UNIT-II

Unit II

Perspective Models - Agile Process Models – Scrum and Extreme Programming (XP) - Requirements Analysis - Unified Modelling Language – Design Engineering – Test Engineering.

- A software process model is an **abstract representation** of the development process.
- The models specify the stages and order of a process. So, think of this as a representation of the **order of activities** of the process and the **sequence** in which they are performed.
- **A model will define the following:**
- The tasks to be performed
- The input and output of each task
- The pre and post-conditions for each task
- The flow and sequence of each task

- The goal of a software process model is to provide guidance for controlling and coordinating the tasks to achieve the end product and objectives as effectively as possible.

- There are many kinds of process models for meeting different requirements. We refer to these as **SDLC models** (Software Development Life Cycle models).
- The most popular and important SDLC models are as follows:
- Waterfall model
- V model
- Incremental model
- RAD model
- Agile model
- Iterative model
- Prototype model
- Spiral model

Factors in choosing a software process

- **Project requirements**
- Before you choose a model, take some time to go through the project requirements and clarify them alongside your organization’s or team’s expectations.
- Will the user need to specify requirements in detail after each iterative session?
- Will the requirements *change* during the development process?

- **Project size**
- Consider the size of the project you will be working on.
- Larger projects mean bigger teams, so you'll need more extensive and elaborate project management plans.
- **Project complexity**
- Complex projects may not have clear requirements. The requirements may change often, and the cost of delay is high. Ask yourself if the project requires constant monitoring or feedback from the client.

- **Cost of delay**
- Is the project highly time-bound with a huge cost of delay, or are the timelines flexible?
- **Customer involvement**
- Do you need to consult the customers during the process? Does the user need to participate in all phases?

- **Familiarity with technology**
- This involves the developers’ knowledge and experience with the project domain, software tools, language, and methods needed for development.
- **Project resources**
- This involves the amount and availability of funds, staff, and other resources.

Prescriptive Process Model

- What is it?
- Process models define a distinct set of activities, actions, tasks, milestones and work products that are required to engineer high-quality software.
- Who does it?
- Software engineers and their managers adopt a process model to their needs and then follow it.
- Why is it important?
- Because it provides stability, control and organization to an activity

Why the model is called prescriptive?

- The model prescribes a set of process elements – software engineering actions, tasks, work products, quality assurance, and change control mechanisms for each project.
- Each model prescribes a workflow.
- Though generic process framework is the major idea behind, each model applies a different style to those activities and defines a workflow.
- Communication -> Planning -> Modeling -> Construction -> Deployment

- **There are three types of prescriptive process models. They are:**

  1. The Waterfall Model
  2. Incremental Process model
  3. RAD model

The Waterfall Model

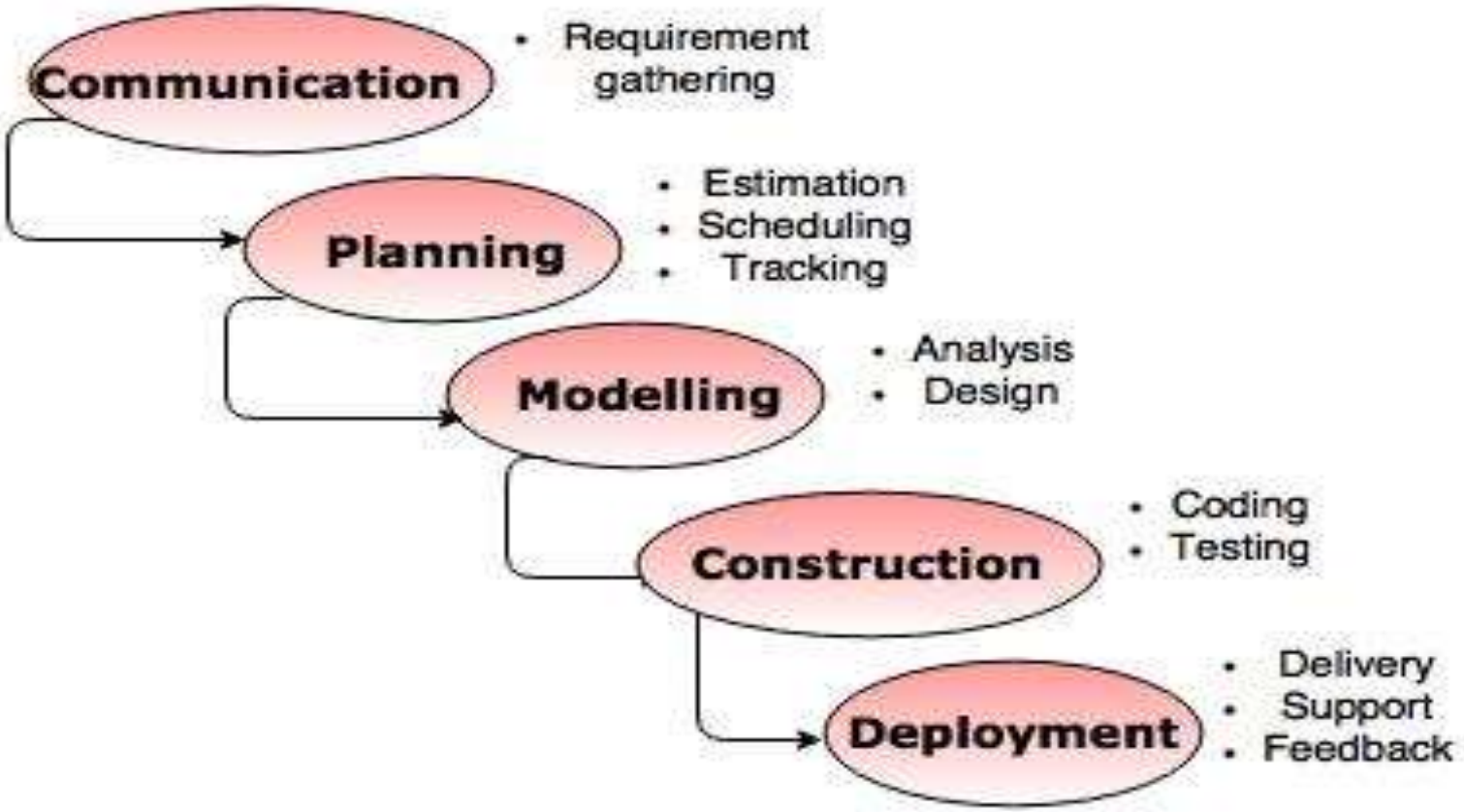


Fig. - The Waterfall model

- The waterfall model is also called as **'Linear sequential model'** or **'Classic life cycle model'.**
- In this model, each phase is fully completed before the beginning of the next phase.
- This model is used for the small projects.
- In this model, feedback is taken after each phase to ensure that the project is on the right path.
- Testing part starts only after the development is complete.

- **Advantages of waterfall model**
- The waterfall model is simple and easy to understand, implement, and use.
- All the requirements are known at the beginning of the project, hence it is easy to manage.
- It avoids overlapping of phases because each phase is completed at once.
- This model works for small projects because the requirements are understood very well.
- This model is preferred for those projects where the quality is more important as compared to the cost of the project.

- **Disadvantages of the waterfall model**
- This model is not good for complex and object oriented projects.
- It is a poor model for long projects.
- The problems with this model are uncovered, until the software testing.
- The amount of risk is high.

