

Software Engineering

T.E. Semester –VI
Choice Based Credit Grading Scheme with Holistic Student Development
Proposed Syllabus under Autonomy

BE (Computer Engineering)					SEM: VI				
Course Name: Software Engineering					Course Code: PCC-CS503				
Teaching Scheme (Program Specific)					Examination scheme				
Modes of Teaching / Learning / Weightage					Modes of Continuous Assessment / Evaluation				
Hours Per Week-					Theory (100)		Practical/Oral (25)	Term-work	Total
Theory	Tu.	Practical	Contact Hours	Credits	ISE	IE	ESE	PR/OR	TW
3	-	2	5	4	20	20	60	25	25
150									
IA: In-Semester Assessment - Paper Duration – 1 Hour									
ESE: End Semester Examination - Paper Duration - 2 Hours									
Prerequisite: Object Oriented Programming, Frontend Backend connectivity									

Course Objective: The objective of the course is to introduce to the students about the development of software product, the processes that provides a framework for the engineering methodologies and practices. Also to give the information regarding the including the analysis, design, testing methodologies and quality assurance.

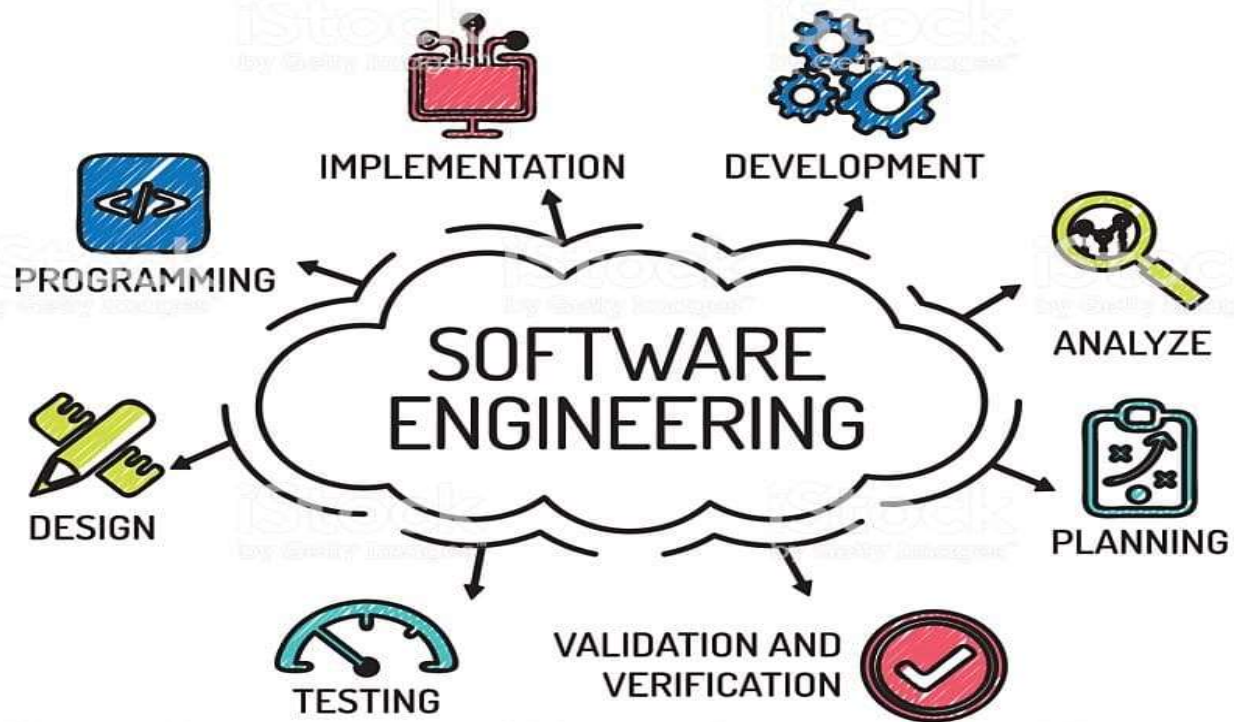
Course Outcomes: Students will be able to:

SN	Course Outcomes	RBT level
1	Understand the use of basic and advanced models in software Engineering.	L1, L2
2	Understand and apply the scenarios to design the UML diagrams.	L1, L2, L3
3	Understand and apply the different techniques of project estimation and understand the tracking methods.	L1, L2, L3
4	Identify the design concepts and apply them to the project.	L1, L2, L3, L4
5	Identify and estimate risks, manage the change to assure quality in software project.	L1, L2, L3, L4, L5
6	Apply the principles of testing and develop test plan for the project.	L1, L2, L3, L4, L5, L6

Module No.	Topics	Hrs.	RBT Levels
1.0	Introduction Introduction to software engineering, Importance of Software engineering Software Process, Various models for Software Development (Waterfall, Spiral, Agile (Scrum), V-Model, RAD, DevOps), Capability Maturity Model (CMM).	6	L1, L2
2.0	Requirements Analysis and Modelling Requirement Elicitation, Software requirement specification (SRS), Data Flow Diagram (DFD), Feasibility Analysis, Cost- Benefit Analysis, Developing Use Cases (UML), Requirement Model – Scenario-based model, Class-based model, Behavioural model.	8	L1, L2, L3
3.0	Project Scheduling and Tracking Software Project Estimation: LOC, FP, Software Project Scheduling Principles , Empirical Estimation Models - COCOMO , COCOMO II Model, Estimation for agile: planning poker, user story planning, Benefits of Agile Estimation , Project scheduling: Timeline charts, CPM, Fish-bone diagram	4	L1, L2, L3
4.0	Software Design Design Concepts, Characteristics of Good Design, Effective Modular Design – Cohesion and Coupling. Architectural Styles, UI Design.	8	L1, L2, L3, L4
5.0	Software Risk, Configuration Management & Quality Assurance Risk Identification, Risk Assessment, Risk Projection, RMMM, Software Configuration management, Software Quality Assurance: Software Reliability, Formal Technical Review (FTR), Walk-through, Quality Assurance Standards.	8	L1, L2, L3, L4, L5
6.0	Software Testing and Maintenance Software Testing, Unit testing, Integration testing Verification, Validation Testing, System Testing, Test plan, White-Box Testing, Basis Path Testing, Control Structure Testing, Black-Box Testing, Software maintenance and its types, Software Re-engineering, Reverse Engineering, Case Study on Artificial Intelligence and Computer Networks and Security, Real Time Applications of Software Engineering.	11	L1, L2, L3, L4, L5, L6
Total Hours		45	

Software Engineering ...?

