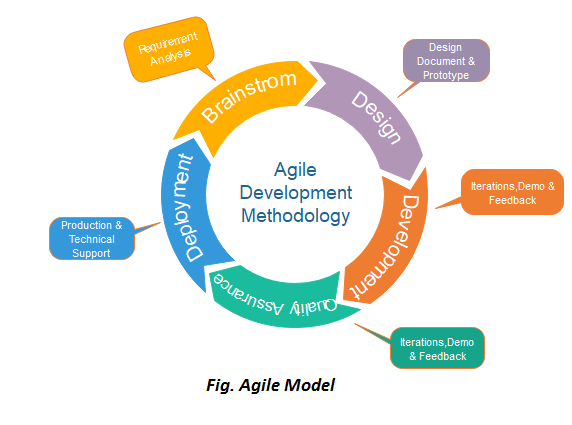


### What is agile process:

* Agile is an iterative and incremental software development methodology that focuses on delivering software products in small, frequent releases.
* Agile methodology emphasizes collaboration, flexibility, and customer satisfaction.
* Agile methodology is based on the Agile Manifesto, which values individuals and interactions, working software, customer collaboration, and responding to change.
* The Agile process typically involves breaking down the software development process into smaller iterations, called sprints, that last anywhere from one to four weeks.
* Each sprint involves planning, design, development, testing, and review activities.
* The Agile process encourages continuous feedback and communication among the development team, stakeholders, and customers.
* Agile methodology includes various frameworks such as Scrum, Kanban, and Extreme Programming (XP).



* Phases in agile development process:
  + Planning
  + Design
  + Development
  + Testing
  + Deployment
  + Release
  + Maintenance
* As agile is an iterative and incremental process and all phrases can be carried out in any order, multiple times in the course of the development process

### What are Agility principles:

* Customer satisfaction through early and continuous software delivery.
* Embrace changes in customer requirements, even late in the development cycle.
* Deliver working software frequently, with a preference for shorter timescales.
* Close and continuous collaboration between business stakeholders and the development team.
* Build projects around motivated individuals and give them the resources and support they need.
* Face-to-face communication is the most efficient and effective means of conveying information within a development team.
* Working software is the primary measure of progress.
* Maintain a sustainable development pace by promoting a healthy work-life balance.
* Continuous attention to technical excellence and good design enhances agility.
* Simplicity, as the art of maximizing the amount of work not done, is essential.
* Self-organizing teams are most effective in delivering software solutions.
* Regular retrospectives that focus on how to improve and adjust the team’s behavior are necessary for growth and development.

### What is white box and black box testing?

Black Box Testing:

* Black box testing is a software testing technique that focuses on the functionality of the software, without any knowledge of the internal workings of the system.
* The tester does not have access to the source code, and testing is based on the input and output of the software.
* The tester does not need any programming knowledge to perform black box testing.
* Black box testing is used to validate the software's functional requirements and ensure that it meets the customer's needs.
* The testing techniques used in black box testing include boundary value analysis, equivalence partitioning, and error guessing.
* Black box testing is useful for uncovering defects related to incorrect or missing functionality, user interface errors, and performance issues.

White Box testing:

* White box testing is a software testing technique that focuses on the internal workings of the system, including the source code, structure, and logic.
* The tester has access to the source code and can use programming knowledge to design and execute test cases.
* The goal of white box testing is to ensure that the software functions correctly at the code level and adheres to the specified coding standards.
* White box testing is useful for detecting defects related to control flow errors, data flow errors, and boundary value errors.
* The testing techniques used in white box testing include statement coverage, branch coverage, and path coverage.
* White box testing is usually performed by developers as part of the software development process, but it can also be performed by dedicated software testers

### What is SRS - Software requirement specification

Software Requirements Specification (SRS) is a document that describes the software product's requirements and specifications. It provides a clear and detailed description of what the software is supposed to do, how it should function, and how it should be designed.

The SRS document outlines the software's functional and non-functional requirements, including its features, performance, user interface, and other technical specifications. It is usually created by the business analyst or the project manager in collaboration with the software development team and the stakeholders.

The SRS document typically includes the following components:

* Introduction: Provides an overview of the software product, its purpose, scope, and the intended audience.
* Functional Requirements: Describes the software's functional requirements, including its features, user interface, data management, and other specifications.
* Non-Functional Requirements: Describes the software's non-functional requirements, such as its performance, reliability, security, and compatibility.
* User Interface: Describes the user interface of the software, including its design, layout, and navigation.
* Assumptions and Dependencies: Describes the assumptions and dependencies that are necessary for the software product to work as intended.
* Constraints: Describes any constraints that may limit the development or use of the software product, such as budget or time constraints.
* Appendices: Includes any additional information that is relevant to the software requirements, such as diagrams, tables, or supporting documentation.