Incremental Process model

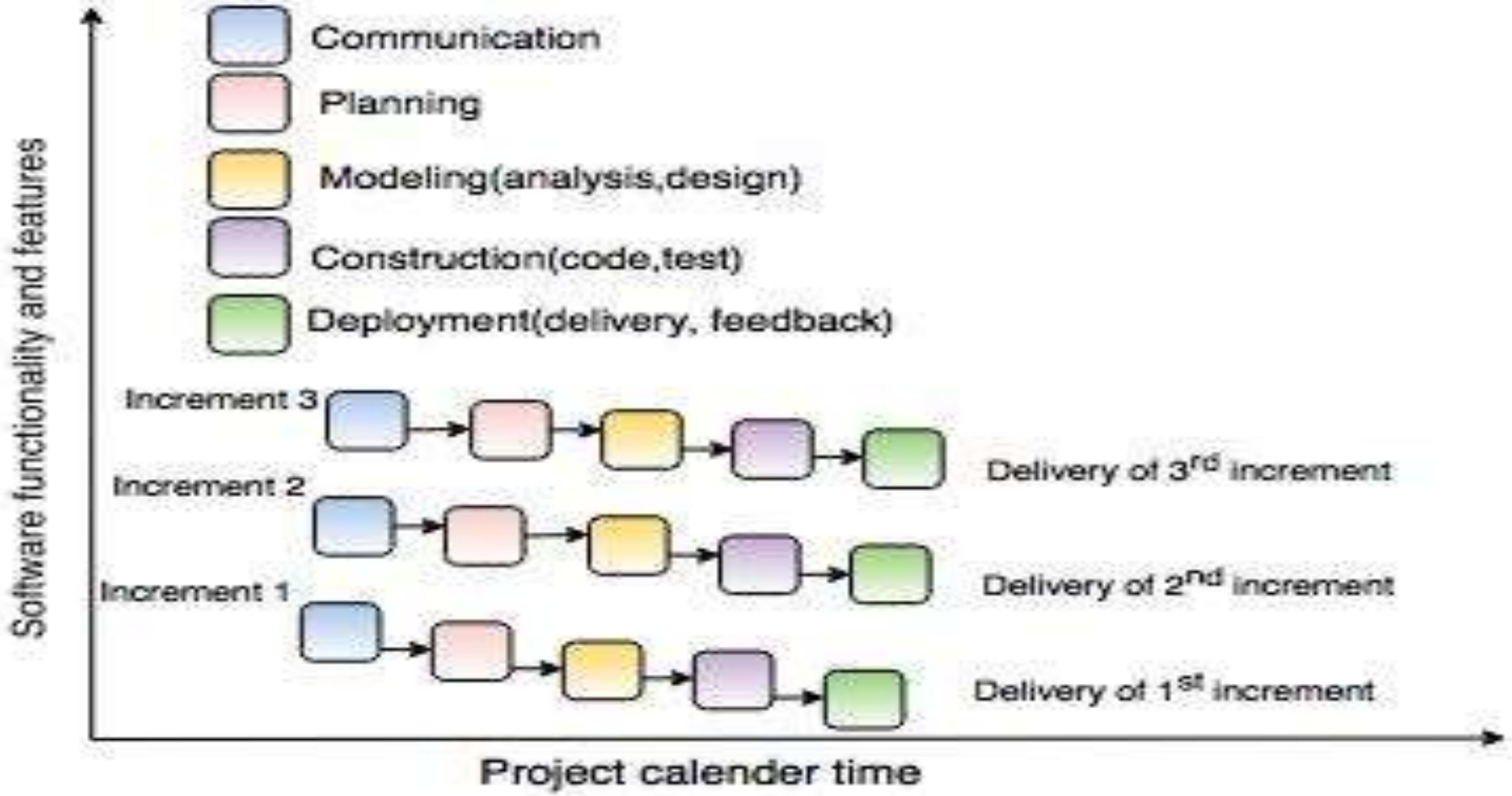


Fig. - Incremental Process Model

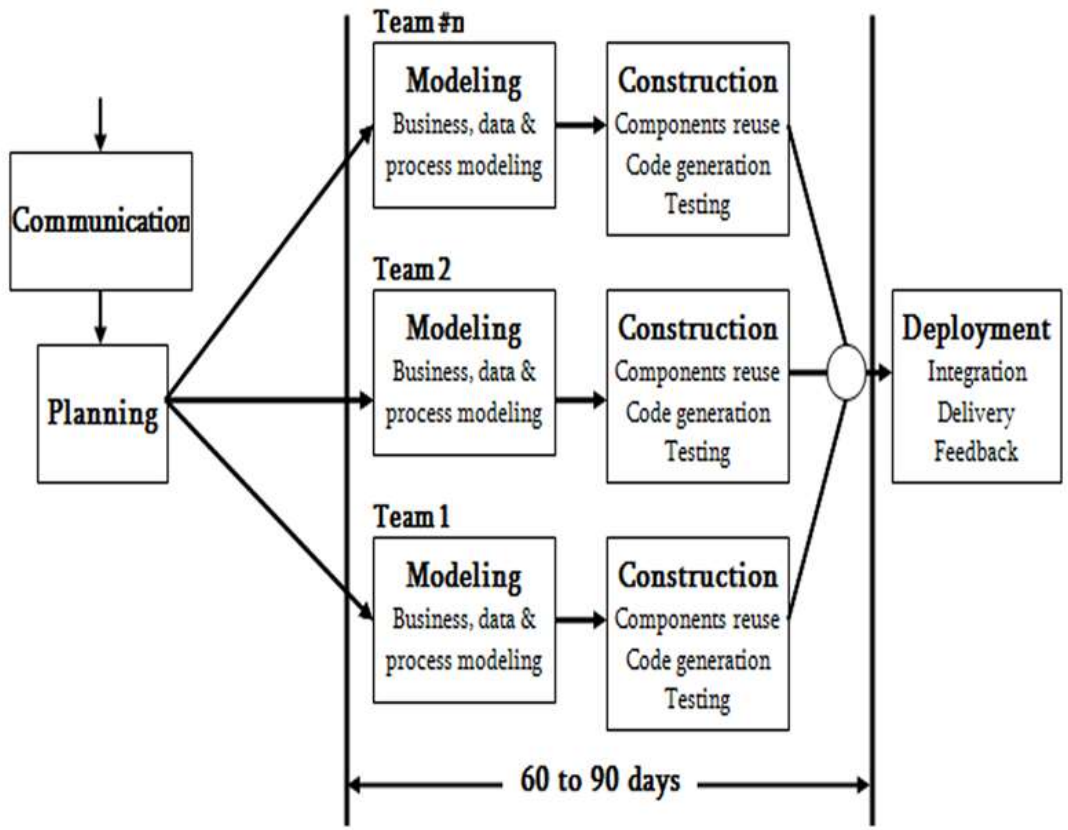
- The incremental model combines the elements of waterfall model and they are applied in an iterative fashion.
- The first increment in this model is generally a core product.
- Each increment builds the product and submits it to the customer for any suggested modifications.
- The next increment implements on the customer's suggestions and add additional requirements in the previous increment.
- This process is repeated until the product is finished.
- **For example,** the word-processing software is developed using the incremental model.

- **Advantages of incremental model**
- This model is flexible because the cost of development is low and initial product delivery is faster.
- It is easier to test and debug during the smaller iteration.
- The working software generates quickly and early during the software life cycle.
- The customers can respond to its functionalities after every increment.

- **Disadvantages of the incremental model**
- The cost of the final product may cross the cost estimated initially.
- This model requires a very clear and complete planning.
- The planning of design is required before the whole system is broken into small increments.
- The demands of customer for the additional functionalities after every increment causes problem during the system architecture.

The RAD Model

- It is an incremental process model that emphasizes a short development cycle.
- It is a high speed adoption of the waterfall model, in which rapid software development is achieved by using a component based development.
- If customer requirements are full understood and project scope is constrained, the RAD process enables a development team to create a “fully functional system” within a very short period of time. (Ex:- 60 to 90 days)


Team #n
Modeling
Business, data & process modeling
Construction
Components reuse
Code generation
Testing
Communication
Team 2
Modeling
Business, data & process modeling
Construction
Components reuse
Code generation
Testing
Deployment
Integration
Delivery
Feedback
Planning
Team 1
Modeling
Business, data & process modeling
Construction
Components reuse
Code generation
Testing
60 to 90 days

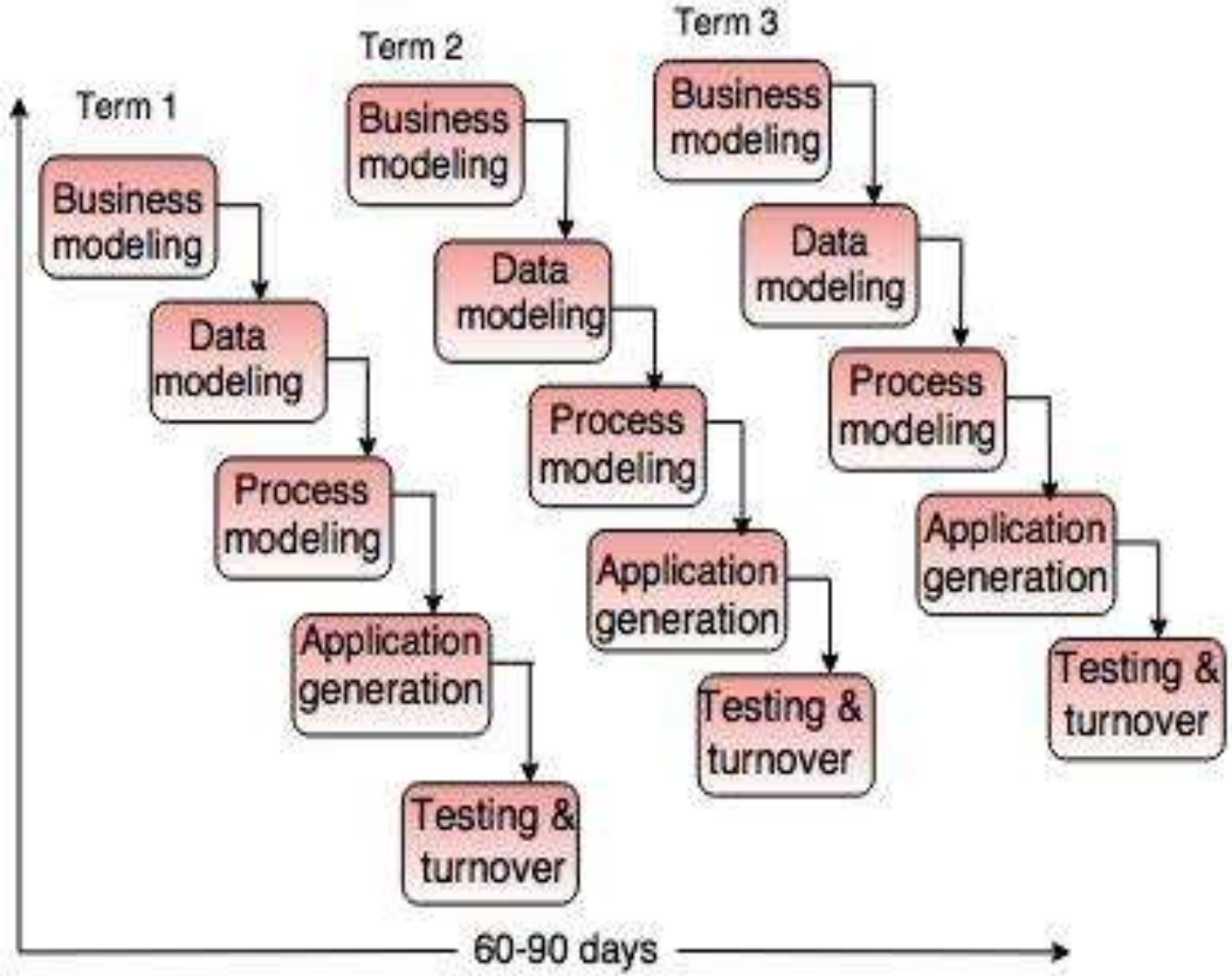


Fig. - RAD Model

The RAD Model

- Like other process models, RAD model maps into process framework activities.
- 1. Communication & Planning: helps to understand business problem and information characteristics that the software must accommodate.
- 2. Modeling: It encompasses three major phases:
- 1. Business modeling
- 2. Data modelling
- 3. Process modeling
- 3. Construction: emphasizes the use of pre-developed software components and automatic code generation.
- 4. Deployment: initiates subsequent iteration

The RAD model consist of following phases:

- **1. Business Modeling**Business modeling consist of the flow of information between various functions in the project.
- For example what type of information is produced by every function and which are the functions to handle that information.
- A complete business analysis should be performed to get the essential business information.
- **2. Data modeling**The information in the business modeling phase is refined into the set of objects and it is essential for the business.
- The attributes of each object are identified and define the relationship between objects.
- **3. Process modeling**The data objects defined in the data modeling phase are changed to fulfil the information flow to implement the business model.
- The process description is created for adding, modifying, deleting or retrieving a data object.