- ▶ Software engineering is defined as a process of analyzing user requirements and then designing, building, and testing software application which will satisfy those requirements.
- ▶ Software Engineering is a collection of techniques, methodologies and tools that help with the production of
 - ▶ a high quality software system
 - ▶ with a given budget
 - ▶ before a given deadline
 - ▶ while change occurs.



Requirement of software engineering as a subject:


- ▶ Students are proficient in programming language , however they have lack of experience in analysis and design of a system.
- ▶ To learn technical aspects of analysis and design of complex systems.


Objective :


- ▶ Develop complex system in context of frequent change.
- ▶ Produce high quality software with in time limits.
- ▶ Develop technical knowledge
- ▶ Develop managerial knowledge.
- ▶ Many software became over budget.
- ▶ Larger software was difficult and quite expensive to maintain.
- ▶ Demand for new software increased faster compared with the ability to generate new software.

Importance :

- ▶ Reduces complexity
- ▶ Reduces the software cost
- ▶ Reduces time
- ▶ Handling big projects
- ▶ Reliable software
- ▶ Effectiveness

- 
- ▶ **Reduces complexity** : Major projects are difficult and complex to develop. This project can be subdivided into smaller parts and each of those parts can be worked upon independently.
 - ▶ **Reduces the software cost**: Utilizing the method of software engineering, the development process can be reduced because of a planned approach by the programmer which helps the programmer to remove the processes that are not necessary. therefore the cost of the software production and the development of the product is highly reduced.

- 
- ▶ **Reducing time** : Software engineering is the process of making a perfect production plan for the development of software.
 - ▶ Creating software according to the software engineering approach then it will decrease the amount of time taken in the completion of the software.
 - ▶ **Handling projects** : The process has to maintain the stages of the development so after every single month it can check its development and track its progress as the process provides many resources to the project and therefore it requires the project to be completed in time. Therefore, in this case, the software engineering approach is very beneficial to handle these break projects without any problem.

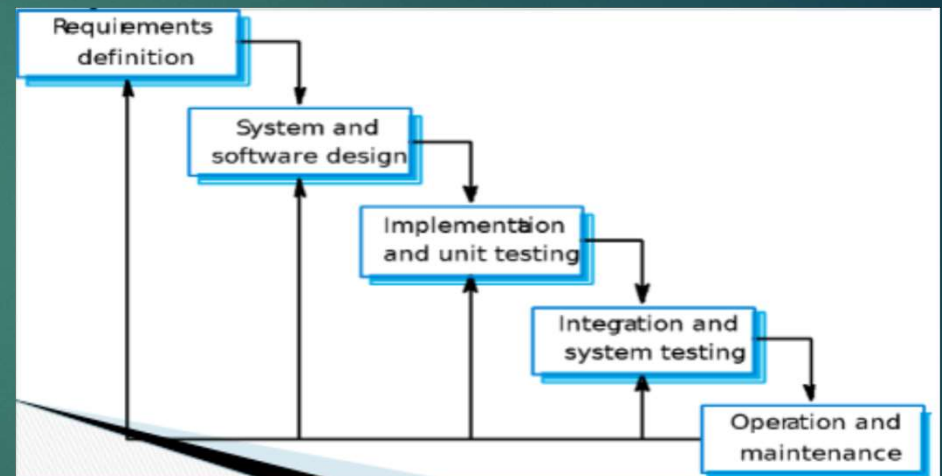
- 
- ▶ **Reliable software** : There should be no bugs and errors and also it should be reliable therefore the software that the developer has developed should work at least for a given span of time.
 - ▶ **Effectiveness** : The software that is developed by the company is only effective if it has followed the standards that the company provides. These standards are the main focus of the company and due to the software standards, the software has become more effective.

Software life cycle models :

- ▶ The following are the software life cycles models,
- ▶ Waterfall
- ▶ RAD
- ▶ Spiral
- ▶ Open source
- ▶ Agile process

Waterfall model :

- ▶ This model is also known as the linear sequential model or the software life cycle.



Why Waterfall model ...?

- ▶ **Requirements are complex**
- ▶ The client does not know the functional requirements in advance
- ▶ **Requirements may be changing**
- ▶ Technology enablers introduce new possibilities to deal with nonfunctional requirements
- ▶ **Frequent changes are difficult to manage**
- ▶ Identifying milestones and cost estimation is difficult
- ▶ **There is more than one software system**
- ▶ New system must be backward compatible with existing system ("legacy system")
- ▶ Phased development: Need to distinguish between the system under development and already released

Advantages :

- ▶ Testing is inherent to every phase of the waterfall model
- ▶ It is an enforced disciplined approach
- ▶ It is documentation driven, that is, documentation is produced at every stage.

Disadvantages :

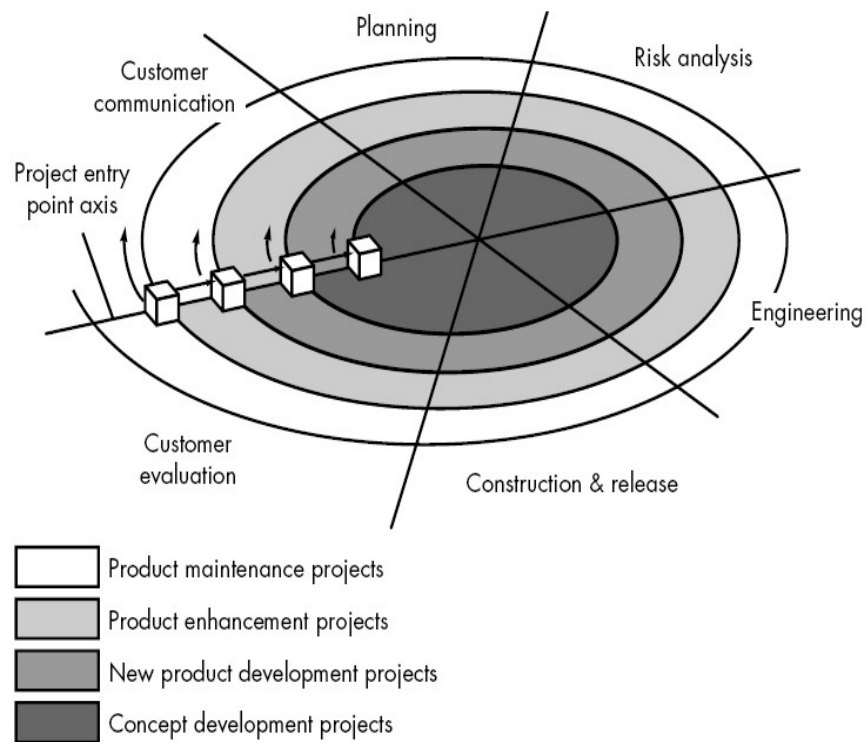
- ▶ Projects rarely flow its sequential flow as we cannot go back to introduce change.
- ▶ Small changes or errors that arise in the completed software may cause a lot of problem.
- ▶ it happens that the client is not very clear of what he exactly wants from the software. Any changes that he mentions in between may cause a lot of confusion.
- ▶ The greatest disadvantage of the waterfall model is that until the final stage of the development cycle is complete, a working model of the software does not lie in the hands of the client