The SRS document is an essential part of the software development process as it provides a common understanding of the software's requirements and specifications among the stakeholders, the development team, and the quality assurance team. It also serves as a reference document throughout the software development lifecycle to ensure that the software meets the specified requirements and is delivered on time and within budget.

### 

### 

### What is the CMM - capability maturity model?

The Capability Maturity Model (CMM) is a model used to assess the maturity of an organization's software development processes. Here are some key points about the model:

The CMM was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University.

The model is divided into five levels of maturity, each representing a different stage in the development of an organization's processes:

* a. Level 1 - Initial: Processes are ad hoc and chaotic, with little or no formal procedures in place.
* b. Level 2 - Managed: Processes are characterized for individual projects and are reactive. The organization has established basic project management practices to track cost, schedule, and functionality.
* c. Level 3 - Defined: Processes are well characterized and understood, and are proactive in nature. The organization has established a standard process for developing and maintaining software across the organization.
* d. Level 4 - Quantitatively Managed: Processes are measured and controlled using statistical and other quantitative techniques. The organization is able to quantitatively predict the quality of software it produces.
* e. Level 5 - Optimizing: Processes are continuously improved based on the feedback received from quantitative measurements and process improvements.

The CMM provides a framework for organizations to assess their current level of maturity and develop a roadmap for improvement.

The model is widely used in the software industry as a benchmark for software process improvement.

The CMM has evolved into the Capability Maturity Model Integration (CMMI), which combines the best practices from several different process improvement models.

### What is PMBOK?

* PMBOK stands for Project Management Body of Knowledge. It is a guidebook or a standard that outlines the best practices, processes, and knowledge areas that are generally accepted as the industry standard for project management.
* The PMBOK guide is developed and maintained by the Project Management Institute (PMI), a non-profit professional association for project managers. It provides a framework for managing projects of different sizes, complexities, and industries.
* The PMBOK guide is currently in its sixth edition and includes ten knowledge areas, five process groups, and 49 project management processes. The guide covers a wide range of topics such as project integration, scope, time, cost, quality, communication, risk, procurement, and stakeholder management.
* The PMBOK guide is widely used by project managers as a reference and a source of guidance for managing projects effectively. It is also the basis for PMI certification programs such as the Project Management Professional (PMP) certification.

### PMBOK Knowledge areas?

1. Project Integration Management: This knowledge area includes the processes and activities required to ensure that all project components are effectively coordinated, integrated, and managed throughout the project lifecycle.
2. Project Scope Management: This knowledge area includes the processes and activities required to define, document, and control the scope of a project, and to ensure that it remains within the defined boundaries and meets the project objectives.
3. Project Schedule Management: This knowledge area includes the processes and activities required to develop and maintain a project schedule, track progress against the schedule, and manage changes to the schedule.
4. Project Cost Management: This knowledge area includes the processes and activities required to estimate, budget, and control costs associated with a project, and to ensure that the project is delivered within the approved budget.
5. Project Quality Management: This knowledge area includes the processes and activities required to ensure that the project meets the defined quality standards and that customer expectations are met or exceeded.
6. Project Resource Management: This knowledge area includes the processes and activities required to identify, acquire, and manage resources, such as people, equipment, and materials, required for project success.
7. Project Communications Management: This knowledge area includes the processes and activities required to effectively communicate with stakeholders, ensure that information is distributed, understood, and acted upon appropriately, and manage stakeholder expectations.
8. Project Risk Management: This knowledge area includes the processes and activities required to identify, assess, and manage risks that may impact the project objectives and outcomes.
9. Project Procurement Management: This knowledge area includes the processes and activities required to acquire goods and services from external vendors, suppliers, or contractors required for project success.
10. Project Stakeholder Management: This knowledge area includes the processes and activities required to identify, manage, and engage stakeholders, such as customers, sponsors, and team members, and to ensure that their needs and expectations are addressed throughout the project lifecycle.