- **4. Application generation**
- In the application generation phase, the actual system is built.
- To construct the software the automated tools are used.
- **5. Testing and turnover**
- The prototypes are independently tested after each iteration so that the overall testing time is reduced.
- The data flow and the interfaces between all the components are are fully tested. Hence, most of the programming components are already tested.

Five Drawbacks of RAD Model

- 1.For large and scalable projects (continuous development) RAD model requires more human resources to create more number of teams.
- 2. If developers and customers are not committed to the rapid-activities that are necessary to complete the system within a short-time, the entire system will fail.
- 3. If a system cannot be properly modularized, the RAD will be problematic.
- 4. RADmaynotbesuitable if technical risks are high.
- 5. If performance is an issue, and high performance can be achieved through tuning the interfaces among components, the RAD is useless

When to use the RAD Model?

1. **Well-understood Requirements:** When project requirements are stable and transparent, RAD is appropriate.
2. **Time-sensitive Projects:** Suitable for projects that need to be developed and delivered quickly due to tight deadlines.
3. **Small to Medium-Sized Projects:** Better suited for smaller initiatives requiring a controllable number of team members.
4. **High User Involvement:** Fits where ongoing input and interaction from users are essential.
5. **Innovation and Creativity**: Helpful for tasks requiring creative inquiry and innovation.
6. **Prototyping:** It is necessary when developing and improving prototypes is a key component of the development process.
7. **Low technological Complexity:** Suitable for tasks using comparatively straightforward technological specifications.

Objectives of Rapid Application Development Model (RAD)

- 1. Speedy Development
- 2. Adaptability and Flexibility
- 3. Stakeholder Participation
- 4. Improved Interaction
- 5. Improved Quality via Prototyping
- 6. Customer Satisfaction

Advantages of Rapid Application Development Model (RAD)

- The use of reusable components helps to reduce the cycle time of the project.
- Feedback from the customer is available at the initial stages.
- Reduced costs as fewer developers are required.
- The use of powerful development tools results in better quality products in comparatively shorter periods.
- The progress and development of the project can be measured through the various stages.
- It is easier to accommodate changing requirements due to the short iteration time spans.
- Productivity may be quickly boosted with a lower number of employees.

Disadvantages of Rapid application development model (RAD)

- The use of powerful and efficient tools requires highly skilled professionals.
- The absence of reusable components can lead to the failure of the project.
- The team leader must work closely with the developers and customers to close the project on time.
- The systems which cannot be modularized suitably cannot use this model.
- Customer involvement is required throughout the life cycle.
- It is not meant for small-scale projects as in such cases, the cost of using automated tools and techniques may exceed the entire budget of the project.
- Not every application can be used with RAD.

Evolutionary Process Models in Software Engineering

- Evolutionary models are iterative type models.
- They allow to develop more complete versions of the software.
- **Following are the evolutionary process models.**

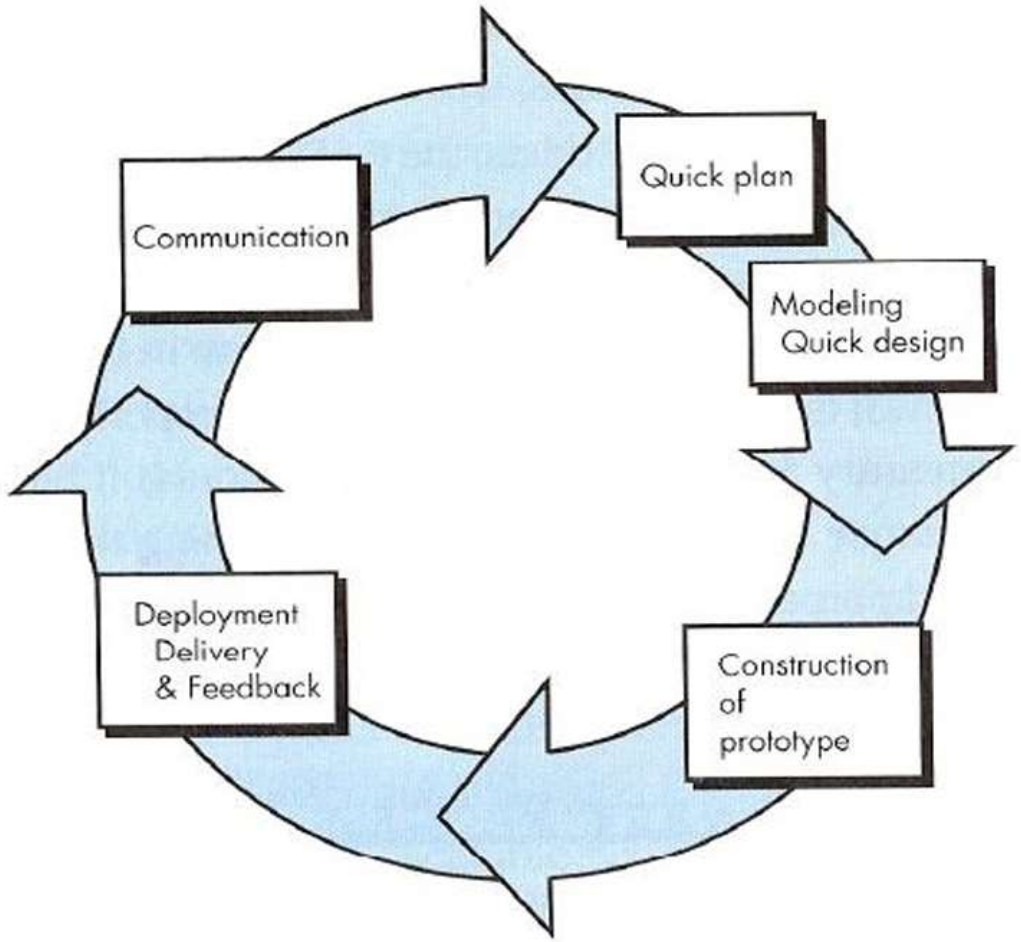
1. The prototyping model
2. The spiral model
3. Concurrent development model

The prototyping model

- Evolutionary process models are iterative.
- They are highly useful to a software engineer to develop more complex version soft wares.
- Three types in evolutionary process model:
- 1. The prototyping model
- 2. The spiral model
- 3. The concurrent development model

- Consider the following two different situations:
- 1. Often, a customer defines a set of general objectives, but does not identify a detailed specifications like input, processing and output requirements. Its a problem in customer side.
- 2. Next is, problems in developer side; a developer may unsure about the efficiency of an algorithm; the adoptability of an operating system; problems in human-computer interaction etc.
- In these non-stabilized cases, the prototyping model may offer a best solution

The prototyping model



- **The different phases of Prototyping model are:**

  1. Communication
  In this phase, developer and customer meet and discuss the overall objectives of the software.

  2. Quick design Quick design is implemented when requirements are known.

- It includes only the important aspects like input and output format of the software.

- It focuses on those aspects which are visible to the user rather than the detailed plan.

- It helps to construct a prototype.

- **3. Modeling quick design**This phase gives the clear idea about the development of software because the software is now built.

- It allows the developer to better understand the exact requirements.

- **4. Construction of prototype**
  The prototype is evaluated by the customer itself.

  5. Deployment, delivery, feedback If the user is not satisfied with current prototype then it refines according to the requirements of the user.

- The process of refining the prototype is repeated until all the requirements of users are met.

- When the users are satisfied with the developed prototype then the system is developed on the basis of final prototype.

Problems in prototyping model

- 1. Customer gets disappointed and lodge more complaints about the product after seeing an initial version of working model, because they see a temporary patch. The customer sees what appears to be a working version of the software, unaware about the prototype. When a developer informed that the product must be rebuilt so that high quality can be maintained, the customer demands that “a few fixes” be applied to make the prototype a working product.
- 2. The developer makes implementation compromises in order to get a working prototype. An inappropriate program / inefficient algorithm / easy tool may be used to demonstrate the capability of working model. Later, the developer become comfortable with these choices and forget all the reasons why they were inappropriate. The less-than-ideal choice has now become an integral part of the system

Advantages of Prototyping Model

- The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.
- New requirements can be easily accommodated as there is scope for refinement.
- Missing functionalities can be easily figured out.
- Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
- The developed prototype can be reused by the developer for more complicated projects in the future.
- Flexibility in design.
- Early feedback from customers and stakeholders can help guide the development process and ensure that the final product meets their needs and expectations.

