Chapter 1 Introduction to software Engineering

1.1 Software Engineering Process Paradigms.

Software

- Software is (1) **instructions** (computer programs) that when executed provide desired function and performance, (2) **data structures** that enable the programs to adequately manipulate information, and (3) **documents** that describe the operation and use of the programs.

- Software products may be developed for a particular customer or may be developed for a general market
- Software products may be
  - **Generic** - developed to be sold to a range of different customers
  - **Custom** - developed for a single customer according to their specification

Software Engineering

- It's a **discipline that is concerned with all aspects of software production.**
- Software engineers should adopt
  - Systematic and organized approach to their work
  - Use appropriate tools and techniques depending on the problem to be solved
  - The development constraints and the resources available

- **The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.**

- **Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.**

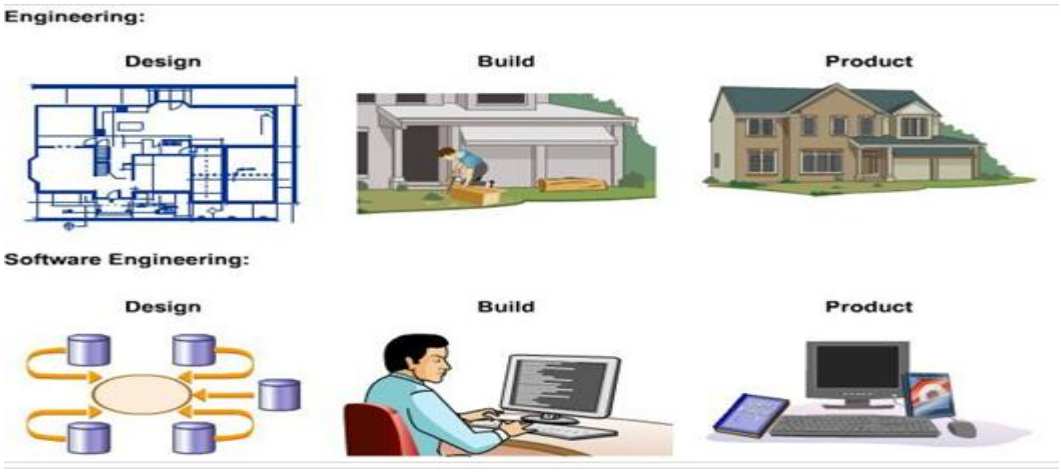


Figure 1.1 Software Engineering

Software Engineering: A Layered Technology:

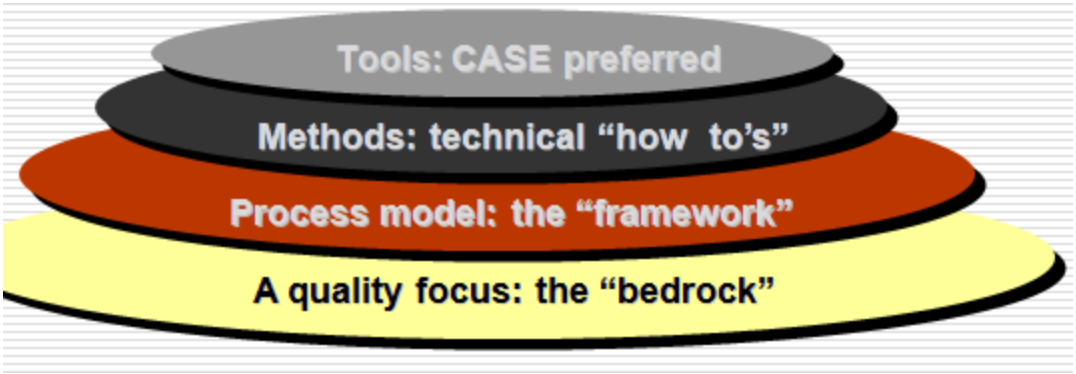


Figure 1.2 Software Engineering a layered technology

1. A quality Focus

- Every organization is rest on its commitment to quality.
- Total quality management, Six Sigma, or similar continuous improvement culture and it is this culture ultimately leads to development of increasingly more effective approaches to software engineering.

