SEPM NOTES

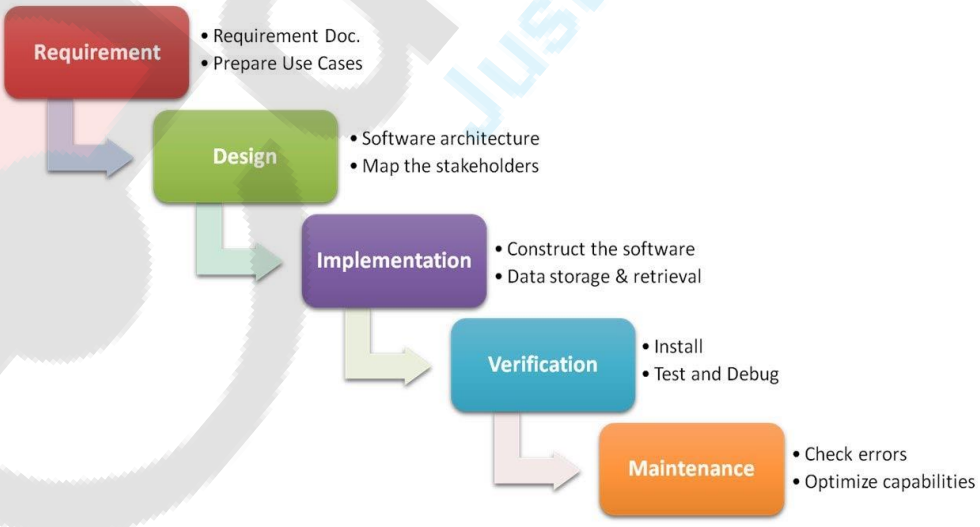
1. EXPLAIN Waterfall MODEL?

The **waterfall model** is a linear sequential (non-iterative) design approach for software development, in which progress flows in one direction downwards (like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, deployment and maintenan ce.

The waterfall development model originated in the manufacturing and construction industries: highly structured physical environments in which after-the-fact changes are impossible or at least prohibitively expensive. At the time it was adopted for software development, there were no recognised alternatives for knowledge-based creative work.

The Waterfall Model was first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed fully before the next phase can begin. This type of **software development model** is basically used for the for the project which is small and there are no uncertain requirements. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model **software testing** starts only after the development is complete. In **waterfall model phases** do not overlap.

Diagram of Waterfall-model:



Advantages of waterfall model:

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are very well understood.

Disadvantages of waterfall model:

- Once an application is in the **testing** stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.

When to use the waterfall model:

- This model is used only when the requirements are very well known, clear and fixed.
- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements
- Ample resources with required expertise are available freely
- The project is short.

Very less customer interaction is involved during the development of the product. Once the product is ready then only it can be demoed to the end users. Once the product is developed and if any failure occurs then the cost of fixing such issues are very high, because we need to update everywhere from document till the logic.

What is Incremental model- advantages, disadvantages and when to use it?

In incremental model the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a **"multi-waterfall" cycle**. Cycles are divided up into smaller, more easily managed

modules. Incremental model is a type of software development model like **V-model**, **Agile model** etc.

In this model, each module passes through the requirements, design, implementation and **testing** phases. A working version of software is produced during the first module, so you have working software early on during the **software life cycle**. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.

For example:

when we work **incrementally** we are adding piece by piece but expect that each piece is fully finished. Thus keep on adding the pieces until it's complete. As in the image above a person has thought of the application. Then he started building it and in the first iteration the first module of the application or product is totally ready and can be demoed to the customers. Likewise in the second iteration the other module is ready and integrated with the first module. Similarly, in the third iteration the whole product is ready and integrated. Hence, the product got ready step by step.

Diagram of Incremental model:

Advantages of Incremental model:

- Generates working software quickly and early during the software life cycle.
- This model is more flexible – less costly to change scope and requirements.
- It is easier to test and debug during a smaller iteration.
- In this model customer can respond to each built.
- Lowers initial delivery cost.
- Easier to manage risk because risky pieces are identified and handled during it'd iteration.

Disadvantages of Incremental model:

- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than **waterfall**.

When to use the Incremental model:

- This model can be used when the requirements of the complete system are clearly defined and understood.
- Major requirements must be defined; however, some details can evolve with time.
- There is a need to get a product to the market early.
- A new technology is being used
- Resources with needed skill set are not available
- There are some high risk features and goals.

2. **What is Prototype model- advantages, disadvantages and when to use it?**

The basic idea in **Prototype model** is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. Prototype model is a **software development model**. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system. Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements.

The prototype are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

Advantages of Prototype model:

- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified Requirements validation, Quick implementation of, incomplete, but functional, application.

Disadvantages of Prototype model:

- Leads to implementing and then repairing way of building systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Incomplete application may cause application not to be used as the full system was designed Incomplete or inadequate problem analysis.

When to use Prototype model:

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.

3. **What is Spiral model- advantages, disadvantages and when to use it?**

The spiral model is similar to the **incremental model**, with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spirals builds on the baseline spiral. Its one of the

software development models like **Waterfall**, **Agile**, **V-Model**.

Planning Phase: Requirements are gathered during the planning phase. Requirements like 'BRS' that is 'Business Requirement Specifications' and 'SRS' that is 'System Requirement specifications'.

Risk Analysis: In the **risk analysis phase**, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

Engineering Phase: In this phase software is **developed**, along with **testing** at the end of the phase. Hence in this phase the development and testing is done.

E**valuation phase:** This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

Advantages of Spiral model:

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the **software life cycle**.

Disadvantages of Spiral model:

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

When to use Spiral model:

- When costs and risk evaluation is important
- For medium to high-risk projects
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex
- New product line
- Significant changes are expected (research and exploration)

4. **What is Agile model – advantages, disadvantages and when to use it?**

Agile development model is also a type of **Incremental model**. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly **tested** to ensure **software quality** is maintained. It is used for time critical applications. Extreme Programming (XP) is currently one of the most well-known agile **development life cycle model**.

Diagram of Agile model:

Advantages of Agile model:

- Customer satisfaction by rapid, continuous delivery of useful software.
- People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
- Working software is delivered frequently (weeks rather than months).
- Face-to-face conversation is the best form of communication.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed

Disadvantages of Agile model:

- In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
- There is lack of emphasis on necessary designing and documentation.
- The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
- Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

When to use Agile model:

- When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
- To implement a new feature, the developers, need to lose only the work of a few days, or even only hours, to roll back and implement it.
- Unlike the **waterfall model** in agile model very limited **planning** is required to get started with the project. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
- Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed

in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

5. What is RAD model- advantages, disadvantages and when to use it.

RAD model is Rapid Application Development model. It is a type of **incremental model**. In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

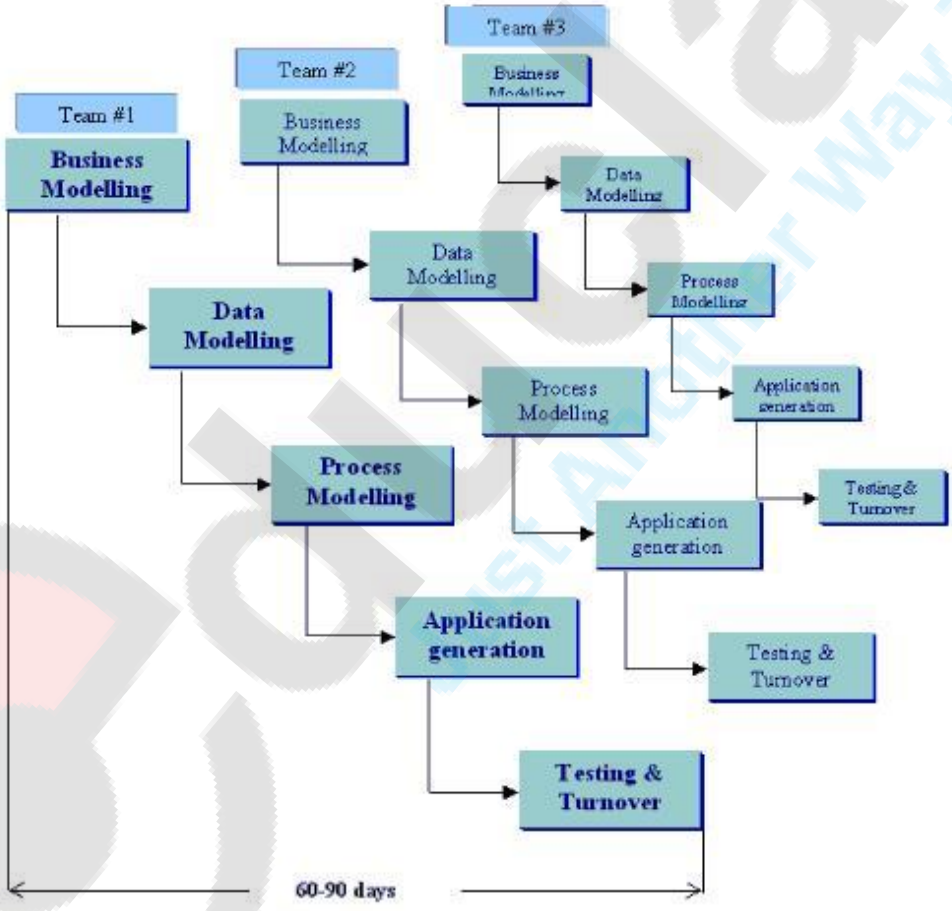


Figure 1.5 – RAD Model

Diagram of RAD-MODEL

The phases in the rapid application development (RAD) model are:

Business modeling: The information flow is identified between various business functions.
Data modeling: Information gathered from business modeling is used to define data objects that are needed for the business.
Process modeling: Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.
Application generation: Automated tools are used to convert process models into code and the actual system.
Testing and turnover: Test new components and all the interfaces.