- Prototyping can be used to test and validate design decisions, allowing for adjustments to be made before significant resources are invested in development.
- Prototyping can help reduce the risk of project failure by identifying potential issues and addressing them early in the process.
- Prototyping can facilitate communication and collaboration among team members and stakeholders, improving overall project efficiency and effectiveness.
- Prototyping can help bridge the gap between technical and non-technical stakeholders by providing a tangible representation of the product.

Disadvantages of the Prototyping Model

- Costly concerning time as well as money.
- There may be too much variation in requirements each time the prototype is evaluated by the customer.
- Poor Documentation due to continuously changing customer requirements.
- It is very difficult for developers to accommodate all the changes demanded by the customer.
- There is uncertainty in determining the number of iterations that would be required before the prototype is finally accepted by the customer.
- After seeing an early prototype, the customers sometimes demand the actual product to be delivered soon.

- Developers in a hurry to build prototypes may end up with sub-optimal solutions.
- The customer might lose interest in the product if he/she is not satisfied with the initial prototype.
- The prototype may not be scalable to meet the future needs of the customer.
- The prototype may not accurately represent the final product due to limited functionality or incomplete features.
- The focus on prototype development may shift away from the final product, leading to delays in the development process.

- The prototype may give a false sense of completion, leading to the premature release of the product.
- The prototype may not consider technical feasibility and scalability issues that can arise during the final product development.
- The prototype may be developed using different tools and technologies, leading to additional training and maintenance costs.
- The prototype may not reflect the actual business requirements of the customer, leading to dissatisfaction with the final product.

Applications of Prototyping Model

- The Prototyping Model should be used when the requirements of the product are not clearly understood or are unstable.
- The prototyping model can also be used if requirements are changing quickly.
- This model can be successfully used for developing user interfaces, high-technology software-intensive systems, and systems with complex algorithms and interfaces.
- The prototyping Model is also a very good choice to demonstrate the technical feasibility of the product.

The Spiral model

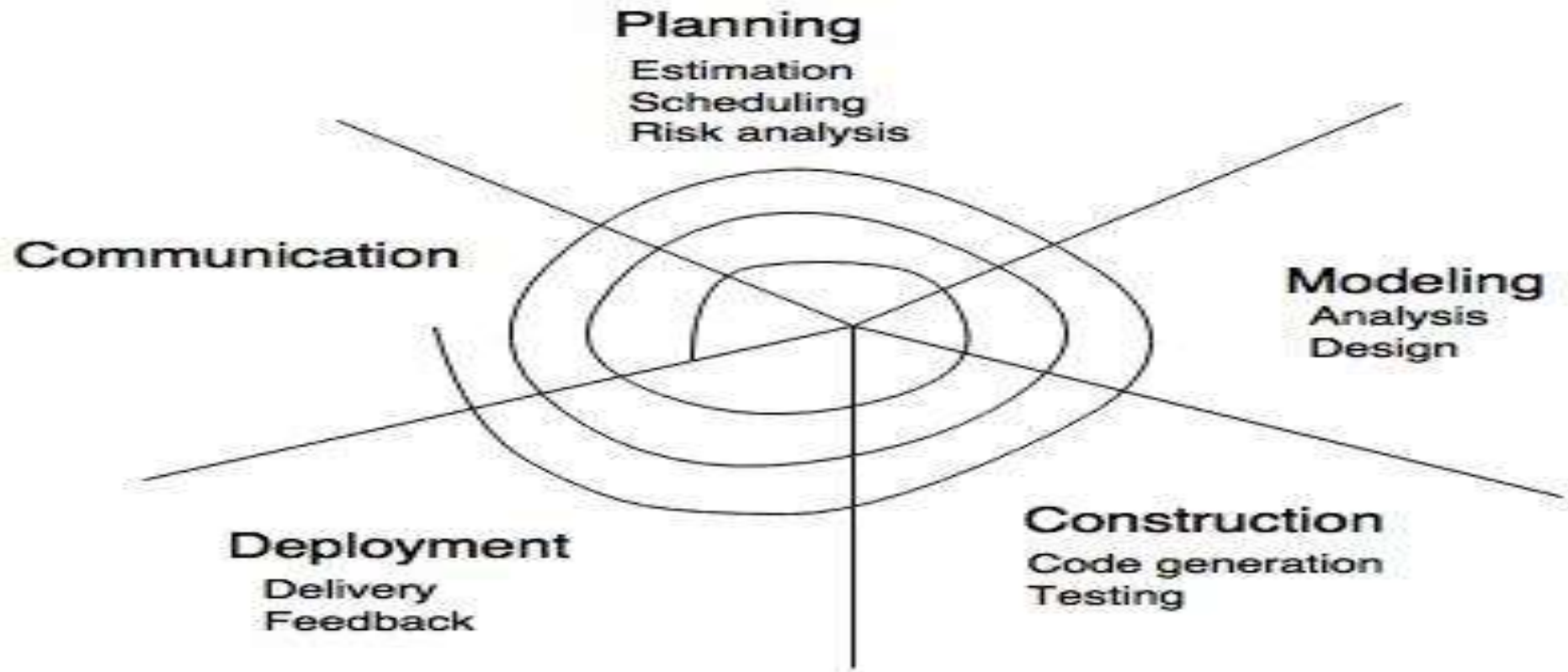


Fig. - The Spiral Model

- 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.

- The first circuit around the spiral might result in the development of a product specification. The subsequent passes around the spiral might be used to develop a prototype and then progressively more mature versions of the software.
- **Planning** is where the objectives, alternatives and other constraints are determined. The alternatives are considered, risks in each alternative are analysed and prototypes are refined in the risk analysis sector. At the development quadrant level risks are known and it proceeds with developing and testing the product. In the assessment sector, customer evaluation of product developed is reviewed and the next phase is planned. This loop continues until acceptable software is built and deployed.
- Hence, the spiral model follows an incremental process methodology and unlike other process models, it deals with the uncertainty by applying a series of risk analysis strategies throughout the process.

- **Advantages of spiral model**
- Reduces risk.
- Recommended for complex projects.
- Changes can be incorporated at a later stage.
- Strong documentation helps in better management.
- **Disadvantages of spiral model**
- Costly and not recommended for small projects.
- Demands risk assessment expertise.
- Looping is a complex process.
- Heavy documentation.

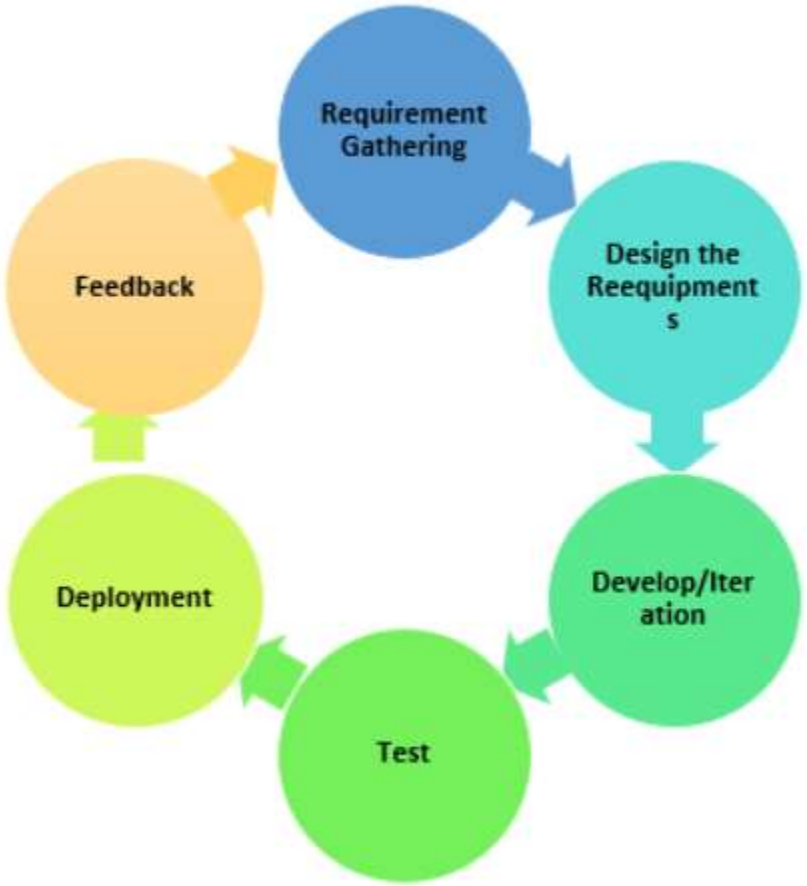
Agile Methodology

- Agile Methodology is a people-focused, results-focused approach to software development that respects our rapidly changing world.
- It’s centered around adaptive planning, self-organization, and short delivery times.
- It’s flexible, fast, and aims for continuous improvements in quality, using tools like *Scrum* and *eXtreme Programming*.

Agile Process Models

- The Agile Model is an incremental and iterative process of software development.
- It defines each iteration's number, duration, and scope in advance. Every iteration is considered a short "frame" in the Agile process model, which mostly lasts from two to four weeks.
- Agile Model divides tasks into time boxes to provide specific functionality for the release.
- Each build is incremental in terms of functionality, with the final build containing all the attributes.
- The division of the entire project into small parts helps minimize the project risk and the overall project delivery time.

Phases of Agile Model



Stages involved in the Agile Model process in the SDLC life cycle:

- **Requirements Gathering:** In this Agile model phase, you must define the requirements. The business opportunities and the time and effort required for the project should also be discussed. By analyzing this information, you can determine a system's economic and technical feasibility.
- **Design the Requirements:** Following the feasibility study, you can work with stakeholders to define requirements. Using the UFD diagram or high-level UML diagram, you can determine how the new system will be incorporated into your existing software system.
- **Develop/Iteration:** The real work begins at this stage after the software development team defines and designs the requirements. Product, design, and development teams start working, and the product will undergo different stages of improvement using simple and minimal functionality.
- **Test:** This phase of the Agile Model involves the testing team. For example, the Quality Assurance team checks the system's performance and reports bugs during this phase.
- **Deployment:** In this phase, the initial product is released to the user.
- **Feedback:** After releasing the product, the last step of the Agile Model is feedback. In this phase, the team receives feedback about the product and works on correcting bugs based on the received feedback.