2. Process Models

- It's a foundation layer for software engineering.
- A software process is a set of activities and associated results which produce a s/w product.

- **A set of activities whose goal** is the development or evolution of software
- The processes define the tasks to be performed and the order in which they are to be performed

- Generic activities in all software processes are:
  - Specification - what the system should do and its development constraints
  - Development - production of the software system
  - Validation - checking that the software is what the customer wants
  - Evolution - changing the software in response to changing demands

3. Methods:

- It provides the technical **way** for building software.
- Methods encompass a broad array of tasks that include requirements analysis, design, program construction, testing, and support.
- There could be more than one technique to perform a task and different techniques could be used in different situations.

4. Tools:

- Software systems which are intended to provide automated support for software process activities. CASE systems are often used for method support
- **Upper-CASE**

  Tools to support the early process activities of requirements and design
- **Lower-CASE**

  Tools to support later activities such as programming, debugging and testing

The Software Process and Process Framework

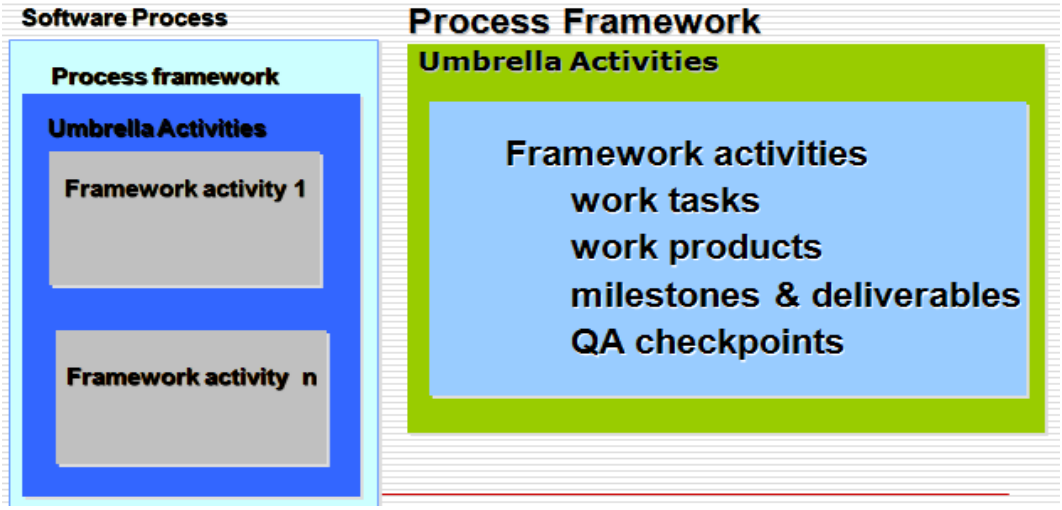


Figure 1.3 Software Process and Process Framework

A process defines who is doing what, when and how to reach a certain goal.

- To build complete software process.
- Identified a small number of framework activities that are applicable to all software projects, regardless of their size or complexity.
- It encompasses a set of umbrella activities that are applicable across the entire software process.

Generic Process Framework Activities

- Communication:
  - Heavy communication with customers, stakeholders, team
  - Encompasses requirements gathering and related activities
- Planning:
  - Workflow that is to follow
  - Describe technical task, likely risk, resources will require, work products to be produced and a work schedule.
- Modeling:
  - Help developer and customer to understand requirements (Analysis of requirements) & Design of software
- Construction
  - Code generation: either manual or automated or both
  - Testing – to uncover error in the code.
- Deployment:

- Delivery to the customer for evaluation
- Customer provide feedback

1.2 Software Process Models:

Process model defines a distinct set of activities, actions, tasks, milestones and work products that are required to engineer high quality software.