Advantages of the RAD model:

- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of **integration issues**.

Disadvantages of RAD model:

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

When to use RAD model:

- RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- RAD **SDLC model** should be chosen only if resources with high business knowledge are available and there is a need to produce the same.

6. Evolutionary Process Models

- 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

1. The Prototyping model

- Prototype is defined as first or preliminary form using which other forms are copied or derived.
- Prototype model is a set of general objectives for software.
- It does not identify the requirements like detailed input, output.
- It is software working model of limited functionality.
- In this model, working programs are quickly produced.

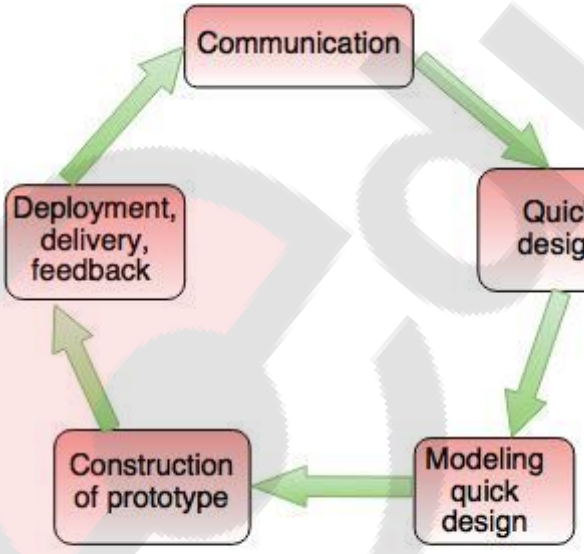


Fig. - The Prototyping Model

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.

Advantages of Prototyping Model

- Prototype model need not know the detailed input, output, processes, adaptability of operating system and full machine interaction.
- In the development process of this model users are actively involved.

- The development process is the best platform to understand the system by the user.
- Errors are detected much earlier.
- Gives quick user feedback for better solutions.
- It identifies the missing functionality easily. It also identifies the confusion or difficult functions.

Disadvantages of Prototyping Model:

- The client involvement is more and it is not always considered by the developer.
- It is a slow process because it takes more time for development.
- Many changes can disturb the rhythm of the development team.
- It is a thrown away prototype when the users are confused with it.

7. **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 framework activities of the spiral model are as shown in the following figure.

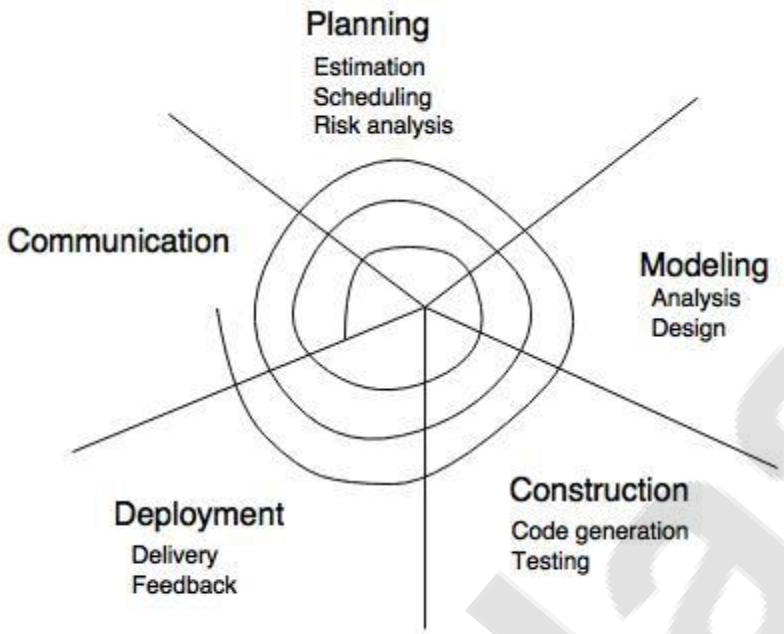


Fig. - The Spiral Model

NOTE: The description of the phases of the spiral model is same as that of the process model.

Advantages of Spiral Model

- It reduces high amount of risk.
- It is good for large and critical projects.
- It gives strong approval and documentation control.
- In spiral model, the software is produced early in the life cycle process.

Disadvantages of Spiral Model

- It can be costly to develop a software model.
- It is not used for small projects.

8. **The concurrent development model**

- The concurrent development model is called as concurrent model.
- The communication activity has completed in the first iteration and exits in the awaiting changes state.

- The modeling activity completed its initial communication and then go to the underdevelopment state.
- If the customer specifies the change in the requirement, then the modeling activity moves from the under development state into the awaiting change state.
- The concurrent process model activities moving from one state to another state.

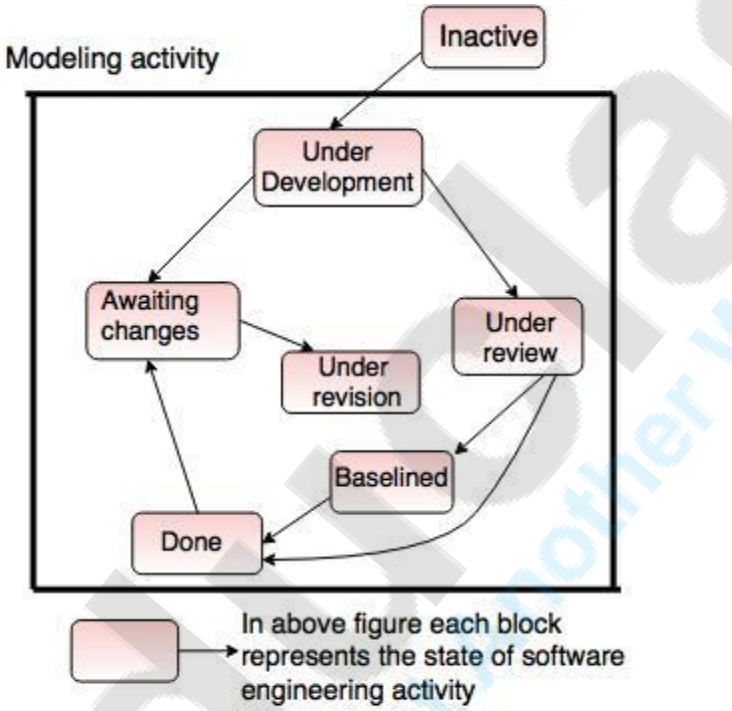


Fig. - One element of the concurrent process model

Advantages of the concurrent development model

- This model is applicable to all types of software development processes.
- It is easy for understanding and use.
- It gives immediate feedback from testing.
- It provides an accurate picture of the current state of a project.

Disadvantages of the concurrent development model

- It needs better communication between the team members. This may not be achieved all the time.
- It requires to remember the status of the different activities.

9. What is Extreme programming

Planning and feedback loops in extreme programming.

Extreme programming (**XP**) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type of agile software development, it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted.
Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, code simplicity and clarity, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers. The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels. As an example, code reviews are considered a beneficial practice; taken to the extreme, code can be reviewed *continuously*, i.e. the practice of pair programming.

Extreme programming has been described as having 12 practices, grouped into four areas:

Fine-scale feedback

- Pair programming
- Planning game
- Test-driven development
- Whole team

Continuous process

- Continuous integration
- Refactoring or design improvement
- Small releases

Shared understanding

- Coding standards
- Collective code ownership
- Simple design
- System metaphor

Programmer welfare

- Sustainable pace

Coding

- The customer is always available
- Code the unit test first
- Only one pair integrates code at a time
- Leave optimization until last
- No overtime

Testing

- All code must have unit tests
- All code must pass all unit tests before it can be released.
- When a bug is found, tests are created before the bug is addressed (a bug is not an error in logic, it is a test that was not written)
- Acceptance tests are run often and the results are published

- **Advantages of Extreme Programming**
- The greatest advantage of Extreme Programming is that this methodology allows software development companies to save costs and time required for project realization. Time savings are available because of the fact that XP focuses on timely delivery of final products.

Extreme Programming teams save lots of money because they don't use too much documentation. They usually solve problems through discussions inside of the team.

- Simplicity is one more advantage of Extreme Programming projects. The developers who prefer to use this methodology create extremely simple code that can be improved at any moment.
- **Extreme Programming disadvantages**
- Some specialists say that Extreme Programming is focused on the code rather than on design. That may be a problem because good design is extremely important for software applications. It helps sell them in the software market. Additionally, in XP projects the defect documentation is not always good. Lack of defect documentation may lead to occurrence of similar bugs in the future.
- One more disadvantage of XP is that this methodology does not measure code quality assurance. It may cause defects in the initial code.

10. What is scrum

Scrum is an Agile framework for completing complex projects. Scrum originally was formalized for software development projects, but it works well for any complex, innovative scope of work. The possibilities are endless. The Scrum framework is deceptively simple.