- When to use the Agile Model?
- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

- Advantage(Pros) of Agile Method:

1.Frequent Delivery

2.Face-to-Face Communication with clients.

3.Efficient design and fulfils the business requirement.

4.Anytime changes are acceptable.

5.It reduces total development time.

- Disadvantages(Cons) of Agile Model:

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

Agile Testing Methods:

- Scrum
- Crystal
- Dynamic Software Development Method(DSDM)
- Feature Driven Development(FDD)
- Lean Software Development
- Extreme Programming (XP)

Benefits of Agile Methodology

- **Faster**. Speed is one of the biggest benefits of Agile Methodology. A faster software development life cycle means less time between paying and getting paid. That, in turn, means a more profitable business.
- **Increased customer satisfaction**. With Agile, customers don’t wait for months or years, only to get exactly what they didn’t want. Instead, they get iterations of something very close to what they want, very fast. The system adjusts quickly to refine the successful customer solution, adapting as it goes to changes in the overall environment.
- **Values employees**. Employees whose ideas are valued are vastly more productive than those who are ordered to follow a set of rules. The Agile Methodology respects employees by giving them the goal, then trusting them to reach it. Since they’re the ones with their hands on the controls and the ones who see the obstacles that crop up every day, employees are in the best position to respond to challenges and meet the goals at hand.
- **Eliminates rework.** By involving the customer at more than just the phases of requirements and delivery, the project remains on-task and in-tune with customer needs at every step. This means less backtracking and less “out on a limb” time between the time we do the work and the time the customer suggests revisions.

Best Practices

- **Set priorities**. A *product backlog* is a list of prioritized tasks maintained by a *product owner.*
- **Maintain small release cycles.** The product should be released in increments every 2-4 weeks, with stakeholders giving feedback before proceeding.
- **Use pair programming.** Two programmers work side-by-side at a single computer. This technique actually results in an identical degree of productivity to separate programming but delivers higher quality.
- **Refactor.** Rework code regularly to achieve the same result with greater efficiency and clarity.
- **Use test-driven development.** Code the unit test first to keep the project on task throughout. Test-driven development as an Agile best practice also produces greater employee engagement, since it transforms testing from a boring grind to a coding challenge.

Scrum

- SCRUM is an agile development method which concentrates specifically on how to manage tasks within a team-based development environment.
- Scrum believes in empowering the development team and advocates working in small teams (say- 7 to 9 members).

- There are three roles in it, and their responsibilities are:
- **Scrum Master:** The scrum can set up the master team, arrange the meeting and remove obstacles for the process
- **Product owner:** The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.
- A Product Owner is part of the scrum team. The key responsibilities of a Product Owner are to define user stories and create a product backlog. The Product Owner is the primary point of contact on behalf of the customer to identify the product requirements for the development team.
- **Scrum Team:** The team manages its work and organizes the work to complete the sprint or cycle.

- **Product Backlog**
- This is a repository where requirements are tracked with details on the no of requirements(user stories) to be completed for each release.
- It should be maintained and prioritized by Product Owner, and it should be distributed to the scrum team.
- Team can also request for a new requirement addition or modification or deletion

What is the difference between Sprint and scrum?

- Scrum is a framework often used in Agile methodology, and a Sprint is part of Scrum's framework structure.
- Scrum gives meetings, tools, and roles, while a Sprint is a defined period for creating a feature.


Scrum Master
Scrum Meeting
Product Owner
Prepare product backlog
Team organizes tasks

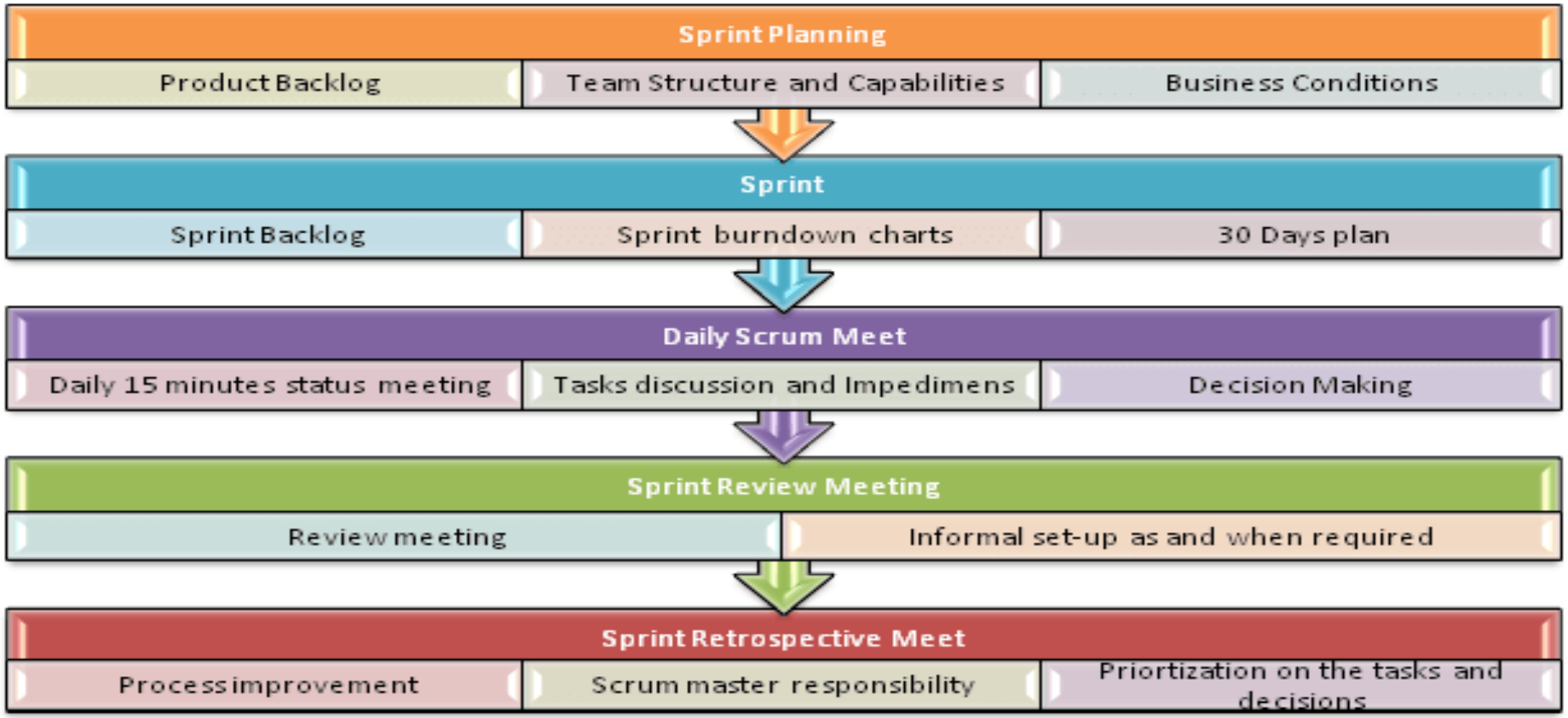
SCRUM PROCESS

- **Scrum** is a hands-on system consisting of simple interlocking steps and components:
- A product owner makes a prioritized wish list known as a product backlog.
- The *scrum team* takes one small piece of the top of the wish list called a *sprint backlog* and plans to implement it.
- The team completes their sprint backlog task in a *sprint* (a 2-4 week period). They assess progress in a meeting called a *daily scrum.*
- The *ScrumMaster* keeps the team focused on the goal.
- At the sprint's end, the work is ready to ship or show. The team closes the sprint with a review, then starts a new sprint.

- **How Does Scrum Work?**
- The work in scrum occurs in events:
- **The Sprint**: the short cycle in which every work occurs. This can take up to a month, with preference given to the shortest length.
- **Sprint Planning**: the event that starts the Sprint. This consists of a meeting where the team decides how to add value to the customers, the amount of work that can be done, and how it will be done.
- **Daily Scrum**: a fifteen-minute meeting at the start of each day in which the developers talk about progress toward the goal of the sprint, making adjustments as necessary.
- **Sprint Review**: a meeting that occurs at the end of each Sprint in which the team presents to stakeholders the results accomplished during the Sprint.
- **Sprint Retrospective**: represents the end of the Sprint. It's an opportunity for the team to reflect upon the Sprint and how well it went and decide which changes could be made to improve the team's work.

- Here’s an example of how Scrum works:
- Bill meets with a customer to discuss her company’s needs. Those needs are the product backlog.
- Bill chooses the most important tasks to work on in the next two weeks.
- His team meets in a daily scrum to target work for the day ahead and address roadblocks.
- At the end of the sprint, Bill delivers the work, reviews the backlog, and sets the goal for the next sprint.
- The cycle repeats until the software is complete.

Scrum Practices



Process flow of Scrum Methodologies:

- Each iteration of a scrum is known as Sprint
- Product backlog is a list where all details are entered to get the end-product
- During each Sprint, top user stories of Product backlog are selected and turned into Sprint backlog
- Team works on the defined sprint backlog
- Team checks for the daily work
- At the end of the sprint, team delivers product functionality

Extreme Programming (XP)

- This type of methodology is used when customers are constantly changing demands or requirements, or when they are not sure about the system's performance.
- It advocates frequent “releases” of the product in short development cycles, which inherently improves the productivity of the system and also introduces a checkpoint where any customer requirements can be easily implemented.
- The XP develops software keeping customer in the target.