### Explain software testing process

* Test Planning: In this phase, the testing team identifies the scope of the testing, objectives, and timelines, and develops a test plan that includes details about the testing approach, test cases, and the resources required.
* Test Case Design: In this phase, the testing team creates detailed test cases for each identified functional and non-functional requirement of the software system. Test cases typically include inputs, expected outputs, and test procedures.
* Test Environment Setup: In this phase, the testing team sets up the test environment, including hardware, software, and testing tools required to execute the test cases.
* Test Execution: In this phase, the testing team executes the test cases to validate the software system. The testing team may use manual or automated testing techniques.
* Test Reporting: In this phase, the testing team documents the results of the testing, including the test cases executed, defects identified, and their severity. The testing team also provides recommendations on how to fix the identified issues.
* Defect Retesting: In this phase, the testing team retests the defects that were identified and fixed by the development team to ensure that they have been resolved.
* Regression Testing: In this phase, the testing team executes all previously passed test cases to ensure that the software system is still working as expected after the fixes or changes have been made.
* Test Closure: In this phase, the testing team prepares a test closure report that includes the testing summary, testing effort, and recommendations for future improvements.

### Software testing strategies:

Software testing strategies are a set of approaches and methods used to ensure that software products meet quality standards and perform as expected. Here are some key points about software testing strategies

Software testing strategies are an integral part of the software development life cycle (SDLC) and are used to evaluate software quality at various stages of development.

Different software testing strategies are used depending on the nature of the software product, the project goals, and the budget and timeline constraints.

Some common software testing strategies include manual testing, automated testing, and exploratory testing.

1. Manual testing:
   1. Manual testing is a fundamental part of the software development life cycle (SDLC) and is typically performed at various stages of development to ensure software quality.
   2. Manual testing involves a human tester executing test cases that are designed to evaluate the software product's functionality, usability, performance, and other quality attributes.
   3. Manual testing is usually performed by a dedicated testing team or a tester who is responsible for ensuring that the software product meets quality standards.
   4. Manual testing requires testers to have domain expertise and knowledge of the software product to be able to identify quality issues effectively.
   5. Manual testing can be time-consuming and labor-intensive, and it may not be feasible to test all possible scenarios manually.
   6. Manual testing may involve different types of testing, including functional testing, usability testing, performance testing, and security testing.
2. Automated testing:
   1. Is typically performed after manual testing to ensure software quality.
   2. Automated testing involves using software tools to execute pre-written test cases that are designed to evaluate the software product's functionality, usability, performance, and other quality attributes.
   3. Automated testing is usually performed by a dedicated testing team or a tester who is responsible for ensuring that the software product meets quality standards.
   4. Automated testing can be faster and more efficient than manual testing, as it allows testers to execute large numbers of test cases quickly and accurately.
   5. Automated testing requires testers to have programming skills and knowledge of the software tools used for testing.
   6. Automated testing can involve different types of testing, including functional testing, regression testing, performance testing, and security testing.
3. Exploratory testing:

Exploratory testing is a type of software testing in which testers use their domain expertise and creativity to explore the software product and identify potential quality issues. Here are some key points about exploratory testing:

* 1. Exploratory testing is an approach to software testing that emphasizes creativity and critical thinking.
  2. Exploratory testing is typically performed by a tester who has knowledge of the software product and its intended use.
  3. Exploratory testing involves using ad-hoc testing techniques to evaluate the software product for defects, errors, and other quality issues.
  4. Exploratory testing is not scripted, and the tester is free to explore the software product in any way they see fit.
  5. Exploratory testing can be used to supplement other types of testing, such as manual and automated testing.
  6. Exploratory testing is particularly useful for identifying defects and errors that may not be identified through other types of testing

### Debugging:

Debugging is the process of finding and fixing defects, errors, and other issues in software code.

* Debugging is a critical part of the software development life cycle (SDLC) and is typically performed after coding and testing to ensure that the software product is free of defects and errors.
* Debugging involves identifying the root cause of a defect or error and then modifying the software code to correct the issue.
* Debugging can be performed manually or using software tools, such as debuggers, profilers, and code analyzers.
* Debugging requires testers and developers to have strong analytical and problem-solving skills and to be able to understand and interpret software code.
* Debugging can be time-consuming and challenging, particularly when defects or errors are difficult to reproduce or are caused by complex interactions between different parts of the software code.
* Debugging can involve different types of defects and errors, including syntax errors, logical errors, runtime errors, and memory leaks.
* Debugging is usually performed in an iterative process, where testers and developers identify and fix defects and then test the software product again to ensure that the issues have been resolved.
* Debugging is an ongoing process throughout the software development life cycle, as defects and errors can be introduced at any stage of development.
* Debugging is important for ensuring the quality and reliability of software products and for meeting user expectations for functionality and performance.