1. Waterfall Model :(Linear Sequential model or Classic life cycle model)

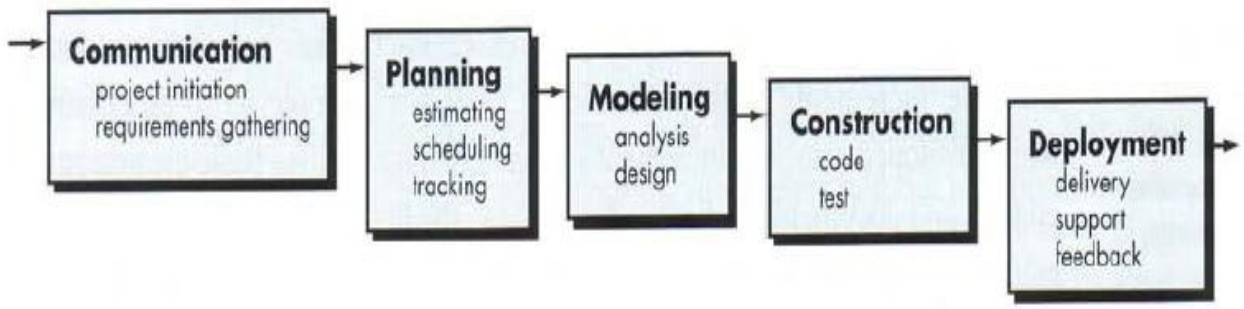
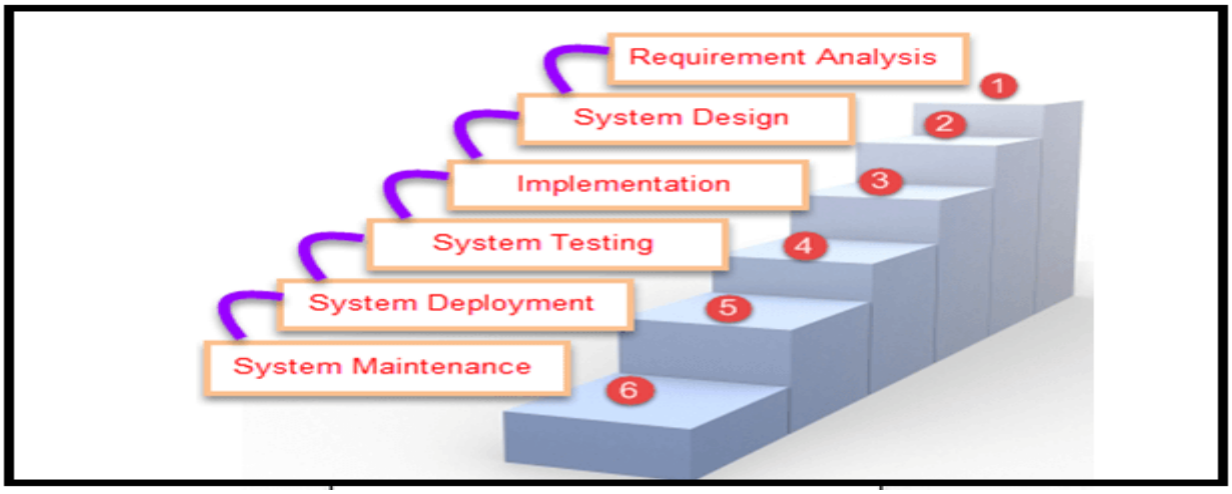


Figure 1.4 Waterfall model



Sometimes called the ***classic life cycle*** or the ***waterfall model***, the ***linear sequential model*** suggests a systematic, sequential approach5 to software development that begins at the system level and progresses through analysis, design, coding, testing, and support. The linear sequential model is the oldest and the most widely used paradigm for software engineering.

The waterfall model is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one and corresponds to a specialization of tasks. The approach is typical for certain areas of engineering design.

