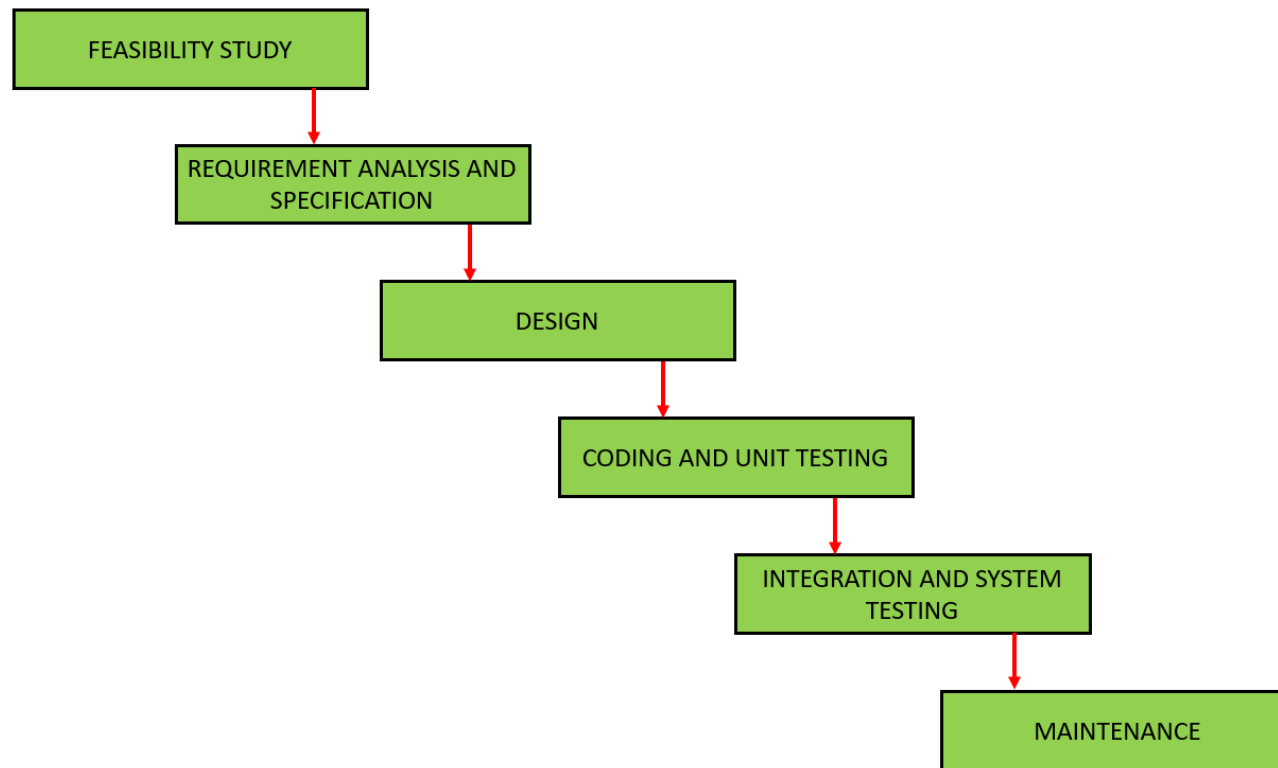
SOFTWARE PROCESS MODELS

- ▶ Software Process models may contain activities, which are part of the software process, software product, and the roles of people involved in software engineering.
- ▶ Different Types of models
 1. Waterfall model/classical waterfall model
 2. Incremental model
 3. Prototype model
 4. RAD
 5. Spiral model
 6. Agile models etc

Water Fall model

- ▶ The classical waterfall model is the basic software development life cycle model. It is very simple but idealistic.
- ▶ Earlier this model was very popular but nowadays it is not used.
- ▶ But it is very important because all the other software development life cycle models are based on the classical waterfall model.

Phases of Classical Waterfall Model



- ▶ **Requirements Gathering and Analysis:** The first phase involves gathering requirements from stakeholders and analyzing them to understand the scope and objectives of the project.
- ▶ **Design:** Once the requirements are understood, the design phase begins. This involves creating a detailed design document that outlines the software architecture, user interface, and system components.
- ▶ **Implementation:** The implementation phase involves coding the software based on the design specifications. This phase also includes unit testing to ensure that each component of the software is working as expected

- ▶ **Testing:** In the testing phase, the software is tested as a whole to ensure that it meets the requirements and is free from defects.
- ▶ **Deployment:** Once the software has been tested and approved, it is deployed to the production environment.
- ▶ **Maintenance:** The final phase of the Waterfall Model is maintenance, which involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.

Advantages of the Classical Waterfall Model

- ▶ **Easy to Understand:** Classical Waterfall Model is very simple and easy to understand.
- ▶ **Individual Processing:** Phases in the Classical Waterfall model are processed one at a time.
- ▶ **Properly Defined:** In the classical waterfall model, each stage in the model is clearly defined.
- ▶ **Properly Documented:** Processes, actions, and results are very well documented.
- ▶ **Working:** Classical Waterfall Model works well for smaller projects and projects where requirements are well understood.


Disadvantages of the Classical Waterfall Model

- ▶ **No Feedback Path:** In the classical waterfall model evolution of software from one phase to another phase is like a waterfall. It assumes that no error is ever committed by developers during any phase. Therefore, it does not incorporate any mechanism for error correction.
- ▶ **Difficult to accommodate Change Requests:** This model assumes that all the customer requirements can be completely and correctly defined at the beginning of the project, but actually customer's requirements keep on changing with time. It is difficult to accommodate any change requests after the requirements specification phase is complete.
- ▶ **No Overlapping of Phases:** This model recommends that a new phase can start only after the completion of the previous phase. But in real projects, this can't be maintained. To increase efficiency and reduce cost, phases may overlap.
- ▶ **Limited Flexibility:** The Waterfall Model is a rigid and linear approach to software development, which means that it is not well-suited for projects with changing or uncertain requirements. Once a phase has been completed, it is difficult to make changes or go back to a previous phase.

- ▶ **Limited Stakeholder Involvement:** The Waterfall Model is a structured and sequential approach, which means that stakeholders are typically involved in the early phases of the project (requirements gathering and analysis) but may not be involved in the later phases (implementation, testing, and deployment).
- ▶ **Late Defect Detection:** In the Waterfall Model, testing is typically done toward the end of the development process. This means that defects may not be discovered until late in the development process, which can be expensive and time-consuming to fix.
- ▶ **Lengthy Development Cycle:** The Waterfall Model can result in a lengthy development cycle, as each phase must be completed before moving on to the next. This can result in delays and increased costs if requirements change or new issues arise.
- ▶ **Not Suitable for Complex Projects:** The Waterfall Model is not well-suited for complex projects, as the linear and sequential nature of the model can make it difficult to manage multiple dependencies and interrelated components.