- Business requirements are gathered in terms of stories. All those stories are stored in a place called the parking lot.
- In this type of methodology, releases are based on the shorter cycles called Iterations with span of 14 days time period.
- Each iteration includes phases like coding, unit testing and system testing where at each phase some minor or major functionality will be built in the application.

- ***Phases of Extreme Programming (XP) :***
- There are 6 phases available in Agile XP method, and those are explained as follows:
- ***Planning***
- Identification of stakeholders and sponsors
- Infrastructure Requirements
- Security related information and gathering
- Service Level Agreements and its conditions
- ***Analysis***
- Capturing of Stories in Parking lot
- Prioritize stories in Parking lot
- Scrubbing of stories for estimation
- Define Iteration SPAN(Time)
- Resource planning for both Development and QA teams

- ***Design***
- Break down of tasks
- Test Scenario preparation for each task
- Regression Automation Framework
- ***Execution***
- Coding
- Unit Testing
- Execution of Manual test scenarios
- Defect Report generation
- Conversion of Manual to Automation regression test cases
- Mid Iteration review
- End of Iteration review

- ***Wrapping***
- Small Releases
- Regression Testing
- Demos and reviews
- Develop new stories based on the need
- Process Improvements based on end of iteration review comments
- ***Closure***
- Pilot Launch
- Training
- Production Launch
- SLA Guarantee assurance
- Review SOA strategy
- Production Support

- There are two storyboards available to track the work on a daily basis, and those are listed below for reference.
- Story Cardboard
  - This is a traditional way of collecting all the stories in a board in the form of stick notes to track daily XP activities. As this manual activity involves more effort and time, it is better to switch to an online form.
- Online Storyboard
  - Online tool Storyboard can be used to store the stories. **Several teams can use it** for different purposes.

Agile Process Models

- **The Agile Model** was primarily designed to help a project adapt quickly to change requests.
- So, the main aim of the Agile model is to facilitate quick project completion.
- To accomplish this task, agility is required. Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project.
- The Agile Model refers to a group of development processes.

Agile SDLC Models/Methods

- Scrum
- Extreme Programming (XP)
- Lean Development
- Unified Process

Steps in the Agile Model

- The agile model is a combination of iterative and incremental process models. The steps involve in agile SDLC models are:
- Requirement gathering
- Design the Requirements
- Construction / Iteration
- Testing / Quality Assurance
- Deployment
- Feedback

1. **Requirement Gathering:-** In this step, the development team must gather the requirements, by interaction with the customer. development team should plan the time and effort needed to build the project. Based on this information you can evaluate technical and economical feasibility.

2. **Design the Requirements:-** In this step, the development team will use user-flow-diagram or high-level UML diagrams to show the working of the new features and show how they will apply to the existing software. Wireframing and designing user interfaces are done in this phase.

3. **Construction / Iteration:-** In this step, development team members start working on their project, which aims to deploy a working product.

4.Testing / Quality Assurance:- Testing involves Unit Testing, Integration Testing, and System Testing. A brief introduction of these three tests is as follows:

Unit Testing:- Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own. Unit testing is used to test individual blocks (units) of code.

Integration Testing:- Integration testing is used to identify and resolve any issues that may arise when different units of the software are combined.

System Testing:- Goal is to ensure that the software meets the requirements of the users and that it works correctly in all possible scenarios.

5. **Deployment:-** In this step, the development team will deploy the working project to end users.

6. **Feedback:-** This is the last step of the **Agile Model.** In this, the team receives feedback about the product and works on correcting bugs based on feedback provided by the customer.

Principles of the Agile Model

- To establish close contact with the customer during development and to gain a clear understanding of various requirements, each Agile project usually includes a customer representative on the team. At the end of each iteration stakeholders and the customer representative review, the progress made and re-evaluate the requirements.
- The agile model relies on working software deployment rather than comprehensive documentation.
- Frequent delivery of incremental versions of the software to the customer representative in intervals of a few weeks.
- Requirement change requests from the customer are encouraged and efficiently incorporated.

- It emphasizes having efficient team members and enhancing communications among them is given more importance. It is realized that improved communication among the development team members can be achieved through face-to-face communication rather than through the exchange of formal documents.

- It is recommended that the development team size should be kept small (5 to 9 people) to help the team members meaningfully engage in face-to-face communication and have a collaborative work environment.

- The agile development process usually deploys Pair Programming. In Pair programming, two programmers work together at one workstation. One does coding while the other reviews the code as it is typed in. The two programmers switch their roles every hour or so.

Characteristics of the Agile Process

- Agile processes must be adaptable to technical and environmental changes. That means if any technological changes occur, then the agile process must accommodate them.
- The development of agile processes must be incremental. That means, in each development, the increment should contain some functionality that can be tested and verified by the customer.
- The customer feedback must be used to create the next increment of the process.
- The software increment must be delivered in a short span of time.
- It must be iterative so that each increment can be evaluated regularly.

When To Use the Agile Model?

- When frequent modifications need to be made, this method is implemented.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with the team all the time.
- when the project needs to be delivered quickly.
- Projects with few regulatory requirements or not certain requirements.
- projects utilizing a less-than-strict current methodology
- Those undertakings where the product proprietor is easily reachable
- Flexible project schedules and budgets.

Advantages of the Agile Model

- Working through Pair programming produces well-written compact programs which have fewer errors as compared to programmers working alone.
- It reduces the total development time of the whole project.
- Agile development emphasizes face-to-face communication among team members, leading to better collaboration and understanding of project goals.
- Customer representatives get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.
- Agile development puts the customer at the center of the development process, ensuring that the end product meets their needs.

Disadvantages of the Agile Model

- The lack of formal documents creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.
- It is not suitable for handling complex dependencies.
- The agile model depends highly on customer interactions so if the customer is not clear, then the development team can be driven in the wrong direction.
- Agile development models often involve working in short sprints, which can make it difficult to plan and forecast project timelines and deliverables. This can lead to delays in the project and can make it difficult to accurately estimate the costs and resources needed for the project.
- Agile development models require a high degree of expertise from team members, as they need to be able to adapt to changing requirements and work in an iterative environment. This can be challenging for teams that are not experienced in agile development practices and can lead to delays and difficulties in the project.
- Due to the absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.

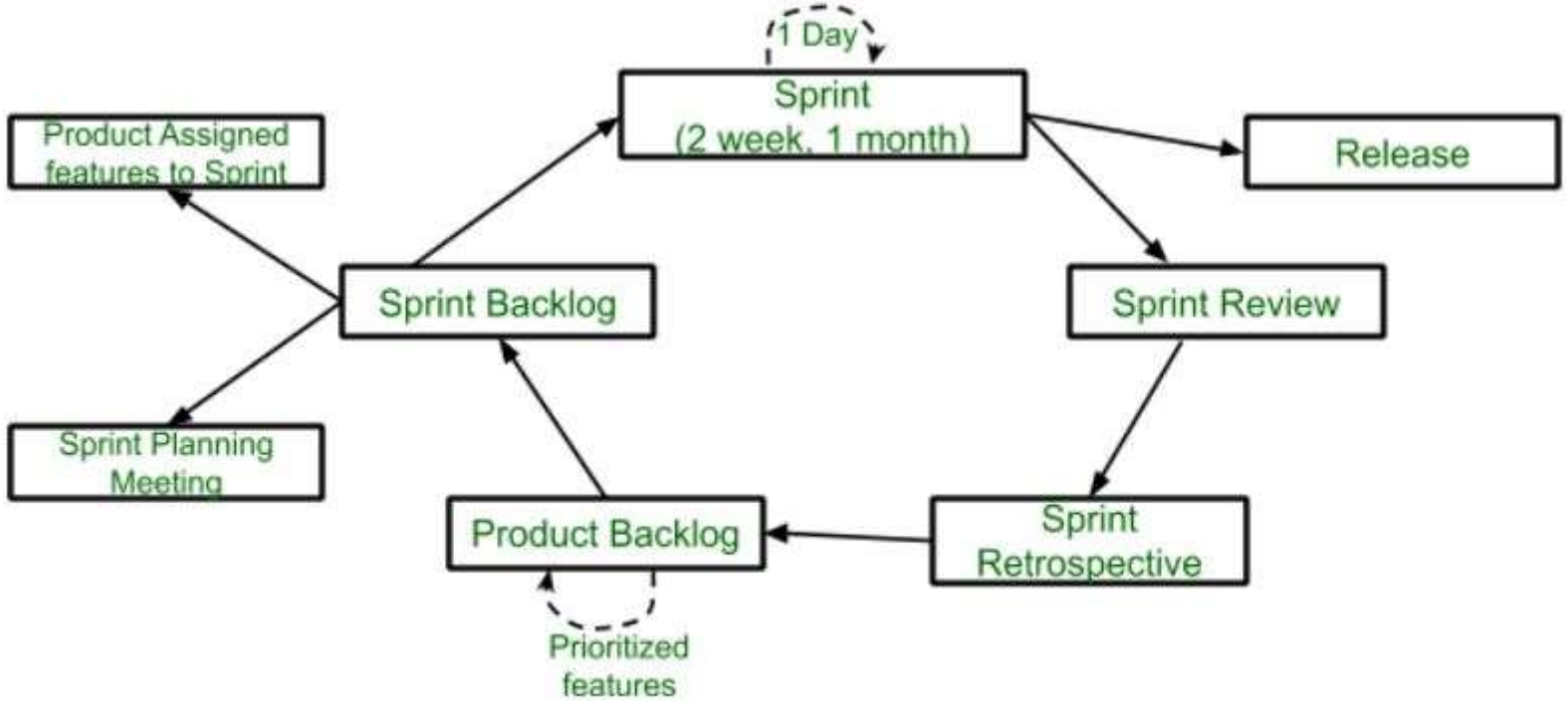
What is a scrum in software development?