Phases of waterfall model

Different phases	Activities performed in each stage
Requirement Gathering stage	• During this phase, detailed requirements of the software system to be developed are gathered from client
Design Stage	• Plan the programming language, for Example Java, PHP, .net • or database like Oracle, MySQL, etc. • Or other high-level technical details of the project
Built Stage	• After design stage, it is built stage, that is nothing but coding the software
Test Stage	• In this phase, you test the software to verify that it is built as per the specifications given by the client.
Deployment stage	• Deploy the application in the respective environment
Maintenance stage	• Once your system is ready to use, you may later require change the code as per customer request

Advantages and disadvantages

Advantages	Dis-Advantages
• Before the next phase of development, each phase must be completed	• Error can be fixed only during the phase
• Suited for smaller projects where requirements are well defined	• It is not desirable for complex project where requirement changes frequently
• They should perform quality assurance test (Verification and Validation) before completing each stage	• Testing period comes quite late in the developmental process
• Elaborate documentation is done at every phase of the software's development cycle	• Documentation occupies a lot of time of developers and testers
• Project is completely dependent on project team with minimum client intervention	• Clients valuable feedback cannot be included with ongoing development phase
• Any changes in software is made during the process of the development	• Small changes or errors that arise in the completed software may cause a lot of problems

Disadvantages (Problems that are sometimes encountered when the linear sequential model is applied are):

1. Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.

2. It is often difficult for the customer to state all requirements explicitly. The linear sequential model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.

3. The customer must have patience. A working version of the program(s) will not be available until late in the project time-span. A major blunder, if undetected until the working program is reviewed, can be disastrous.

2. Incremental process Models

- **Incremental Model**
- **RAD model**

■ Incremental Model

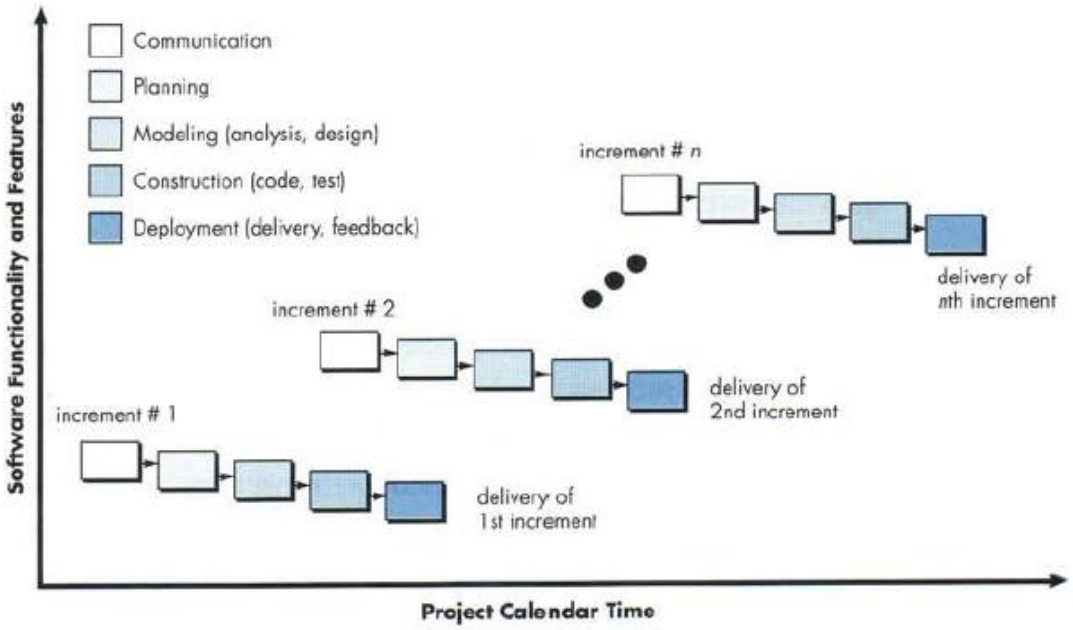


Figure 1.5 Incremental model

- The first increment is often a *core product.* That is, basic requirements are addressed, but many supplementary features (some known, others unknown) remain undelivered. The core product is used by the customer (or undergoes detailed review). As a result of use and/or evaluation, a plan is developed for the next increment. The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced.
- **The incremental process model is iterative in nature**.
- Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.
- Increments can be planned to manage technical risks.
- **Phases of incremental model**

- System development is broken down into many mini development projects
- Partial systems are successively built to produce a final total system
- Highest priority requirement is tackled first
- Once the requirement is developed, requirement for that increment are frozen