Spiral Model

- ▶ The *spiral model*, originally proposed by Boehm , is an evolutionary software process model that couples the **iterative nature of prototyping** with the **controlled and systematic aspects of the linear sequential model**.
- ▶ It provides the potential for rapid development of incremental versions of the software.
- ▶ Using the spiral model, software is developed in a series of incremental releases.
- ▶ During early iterations, the incremental release might be a paper model or prototype.
- ▶ Each loop in the spiral represents a phase in the process.
- ▶ Risks are explicitly assessed and resolved throughout the process.

Spiral Model Diagram

FIGURE 2.8
A typical spiral model



Spiral model sectors

- ▶ A spiral model is divided into a number of framework activities, also called *task Regions*.
- ▶ **Customer communication**—tasks required to establish effective communication between developer and customer.
- ▶ **Planning**—tasks required to define resources, timelines, and other project related information.
- ▶ **Risk analysis**—tasks required to assess both technical and management risks.
- ▶ **Engineering**—tasks required to build one or more representations of the application.
- ▶ **Construction and release**—tasks required to construct, test, install, and provide user support (e.g., documentation and training).

Advantages of Spiral model

- ▶ It promotes reuse of existing software in early stages of development.
- ▶ Allows quality objectives to be formulated during development.
- ▶ Provides preparation for eventual evolution of the software product.
- ▶ Eliminates errors and unattractive alternatives early.

Disadvantages of Spiral Model

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

V Model

- ▶ The V-model is a type of SDLC model where the process executes in a sequential manner in a V-shape.
- ▶ It is also known as the Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage.
- ▶ The development of each step is directly associated with the testing phase.
- ▶ The next phase starts only after completion of the previous phase i.e., for each development activity, there is a testing activity corresponding to it.

Verification Phases:

- ▶ **Business Requirement Analysis:**
- ▶ **System Design:**
- ▶ **Architectural Design:**
- ▶ **Module Design**
- ▶ **Coding Phase**

Validation Phases

- ▶ Unit Testing
- ▶ Integration testing:
- ▶ System Testing
- ▶ User Acceptance Testing (UAT)