Scrum is the leading agile development methodology, used by Fortune 500 companies around the world. The Scrum Alliance exists to transform the way we tackle complex projects, bringing the Scrum framework and agile principles beyond software development to the broader world of work.

The Scrum framework in 30 seconds

- A product owner creates a prioritized wish list called a product backlog.
- During sprint planning, the team pulls a small chunk from the top of that wish list, a sprint backlog, and decides how to implement those pieces.

- The team has a certain amount of time — a sprint (usually two to four weeks) — to complete its work, but it meets each day to assess its progress (daily Scrum).
- Along the way, the Scrum Master keeps the team focused on its goal.
- At the end of the sprint, the work should be potentially shippable: ready to hand to a customer, put on a store shelf, or show to a stakeholder.
- The sprint ends with a sprint review and retrospective.
- As the next sprint begins, the team chooses another chunk of the product backlog and begins working again.

Scrum is a management and control process that cuts through complexity to focus on building products that meet business needs. Management and teams are able to get their hands around the requirements and technologies, never let go, and deliver working products, incrementally and empirically.

Scrum itself is a simple framework for effective team collaboration on complex products.

Scrum Glossary

The Scrum Glossary is meant to represent an overview of Scrum-related terms. Some of the mentioned terms are not mandatory in Scrum, but have been added because they are commonly used in Scrum.

To learn more about the Scrum framework, to identify which of these terms are required elements of Scrum and to understand how the mentioned elements are connected, we highly recommend that you reference The Scrum Guide. To learn more about terms specific to software development teams using Scrum and agile software development techniques, reference the Professional Scrum Developer glossary.

The Scrum Framework

Scrum is simple. It is the opposite of a big collection of interwoven mandatory components. Scrum is not a *methodology*. Scrum implements the scientific *method* of empiricism. Scrum replaces a programmed algorithmic approach with a heuristic one, with respect for people and self-organization to deal with unpredictability and solving complex problems. The below graphic represents Scrum in Action as described by Ken Schwaber and Jeff Sutherland in their book *Software in 30 Days*taking us from planning through software delivery.

The Roles of the Scrum Team

The Scrum Team consists of a Product Owner, the Development Team, and a Scrum Master. Scrum Teams are self-organizing and cross-functional. Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team. Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team. The team model in Scrum is designed to optimize flexibility, creativity, and productivity.

The Scrum Events

Prescribed events are used in Scrum to create regularity and to minimize the need for meetings not defined in Scrum. All events are time-boxed. Once a Sprint begins, its duration is fixed and cannot be shortened or lengthened. The remaining events may end whenever the purpose of the event is achieved, ensuring an appropriate amount of time is spent without allowing waste in the process. The Scrum Events are:

- Sprint
- Sprint Planning
- Daily Scrum
- Sprint Review
- Sprint Retrospective

UNIT 3

1) What is Software Requirement?

Requirement is a condition or capability possessed by the software or system component in order to solve a real world problem. The problems can be to automate a part of a system, to correct shortcomings of an existing system, to control a device, and so on. **IEEE** defines requirement as (1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or

system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2).'

Requirements describe how a system should act, appear or perform. For this, when users request for software, they provide an approximation of what the new system should be capable of doing. Requirements differ from one user to another and from one business process to another.

Guidelines for Expressing Requirements

The purpose of the requirements document is to provide a basis for the mutual understanding between the users and the designers of the initial definition of the software development life cycle (SDLC) including the requirements, operating environment and development plan.

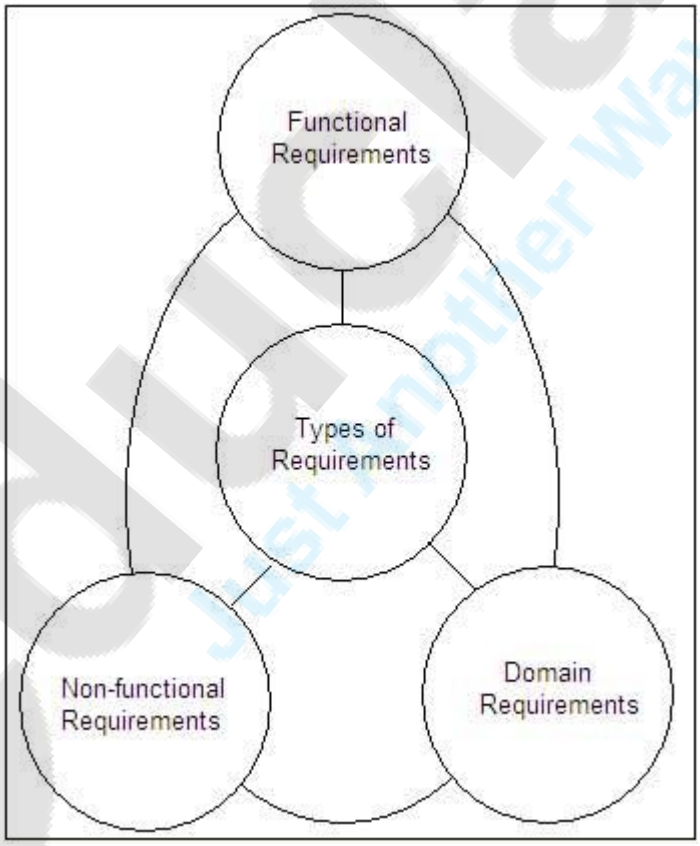
The requirements document should include the overview, the proposed methods and procedures, a summary of improvements, a summary of impacts, security, privacy, internal control considerations, cost considerations, and alternatives. The requirements section should state the functions required in the software in quantitative and qualitative terms and how these functions will satisfy the performance objectives. The requirements document should also specify the performance requirements such as accuracy, validation, timing, and flexibility. Inputs, outputs, and data characteristics need to be explained. Finally, the requirements document needs to describe the operating environment and provide (or make reference to) a development plan.

There is no standard method to express and document requirements. Requirements can be stated efficiently by the experience of knowledgeable individuals, observing past requirements, and by following guidelines. Guidelines act as an efficient method of expressing requirements, which also provide a basis for software development, system testing, and user satisfaction. The guidelines that are commonly followed to document requirements are listed below.

1. Sentences and paragraphs should be short and written in active voice. Also, proper grammar, spelling, and punctuation should be used.
2. Conjunctions such as 'and' and 'or' should be avoided as they indicate the combination of several requirements in one requirement.
3. Each requirement should be stated only once so that it does not create redundancy in the requirements specification document.

Types of Requirements

Requirements help to understand the behavior of a system, which is described by various tasks of the system. For example, some of the tasks of a system are to provide a response to input values, determine the state of data objects, and so on. Note that requirements are considered prior to the development of the software. The requirements, which are commonly considered, are classified into three categories, namely, functional requirements, non-functional requirements, and domain requirements.



IEEE defines functional requirements as 'a function that a system or component must be able to perform.' These requirements describe the interaction of software with its environment and specify the inputs, outputs, external interfaces, and the functions that should be included in the software. Also, the services provided byfunctional requirements specify the

procedure by which the software should reactto particular inputs or behave in particular situations.

To understand functional requirements properly, let us consider the following example of an online banking system.

1. The user of the bank should be able to search the desired services from the available ones.
2. There should be appropriate documents' for users to read. This implies that when a user wants to open an account in the bank, the forms must be available so that the user can open an account.
3. After registration, the user should be provided with a unique acknowledgement number so that he can later be given an account number.

The above mentioned functional requirements describe the specific services provided by the online banking system. These requirements indicate user requirements and specify that functional requirements may be described at different levels of detail in an online banking system. With the help of these functional requirements, users can easily view, search and download registration forms and other information about the bank. On the other hand, if requirements are not stated properly, they are misinterpreted by software engineers and user requirements are not met.

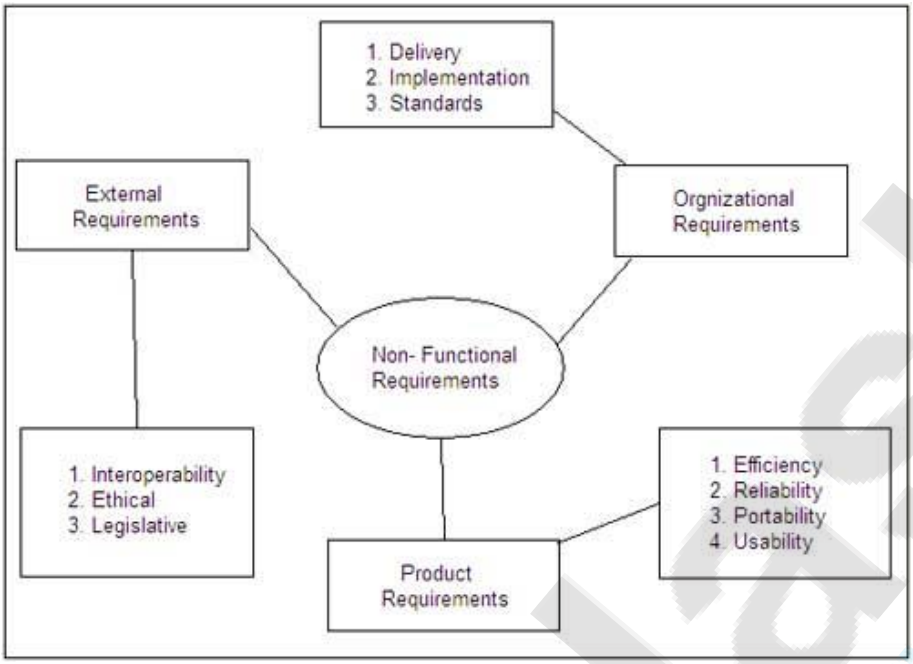
The functional requirements should be complete and consistent. Completeness implies that all the user requirements are defined. Consistency implies that all requirements are specified clearly without any contradictory definition. Generally, it is observed that completeness and consistency cannot be achieved in large software or in a complex system due to the problems that arise while defining the functional requirements of these systems. The different needs of stakeholders also prevent the achievement of completeness and consistency. Due to these reasons, requirements may not be obvious when they are,'first specified and may further lead to inconsistencies in the requirements specification.