Applications of Classical Waterfall Model

- ▶ **Large-scale Software Development Projects:** The Waterfall Model is often used for large-scale software development projects, where a structured and sequential approach is necessary to ensure that the project is completed on time and within budget.
- ▶ **Safety-Critical Systems:** The Waterfall Model is often used in the development of safety-critical systems, such as aerospace or medical systems.
- ▶ **Government and Defense Projects:** The Waterfall Model is also commonly used in government and defense projects, where a rigorous and structured approach is necessary to ensure that the project meets all requirements and is delivered on time.

- 
- ▶ **Projects with well-defined Requirements:** The Waterfall Model is best suited for projects with well-defined requirements, as the sequential nature of the model requires a clear understanding of the project objectives and scope.
 - ▶ **Projects with Stable Requirements:** The Waterfall Model is also well-suited for projects with stable requirements, as the linear nature of the model does not allow for changes to be made once a phase has been completed.

Incremental model

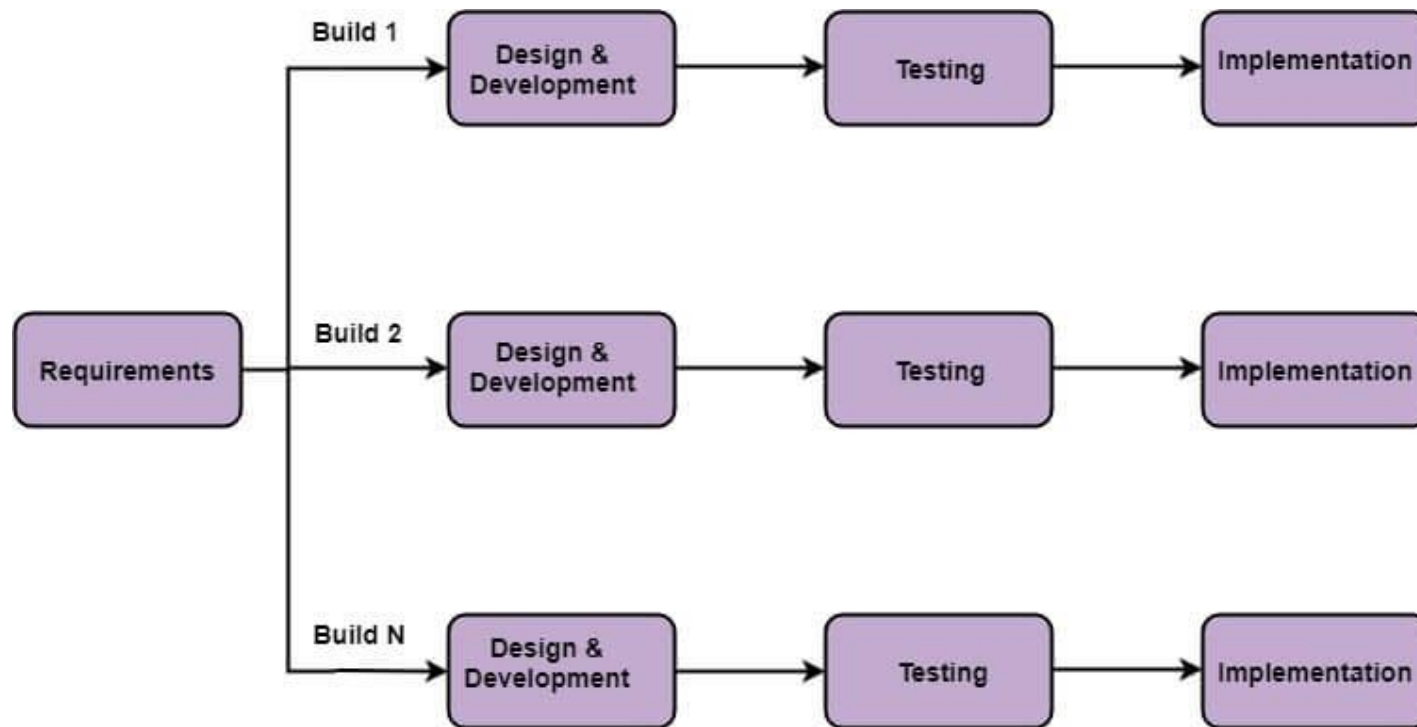
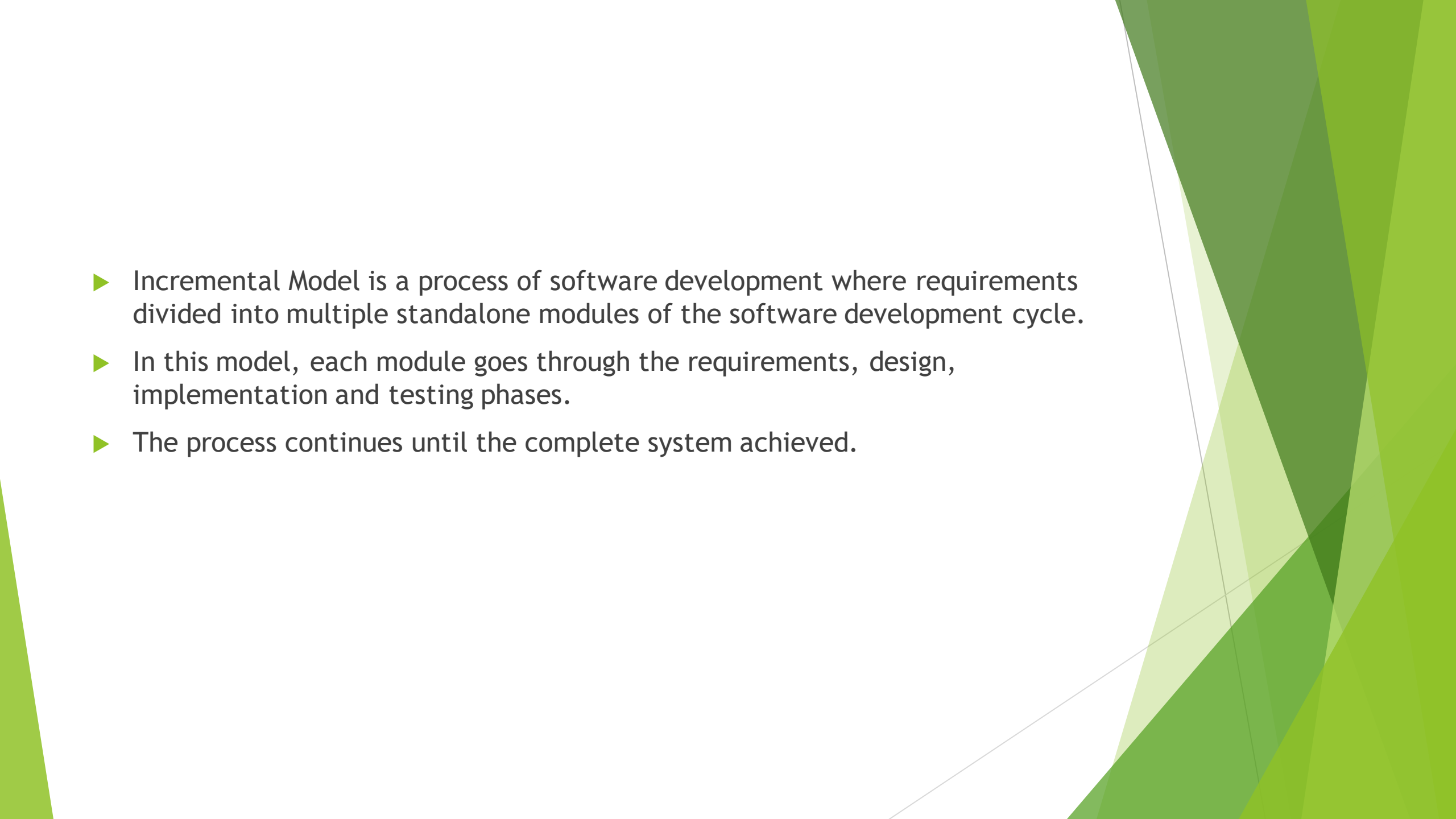