Incremental Phases	Activities performed in incremental phases
Requirement Analysis	• Requirement and specification of the software are collected
Design	• Some high-end function are designed during this stage
Code	• Coding of software is done during this stage
Test	• Once the system is deployed, it goes through the testing phase

- **Advantages and disadvantages:**

Advantages	Disadvantages
• The software will be generated quickly during the software life cycle	• It requires a good planning designing
• It is flexible and less expensive to change requirements and scope	• Problems might cause due to system architecture as such not all requirements collected up front for the entire software lifecycle
• Throughout the development stages changes can be done	• Each iteration phase is rigid and does not overlap each other
• This model is less costly compared to others	• Rectifying a problem in one unit requires correction in all the units and consumes a lot of time
• A customer can respond to each building	

■ The RAD Model:

Extremely short development cycle + component based construction

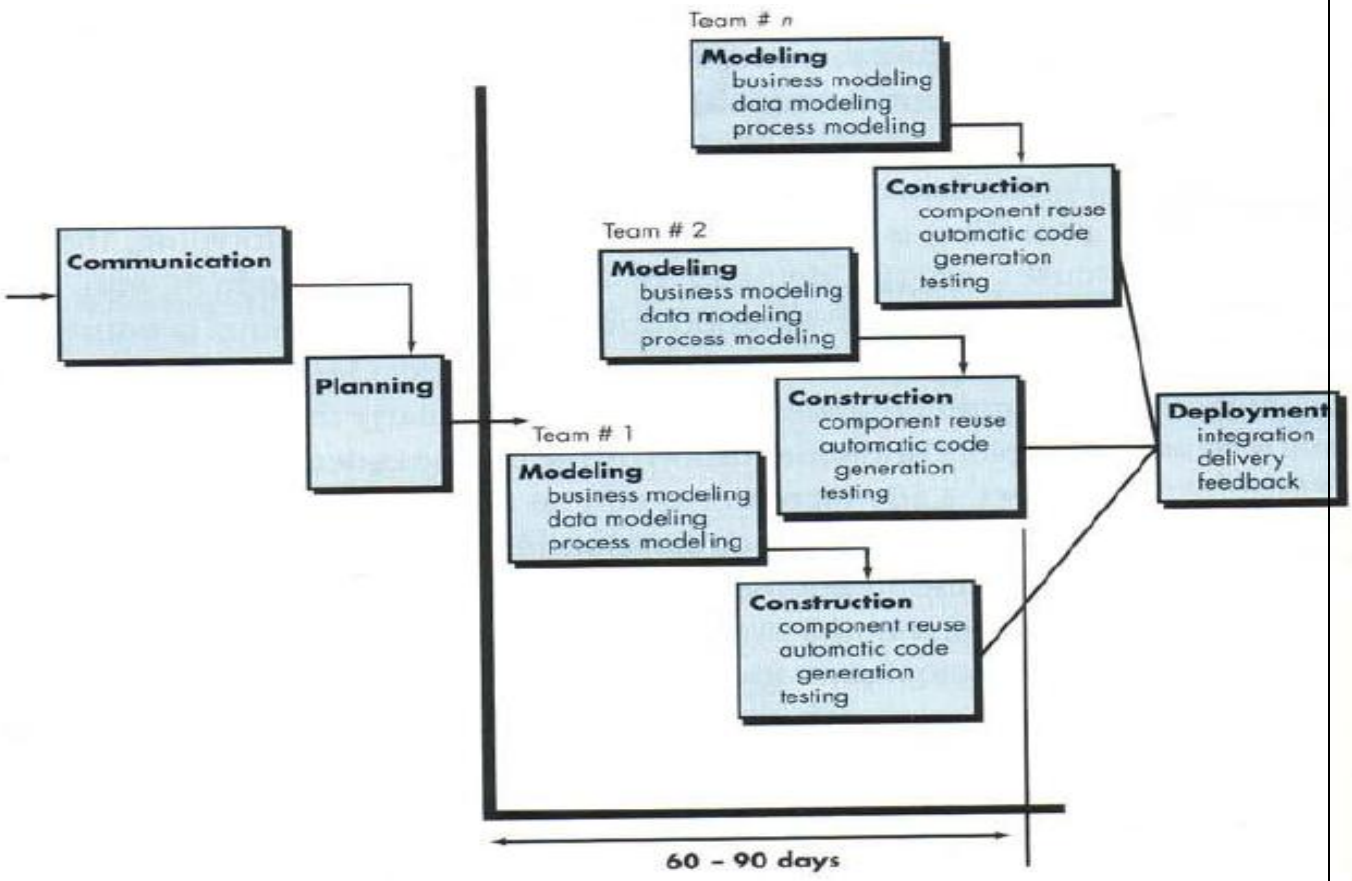


Figure 1.6 RAD Model

Phases of RAD

Different Phases of RAD Model

There are following five major phases of Rapid Application Development Model

RAD Model Phases	Activities performed in RAD Modeling
Business Modeling	• On basis of the flow of information and distribution between various business channels, the product is designed
Data Modeling	• The information collected from business modeling is refined into a set of data objects that are significant for the business
Process Modeling	• The data object that is declared in the data modeling phase is transformed to achieve the information flow necessary to implement a business function
Application Generation	• Automated tools are used for the construction of the software, to convert process and data models into prototypes
Testing and Turnover	• As prototypes are individually tested during every iteration, the overall testing time is reduced in RAD.

- **Business modeling.** The information flow among business functions is modeled in a way that answers the following questions: What information drives the business process? What information is generated? Who generates it? Where does the information go? Who processes it?