The non-functional requirements (also known as **quality requirements)** are related to system attributes such as reliability and response time. Non-functional requirements arise due to user requirements, budget constraints, organizational policies, and so on. These requirements are not related directly to any particular function provided by the system.

Non-functional requirements should be accomplished in software to make it perform efficiently. For example, if an aeroplane is unable to fulfill reliability requirements, it is not approved for safe operation. Similarly, if a real time control system is ineffective in accomplishing non-functional requirements, the control functions cannot operate correctly.

The description of different types of non-functional requirements is listed below.

1. **Product requirements:** These requirements specify how software product performs. Product requirements comprise the following.
2. **Efficiency requirements:** Describe the extent to which the software makes optimal use of resources, the speed with which the system executes, and the memory it consumes for its operation. For example, the system should be able to operate at least three times faster than the existing system.
3. **Reliability requirements:** Describe the acceptable failure rate of the software. For example, the software should be able to operate even if a hazard occurs.
4. **Portability requirements:** Describe the ease with which the software can be transferred from one platform to another. For example, it should be easy to port the software to a different operating system without the need to redesign the entire software.
5. **Usability requirements:** Describe the ease with which users are able to operate the software. For example, the software should be able to provide access to functionality with fewer keystrokes and mouse clicks.
6. **Organizational requirements:** These requirements are derived from the policies and procedures of an organization. Organizational requirements comprise the following.
7. **Delivery requirements:** Specify when the software and its documentation are to be delivered to the user.
8. **Implementation requirements:** Describe requirements such as programming language and design method.
9. **Standards requirements:** Describe the process standards to be used during software development. For example, the software should be developed using standards specified by the ISO and IEEE standards.
10. **External requirements:** These requirements include all the requirements that affect the software or its development process externally. External requirements comprise the following.
11. **Interoperability requirements:** Define the way in which different computer based systems will interact with each other in one or more organizations.
12. **Ethical requirements:** Specify the rules and regulations of the software so that they are acceptable to users.
13. **Legislative requirements:** Ensure that the software operates within the legal jurisdiction. For example, pirated software should not be sold.



Non-functional requirements are difficult to verify. Hence, it is essential to write non-functional requirements quantitatively, so that they can be tested. For this, non-functional requirements metrics are used. These metrics are listed in Table.

Metrics for Non-functional Requirements

Features	Measures
Speed	Processed transaction/ second User/event response time Screen refresh rate
Size	Amount of memory (KB) Number of RAM chips.
Ease of use	Training time Number of help windows
Reliability	Mean time to failure (MTTF) Portability of unavailability Rate of failure occurrence
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target-dependent statements Number of target systems

Requirements which are derived from the application domain of the system instead from the needs of the users are known as **domain requirements.** These requirements may be new functional requirements or specify a method to perform some particular computations. In addition, these requirements include any constraint that may be present in the existing functional requirements. As domain requirements reflect the fundamentals of the application domain, it is important to understand these requirements. Also, if these requirements are not fulfilled, it may be difficult to make .the system work as desired.

A system can include a number of domain requirements. For example, it may comprise a design constraint that describes the user interface, which is capable of accessing all the databases used in a system. It is important for a development team to create databases and interface designs as per established standards. Similarly, the requirements of the user such as copyright restrictions and security mechanism for the files and documents used in the system are also domain requirements. When domain

requirements are not expressed clearly, it can result in the following difficulties.

Problem of understandability: When domain requirements are specified in the language of application domain (such as mathematical expressions), it becomes difficult for software engineers to understand them.

Problem of implicitness: When domain experts understand the domain requirements but do not express these requirements clearly, it may create a problem (due to incomplete information) for the development team to understand and implement the requirements in the system.

11. WHAT is Feasibility study

A project feasibility study is a comprehensive report that examines in detail the five frames of analysis of a given project. It also takes into consideration its four Ps, its risks and POVs, and its constraints (calendar, costs, and norms of quality). The goal is to determine whether the project should go ahead, be redesigned, or else abandoned altogether.

The five frames of analysis are: The frame of definition; the frame of contextual risks; the frame of potentiality; the parametric frame; the frame of dominant and contingency strategies.

The four Ps are traditionally defined as Plan, Processes, People, and Power. The risks are considered to be external to the project (e.g., weather conditions) and are divided in eight categories: (Plan) financial and organizational (e.g., government structure for a private project); (Processes) environmental and technological; (People) marketing and sociocultural; and (Power) legal and political. POVs are Points of Vulnerability: they differ from risks in the sense that they are internal to the project and can be controlled or else eliminated.

The constraints are the standard constraints of calendar, costs and norms of quality that can each be objectively determined and measured along the entire project lifecycle. Depending on projects, portions of the study may suffice to produce a feasibility study; smaller projects, for example, may not require an exhaustive environmental assessment.

Technical feasibility
This assessment is based on an outline design of system requirements, to determine whether the company has the technical expertise to handle completion of the project. When writing a feasibility report, the following should be taken to consideration:

- A brief description of the business to assess more possible factors which could affect the study

- The part of the business being examined
- The human and economic factor
- The possible solutions to the problem

At this level, the concern is whether the proposal is both *technically* and legally feasible (assuming moderate cost).
The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system

Method of production

Legal feasibility

Determines whether the proposed system conflicts with legal requirements, e.g., a data processing system must comply with the local data protection regulations and if the proposed venture is acceptable in accordance to the laws of the land.

Operational feasibility

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.
The operational feasibility assessment focuses on the degree to which the proposed development project fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture and existing business processes.
To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability and others. These parameters are required to be considered at the early stages of design if desired operational behaviours are to be realised. A system design and development requires appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases.

Schedule feasibility

A project will fail if it takes too long to be completed before it is useful. Typically this means estimating how long the system will take to develop, and if it can be completed in a given time period using some methods like payback period. Schedule feasibility is a measure of how reasonable the project timetable is. Given our technical expertise, are the project deadlines reasonable? Some projects are initiated with specific deadlines. It is necessary to determine whether the deadlines are mandatory or desirable Other feasibility factors

Resource feasibility

This involves questions such as how much time is available to build the new system, when it can be built, whether it interferes with normal business operations, type and amount of resources required, dependencies, and developmental procedures with company revenue prospectus.

Financial feasibility

In case of a new project, financial viability can be judged on the following parameters:

- Total estimated cost of the project
- Financing of the project in terms of its capital structure, debt to equity ratio and promoter's share of total cost
- Existing investment by the promoter in any other business
- Projected cash flow and profitability

The financial viability of a project should provide the following information:

- Full details of the assets to be financed and how liquid those assets are.
- Rate of conversion to cash-liquidity (i.e., how easily the various assets can be converted to cash).
- Project's funding potential and repayment terms.
- Sensitivity in the repayments capability to the following factors:
  - Mild slowing of sales.
  - Acute reduction/slowing of sales.
  - Small increase in cost.
  - Large increase in cost.

- Adverse economic conditions.

In 1983 the first generation of the Computer Model for Feasibility Analysis and Reporting (COMFAR), a computation tool for financial analysis of investments, was released. Since then, this United Nations Industrial Development Organization (UNIDO) software has been developed to also support the economic appraisal of projects. The COMFAR III Expert is intended as an aid in the analysis of investment projects. The main module of the program accepts financial and economic data, produces financial and economic statements and graphical displays and calculates measures of performance. Supplementary modules assist in the analytical process. Cost-benefit and value-added methods of economic analysis developed by UNIDO are included in the program and the methods of major international development institutions are accommodated. The program is applicable for the analysis of investment in new projects and expansion or rehabilitation of existing enterprises as, e.g., in the case of reprivatisation projects. For joint ventures, the financial perspective of each partner or class of shareholder can be developed. Analysis can be performed under a variety of assumptions concerning inflation, currency revaluation and price escalations

12. Requirements Elicitation and Analysis

Requirements elicitation is the process of gathering the requirements. In this process the technical professional in the organization, like software developers and the system engineers, work together with the users of the system and the customers

This process is useful in finding the problems that has to be solved. The problems include,

1. What the proposed system should provide?
2. What are the expected services form the system?
3. What are the required characteristics of the system?
4. What are the required hardware and software constraints of the system?

Requirements elicitation process includes a chain of processes that interact with each other to produce requirements documentation. The lifecycle of the requirements elicitation process is represented in figure 3 and the steps involved are explained Diagrammatic Representation of Requirements elicitation process

Back ground Knowledge

The analyst must understand the back ground and the domain knowledge of the application that is being developed.

Example: If the system is developed for ATM, the developer should have some basic knowledge of how the ATM works.