Fig: Incremental Model

- 
- The background of the slide features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.
- ▶ Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle.
 - ▶ In this model, each module goes through the requirements, design, implementation and testing phases.
 - ▶ The process continues until the complete system achieved.

Phases

- ▶ **1. Requirement analysis:** In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.
- ▶ **2. Design & Development Methods:** In this phase of the Incremental model the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

- ▶ **3. Testing:** In the incremental model, the testing phase checks the **importance of each existing function** as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task
- ▶ **4. Implementation:** Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

When we use the Incremental Model?

- ▶ When the requirements are superior.
- ▶ A project has a lengthy development schedule.
- ▶ When Software team are not very well skilled or trained.
- ▶ When the customer demands a quick release of the product.
- ▶ You can develop prioritized requirements first.

Advantage of Incremental Model

- ▶ Errors are easy to be recognized.
- ▶ Easier to test and debug
- ▶ More flexible.
- ▶ Simple to manage risk because it handled during its iteration.
- ▶ The Client gets important functionality early

Disadvantage of Incremental Model

- ▶ Need for good planning
- ▶ Total Cost is high.
- ▶ Well defined module interfaces are needed

Process Activity

SPECIFICATION

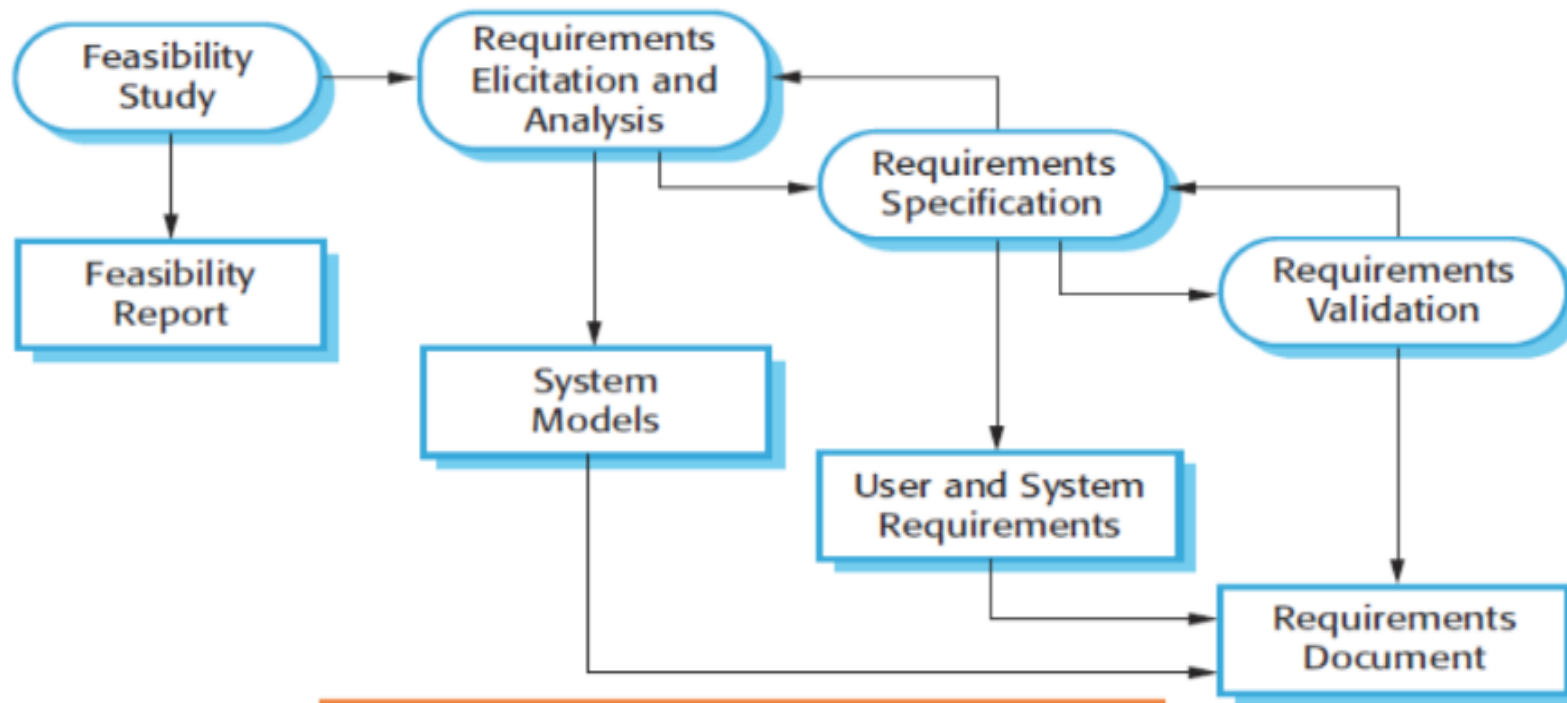
DEVELOPMENT

VALIDATION

EVOLUTION

Software Specification

- ▶ Software specification or **requirements engineering** is the process of understanding and defining what services are required from the system and identifying the constraints on the system's operation and development.
- ▶ Requirements engineering is a particularly critical stage of the software process as errors at this stage unavoidably lead to later problems in the system design and implementation.




The requirements engineering process

- ▶ 1. Feasibility study: The study considers whether the proposed system will be cost-effective from a business point of view and if it can be developed within existing budgetary constraints.
- ▶ A feasibility study should be relatively cheap and quick. The result should inform the decision of whether or not to go ahead with a more detailed analysis.
- ▶ 2. Requirements elicitation and analysis: This is the process of deriving the system requirements through observation of existing systems, discussions with potential users and buyer, task analysis.

- ▶ 3. Requirements specification: Requirements specification is the activity of translating the information gathered during the analysis activity into a document that defines a set of requirements.
- ▶ 4. Requirements validation: This activity checks the requirements for **realism consistency, and completeness**. During this process, errors in the requirements document are inevitably discovered. It must then be modified to correct these problems.