- Scrum is a management framework that teams use to self-organize and work towards a common goal.
- Scrum allows us to develop products of the highest value while making sure that we maintain creativity and productivity.
- The iterative and incremental approach used in scrum allows the teams to adapt to the changing requirements.

Silent features of Scrum

- Scrum is a light-weighted framework
- Scrum emphasizes self-organization
- Scrum is simple to understand
- Scrum framework helps the team to work together

Lifecycle of Scrum



- **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, it goes to the Release stage.
- **Sprint Review:** If the product still has some non-achievable features, it will be checked in this stage and then 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.

Advantage of Scrum framework

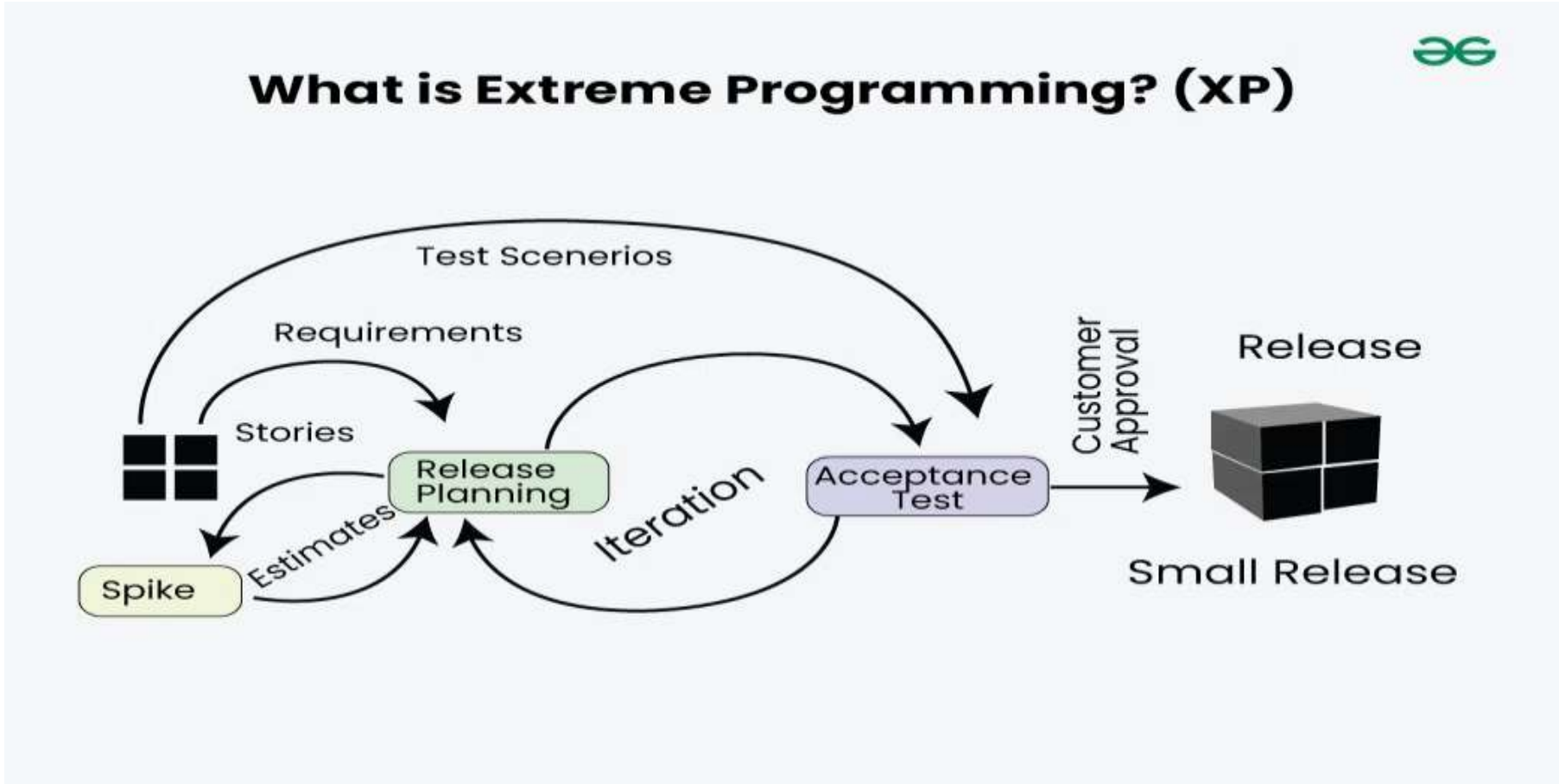
- Scrum framework is fast moving and money efficient.
- Scrum framework works by dividing the large product into small sub-products. It’s like a divide and conquer strategy
- In Scrum customer satisfaction is very important.
- Scrum is adaptive in nature because it have short sprint.
- As Scrum framework rely on constant feedback therefore the quality of product increases in less amount of time

Disadvantage of Scrum framework

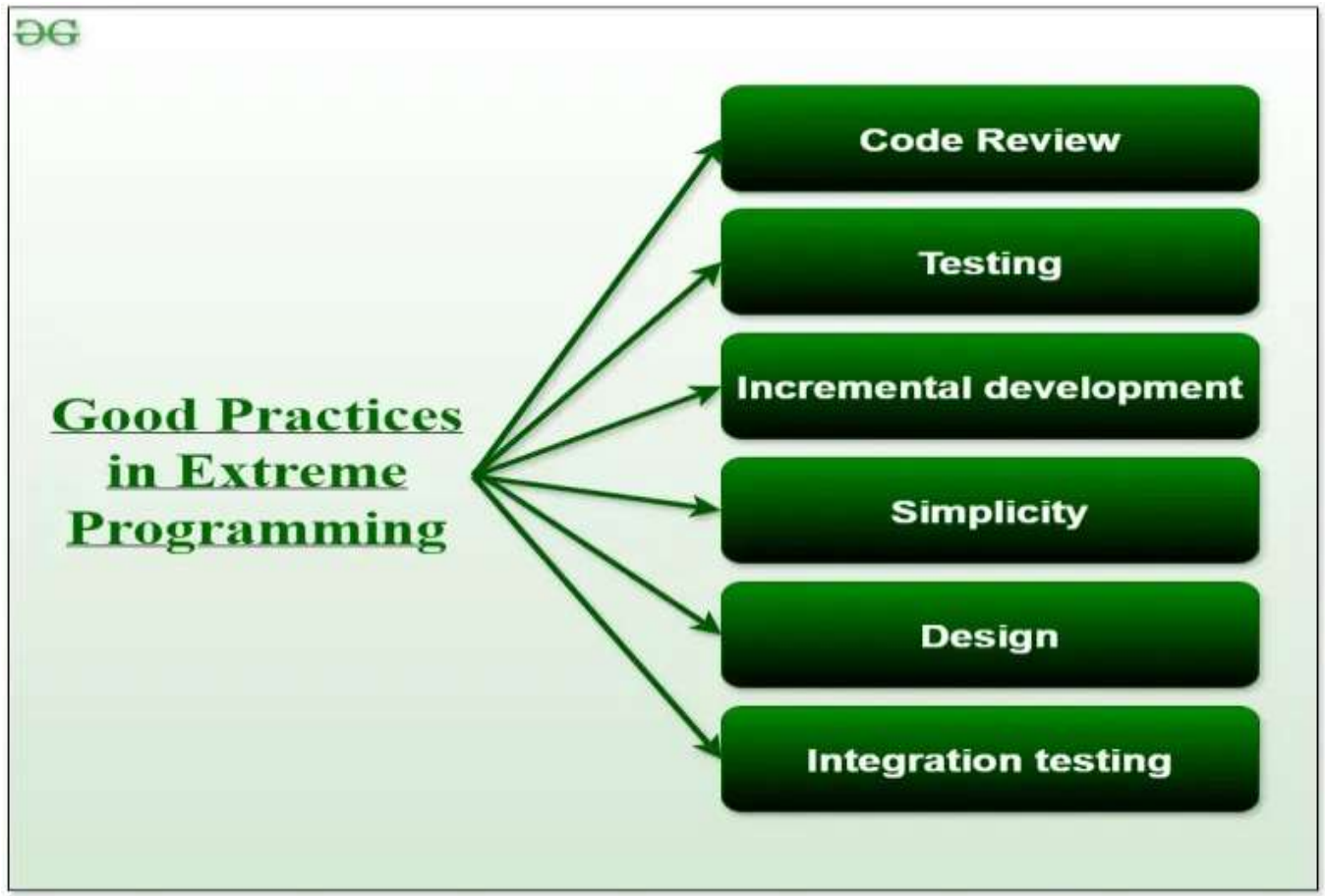
- Scrum framework do not allow changes into their sprint.
- It can be difficult for the Scrum to plan, structure and organize a project that lacks a clear definition.
- The daily Scrum meetings and frequent reviews require substantial resources.

What is Extreme Programming (XP)?

- Extreme Programming (XP) is an Agile software development methodology that focuses on delivering high-quality software through frequent and continuous feedback, collaboration, and adaptation.
- XP emphasizes a close working relationship between the development team, the customer, and stakeholders, with an emphasis on rapid, iterative development and deployment.


What is Extreme Programming? (XP)
Test Scenerios
Requirements
Stories
Release Planning
Iteration
Acceptance Test
Customer Approval
Release
Small Release
Estimates
Spike

- Extreme programming is one of the most popular and well-known approaches in the family of agile methods.
- An XP project starts with user stories which are short descriptions of what scenarios the customers and users would like the system to support.
- Each story is written on a separate card, so they can be flexibly grouped.