Gathering the requirements

This is the activity of discovering the requirements by involving with the stakeholders and users.

Requirements classification

This activity includes the organizing of the requirements gathered from different sources.

Requirements conflict

This activity involves with the stakeholders and requirements engineers. This is used to solve the problems in the requirements that contradict the organization and business rules.

Requirements Prioritization

Discovering the important requirements by interacting with the stake holders and organize them in to most priority order.

Requirements check

This activity involves checking the stake holder's expectations on the system with the gathered requirements.

Requirements elicitation process not only helps the organization to gather the requirements, but also it analyses the requirements and the business procedures of the organization. The requirements elicitation and analysis is a difficult activity in the requirements engineering, due to the following reasons

1. Lack of technical knowledge and unawareness of the technical aspects of the system from the stake holder's side.

2. Sometimes they may demand unrealistic things and they do not know what they exactly expecting form, the system.

3. Stakeholders express their requirements in the most general terms; **it is difficult** to find the technical aspects of the system form the general terms.

4. Different people expect different services form the proposed system, requirements engineers has to discover the good sources of requirements and commonalities.

5. Business procedures, political influences and the budget matters where the clear analysis is required.

4. Classic Requirements Elicitation techniques

These requirements elicitation techniques have been used for a long time. These are tested and proven methods. The different classic requirements elicitation processes are,

4.1 Interviews

Interviews are the commonly used and most popular method for requirements elicitation

In this method the analyst and the engineers of the requirements engineering process discuss with the different types of stake holders to understand the requirements of the system and the objective they have to fulfil in the system. There are typically 2 main types of interviews, which will be described in the following sections

1. Closed Interview: In this interview the requirements engineer prepares some predefined questions and he tries to get the answers for these questions from the stake holders.

2. Open interview: In this interview the requirements engineer does not prepare any predefined questions, and he tries to get the information from the stakeholders in open discussions. He mostly concentrates on finding the stake holders expectations on the system.

Generally the interviews start with the predefined questions. However, in the process of the interview, a lot of different considerable things may arise, that leads to open discussion. Interviews are effective for understating the problems in the existing system and to find the general requirements of the stakeholders. But, it is difficult to decide the boundaries of the proposed system and the organization procedures using this method. To make the effective interview the requirement engineer and the stake holders has to perform in the following ways

1. Interviewer should be patient enough to listen to the stake holder’s views and the requirements. He should be open-minded.

2. Stake holders should be expressive in the interview; they should express their views in definite context.

4.2 Questionnaire

Questionnaires are one of the methods of gathering requirements in less cost.

Questionnaires reach a large number of people, not only in less time but also in a lesser cost. But the results extracted from the questionnaires should be clearly analyzed. The result from the questionnaires mainly depends on the two factors,

1. Effectiveness and the design of the questionnaire

2. Honesty of the respondent.

A well designed and effective questionnaire can be used to decide the actual user requirements, objectives and the constraints [22]. A good structured questionnaire influences people to answer honestly thus making it possible to gather reliable results from a large group of people. The data collected through questionnaires can

be used to analyze the obtained results, both systematically and quantitatively.

The designing of questionnaire is a multi-stage process and should be viewed accordingly.

The steps involved in designing and administering a questionnaire

1. The purpose of the survey should be defined

2. The sampling group (respondents to the survey) should be decided

3. Preparing and developing the Questionnaire

4. Conducting the Questionnaire process

5. Gathering and analysing the results

Steps in arranging a questionnaire

• The questions should be arranged well, so that general questions are followed by particular questions.

• Arrange the questions such that, easy questions comes first.

• Arrange the questions in a order of known to unknown

• Try to use closed format questions in the beginning.

• The questions relevant to the main subject should be given high priority and stated at the start of the questionnaire.

• Avoid personal and intimate questions at the beginning

The general factors which affect the usage of the questionnaire are

1. The available resources to gather the requirements: The usage off the questionnaire mainly depends on the available resources. When the resources and the funds are less, the questionnaire is the best method to gather the requirements because the cost of the administrational is very less. The questionnaire can also be used to save the time by gathering the requirements from a large number of people in a very short interval.

2. Type of Requirements that has to be gathering: The type of the information that has to be gathered depends on the level of the respondent's knowledge and background.

3. Anonymity provided to the respondent: These are the general factors to use the questionnaire but there is no particular rule as and when to use the questionnaire for gathering requirements.

4.3 Social analysis

Social analysis is also known as Observation. Observation is the method of collecting requirements by observing the people doing their normal work. This method is generally used to find the additional requirements needed by the user, when the user is unable to explain their expected requirements from the new product and problems with the existing product.

This social analysis is of four types. They are,

Passive observations- This social analysis is carried out without direct involvement of the observer in the society. The observation of the peoples work is carried out by recording using videotapes, video cameras and surveillance cameras. The documentation of the problems and the requirements are prepared from the recorded data.

Active observation- This observation is carried out with the direct involvement of the observer. The people are provided with the new product prototype or existing product to perform the operations on the product. The observer provides the domain knowledge to the user to work with the product and he makes the report of the requirements of the people by observing their work with the product.

Explanatory Observations- In this type of observation, the users talk loudly, explaining what they are doing, while using the product. The observer takes notes using the explanation given by the user.

Ethnography - In this method the observer is completely immersed in the society. The observer goes through in depth observation of the society and their works.

There is no particular formula to carry out this method but it is time consuming and expensive method to gather the requirements.

5. Modern Requirements Elicitation Techniques

There are different types of modern elicitation techniques, which will be described in the following section.

5.1 Prototyping

Prototype is the representations or visualizations of the actual system parts.

The prototype is designed in the early stages of the implementation of the project. It provides the General idea of the actual system functions and the work flow. Prototyping is used to gather the requirements from the users by presenting GUI based system functions.

The main aim of Requirements Elicitation is to gather the requirements before the product is developed. But it is difficult to discover the additional requirements until it comes in to usage or somebody is actually using it. The process of gathering the requirements from the stakeholders and the end users is limited and it is difficult to discover their expectations and the requirements on the new product without providing some model that resembles the appearance of the real product.

A prototype represents the actual product in both functional and graphical sense.

It provides the flexibility to the users and the stake holders to work with the initial version of the product to understand the system and discuss them to think of the additional and missed requirements. Prototyping is a most expensive than the all other methods of requirements elicitation.

5.2 Requirements reuse

In the field of software engineering reusing the requirements of the existing system is common method of requirements elicitation [2,4,33, 34]. Using the existing Knowledge to develop the new product has many advantages that include low cost and less time.

Though each product has their own type of stake holders and users, there is still number of situations that the reusing of the requirements takes place.

Example,

1. User interface designs of the application domain information

2. Different companies' database and security policies.

Nowadays in software industries the more than half of the requirements for the new project requirements are acquired from the existing projects. Although there is need to check the requirements before they are used in the proposed product, the reused requirements are already validated and analysed thus reducing the time of testing.

"Successful reuse starts with the having an organizational culture that consciously encourage reuse rather than reinvention".

The various questions that help us to find the reusable requirements are,

• What are the problems in the existing product?

• What the proposed product should provide to overcome the difficulties in the existing product?

• Who are the users and the stakeholders of the existing and proposed products?

It is difficult to say that particular proposed product is completely different from the existing product, because it is easy to find reused requirements in any project requirement specification

Stake holders will provide the related documents and the existing system documents to find the reusable requirements. Sometimes the documents of the existing product questionnaire are also helpful in finding the source of reusable requirements. Finding and using the reusable requirements for the projects is the best way of requirements elicitation.

5.3 Scenarios

The scenarios of the particular system will give the working method of different interaction sessions or the situations of the system [38, 39]. These scenarios are helpful for requirements elicitation in two ways. They are.

1. Analyzing the different sessions of the system gives the flexibility to find the requirements

2. The user response after interaction with the scenarios will give the flexibility to find the requirements.

Scenarios are generally used after the initial requirements specification is finished. The scenarios are produced by the developers when the initial requirements are collected and the basic idea about the functions to be provided by the system is prepared. The developed scenario**s** will be used to find and prepare the detailed requirements specification.

The method to prepare the scenario is as follows

1. Scenario should start with the particular state called the starting point

2. Every state should be connected with the next state.

3. Every state should contain the Normal, exceptional and alternative flow of events.

4. Every state should describe the actions performed on that state.

5. The interaction should be ended with the final state.

Generally, the software industries use the text based and picture based scenarios.

The developer provides the scenario model for the system. The user and the requirements engineer work with the scenarios of the proposed system. The requirements engineer takes the notes of the user comments, suggestions and difficulties that user faced when interacting with the scenario. Scenarios of the proposed system are always prepared with the involvement of the stake holders to clarify what they require in each interaction. Use cases developed for the system are the basic guidelines for the scenario models.

5.4 Brainstorming

Brainstorming is a techniques used to generate new ideas and find the solution to a specific issue. This is conducted as a conference with six to ten members.

The members are from the different departments and domain experts are also included. This conference is headed by the organizer, who states the issue to be discussed. The conference is generally held in a round table fashion. Every member person is allotted with certain time interval to explain their ideas. Notepads are provided to all members to write their ideas and suggestions. The team of brainstorming will then decide the best

idea by voting from the group and that idea is selected as the solution to the issue discussed in the conference.