Software design and implementation

- ▶ A software design is a **description of the structure of the software** to be implemented, the data models and structures used by the system, the interfaces between system components and, the algorithms used.

- 
- ▶ There are four activities that may be part of the design process for information systems as shown in the previous figure:
 - ▶ 1. Architectural design, where the software engineer identifies the **overall structure of the system**, the principal components (sometimes called sub-systems or modules), their relationships, and how they are distributed.
 - ▶ 2. Interface design, where the software engineer defines the **interfaces between system components**.

- ▶ 3. Component design, where the software engineer takes each system component and design how it will operate. This may be a simple statement of the **expected functionality** to be implemented, with the specific design left to the programmer. Alternatively, it may be a list of changes to be made to a reusable component or a detailed design model
- ▶ 4. Database design, where the software engineer designs the system data structures and how these are to be represented in a database.

Implementation

- ▶ The development of a program to implement the system follows naturally from the system design processes. Software development tools may be used to generate a skeleton program from a design.
- ▶ This includes code to define and implement interfaces, and, in many cases, the developer need only add details of the operation of each program component.

Software validation

- ▶ Software validation or, more generally, verification and validation (V&V) is intended to show that a system both conforms to its specification and that it meets the expectations of the system customer.
- ▶ Program testing, where the system is executed using simulated test data, is the principal validation technique.
- ▶ Validation may also involve checking processes, such as inspections and reviews, at each stage of the software process from user requirements definition to program development.
- ▶ The majority of validation costs are incurred during and after implementation.

The stages in the testing process are:

- ▶ 1. Component (or unit) testing: Individual components are tested to ensure that they operate correctly.
- ▶ Each component is tested independently, without other system components. Components may be simple entities such as functions or object classes, or may be coherent groupings of these entities.
- ▶ 2. System testing: System components are integrated to create a complete system. This process is concerned with finding errors that result from not expected interactions between components and component interface problems.
- ▶ It is also concerned with showing that the system meets its functional and non-functional requirements, and testing the emergent system properties.


- ▶ Acceptance testing: This is the final stage in the testing process before the system is accepted for operational use.
- ▶ The system is tested with data supplied by the system customer rather than with simulated test data.

Software evolution

- ▶ The development and maintenance are relevant.
- ▶ Few software systems are now completely new systems and it makes much more sense to see development and maintenance as a continuous sequence.
- ▶ Software engineering as an evolutionary process where **software is continually changed over its lifetime** in response to changing requirements and customer needs.


Coping with change

- ▶ Coping with change or sometime called “Process iteration” means that, change is sure to happen in all large software projects.
- ▶ As new technologies become available, new design and implementation possibilities emerge.
- ▶ Therefore whatever software process model is used, it is essential that it can accommodate changes to the software being developed.

- 
- ▶ Change adds to the costs of software development because it usually means that work that has been completed has to be redone. This is called rework.
 - ▶ For example, if the relationships between the requirements in a system have been analyzed and new requirements are then identified, some or all of the requirements analysis has to be repeated.
 - ▶ It may then be necessary to redesign the system to deliver the new requirements, change any programs that have been developed, and re-test the system.

There are two related approaches that may be used to reduce the costs of rework:

- ▶ 1. Change avoidance, where the software process includes activities that can anticipate possible changes before significant rework is required.
- ▶ For example, a **prototype system may be developed to show some key features of the system to customers.**
- ▶ They can experiment with the prototype and refine their requirements before committing to high software production costs.

- 
- ▶ 2. Change tolerance: where the process is designed so that **changes can be accommodated at relatively low cost.**
 - ▶ This normally involves some form of incremental development.
 - ▶ Proposed changes may be implemented in increments that have not yet been developed.

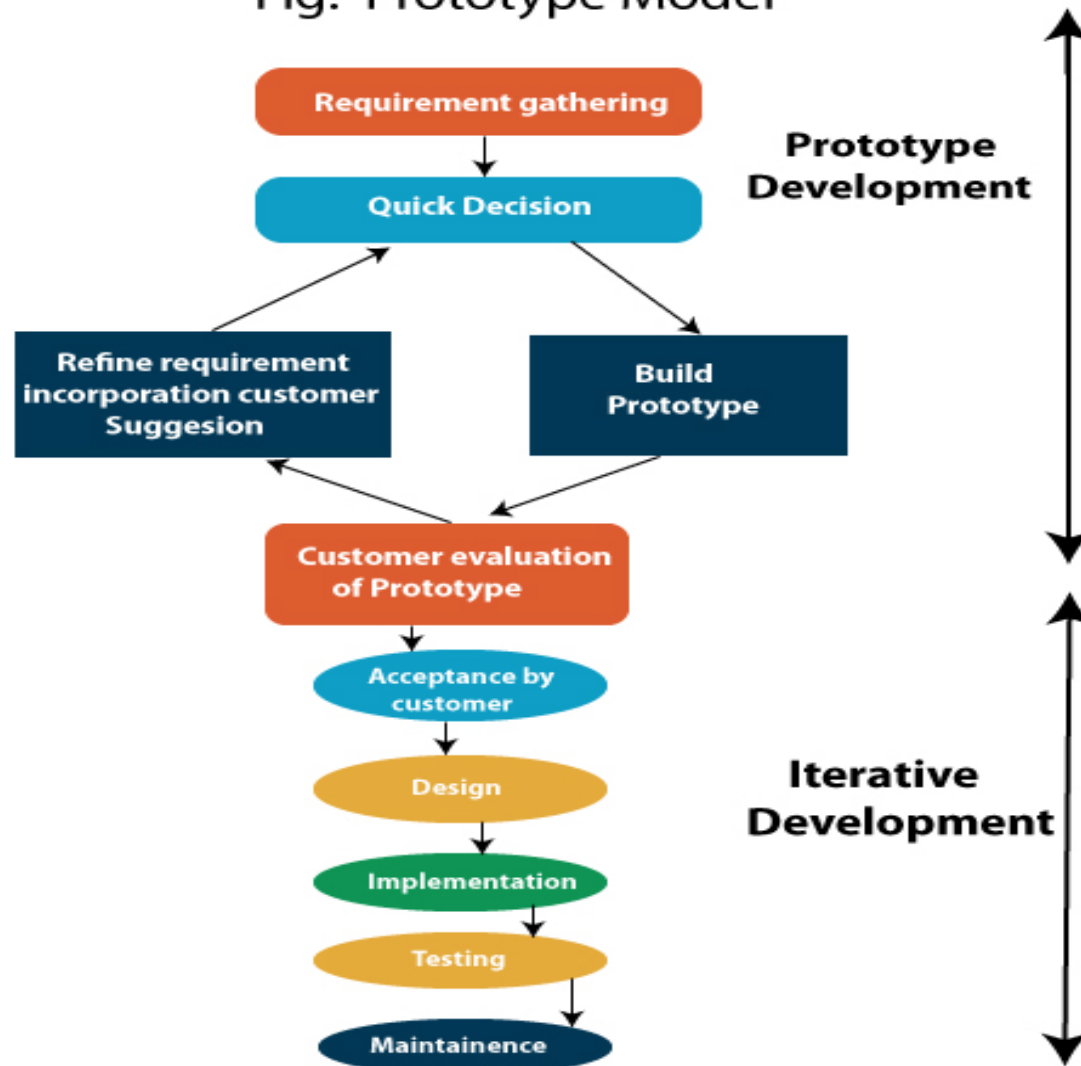
EVALUATIONARY PROCESS MODEL

- ▶ Prototype Model
- ▶ Spiral Model

Prototype Model

- ▶ The prototype model requires that **before carrying out the development of actual software, a working prototype of the system should be built.**
- ▶ A prototype is a toy implementation of the system.
- ▶ A prototype usually turns out to be a very crude version of the actual system, possible exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to actual software.