- **Process modeling.** The data objects defined in the data modeling phase are transformed to achieve the information flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.
- **Application generation.** RAD assumes the use of fourth generation techniques. Rather than creating software using conventional third generation programming languages the RAD process works to reuse existing program components or create reusable components. In all cases, automated tools are used to facilitate construction of the software.

- **Testing and turnover.** **Since the RAD process emphasizes reuse, many of the program components have already been tested**. This reduces overall testing time. However, new components must be tested and all interfaces must be fully exercised.

- **The RAD approach has drawbacks:**
  - For large but scalable projects, **RAD requires sufficient human resources to create the right number of RAD teams.**
  - **RAD requires developers and customers who are committed** to the rapid-fire activities necessary to get a system complete in a much abbreviated time frame. If commitment is lacking from either constituency, RAD projects will fail.
  - Not all types of applications are appropriate for RAD. **If a system cannot be properly modularized, building the components necessary for RAD will be problematic**. If high performance is an issue and performance is to be achieved through tuning the interfaces to system components, the RAD approach may not work.
  - **RAD is not appropriate when technical risks are high.** This occurs when a new application makes heavy use of new technology or when the new software requires a high degree of interoperability with existing computer programs.

- **Advantages**
  - **Due to emphasis on rapid development it results in the delivery of fully functional project in short time period**

3. Evolutionary Process Models

- **The Prototyping Model:**
- **The Spiral Model**
- **Concurrent Development model**

■ The Prototyping Model:

- **When prototyping model is applicable:**
  Prototype Model is a software development life cycle model which is used when the customer is not known completely about **how the end product should be** and **its requirements.** So in this model, **a prototype of the end product is first developed by the developers** and then tested and changes were made as per customer feedback **until the customer is satisfied with the prototype**.