Brainstorming can be used to answer the following questions,

• What exactly the system should provide

• What are the business and organization rules required to follow

• What type of questions should be there in the interviews and questionnaires?

• What are the risk factors effect the proposed system development and what to do avoid that?

5.5 Joint Application Development[JAD]:

It is an organized and structured technique for requirements elicitation. This is conducted in the same manner as brainstorming, except that the stakeholders and the users are also allowed to participate and discuss on the design of the proposed system.

The participant in these sessions does not exceed 20 to 30.

The requirements engineers start the session by providing the general overview of the system. The discussion with the stakeholders and the users continues until the final requirements are gathered. This leads to elicitation of better requirements in the first attempt and it reduces the time spent on the requirements phase.

The success of the JAD depends on

1. leader of the JAD session

2. Developers, end-users and the stakeholders of the product.

3. Group involvement.

5.6 User Centered Design

This method is similar to Joint Application Development. The one difference is that the user acts as the part of the development team. The user takes active part in the designing and development of the system. The user provides his ideas and contributes his suggestions as a member of the development team.

Advantages of User Centered Design

• The user is closely involved with the development

- **The user feedback can be obtained immediately**
- **Reduces time and cost spent on gathering user feedback.**

FACILITATED APPLICATION SPECIFICATION TECHNIQUES

Introduction to Facilitated Application Specification Techniques (FAST)

While working simultaneously the customer and the software engineer feel that they are working separately with each other and they are us and them in place of awe. In this case, there will be much formal work, paper work, documents and question and answer sessions misunderstandings and unsuccessful working relationship.

So they have to set their minds that they are working together as single team members. And will complete this whole work simultaneously. So they must have the proper discussions in spite of this Q& A session and they should think it as their own work.

Now such type of team oriented approach should be followed. A team should be established during early stages of analysis and specification. This approach is also called as the facilitated application specification technique (FAST), which encourages the creation of a joint team of customers and developers who work together to identify the problem propose elements of the solution negotiate different approaches and specify a preliminary set of solution requirements.

FAST is now becoming a dominating approach in the market. If follows the team oriented approach and set the good and proper discussion type communications. So by using such approach the communication between them can be improved very much which has the direct effect on the whole sole software.

Many different approaches to FAST have been proposed like joint application development (JDA), which was developed by IBM and the METHOD, which was developed by Inc. Falls church, VA. Each approach has some variation with the other approaches, but all follow these basic guidelines.

- A meeting is conducted at a neutral site and attended by both software engineering analysts and customers.

- Rules for preparation and participation are established.
- An agenda is suggested that is formal enough to cover all important point but informal enough to encourage the free flow of ideas.
- A facilitator (can be a customer a developer or an outsider) is assigned to control the meeting.
- A definition mechanism (can be work sheets flip charts or wall stickers or chat room) is used where how meeting will be conducted.
- The goal of this meeting is to identify the problem propose elements of the solution, negotiate different approaches and specify a preliminary set of solution requirement in an atmosphere for the achievement of the goal.

Using these guidelines as base a serried of the steps activities are performed that idea up to the meeting occur during the follow the meeting. These steps are discussed one by one:

- In the beginning a formal meeting is conducted between developer and customer, Q&A session is started to have the general idea about the problem, desired solution, customer and benefits.
- Then, customer and developer both write a separate one or two page. Product request document. This is just like a small report with brief description.
- Next, step is to set a meeting place, time and date for FAST. Parallel a facilitator and attendees of the FAST (both the development and customer user organization sides) are selected.
- After that the product request reports are distributed to all the attendees before the meeting starts.
- Additionally, before the meeting each attendee is asked to make a list of three objects or instances:
- Objects that will be used by the system to perform its functions (Objects of input).
- Objects that will be produced by the system (objects of output) and,
- Objects that will be part of the environment that surrounds the system.

In addition to this each attendee is asked to make another list of service that manipulate or interact with the objects. Finally lists of constraints and performance criterion are also developed.

The attendees are informed that these lists have to be made before the meeting and it should not be exhaustive. In reality, more information would be provided at this stage. But even with additional information, ambiguity would be present, omissions would likely exist, and errors might occur.

- The FAST team is composed of representatives from marketing software and hardware engineering and manufacturing. An outside facilitator is to be called. Each member of the FAST team produces that lists described previously.
- As the FAST meeting begins, the followings steps are performed in sequence.
- Discussion is started with the need and justification for the new product and everyone should agree that the product is justified.
- After establishing the agreement, each participant presents his lists for discussion. All the lists are either pinned on the walls of the room, or posted on a chat room environment for review prior to the meeting.
- Ideally, each list entry should be capable of being manipulate separately so that lists can be combined, errors can be deleted, additions can be made and modifications can be done.
- After making these individual lists, a combined list is created by the group which eliminates the redundant entries, ads that come up during discussions but does not remove any things.
- Now to begin the meeting the facilitator will coordinate the discussion of the meeting
- During meeting. Consensus list of each topic area is developed which can be shortened, lengthened or reworded.
- Now the teams are divided into the sub teams each of the sub team works to develop mini specifications for one more entries on each of the lists. Each mini specification is an elaboration of the work or phrase contained on a list.
- Each sub team then presents each of its mini specifications to all FAST attendees for discussion. Updating, addition and deletion operations are made in minis specification so that this mini specification will uncover new objects, services, constraints or performance requirements that will be added to the original lists.
- During the discussion, an issues list is maintained.
- After the mini specifications are completed, each FAST attendee makes a list of validation criteria for the product and presents it to its team. Then a consensus of validation criteria is created.
- Finally, a complete draft specification, using all inputs from the FAST meeting is assigned to one /more participant.

Such type of team oriented approach provides many benefits improves discussions and communication and makes a strong step toward the development of a specification

Use case approach

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined. The use case should contain all system activities that have significance to the users. A use case can be thought of as a collection of possible scenarios related to a particular goal, indeed, the use case and goal are sometimes considered to be synonymous

A use case (or set of use cases) has these characteristics:

- Organizes functional requirements
- Models the goals of system/actor (user) interactions
- Records paths (called *scenarios*) from trigger events to goals
- Describes one main flow of events (also called a basic course of action), and possibly other ones, called *exceptional* flows of events (also called alternate courses of action)
- Is multi-level, so that one use case can use the functionality of another one.

Use cases can be employed during several stages of software development, such as planning system requirements, validating design, testing software, and creating an outline for online help and user manuals
To explore how use case is used in the enterprise, here are some additional resources for learning about **use case methodology**:

<u>The pros and cons of use case diagrams:</u>

Putting too much into a use case diagram can often render the otherwise useful technique of use cases almost useless. Kevlin Henney recommends a more balanced and restrained approach in order to not lose readers in a myriad of bubbles and microscopic text.

Use Case Based Requirement Elicitation

Use case approach uses a combination of text and pictures. In order to improve the understanding of requirement. The two component, actor and use case, are used for design of use case approach. An actor can be a person, machine or a Information System that is external to the system model. Use Case describe the sequence of interaction between actor and the system necessary to deliver the services that satisfies the goal.

Software Analysis & Design Tools

Software analysis and design includes all activities, which help the transformation of requirement specification into implementation. Requirement specifications specify all functional and non-functional expectations from the software. These requirement specifications come in the shape of human readable and understandable documents, to which a computer has nothing to do.

Software analysis and design is the intermediate stage, which helps human-readable requirements to be transformed into actual code.

Let us see few analysis and design tools used by software designers:

Data Flow Diagram

Data flow diagram is graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data. The DFD does not mention anything about how data flows through the system.

There is a prominent difference between DFD and Flowchart. The flowchart depicts flow of control in program modules. DFDs depict flow of data in the system at various levels. DFD does not contain any control or branch elements.

Types of DFD

Data Flow Diagrams are either Logical or Physical.

Logical DFD - This type of DFD concentrates on the system process, and flow of data in the system.For example in a Banking software system, how data is moved between different entities.

Physical DFD - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

DFD Components

DFD can represent Source, destination, storage and flow of data using the following set of components -

Entities - Entities are source and destination of information data. Entities are represented by a rectangles with their respective names.

Process - Activities and action taken on the data are represented by Circle or Round-edged rectangles.

Data Storage - There are two variants of data storage - it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.

Data Flow - Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

Levels of DFD

Level 0 - Highest abstraction level DFD is known as Level 0 DFD, which depicts the entire information system as one diagram concealing all the underlying details. Level 0 DFDs are also known as context level DFDs.

Level 1 - The Level 0 DFD is broken down into more specific, Level 1 DFD. Level 1 DFD depicts basic modules in the system and flow of data among various modules. Level 1 DFD also mentions basic processes and sources of information.

Level 2 - At this level, DFD shows how data flows inside the modules mentioned in Level

1.Higher level DFDs can be transformed into more specific lower level DFDs with deeper level of understanding unless the desired level of specification is achieved.

HIPO Diagram

HIPO (Hierarchical Input Process Output) diagram is a combination of two organized method to analyze the system and provide the means of documentation. HIPO model was developed by IBM in year 1970.

HIPO diagram represents the hierarchy of modules in the software system. Analyst uses HIPO diagram in order to obtain high-level view of system functions. It decomposes functions into sub-functions in a hierarchical manner. It depicts the functions performed by system.

HIPO diagrams are good for documentation purpose. Their graphical representation makes it easier for designers and managers to get the pictorial idea of the system structure.