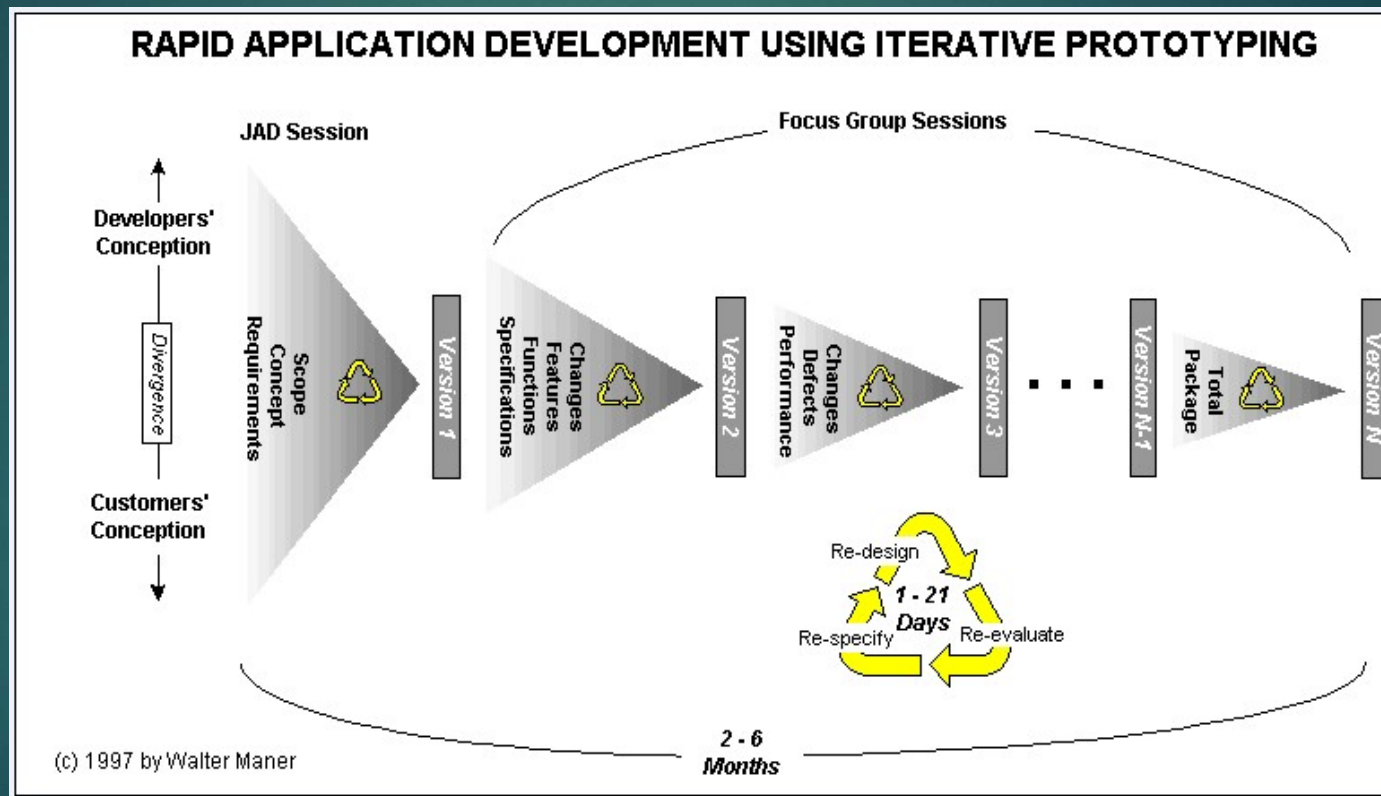
Advantages

- ▶ This is a highly disciplined model and Phases are completed one at a time.
- ▶ V-Model is used for small projects where project requirements are clear.
- ▶ Simple and easy to understand and use.
- ▶ It enables project management to track progress accurately.
- ▶ Clear and Structured Process: The V-Model provides a clear and structured process for software development, making it easier to understand and follow.
- ▶ Emphasis on Testing: The V-Model places a strong emphasis on testing, which helps to ensure the quality and reliability of the software.
- ▶ Improved Traceability: The V-Model provides a clear link between the requirements and the final product, making it easier to trace and manage changes to the software.
- ▶ Better Communication: The clear structure of the V-Model helps to improve communication between the customer and the development team.

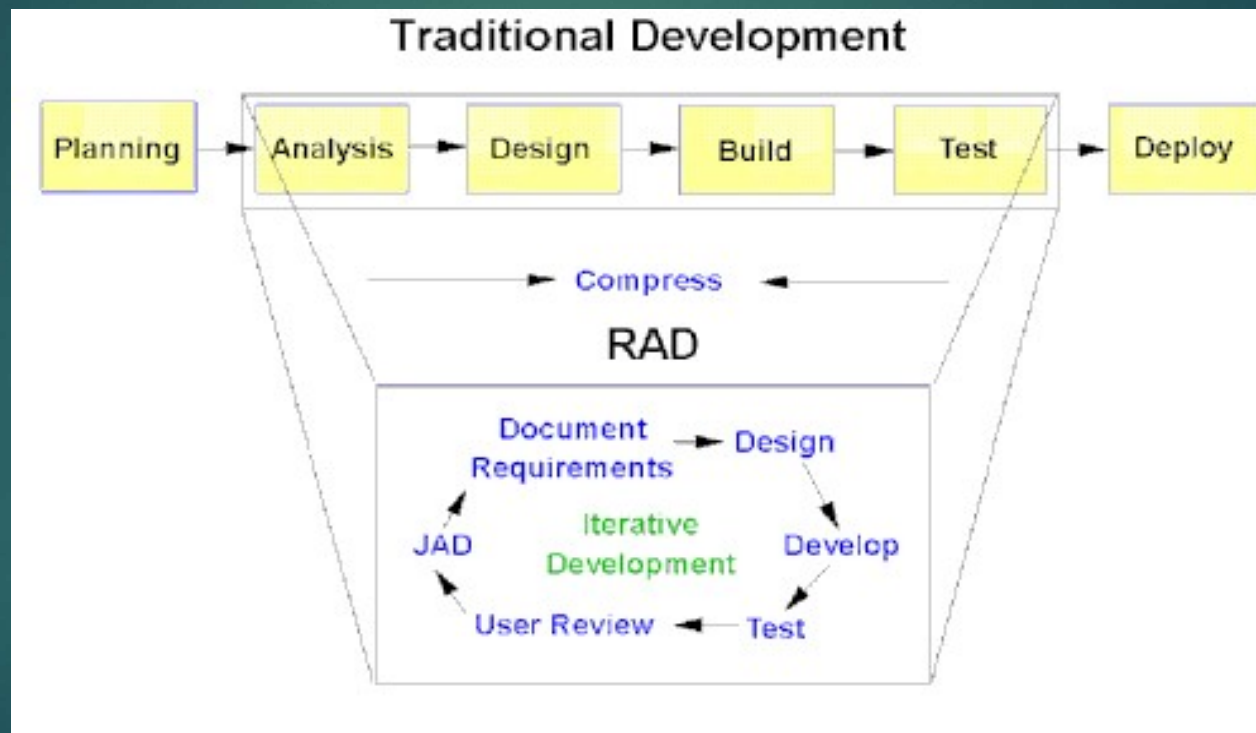
Disadvantages

- ▶ High risk and uncertainty.
- ▶ It is not a good for complex and object-oriented projects.
- ▶ It is not suitable for projects where requirements are not clear and contains high risk of changing.
- ▶ This model does not support iteration of phases.
- ▶ It does not easily handle concurrent events.
- ▶ Inflexibility: The V-Model is a linear and sequential model, which can make it difficult to adapt to changing requirements or unexpected events.
- ▶ Time-Consuming: The V-Model can be time-consuming, as it requires a lot of documentation and testing.
- ▶ Overreliance on Documentation: The V-Model places a strong emphasis on documentation, which can lead to an overreliance on documentation at the expense of actual development work.

RAD Model



Traditional VS RAD



RAD Definition

- ▶ Rapid Application development (RAD) is an incremental software development process model that emphasizes an extremely short development cycle (anywhere from 60-90 days). The RAD model is a high-speed adaptation of the linear sequential model / Waterfall model. The RAD approach encompasses the following phases:

RAD approach encompasses the following phases:

1. Business Modelling

What information drives the business processes?

What information is generated?

Who generates it?

Where does the information flow?

Who processes it?

RAD approach encompasses the following phases: contd..

2.Data Modelling

The information flow defined as part of the business modeling phase is refined into **a set of data objects** that are needed to support the business.

3. Process modeling

Processing descriptions are created for **adding, modifying, deleting or retrieving** a data object.

RAD approach encompasses the following phases: contd..

4. Application generation

RAD assumes the use of fourth generation techniques. Rather than creating software using conventional third generation programming languages,

RAD process **works to use the automated tools** to facilitate the construction of the software.

4. Testing and turnover

This saves time, money and the overall time to test an application also reduces considerably.