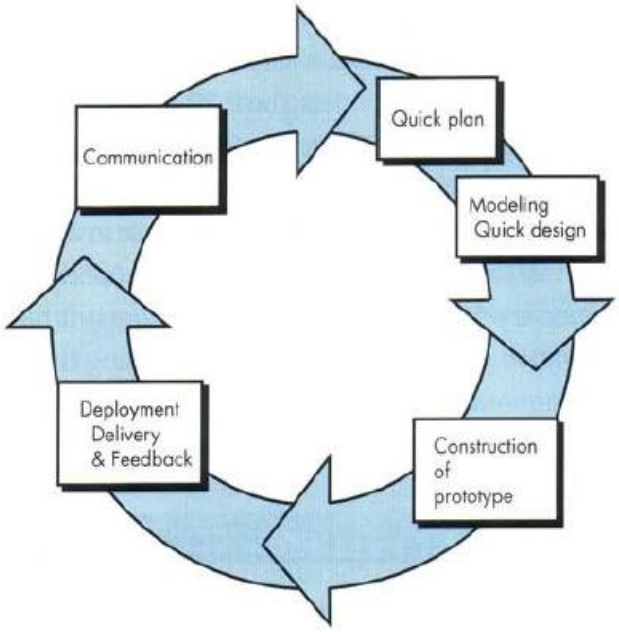


Figure 1.7 Prototyping Model

- Begins with requirements gathering. Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory.

- **A "quick design" then occurs**. The quick design focuses on a representation of those aspects of the software that will be visible to the customer/user (e.g., input approaches and output formats).

- The quick design leads to the construction of a prototype.

- The prototype is evaluated by the customer/user and used to refine requirements for the software to be developed. Iteration occurs as the prototype is tuned to satisfy the needs of the customer, while at the same time enabling the developer to better understand what needs to be done.

- **Advantages:**
  - ✓ This **model** is flexible in design.
  - ✓ It is easy to detect errors.
  - ✓ We can find missing functionality easily.
  - ✓ There is scope of refinement; it means new requirements can be easily accommodated.

- **Disadvantages:** **Prototyping can also be problematic for the following reasons:**
  1. The customer sees what appears to be a working version of the software.
  2. The developer often makes implementation compromises in order to get a prototype working quickly.

■ The Spiral Model(waterfall+ incremental +prototyping)

- **When Spiral Model is applicable:**

  Spiral Model is a software development life cycle model which is highly used for risk driven models. Based on the risk patterns of a given project, the spiral model helps developers to increase the efficiency of model as most risk already handled. It consist of number of loops which are forming a spiral shape **where each loop is called phase of software development cycle.**

- This **Spiral model** is a combination of iterative **development** process **model** and sequential linear **development model** i.e. the waterfall **model** with a very high emphasis on risk analysis. ... **It allows incremental releases of the product or incremental refinement through each iteration around the spiral.**

- In its diagrammatic representation, it looks like a spiral with many loops. **The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process.** The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using spiral model.

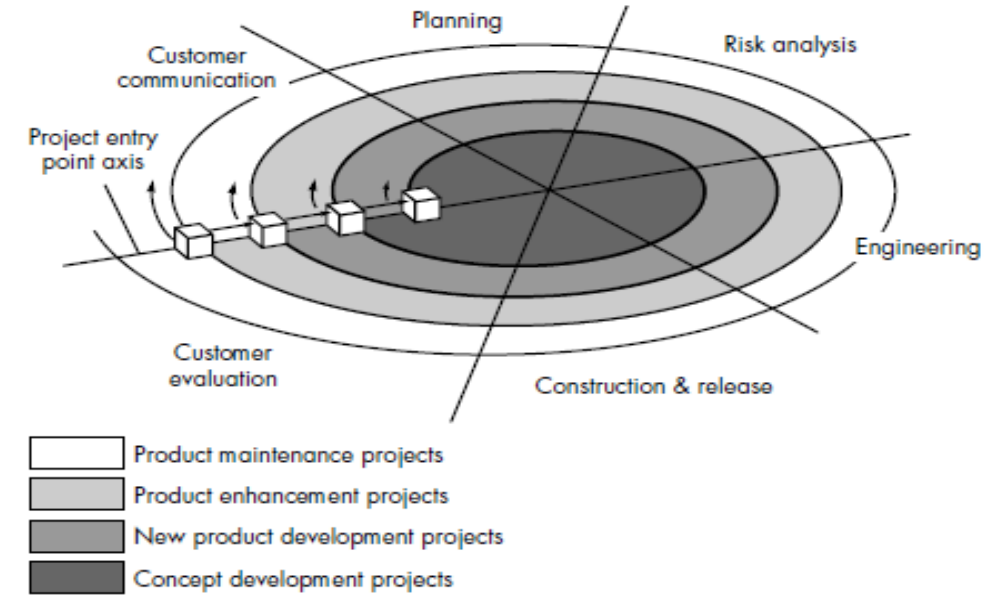


Figure 1.8 Spiral Model

- **Phases of waterfall Model:**

1. **Communication** (customer evaluation)
2. **Planning**
   a. **Estimation**
   b. **Scheduling**
   c. **Risk analysis**
3. **Modeling**
   a. **Analysis**
   b. **Design**
4. **Construction**
   a. **Coding**
   b. **Testing**
5. **Deployment**
   a. **Delivery**
   b. **Feedback**