In contrast to IPO (Input Process Output) diagram, which depicts the flow of control and data in a module, HIPO does not provide any information about data flow or control flow.

Example

Both parts of HIPO diagram, Hierarchical presentation and IPO Chart are used for structure design of software program as well as documentation of the same.

Data Dictionary-Data dictionary is the centralized collection of information about data. It stores meaning and origin of data, its relationship with other data, data format for usage etc. Data dictionary has rigorous definitions of all names in order to facilitate user and software designers.

Data dictionary is often referenced as meta-data (data about data) repository. It is created along with DFD (Data Flow Diagram) model of software program and is expected to be updated whenever DFD is changed or updated.

Requirement of Data Dictionary

The data is referenced via data dictionary while designing and implementing software. Data dictionary removes any chances of ambiguity. It helps keeping work of programmers and designers synchronized while using same object reference everywhere in the program.

Data dictionary provides a way of documentation for the complete database system in one place. Validation of DFD is carried out using data dictionary.

Contents

Data dictionary should contain information about the following

Data Flow

Data Structure

Data Elements

Data Stores

Data Processing

Warnier/Orr diagram:

A **Warnier/Orr diagram** (also known as a logical construction of a program/system) is a kind of hierarchical flowchart that allows the description of the organization of data and procedures. They were initially developed in France by Jean-Dominique Warnier and in the United States by Kenneth Orr. This method aids the design of program structures by identifying the output and processing results and then working backwards to determine the steps and combinations of input needed to produce them. The simple graphic method used in Warnier/Orr diagrams makes the levels in the system evident and the movement of the data between them vivid.

Basic Elements

Warnier/Orr diagrams show the processes and sequences in which they are performed. Each process is defined in a hierarchical manner i.e. it consists of sets of sub processes, that define it. At each level, the process is shown in bracket that groups its components.

Since a process can have many different sub processes, Warnier/Orr diagram uses a set of brackets to show each level of the system. Critical factors in s/w definition and development are iteration or repetition and alteration. Warnier/Orr diagrams show this very well.

Using Warnier/Orr diagrams

To develop a Warnier/Orr diagram, the analyst works backwards, starting with systems output and using output oriented analysis. On paper, the development moves from the set to the element (from left to right) . First, the intended output or results of the processing are defined. At the next level, shown by inclusion with a bracket, the steps needed to produce the output are defined. Each step in turn is further defined. Additional brackets group the processes required to produce the result on the next level.

Warnier/Orr diagrams offer some distinct advantages to systems experts. They are simple in appearance and easy to understand. Yet they are powerful design tools. They have advantage of showing groupings of processes and the data that must be passed from level to level. In addition, the sequence of working backwards ensures that the system will be result oriented. This method is useful for both data and process definition. It can be used for each independently, or both can be combined on the same diagram.

Constructs in Warnier/Orr diagrams

There are four basic constructs used on Warnier/Orr diagrams: hierarchy, sequence, repetition, and alternation. There are also two slightly more advanced concepts that are occasionally needed: concurrency and recursion.

Hierarchy

Hierarchy is the most fundamental of all of the Warnier/Orr constructs. It is simply a nested group of sets and subsets shown as a set of nested brackets. Each bracket on the diagram (depending on how you represent it, the character is usually more like a brace "{" than a bracket "[", but we call them "brackets") represents one level of hierarchy. The hierarchy or structure that is represented on the diagram can show the organization of data or processing. However, both data and processing are never shown on the same diagram.

Sequence

Sequence is the simplest structure to show on a Warnier/Orr diagram. Within one level of hierarchy, the features listed are shown in the order in which they occur. In other words, the step listed first is the first that will be executed (if the diagram reflects a process), while the step listed last is the last that will be executed. Similarly with data, the data field listed first is the first that is encountered when looking at the data, the data field listed last is the final one encountered.

Repetition

Repetition is the representation of a classic "loop" in programming terms. It occurs whenever the same set of data occurs over and over again (for a data structure) or whenever the same group of actions is to occur over and over again (for a processing structure). Repetition is indicated by placing a set of numbers inside parentheses beneath the repeating set.

Typically there are two numbers listed in the parentheses, representing the fewest and the most number of times the set will repeat. By convention the first letter of the repeating set is the letter chosen to represent the maximum.

While the minimum bound and maximum bound can technically be anything, they are most often either "(1,n)" as in the example, or "(0,n)." When used to depict processing, the "(1,n)" repetition is classically known as a "DoUntil" loop, while the "(0,n)" repetition is called a "DoWhile" loop. On the

Warnier/Orr diagram, however, there is no distinction between the two different types of repetition, other than the minimum bound value.

On occasion, the minimum and maximum bound are predefined and not likely to change: for instance the set "Day" occurs within the set "Month" from 28 to 31 times (since the smallest month has 28 days, the largest months, 31). This is not likely to change. And on occasion, the minimum and maximum are fixed at the same number.

In general, though, it is a bad idea to "hard code" a constant other than "0" or "1" in a number of times clause—the design should be flexible enough to allow for changes in the number of times without changes to the design. For instance, if a company has 38 employees at the time a design is done, hard coding a "38" as the "number of employees" within company would certainly not be as flexible as designing "(1,n)".

The number of times clause is always an operator attached to some set (i.e., the name of some bracket), and is never attached to an element (a diagram feature which does not decompose into smaller features). The reason for this will become more apparent as we continue to work with the diagrams. For now, you will have to accept this as a formation rule for a correct diagram.

Alternation

Alternation, or selection, is the traditional "decision" process whereby a determination is made to execute one process or another. The Exclusive OR symbol (the plus sign inside the circle) indicates that the sets immediately above and below it are mutually exclusive (if one is present the other is not). This diagram indicates that an Employee is either Management or Non-Management, one Employee cannot be both. It is also permissible to use a "negation bar" above an alternative in a manner similar to engineering notation. The bar is read by simply using the word "not".

Alternatives do not have to be binary as in the previous examples, but may be many-way alternatives.

Concurrency

Concurrency is one of the two advanced constructs used in the methodology. It is used whenever sequence is unimportant. For instance, years and weeks operate concurrently (or at the same time) within our calendar. The concurrency operator is rarely used in program design (since most languages do not support true concurrent processing anyway), but does come into play when resolving logical and physical data structure clashes.

Recursion

Recursion is the least used of the constructs. It is used to indicate that a set contains an earlier or a less ordered version of itself. In the classic "bill of materials" problem components contain parts and other sub-components. Sub-components also contain sub-sub-components, and so on. The doubled bracket indicates that the set is recursive. Data structures that are truly recursive are rather rare.

What are the project knowledge areas

There's a lot to learn as a project manager! **A Guide to the Project Management Body of Knowledge (PMBOK® Guide) – Fifth Edition** breaks down what project managers need to know to successfully pass their PMP® exam and also be effective in the role.

There are 10 project management knowledge areas covered by the PMBOK® Guide. They cover each of the 47 project management processes. This article provides a high-level view of each of these areas in relation to what you need to know and do as a project manager.

Project Integration Management

This is covered first in the PMBOK® Guide, but it's about bringing together everything you know so that you are managing your project holistically and not in individual process chunks. Because of that, it's easier to study this knowledge area last. Skip this section of the book and come back to it later!

Project Scope Management

'Scope' is the way to define what your project will deliver. Scope management is all about making sure that everyone is clear about what the project is for and what it includes. It covers collecting requirements and preparing the work breakdown structure. (**Here's how to breakdown tasks.**)

Project Time Management

Project time management isn't about being personally more effective. It relates to how you manage the time people are spending on their project tasks, and how long the project takes overall. This knowledge area helps you understand the activities in the project, the sequence of those activities, and how long they are going to take.

It's also where you prepare your project schedule.

Project Cost Management

Cost management is, as you'd expect, all about handling the project's finances. The big activity in this knowledge area is preparing your budget which includes working out how much each task is going to cost and then determining your project's overall budget forecast.

And, of course, it covers tracking the project's expenditure against that budget and making sure that you are still on track not to overspend.

Project Quality Management

Project quality management is quite a small knowledge area, as it only covers three processes. This area is where you'll learn about and set up the quality control and quality management activities on your project so that you can be confident the result will meet your customers' expectations.

Project Human Resource Management

Project human resource management relates to how you run your project team. First, you have to understand what resources you need to be able to complete your project, then you put your team together. After that, it's all about managing the people on the team including giving them extra skills to do their jobs, if they need it, and learning **how to motivate your team**.

Project Communications Management

Given that a project manager's job is often said to be about 80% communication, this is another small knowledge area. The three processes are planning, managing and controlling project communications. It's here that you'll write your communications plan for the project and monitor all the incoming and outgoing communications.

There are strong links with human resource management and stakeholder management too, even if these aren't explicit as I think they should be in the PMBOK® Guide.

Project Risk Management

The first step in project risk management is planning your risk management work, and then you quickly move on to identifying risks and understanding **how to assess risks on your project**.

There is a lot of detail in this knowledge area, specifically around how you perform quantitative and qualitative risk assessments. Risk management isn't a one-off activity, though, and this knowledge area also covers controlling your project risks going forward through **the project life cycle**.

Project Procurement Management

Procurement management isn't something that you'll have to do on all projects, but it is common. This knowledge area supports all your procurement and supplier work from planning what you need to buy, to going through the tendering and purchasing process to managing the work of the supplier and closing the contract when the project is finished.