Fig: Prototype Model



Steps of Prototype Model

- ▶ Requirement Gathering and Analysis
- ▶ Quick Decision
- ▶ Build a Prototype
- ▶ Assessment or User Evaluation
- ▶ Prototype Refinement
- ▶ Engineer Product

Advantage of Prototype Model

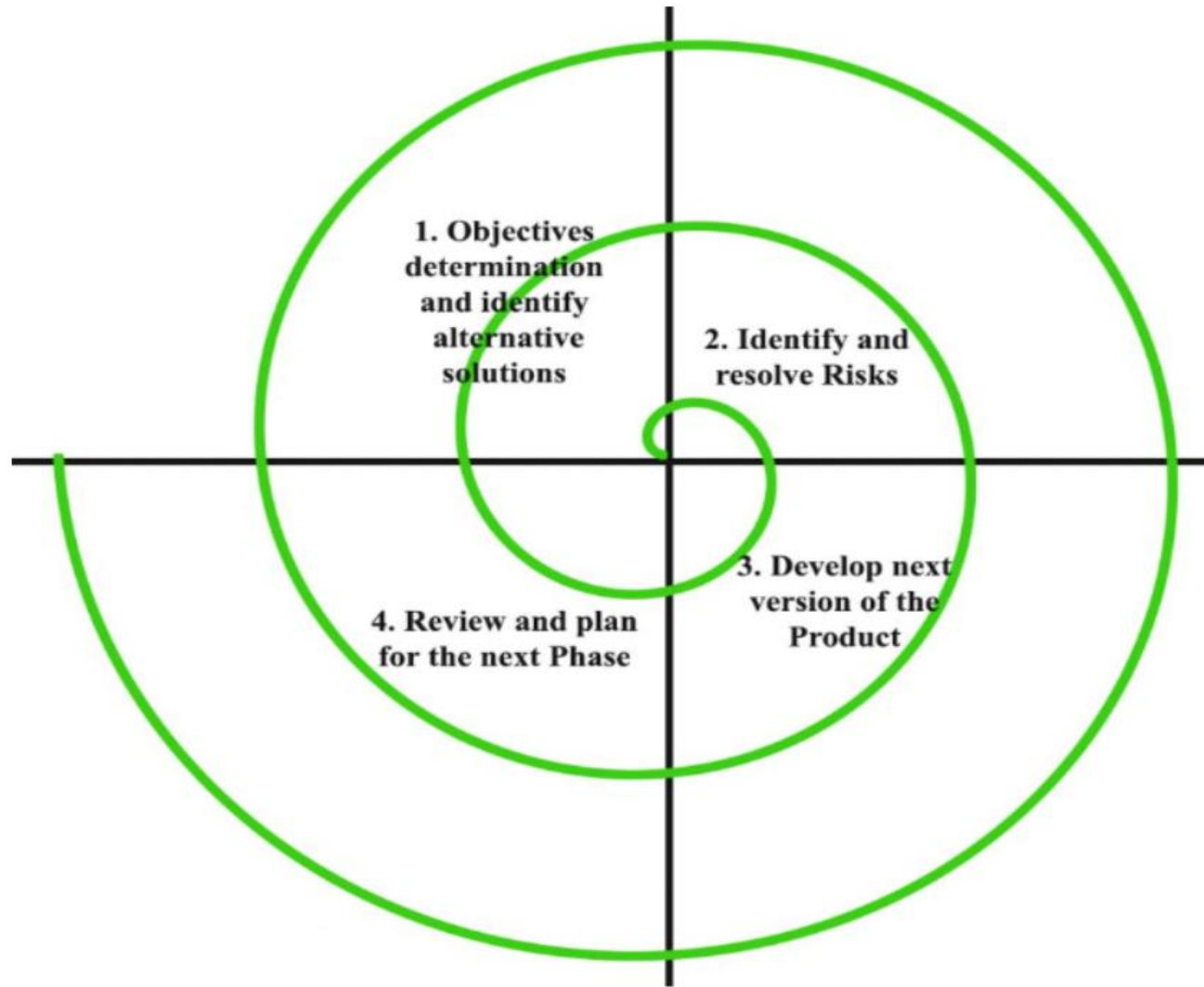
- ▶ Reduce the risk of incorrect user requirement
- ▶ Good where requirement are changing/uncommitted
- ▶ Reduce Maintenance cost.
- ▶ Errors can be detected much earlier as the system is made side by side

Disadvantage of Prototype Model

- ▶ An unstable/badly implemented prototype often becomes the final product.
- ▶ Require extensive customer collaboration
 - ▶ Costs customer money
 - ▶ Needs committed customer
 - ▶ Difficult to finish if customer withdraw
 - ▶ May be too customer specific.
 - ▶ Difficult to know how long the project will last.
- ▶ Prototyping tools are expensive.
- ▶ Special tools & techniques are required to build a prototype.
- ▶ It is a time-consuming process.

The Spiral model

- ▶ Spiral model is a **risk driven** process model.
- ▶ It is used for generating the software projects.
- ▶ In spiral model, an alternate solution is provided if the risk is found in the risk analysis, then alternate solutions are suggested and implemented.
- ▶ It is a combination of prototype and sequential model or waterfall model.
- ▶ In one iteration all activities are done, for large project's the output is small



- ▶ **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
- ▶ **Identify and resolve Risks:** During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the **risks associated with that solution are identified and the risks are resolved using the best possible strategy**. At the end of this quadrant, the Prototype is built for the best possible solution.

- ▶ **Develop the next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
- ▶ **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so-far developed version of the software. In the end, planning for the next phase is started.