Advantages for using RAD

- ▶ To converge early toward a design acceptable to the customer and feasible for the developers
- ▶ To limit a project's exposure to the forces of change
- ▶ To save development time, possibly at the expense of economy or product quality

Disadvantages for using RAD

- ▶ To prevent cost overruns
(RAD needs a **team already disciplined in cost management**)
- ▶ To prevent runaway schedules
(RAD needs a team already disciplined **in time management**)

Agile Process

▶ Agility

- ▶ The ability to both create and respond to change in order to profit in a turbulent business environment
 - ▶ Companies need to determine the amount of agility so they need to be competitive
- ▶ Agile and Scrum are two closely related methodologies often used together in software development and project management.
- ▶ Agile is a broader set of principles and values, while Scrum is a specific framework that falls under the Agile umbrella.

Agile Principles



- 1) Customer Satisfaction through Continuous Delivery- Agile focuses on delivering valuable software to customers in smaller increments, ensuring that they receive continuous value.
- 2) Embrace Changes - Agile methodologies welcome changes in requirements, even late in the development process, to provide the customer with a competitive advantage.
- 3) Deliver Working Software Frequently - Agile emphasizes the importance of delivering functional software frequently, with a preference for shorter development cycles.
- 4) Collaboration and Communication - Agile encourages frequent communication and collaboration among team members, stakeholders, and customers to ensure everyone is aligned.
- 5) Self-Organizing Teams - Teams in Agile are expected to be self-organizing, making decisions collectively and adapting to changes without relying on external direction.

Scrum

- ▶ Scrum is an agile framework for managing and organizing complex projects.
- ▶ It was originally developed for software development, but its principles and practices have been applied successfully in various industries.
- ▶ Scrum provides a structured yet flexible approach to project management, with a focus on iterative development, collaboration, and responsiveness to change.
- ▶ Scrum-it is iterative in nature
- ▶ Sprints- basic unit of development in scrum. It produces a working and tested software within given time limits.
- ▶ Sprints tend to last between one week to month.

Scrum Framework

1. Roles:

- Product Owner - Represents the customer and prioritizes the product backlog.
- Scrum Master - Ensures the team follows Scrum practices and removes impediments.
- Development Team- Cross-functional and self-organizing team responsible for delivering the product

2. Artifacts:

- Product Backlog - A prioritized list of features, enhancements, and bug fixes.
- Sprint Backlog- The subset of items from the product backlog selected for a sprint.
- Increment- The sum of all completed product backlog items at the end of a sprint.

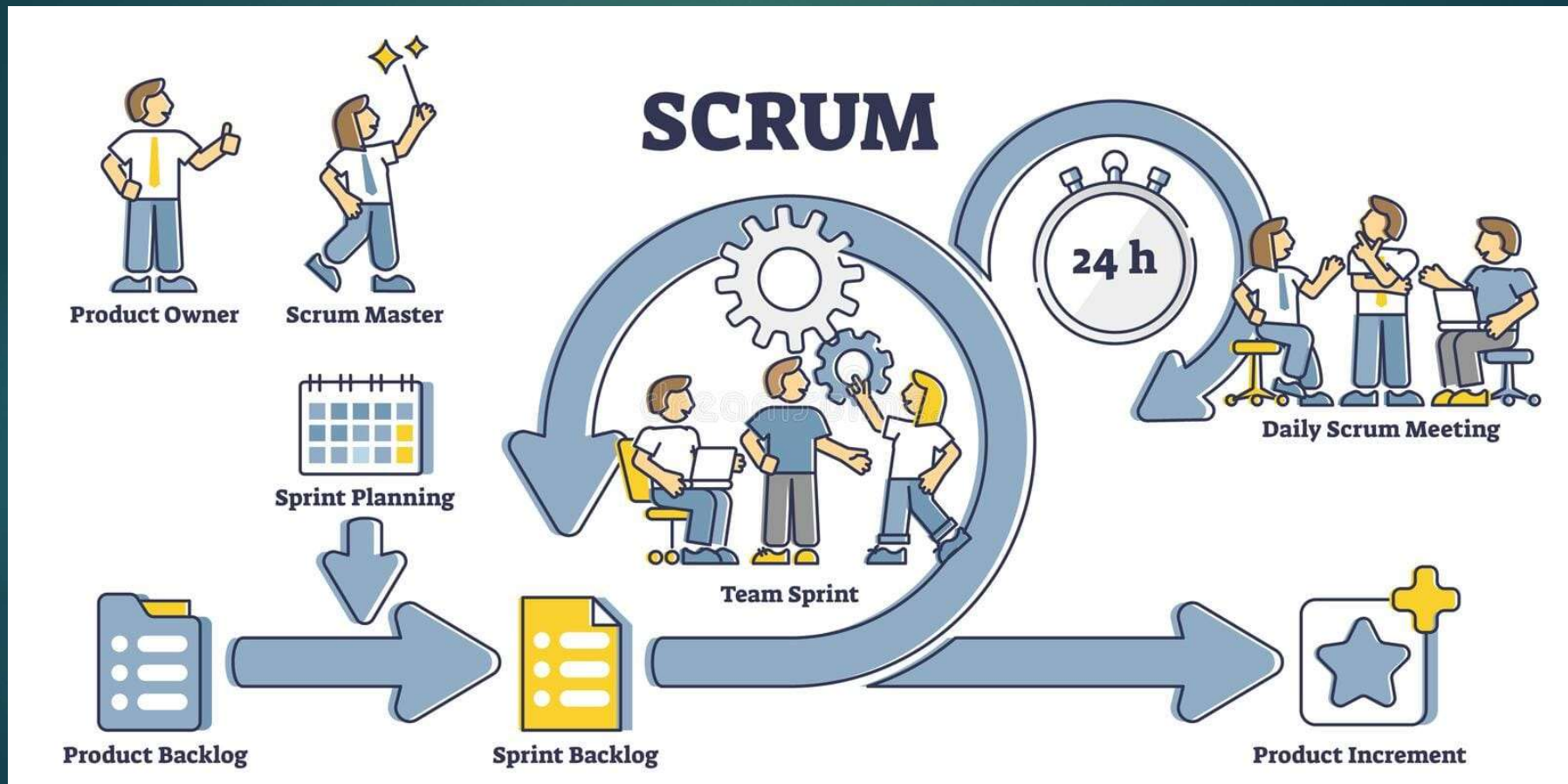
Scrum Framework



3. Events:

- Sprint Planning - Determines what can be delivered in the upcoming sprint.
- Daily Stand-up - Brief daily meeting for team members to sync up on progress.
- Sprint Review - Review of the completed work at the end of a sprint.
- Sprint Retrospective - Reflection on the sprint to improve processes for the next sprint.