Good Practices in Extreme Programming
Code Review
Testing
Incremental development
Simplicity
Design
Integration testing

- **Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their work between them every hour.

- **Testing:** Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach, test cases are written even before any code is written.

- **Incremental development:** Incremental development is very good because customer feedback is gained and based on this development team comes up with new increments every few days after each iteration.

- **Simplicity:** Simplicity makes it easier to develop good-quality code as well as to test and debug it.

- **Design:** Good quality design is important to develop good quality software. So, everybody should design daily.

- **Integration testing:** Integration Testing helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.

Basic Principles of Extreme programming

- XP is based on the frequent iteration through which the developers implement User Stories.
- User stories are simple and informal statements of the customer about the functionalities needed.
- It does not mention finer details such as the different scenarios that can occur.
- Based on User stories, the project team proposes Metaphors.
- Metaphors are a common vision of how the system would work.
- The development team may decide to build a Spike for some features. A Spike is a very simple program that is constructed to explore the suitability of a solution being proposed. It can be considered similar to a prototype.

Basic activities that are followed during software development by using the XP model

- **Coding:** The concept of coding which is used in the XP model is slightly different from traditional coding. Here, the coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system, and choosing among several alternative solutions.

- **Testing:** The XP model gives high importance to testing and considers it to be the primary factor in developing fault-free software.

- **Listening:** The developers need to carefully listen to the customers if they have to develop good quality software. Sometimes programmers may not have the depth knowledge of the system to be developed. So, the programmers should understand properly the functionality of the system and they have to listen to the customers.

- **Designing:** Without a proper design, a system implementation becomes too complex, and very difficult to understand the solution, thus making maintenance expensive. A good design results elimination of complex dependencies within a system. So, effective use of suitable design is emphasized.

- **Feedback:** One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs. Frequent contact with the customer makes the development effective.

- **Simplicity:** The main principle of the XP model is to develop a simple system that will work efficiently in the present time, rather than trying to build something that would take time and may never be used. It focuses on some specific features that are immediately needed, rather than engaging time and effort on speculations of future requirements.

- **Pair Programming:** XP encourages pair programming where two developers work together at the same workstation. This approach helps in knowledge sharing, reduces errors, and improves code quality.

- **Continuous Integration:** In XP, developers integrate their code into a shared repository several times a day. This helps to detect and resolve integration issues early on in the development process.

- **Refactoring:** XP encourages refactoring, which is the process of restructuring existing code to make it more efficient and maintainable. Refactoring helps to keep the codebase clean, organized, and easy to understand.

- **Collective Code Ownership:** In XP, there is no individual ownership of code. Instead, the entire team is responsible for the codebase. This approach ensures that all team members have a sense of ownership and responsibility towards the code.

- **Planning Game:** XP follows a planning game, where the customer and the development team collaborate to prioritize and plan development tasks. This approach helps to ensure that the team is working on the most important features and delivers value to the customer.

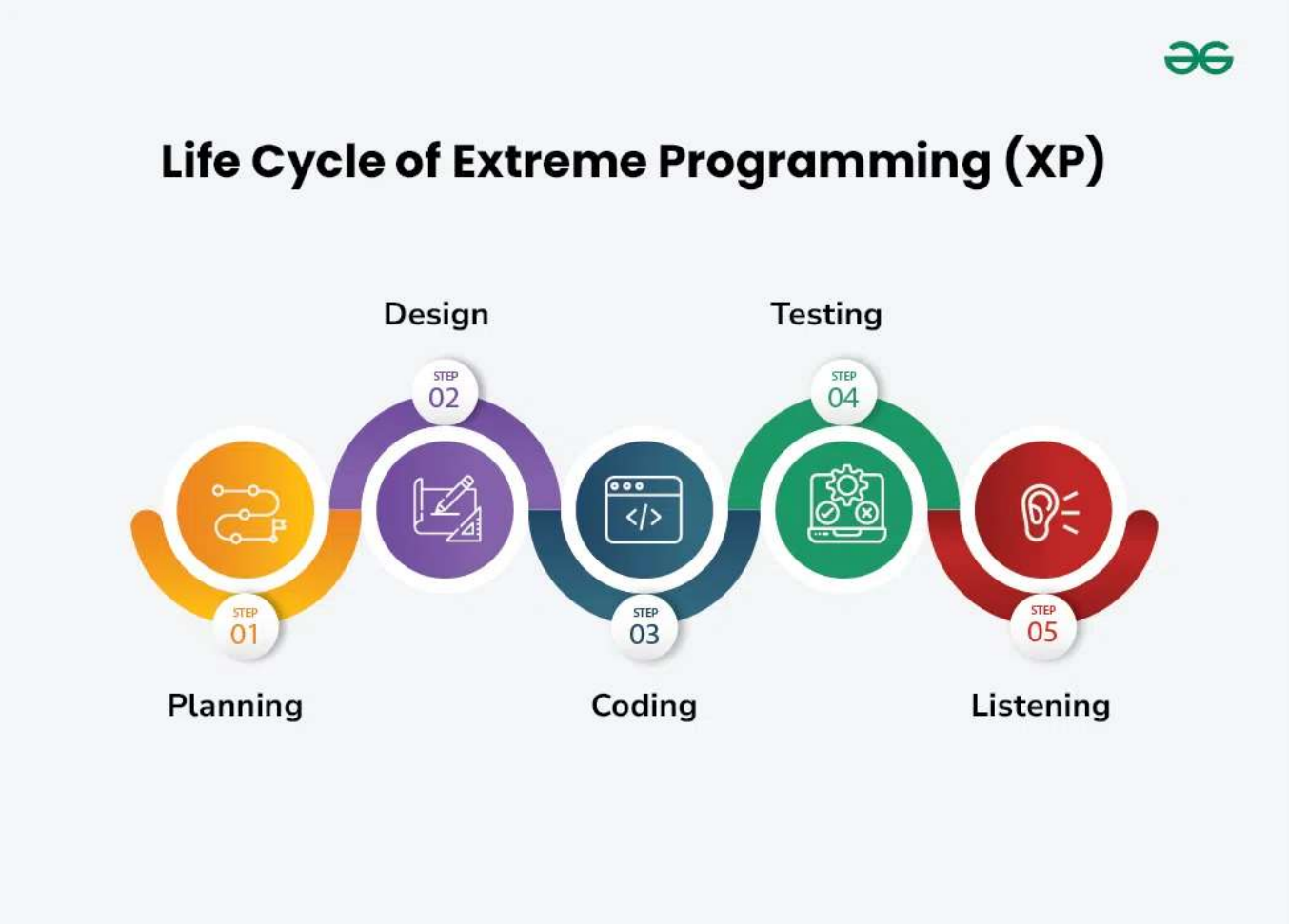
- **On-site Customer:** XP requires an on-site customer who works closely with the development team throughout the project. This approach helps to ensure that the customer's needs are understood and met, and also facilitates communication and feedback.

Applications of Extreme Programming (XP)

- Small projects
- Projects involving new technology or Research projects
- Web development projects
- Collaborative projects
- Projects with tight deadlines
- Projects with rapidly changing requirements
- Projects where quality is a high priority

Life Cycle of Extreme Programming (XP)

- The Extreme Programming Life Cycle consist of five phases:



1. **Planning:** The first stage of Extreme Programming is planning. During this phase, clients define their needs in concise descriptions known as user stories. The team calculates the effort required for each story and schedules releases according to priority and effort.

2. **Design:** The team creates only the essential design needed for current user stories, using a common analogy or story to help everyone understand the overall system architecture and keep the design straightforward and clear.

3. **Coding:** Extreme Programming (XP) promotes pair programming i.e**.** wo developers work together at one workstation, enhancing code quality and knowledge sharing. They write tests before coding to ensure functionality from the start (TDD), and frequently integrate their code into a shared repository with automated tests to catch issues early.

4. **Testing:** Extreme Programming (XP) gives more importance to testing that consist of both unit tests and acceptance test**.** Unit tests, which are automated, check if specific features work correctly. Acceptance tests, conducted by customers, ensure that the overall system meets initial requirements. This continuous testing ensures the software's quality and alignment with customer needs.

5. **Listening:** In the listening phase regular feedback from customers to ensure the product meets their needs and to adapt to any changes.

Values of Extreme Programming (XP)

1. **Communication:** The essence of communication is for information and ideas to be exchanged amongst development team members so that everyone has an understanding of the system requirements and goals. Extreme Programming (XP) supports this by allowing open and frequent communication between members of a team.

2. **Simplicity:** Keeping things as simple as possible helps reduce complexity and makes it easier to understand and maintain the code.

3. **Feedback:** Feedback loops which are constant are among testing as well as customer involvements which helps in detecting problems earlier during development.

4. **Courage:** Team members are encouraged to take risks, speak up about problems, and adapt to change without fear of repercussions.

5. **Respect**: Every member's input or opinion is appreciated which promotes a collective way of working among people who are supportive within a certain group.

Advantages of Extreme Programming (XP)

- Slipped schedules
- Misunderstanding the business and/or domain
- Canceled projects
- Staff turnover
- Costs incurred in changes
- Business changes
- Production and post-delivery defects

- Extreme Programming (XP) is a Software Development Methodology, known for its flexibility, collaboration and rapid feedback using techniques like continuous testing, frequent releases, and pair programming, in which two programmers collaborate on the same code.
- XP supports user involvement throughout the development process while prioritizing simplicity and communication.
- Overall, XP aims to deliver high-quality software quickly and adapt to changing requirements effectively.