### RFP - Request for proposal:

RFP stands for Request for Proposal. It is a document that is used to solicit proposals from potential vendors or suppliers for a project or service.

* RFPs are typically used in procurement processes to ensure that potential vendors or suppliers have a clear understanding of the project or service requirements.
* RFPs are typically issued by the organization that is seeking proposals and will include a detailed description of the project or service, along with instructions on how to respond to the RFP.
* RFPs may include evaluation criteria, such as cost, technical expertise, and delivery timelines, that will be used to evaluate proposals.
* RFP responses are typically reviewed by a selection committee, which will evaluate each proposal based on the evaluation criteria and select the vendor or supplier that best meets the project or service requirements.
* RFPs may be issued as open or closed RFPs. In an open RFP, any vendor or supplier can respond to the RFP, while in a closed RFP, only vendors or suppliers that have been pre-qualified can respond.
* RFPs may also include a request for additional information or clarification from vendors or suppliers, which may be used to help evaluate proposals.
* RFPs may be used for a variety of projects or services, including software development, construction projects, and professional services such as consulting and marketing.
* Responding to an RFP can be time-consuming and requires careful attention to the project or service requirements and the evaluation criteria.
* RFPs are an important part of the procurement process and can help organizations find the best vendor or supplier for their needs while ensuring a fair and transparent evaluation process.

### RFP v/s SRS:

* Purpose: The purpose of an RFR is to request resources such as people, materials, or equipment to support a project. An SRS, on the other hand, specifies the requirements for a software system that is to be developed.
* Content: An RFR typically includes information about the project, the type and quantity of resources needed, and any constraints or limitations. An SRS, on the other hand, includes information about the functional and non-functional requirements of the software system, as well as any constraints or assumptions.
* Audience: An RFR is typically directed towards the organization's procurement or resource management department, while an SRS is directed towards the software development team.
* Timing: An RFR is usually issued before a project begins, while an SRS is typically created during the planning phase of a software development project.
* Scope: An RFR is usually focused on a specific project or initiative, while an SRS covers the complete software system being developed.

### Project Charter and Project Scope.

Project Charter and Project Scope are two important documents that help define and establish a clear understanding of a project's objectives, scope, and expectations. Here are some key points about each:

Project Charter:

* A project charter is a high-level document that provides an overview of the project's purpose, objectives, scope, and stakeholders.
* A project charter is usually created by the project sponsor or project manager and is used to communicate the project's goals and objectives to stakeholders and team members.
* A project charter typically includes information such as project timelines, budget, resources, and risks.
* A project charter helps ensure that everyone involved in the project understands the project's purpose and objectives, and it provides a foundation for making decisions throughout the project lifecycle.
* A project charter is usually created early in the project planning process and is used as a reference throughout the project.

Project Scope:

* Project scope defines the boundaries of the project, including the goals, objectives, deliverables, tasks, and requirements that are included in the project.
* Project scope helps ensure that the project team understands what is and isn't included in the project, which can help prevent scope creep.
* Project scope is typically outlined in a project scope statement, which is a detailed document that includes the project's goals, objectives, deliverables, and requirements.
* Project scope is usually defined in collaboration with stakeholders and is used to ensure that the project meets the needs of all parties involved.
* Project scope can be revised throughout the project lifecycle as new information becomes available or requirements change.

SCM and SCM repositories:

* SCM is a set of practices, processes, and tools used to manage changes to software code, documents, and other artifacts during the software development life cycle.
* SCM helps developers track changes, manage versions, and control access to software artifacts.
* SCM tools are used to store and manage software artifacts in SCM repositories. These repositories are typically centralized or distributed.
* Centralized SCM repositories store all artifacts in a single location, and developers must check out and check in artifacts to make changes.
* Distributed SCM repositories, also known as Git repositories, allow developers to make changes locally and then push their changes to a central repository for integration with other developers' changes.
* SCM repositories typically include features such as version control, branching and merging, access control, and change tracking.
* Popular SCM tools include Git, Subversion (SVN), Mercurial, and Perforce.
* SCM is critical for software development teams to manage code changes, collaborate effectively, and maintain the quality and integrity of their software products.
* SCM repositories can also be used to store and manage other software artifacts, such as documentation, configuration files, and test scripts.
* SCM is an important part of DevOps practices, enabling teams to automate and streamline the software development process, from code changes to deployment.