Explain the structured walk through in detail

A structured walkthrough, a static testing technique performed in an organized manner between a group of peers to review and discuss the technical aspects of software development process. The main objective in a structured walkthrough is to find defects inorder to improve the quality of the product.

Structured walkthroughs are usually NOT used for technical discussions or to discuss the solutions for the issues found. As explained, the aim is to detect error and not to correct errors. When the walkthrough is finished, the author of the output is responsible for fixing the issues.

Benefits:

- Saves time and money as defects are found and rectified very early in the lifecycle.
- This provides value-added comments from reviewers with different technical backgrounds and experience.
- It notifies the project management team about the progress of the development process.
- It creates awareness about different development or maintenance methodologies which can provide a professional growth to participants.

Structured Walkthrough Participants:

- **Author -** The Author of the document under review.

- **Presenter -** The presenter usually develops the agenda for the walkthrough and presents the output being reviewed.
- **Moderator -** The moderator facilitates the walkthrough session, ensures the walkthrough agenda is followed, and encourages all the reviewers to participate.
- **Reviewers -** The reviewers evaluate the document under test to determine if it is technically accurate.
- **Scribe -** The scribe is the recorder of the structured walkthrough outcomes who records the issues identified and any other technical comments, suggestions, and unresolved questions.

Define project scope management

Project Scope management includes the processes that is required to ensure successful completion of the project. It includes all the work required to complete the project successfully. There are five processes that take place in Project Scope Management. Lets discuss these process groups of Project Scope Management.

Five Project Management Process Groups in Project Scope Management:

- Collect Requirements
- Define Scope
- Create WBS
- Verify Scope
- Control Scope

Collect Requirements:

This is the first process group in scope management. It is the process of defining and documenting stakeholders need to meet the project activities. The document for collecting requirements is developed in the project planning phase.

Define Scope:

This is the process of developing a detailed description of the Project and product. So while Collecting requirement list, all the different requirements of the Project and the resulting product or service are defined.

Create WBS:

Creating work breakdown structure is done using a technique called decomposition. It is basically the process of subdividing project deliverables and project work into smaller and more manageable components

Verify Scope:

This is the process which is a part of project monitoring and control process group. This process includes reviewing deliverables with the customer or sponsor to ensure that they are completed satisfactorily and obtaining formal acceptance of deliverables by the customer or sponsor.

Control Scope:

Control Scope is the last process group of project scope management. It is again a part of project scope monitoring the status of the project and Product scope and managing changes to the scope baseline. This process ensures that all requested changes and recommended corrective or preventive actions are processed through the integrated change control process

In project management, the term **scope** has two distinct uses: Project Scope and Product Scope.

Scope involves getting information required to start a project, and the features the product would have that would meet its stakeholder's requirements.

- Project Scope: "The work that needs to be accomplished to deliver a product, service, or result with the specified features and functions."[1]
- Product Scope: "The features and functions that characterize a product, service, or result."

Notice that Project Scope is more **work-oriented** (the hows), while Product Scope is more oriented toward **functional requirements** (the whats).
If requirements aren't completely defined and described and if there is no effective change control in a project, scope or requirement creep may ensue.

When a construction site is being built, the constructor raises a fence on the site defining the boundaries of the construction.

This process of building a fence is called scoping. Scope management is the process of defining what work is required and then making sure all of that work – and only that work – is done. Scope management plan should include the detailed process of scope determination, its management and its control.

This needs to be planned in advance before the commencement of the project during mobilization phase. Project manager must seek formal approval on a well-defined and clearly articulated scope. To identify scope, requirements must be gathered from all stakeholders. Gathering requirements from only a few stakeholders or only the sponsor might lead to incorrect definition of scope. Large projects require more time, effort and resources to gather requirements and thus defining the scope is important. Scope definition helps us make sure that we are doing all the work but only the work included in the scope management plan. Gold plating a project (adding extras) is not allowed. Changes in scope must be taken into consideration all the knowledge areas of project management such as time, cost, risk, quality, resources and customer satisfaction. Integrated change management process is required to approve changes to scope of a project. Integrated Change Management includes updating of Change Request Form by Change Originator and also tracking the change on Change Control Register.

Scope Management is the listing of the items to be produced or tasks to be done to the required quantity, quality and variety, in the time and with the resources available and agreed upon, and the modification of those variable constraints by dynamic flexible juggling in the event of changed circumstance called as Scope creep.

Explain RFP and RFQ in brief

A **request for proposal** (**RFP**) is a document that solicits proposal, often made through a bidding process, by an agency or company interested in procurement of a commodity, service, or valuable asset, to potential suppliers to submit business proposals. It is submitted early in the procurement cycle, either at the preliminary study, or procurement stage.

An RFP is used where the request requires technical expertise, specialized capability, or where the product or service being requested does not yet exist, and the proposal may require research and development to create whatever is being requested.

The RFP presents preliminary requirements for the commodity or service, and may dictate to varying degrees the exact structure and format of the supplier's response. Effective RFPs typically reflect the strategy and short/long-term business objectives, providing detailed insight upon which suppliers will be able to offer a matching perspective.

Similar requests include a request for quotation (RFQ), whereby the customer may simply be looking for a price quote, and a request for

information (RFI), where the customer needs more information from vendors before submitting an RFP. An RFI is typically followed by an RFP or RFQ.

In principle, an RFP:

- Informs suppliers that an organization is looking to procure and encourages them to make their best effort.
- Requires the company to specify what it proposes to purchase. If the requirements analysis has been prepared properly, it can be incorporated quite easily into the Request document.
- Alerts suppliers that the selection process is competitive.
- Allows for wide distribution and response.
- Ensures that suppliers respond factually to the identified requirements.
- Is generally expected to follow a structured evaluation and selection procedure, so that an organization can demonstrate impartiality - a crucial factor in public sector procurements.
- RFPs often include specifications of the item, project or service for which a proposal is requested. The more detailed the specifications, the better the chances that the proposal provided will be accurate. Generally, RFPs are sent to an approved supplier or vendor list.

- A **Request for Quotation (RFQ)** is used when discussions with bidders are not required (mainly when the specifications of a product or service are already known) and when price is the main or only factor in selecting the successful bidder. An RFQ may also be used prior to issuing a full-blown RFP to determine general price ranges. In this scenario, products, services or suppliers may be selected from the RFQ results to bring in to further research in order to write a more fully fleshed out RFP. In commercial business practice, the RFQ is the most popularly used form of RFx, with many companies not understanding the distinction between the RFx's, and so defaulting to RFQ.

- A **Request for Qualifications (RFQ)** also known as **Pre-Qualification Questionnaire(PQQ)** is a document often distributed before initiation of the RFP process. It is used to gather vendor information from multiple companies to generate a pool of prospects. This eases the RFP review process by preemptively short-listing candidates which meet the desired qualifications.

- A **Request for Solution (RFS)** is similar to a RFP, but more open and general. This allows the vendor or supplier the most flexibility of all RFx's in expressing their solution, or their product and service combination.

- A **Request for Tender (RFT)**, also known as **Invitation to Tender (ITT)**, is more commonly used by governments.

What is PROJECT CHARTER

Introduction

Project Charter refers to a statement of objectives in a project. This statement also sets out detailed project goals, roles and responsibilities, identifies the main stakeholders, and the level of authority of a project manager.

It acts as a guideline for future projects as well as an important material in the organization's knowledge management system.

The project charter is a short document that would consist of new offering request or a request for proposal. This document is a part of the project management process, which is required by Initiative for Policy Dialogue (IPD) and Customer Relationship Management (CRM).

The Role of Project Charter

Following are the roles of a Project Charter:

- It documents the reasons for undertaking the project.
- Outlines the objectives and the constraints faced by the project.
- Provides solutions to the problem in hand.
- Identifies the main stakeholders of the project.

Benefits of Project Charter

Following are the prominent benefits of Project Charter for a project:

- It improves and paves way for good customer relationships.
- Project Charter also works as a tool that improves project management processes.
- Regional and headquarter communications can also be improved to a greater extent.
- By having a project charter, project sponsorship can also be gained.

- Project Charter recognizes senior management roles.
- Allows progression, which is aimed at attaining industry best practices.

Elements in Project Charter

Since project charter is a project planning tool, which is aimed at resolving an issue or an opportunity, the below elements are essential for a good charter project.

For an effective charter project, it needs to address these key elements:

- Identity of the project.
- Time: the start date and the deadline for the project.
- People involved in the project.
- Outlined objectives and set targets.
- The reason for a project charter to be carried out, often referred to as 'business case'.
- Detailed description of a problem or an opportunity.
- The return expected from the project.
- Results that could be expected in terms of performance.
- The expected date that the objectives is to be achieved.
- Clearly defined roles and responsibilities of the participants involved.
- Requirement of resources that will be needed for the objectives to be achieved.
- Barriers and the risks involved with the project.
- Informed and effective communication plan.

Out of all above elements, there are three most important and essential elements that need further elaboration.

Project Scope

As the name denotes, it refers to the scope that the project will give the business if they undertake the project.

Before doing a project, the following concerns need to be addressed:

- The within scope and out of scope needs to be considered.
- The process that each team will focus upon.
- The start and end points for a process.
- Availability of resources.
- Constraints under which the team will work.
- Time limitations .
- The impact on the normal workload if the project is to be undertaken.