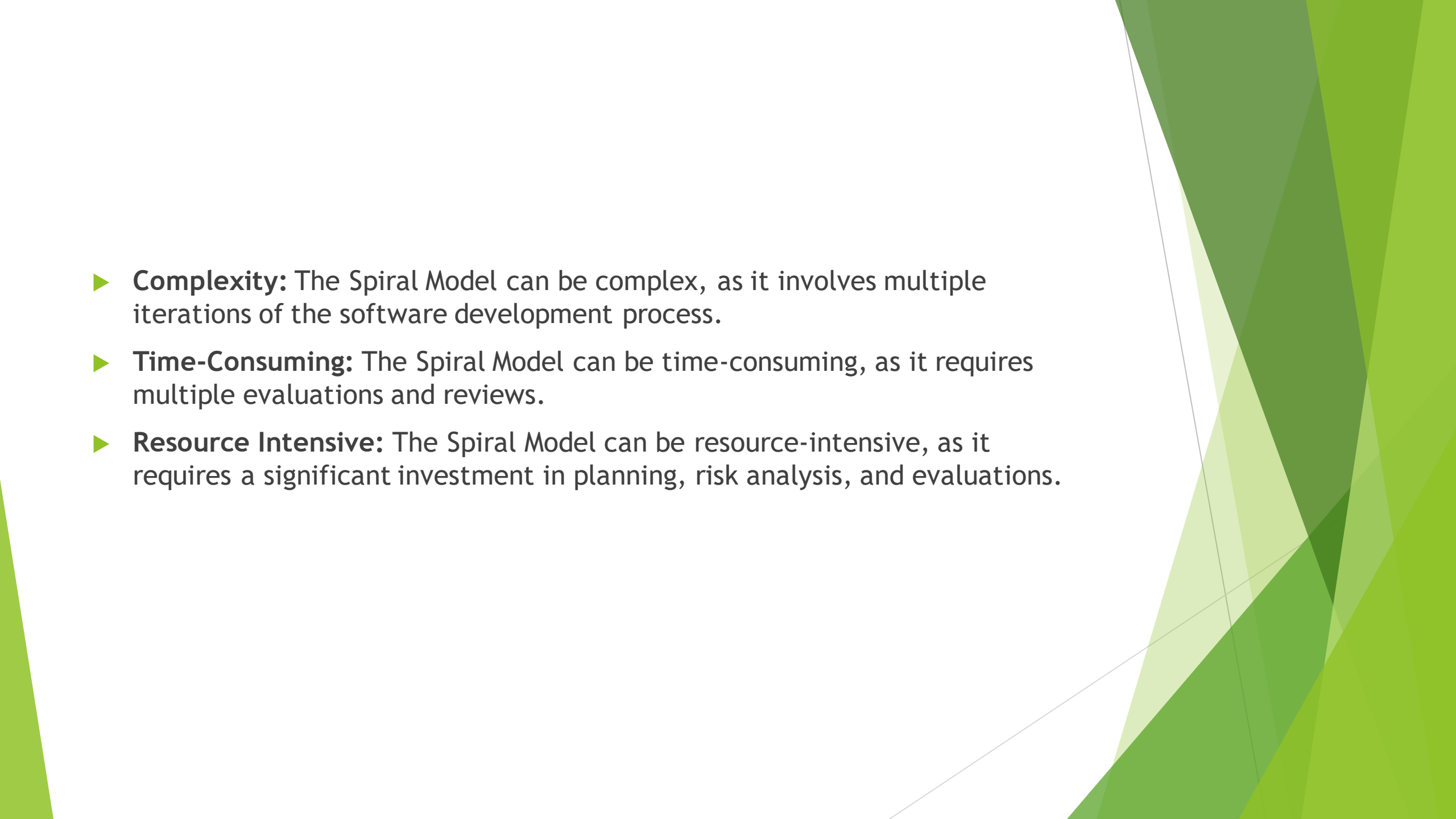
Advantages of the Spiral Model

- ▶ **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
- ▶ **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.
- ▶ **Customer Satisfaction:** Customers can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

- ▶ **Iterative and Incremental Approach:** The Spiral Model provides an iterative and incremental approach to software development, allowing for flexibility and adaptability in response to changing requirements or unexpected events.
- ▶ **Emphasis on Risk Management:** The Spiral Model places a strong emphasis on risk management, which helps to minimize the impact of uncertainty and risk on the software development process.
- ▶ **Improved Communication:** The Spiral Model provides for regular evaluations and reviews, which can improve communication between the customer and the development team.
- ▶ **Improved Quality:** The Spiral Model allows for multiple iterations of the software development process, which can result in improved software quality and reliability

Disadvantages of the Spiral Model

- ▶ **Complex:** The Spiral Model is much more complex than other SDLC models.
- ▶ **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
- ▶ **Too much dependability on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.
- ▶ **Difficulty in time management**

- 
- The background of the slide features abstract, overlapping green geometric shapes, primarily triangles and polygons, in various shades of green, creating a modern and dynamic visual effect.
- ▶ **Complexity:** The Spiral Model can be complex, as it involves multiple iterations of the software development process.
 - ▶ **Time-Consuming:** The Spiral Model can be time-consuming, as it requires multiple evaluations and reviews.
 - ▶ **Resource Intensive:** The Spiral Model can be resource-intensive, as it requires a significant investment in planning, risk analysis, and evaluations.

When To Use the Spiral Model?

- ▶ When a project is vast in software engineering, a spiral model is utilized.
- ▶ A spiral approach is utilized when frequent releases are necessary.
- ▶ When it is appropriate to create a prototype
- ▶ When evaluating risks and costs is crucial
- ▶ The spiral approach is beneficial for projects with moderate to high risk.
- ▶ If modifications are possible at any moment

Agile Development Model

- ▶ **Agile process model** refers to a software development approach based on iterative development.
- ▶ Agile methods break tasks into **smaller iterations, or parts** do not directly involve long term planning.
- ▶ The project scope and requirements are laid down at the beginning of the development process.
- ▶ Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

- ▶ Each iteration is considered as a **short time "frame"** in the Agile process model, which typically lasts from one to four weeks.
- ▶ The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements.
- ▶ Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