Scrum Framework



Agile with Scrum



1. Iterations (Sprints) - Agile promotes the use of iterations, and Scrum defines a specific time-boxed iteration called a sprint, typically lasting 2-4 weeks.
2. User Stories and Backlog - Agile uses user stories to describe features from an end-user perspective, and Scrum maintains a product backlog of these user stories.
3. Adaptation and Inspections - Both Agile and Scrum encourage regular adaptation and inspection of processes and products to improve continuously.
4. Cross-Functional Teams - Agile principles and Scrum both emphasize the importance of having cross-functional teams capable of delivering a complete product increment.

Advantages

- ▶ Superior quality product
- ▶ Customer satisfaction
- ▶ Better control
- ▶ Improved project predictability
- ▶ Reduced risks
- ▶ Increased flexibility
- ▶ Continuous improvement
- ▶ Improved team morale.

Disadvantages

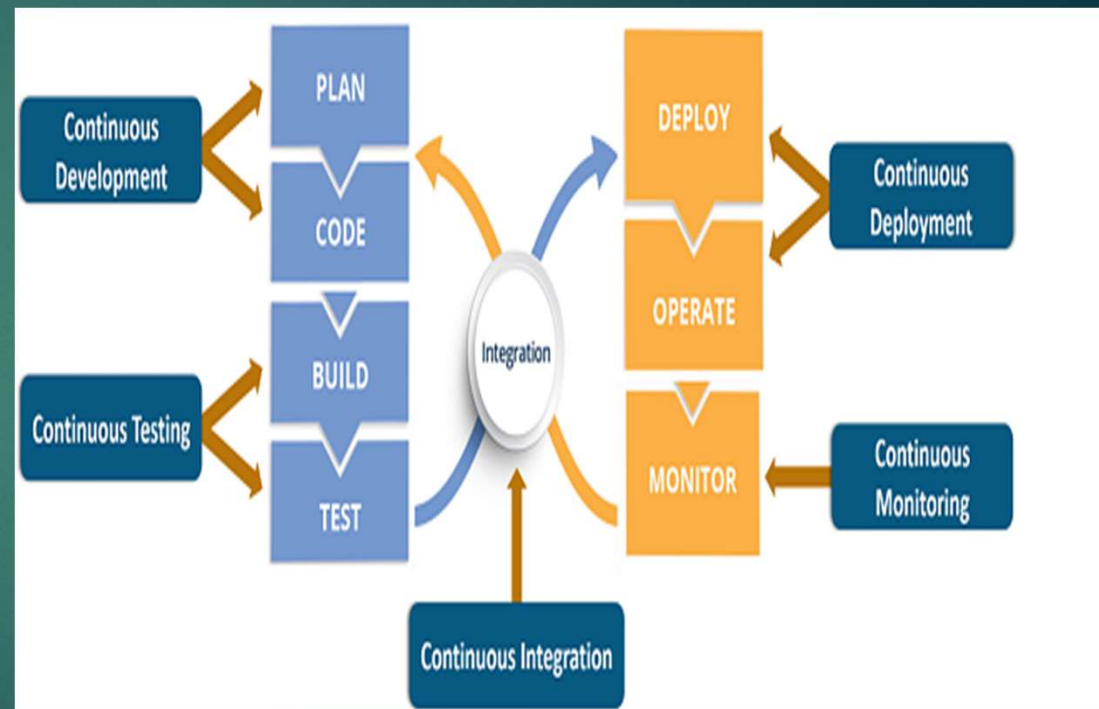
- ▶ Less predictable
- ▶ More time and commitment
- ▶ Greater demands on developers and clients
- ▶ Lack of necessary documentation
- ▶ Projects easily fall off track

► AGILE Methodology



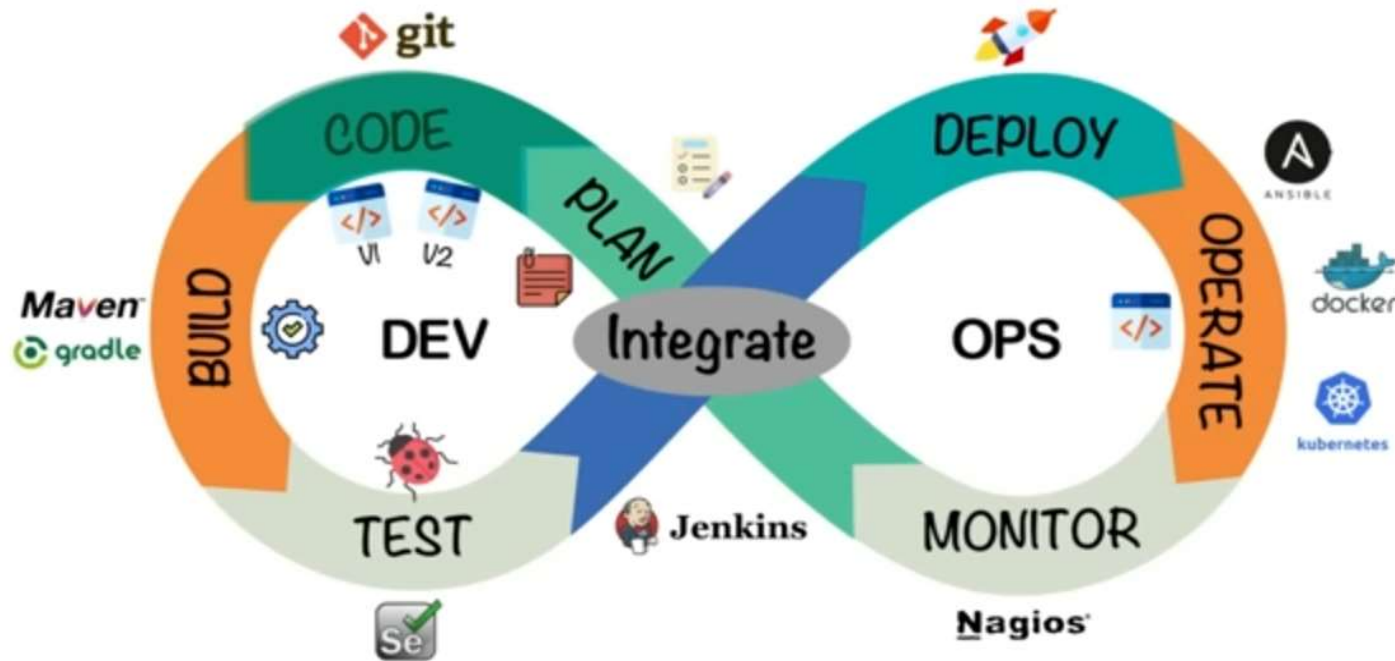
What is DevOps?