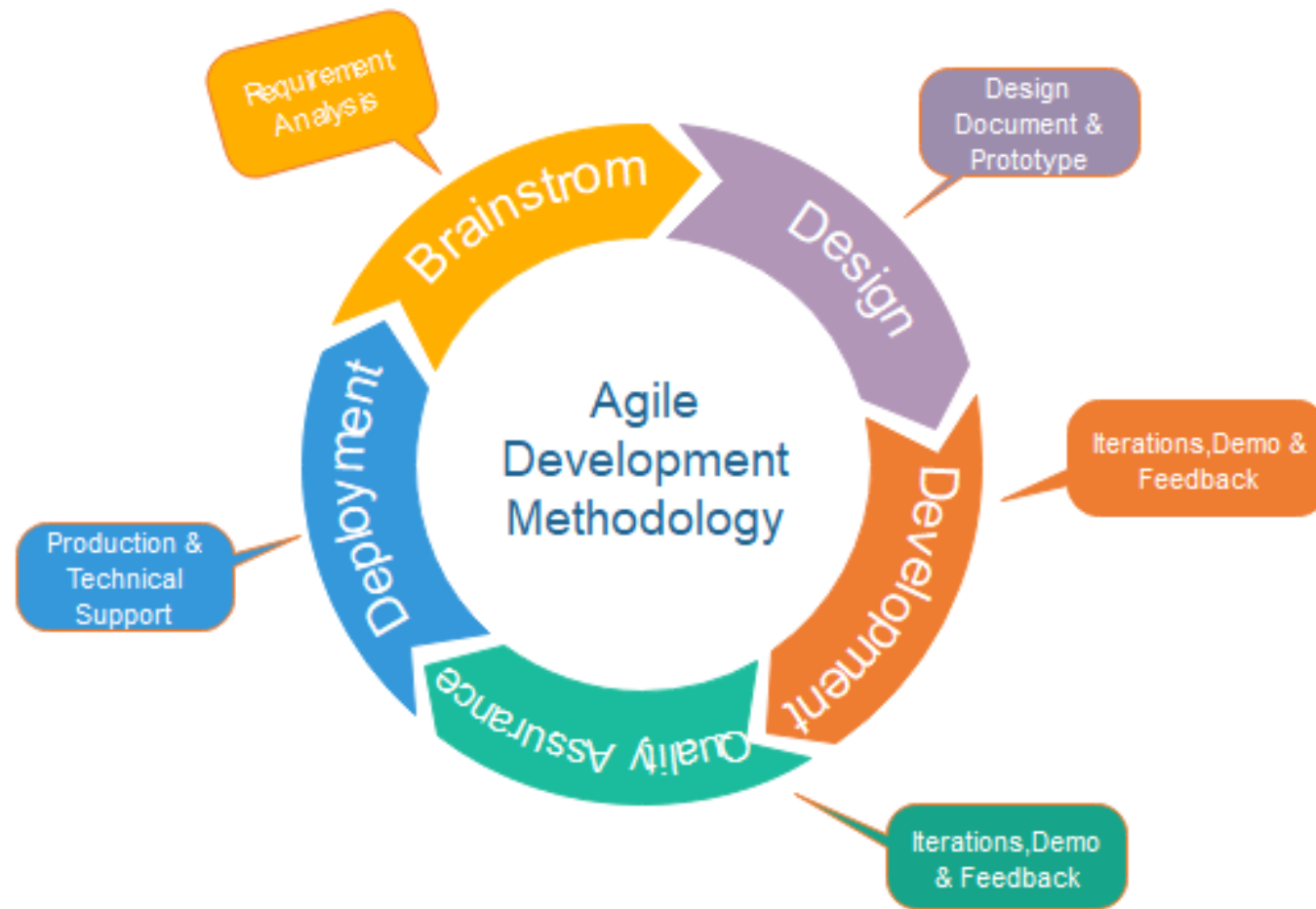


Fig. Agile Model

Phases of Agile Model:

- ▶ Requirements gathering
- ▶ Design the requirements
- ▶ Construction/ iteration
- ▶ Testing/ Quality assurance
- ▶ Deployment
- ▶ Feedback

- ▶ **Requirements gathering:** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility
- ▶ **Design the requirements:** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

- ▶ **Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionalit
- ▶ **Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug
- ▶ **Deployment:** In this phase, the team issues a product for the user's work environment.
- ▶ **Feedback:** After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

Principles of Agile method

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental delivery	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

Agile process model techniques

- ▶ 1. **Extreme Programming (XP)**
- ▶ 2. Adaptive Software Development
- ▶ 3. **Scrum**
- ▶ 4. Dynamic System Development Model
- ▶ 5. Feature Driven Development
- ▶ 6. Crystal
- ▶ 7. Web Engineering

Extreme Programming (XP)

- ▶ XP is a lightweight, efficient, low-risk, flexible, predictable and scientific way to develop a software.
- ▶ eXtreme Programming (XP) was conceived and developed to address the specific needs of software development by small teams in the face of vague and changing requirements

- Extreme programming is an object-oriented development approach and provides four framework activities:

- 1 planning

- 2 design

- 3 coding

- 4 testing.


XP- Planning

- ▶ Planning : begins by creating **user stories**, which are similar to use cases. **User stories relate how the software will be used, and what functionality it will provide.** The customer **then prioritizes** these stories. The development team, in turn, determines the amount of **development time** required to develop the story.
- ▶ The stories are grouped to form **deliverables**. These are the deliverables that will be given to the customer at each increment. Each deliverable is given a delivery date.

- Importantly, the **project velocity** is determined at the end of the first increment: this is essentially the time take to develop the number of stories that were delivered in the first increment. This can be used to better estimate the delivery times for the remaining increments.

XP-Design

- ▶ The design activity in the extreme programming process focuses around **class-responsibility-collaborator (CRC) cards** which the developers use to organize the classes that need to be implemented in the software. These cards are the only design documentation produced using this process model.
- ▶ When the appropriate design is difficult to decide upon, a prototyping method is employed. The design is quickly prototyped to evaluate the risks associated with it, and to estimate the required time needed to implement the story. This prototyping is called a spike-solution.

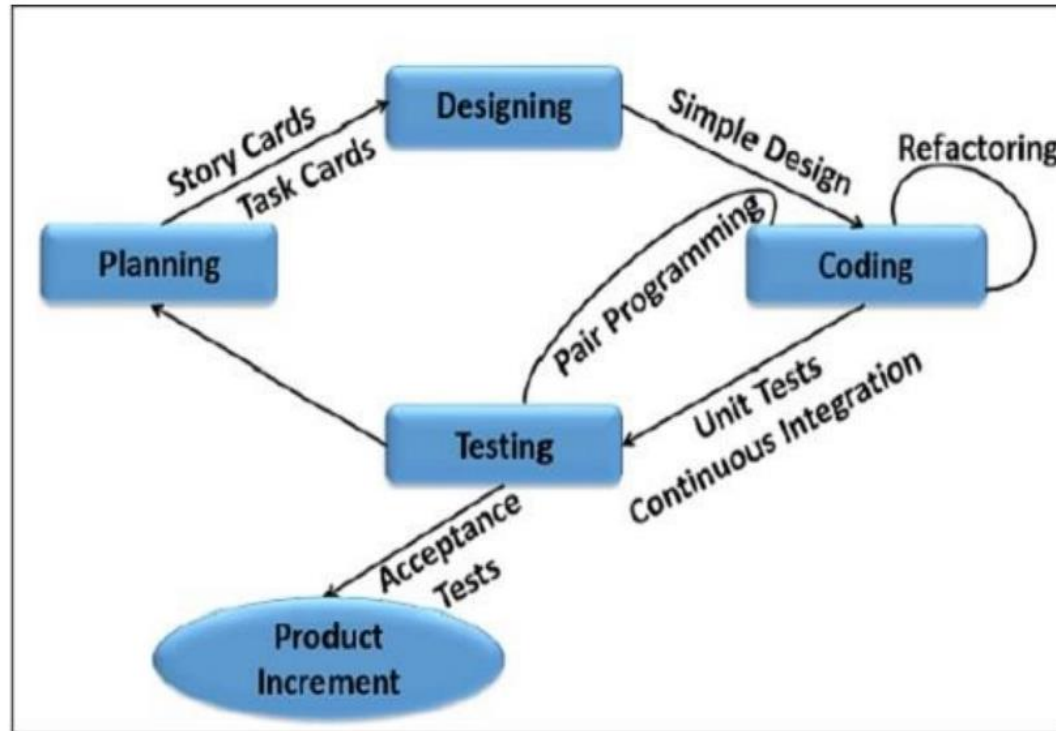
- 
- ▶ Extreme programming encourages developers to reorganize the internal structure of the code (without altering the program's behavior) in order to make developing the software easier, and to make it less likely that bugs will be introduced in future work.
 - ▶ This process of reorganization is called **refactoring**.