- ▶ The term DevOps is a combination of two words namely Development and Operations. DevOps is a practice that allows a single team to manage the entire application development life cycle, that is, development, testing, deployment, operations.
- ▶ The aim of DevOps is to shorten the system's development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives.
- ▶ DevOps is a software development approach through which superior quality software can be developed quickly and with more reliability. It consists of various stages such as continuous development, continuous integration, continuous testing, continuous deployment, and continuous monitoring.



What is DevOps Life cycle?

The DevOps culture is implemented in several phases with the help of several tools

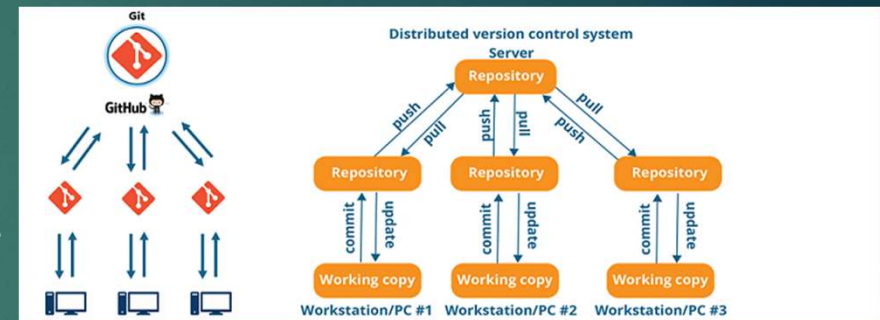




▶ **Continuous Development –**

- ▶ This is the phase that involves ‘planning’ and ‘coding’ of the software. The vision of the project is decided during the planning phase and the developers begin developing the code for the application.
- ▶ There are no DevOps tools that are required for planning, but there are a number of tools for maintaining the code.
- ▶ The code can be written in any language, but it is maintained by using Version Control tools.
- ▶ Maintaining the code is referred to as Source Code Management. The most popular tools used are Git, SVN, Mercurial, CVS, and JIRA. Also tools like Ant, Maven, Gradle can be used in this phase for building/ packaging the code into an executable file that can be forwarded to any of the next phases.

- ▶ Git
- ▶ Git is a distributed version control tool that supports distributed non-linear workflows by providing data assurance for developing quality software. Tools like Git enable communication between the development and the operations team.
- ▶ When you are developing a large project with a huge number of collaborators, it is very important to have communication between the collaborators while making changes in the project.
- ▶ Commit messages in Git play a very important role in communicating among the team. Apart from communication, the most important reason to use Git is that you always have a stable version of the code with you.
- ▶ Hence, Git plays a vital role in succeeding at DevOps.





▶ **Continuous Testing –**

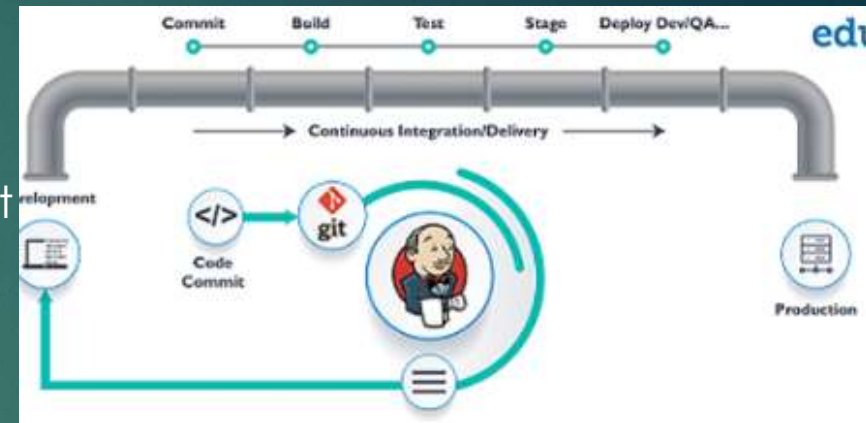
- ▶ This is the stage where the developed software is continuously tested for bugs.
- ▶ For Continuous testing, automation testing tools like Selenium, TestNG, JUnit, etc are used. These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there are no flaws in the functionality.
- ▶ In this phase, Docker Containers can be used for simulating the test environment.
- ▶ Selenium does the automation testing, and the reports are generated by [TestNG](#). This entire testing phase can be automated with the help of a Continuous Integration tool called Jenkins.
- ▶ Suppose you have written a selenium code in Java to test your application. Now you can build this code using ant or maven. Once the code is built, it is tested for User Acceptance Testing (UAT). This entire process can be automated using [Jenkins](#).


- ▶ Automation testing saves a lot of time, effort and labor for executing the tests instead of doing this manually. Besides that, report generation is a big plus. The task of evaluating the test cases that failed in a test suite gets simpler. We can also schedule the execution of the test cases at predefined times. After testing, the code is continuously integrated with the existing code.



► Continuous Integration –

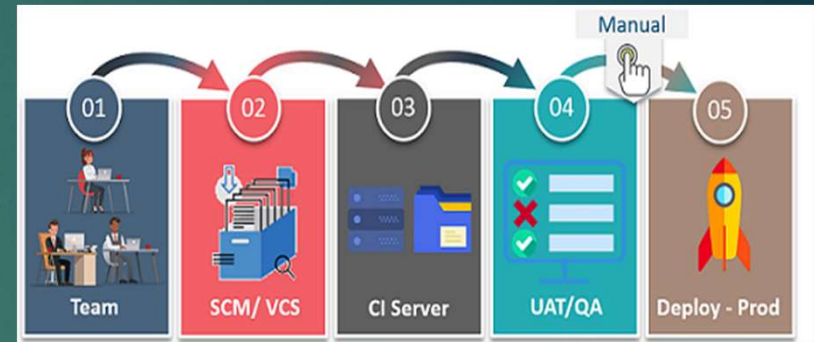
- This stage is the heart of the entire DevOps life cycle. It is a software development practice in which the developers require to commit changes to the source code more frequently. This may be on a daily or a weekly basis. Every commit is then built and this allows early detection of problems if they are present. Building code not only involves compilation but it also includes code review, unit testing, integration testing, and packaging.
- The code supporting new functionality is continuously integrated with the existing code. Since there is continuous development of software, the updated code needs to be integrated continuously as well as smoothly with the systems to reflect changes to the end-users.




- 
- ▶ Jenkins is a very popular tool used in this phase. Whenever there is a change in the [Git repository](#), Jenkins fetches the updated code and it prepares a build of that code which is an executable file in the form of a war or a jar. This build is then forwarded to the test server or the production server.

► Continuous Deployment –

- This is the stage where the code is deployed to the production servers. It is also important to ensure that the code is correctly deployed on all the servers. Before moving on, let us try to understand a few things about Configuration management and Containerization tools. These set of tools here help in achieving Continuous Deployment (CD).
- Configuration Management is the act of establishing and maintaining consistency in an application's functional requirements and performance. Let us put this in simpler words, it is the act of releasing deployments to servers, scheduling updates on all servers and most importantly keeping the configurations consistent across all the servers.
- Since the new code is deployed on a continuous basis, configuration management tools play an important role in executing tasks quickly and frequently. Some popular tools that are used here are Puppet, Chef, SaltStack, and Ansible.



- 
- ▶ **Containerization** is a type of virtualization in which all the components of an application are bundled into a single container image and can be run in isolated user space on the same shared operating system
 - ▶ Containerization refers to the creation of standardized software packages; often, a container will include an application and all of the software components needed to run it
 - ▶ Containerization tools also play an equally important role in the deployment stage. Docker and Vagrant are the popular tools used for this purpose. These tools help produce consistency across Development, Test, Staging and Production environments. Besides this, they also help in scaling-up and scaling-down of instances swiftly.
 - ▶ Containerization tools help in maintaining consistency across the environments where the application is developed, tested and deployed. Using these tools, there is no scope of errors/ failure in the production environment as they package and replicate the same dependencies and packages used in the development/ testing/ staging environment. It makes your application easy to run on different computers.



▶ **Continuous Monitoring –**

- ▶ This is a very crucial stage of the DevOps life cycle where you continuously monitor the performance of your application. Here vital information about the use of the software is recorded. This information is processed to recognize the proper functionality of the application. The system errors such as low memory, server not reachable, etc are resolved in this phase.
- ▶ The root cause of any issue is determined in this phase. It maintains the security and availability of the services. Also if there are network issues, they are resolved in this phase. It helps us automatically fix the problem as soon as they are detected.
- ▶ This practice involves the participation of the Operations team who will monitor the user activity for bugs or any improper behavior of the system. The popular tools used for this are Splunk, ELK Stack, Nagios, NewRelic and Sensu. These tools help you monitor the application's performance and the servers closely and also enable you to check the health of the system proactively.
- ▶ They can also improve productivity and increase the reliability of the systems, which in turn reduces IT support costs. Any major issues if found are reported to the development team so that it can be fixed in the continuous development phase. This leads to a faster resolution of the problems.
- ▶ These DevOps stages are carried out on loop continuously till you achieve the desired product quality. Therefore almost all of the major IT companies have shifted to DevOps for building their products.



ADVANTAGES

1. **Faster Delivery of Software**
2. **Improved Collaboration.**
3. **Increased Efficiency**
4. **Enhanced Quality Assurance**
5. **Greater Stability and Reliability**
6. **Improved Scalability**
7. **Cost Reduction**
8. **Enhanced Security**
9. **Continuous Feedback and Improvement**
10. **Flexibility and Agility**
11. **Alignment with Business Goals**



DISADVANTAGES:

- 1. Initial Implementation Complexity**
- 2. Resistance to Change**
- 3. Dependency on Automation**
- 4. Security Concerns**
- 5. Integration Challenges**

DevOps vs Agile

Features	DevOps	Agile
Agility	Agility in both Development & Operations	Agility in only Development
Processes/ Practices	Involves processes such as CI, CD, CT, etc.	Involves practices such as Agile Scrum, Agile Kanban, etc.
Key Focus Area	Timeliness & quality have equal priority	Timeliness is the main priority
Release Cycles/ Development Sprints	Smaller release cycles with immediate feedback	Smaller release cycles
Source of Feedback	Feedback is from self (Monitoring tools)	Feedback is from customers
Scope of Work	Agility & need for Automation	Agility only

CMM Model

- ▶ The Capability Maturity Model (CMM) is a methodology used to develop and refine an organization's software development process.
- ▶ The model describes a five-level evolutionary path of increasingly organized and systematically more mature processes.
- ▶ CMM was developed and is promoted by the SOFTWARE ENGINEERING INSTITUTE(SEI), a research and development center sponsored by the U.S. Department of Defense (DoD).
- ▶ Capability Maturity Model is used as a benchmark to measure the maturity of an organization's software process.
- ▶ The Capability Maturity Model (CMM) is a framework that outlines the stages of an organization's maturity in developing and improving its processes

Maturity levels within the CMM

1. Initial (Level 1):

1. Processes are ad hoc and chaotic.
2. Success depends on individual efforts.
3. There is no formalized process in place.

2. Managed (Level 2):

1. Basic project management processes are established.
2. Processes are planned, performed, measured, and controlled.
3. There is a focus on project tracking and management.

3. Defined (Level 3):

1. Processes are well characterized and understood.
2. Standard processes are defined and used across the organization.
3. There is an emphasis on process standardization and documentation.


Maturity levels within the CMM

4. Quantitatively Managed (Level 4):

1. Process performance is measured and controlled quantitatively.
2. Statistical methods are employed for process management and improvement.
3. Processes are continuously monitored for improvement opportunities.

5. Optimizing (Level 5):

1. Continuous process improvement is institutionalized.
2. Focus is on improving both the processes and the technologies used.
3. Innovation and optimization are part of the organizational culture.

- 
- ▶ The CMM model is not specific to any particular industry or domain, and it provides a flexible framework that organizations can tailor to their specific needs.
 - ▶ It encourages a systematic and structured approach to process improvement, allowing organizations to advance from lower to higher maturity levels over time.
 - ▶ The goal is to achieve higher maturity levels, which typically result in more predictable and efficient processes, improved product quality, and better overall organizational performance.