XP-CODING

- ▶ Extreme programming has a number of distinctive coding practices.
- ▶ First, before coding begins, the process model recommends developing unit test cases to test each of the stories being developed ,The developers then code towards satisfying those unit tests.
- ▶ • The second distinctive feature is **Pair programming**: all code is written by pairs of programmers, with one developer programming and the other ensuring that the code follows an appropriate coding standard.

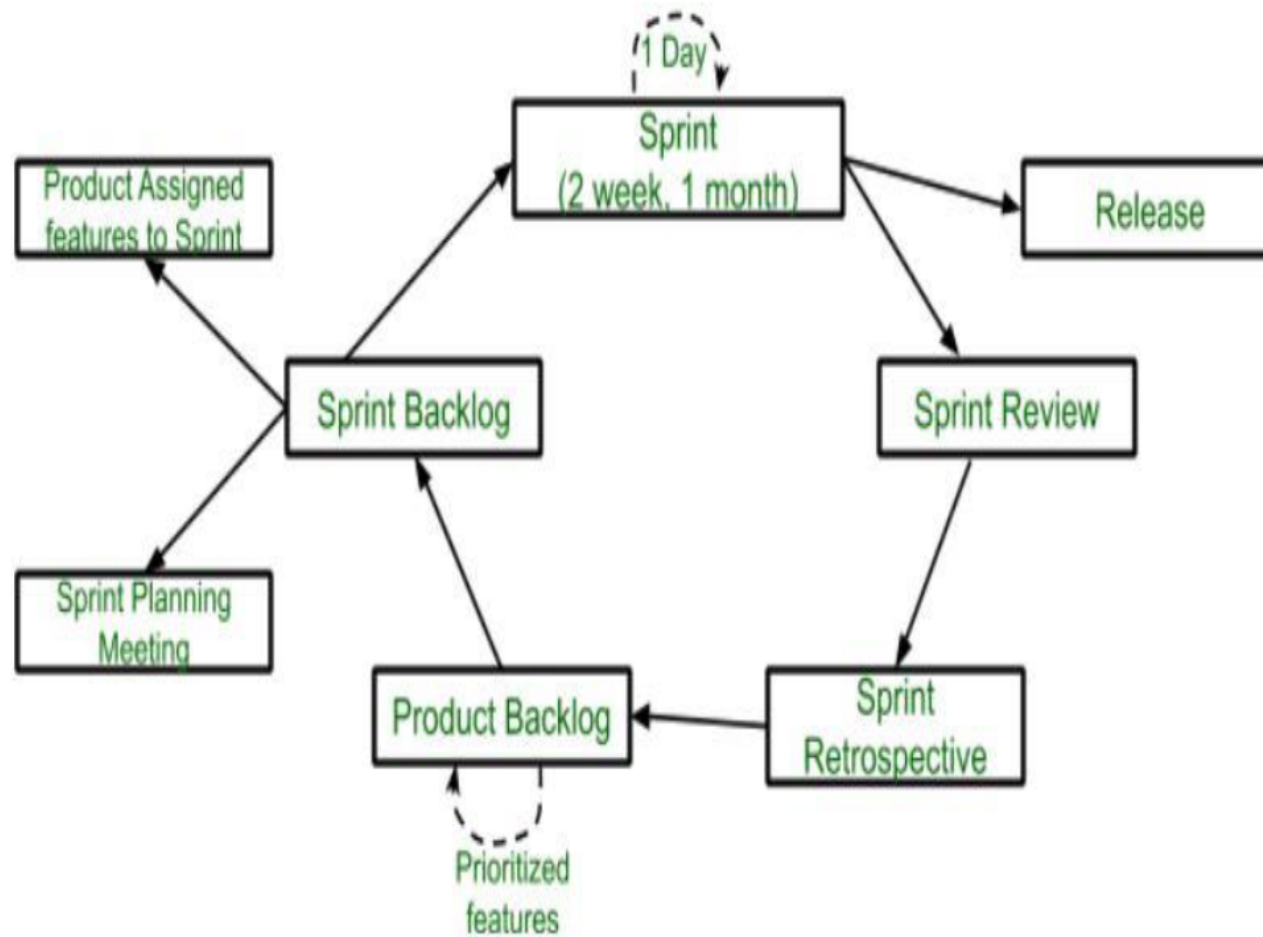
XP-TESTING

- ▶ Unit tests are written before coding begins. It is also recommended that a framework is in place to run all the unit tests, allowing them to be easily and repeatedly run
- ▶ Apart from unit tests, acceptance tests are tests defined by the customer. They focus, by their nature, on the functionality visible to the customer, and will ultimately derive themselves from the stories used to develop the software.



Scrum

- ▶ **Scrum** is the type of **light weighted Agile framework**. It is a framework within which people can address complex adaptive problem while productivity and creativity of delivering product is at highest possible values.
- ▶ Scrum uses **Iterative and incremental developmental model**
- ▶ Scrum encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve.



Terminology

- ▶ **Sprint:** A Sprint is a time-box of one month or less. A new Sprint starts immediately after the completion of the previous Sprint.
- ▶ **Release:** When the product is completed then it goes to the Release stage.
- ▶ **Sprint Review:** If the product still have some non-achievable features then it will be checked in this stage and then the product is passed to the Sprint Retrospective stage.

- ▶ **Sprint Retrospective:** In this stage quality or status of the product is checked.
- ▶ **Product Backlog:** According to the prioritize features the product is organized.
- ▶ **Sprint Backlog:** Sprint Backlog is divided into two parts Product assigned features to sprint and Sprint planning meeting.

Scrum -Advantages

- ▶ Scrum can help teams carry out project deliveries in a **fast and effective** way
- ▶ Scrum makes sure **that money and time are used efficiently**
- ▶ **Large and complex projects** can be separated into practically manageable parts
- ▶ Improvements analyzed during the sprint review
- ▶ Scrum works well for **dynamic and fast-moving project** improvement
- ▶ Scrum meetings allow the team to have neat visibility

Scrum -Disadvantages

- ▶ Scrum framework do not allow changes into their sprint.(time fixed)
- ▶ It can be difficult for the Scrum to plan, structure and organize a project that lacks a clear definition.
- ▶ The daily Scrum meetings and frequent reviews require substantial resources.
- ▶ It is very challenging to adopt the framework of Scrum in large teams
- ▶ Meetings held every day can sometimes disappoint team members
- ▶ A very huge problem will arise if any team member leaves the project in the middle of the progress.