- **Advantages of Spiral Model:** Below are some of the advantages of the Spiral Model.
  - ✓ **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
  - ✓ **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.

- ✓ **Flexibility in Requirements:** Change requests in the Requirements at later phase can be incorporated accurately by using this model.
- ✓ **Customer Satisfaction:** Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

- **Disadvantages of Spiral Model:** Below are some of the main disadvantages of the spiral model.
  - ✓ **Complex:** The Spiral Model is much more complex than other SDLC models.
  - ✓ **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
  - ✓ **Too much dependable on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced expertise, 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.

- **Difference between prototyping and Spiral Model**

S.No.	Prototype Model	Spiral Model
1.	Prototype model is a software development model in which a prototype is built, tested and then refined as per customer needs.	Spiral model is a risk-driven software development model and is made with features of incremental, waterfall or evolutionary prototyping models.
2.	It is also referred to as rapid or closed ended prototype.	It is also referred to as meta model.
3.	It does not give emphasis on risk analysis.	It takes special care about risk analysis and alternative solution is undertaken.
4.	In prototype model, customer interaction is continuous until the final prototype is approved.	In spiral model, there is no continuous customer interaction.
5.	It is best suited when the requirement of the client is not clear and supposed to be changed.	It is best suited when the customer requirements are clear.
6.	Cost effective quality improvement is very much possible.	Cost effective quality improvement is not possible.
7.	In Prototype model, improvement of quality does not increases the cost of product.	In Spiral model, improvement of quality can increase the cost of product.

- **Concurrent Development model**

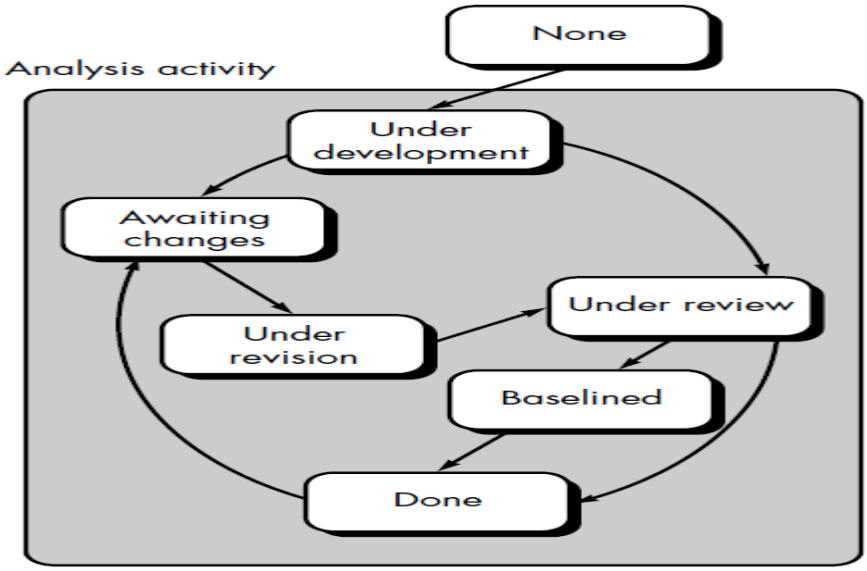


Figure 1.9 Concurrent Development Model

- **Project managers who track project status in terms of the major phases** [of the classic life cycle] have no idea of the status of their projects. Note that although . . . [a large] project is in the coding phase, there are personnel on the project involved in activities typically associated with many phases of development simultaneously. For example. Personnel are writing requirements, designing, coding, testing, and integration testing [all at the same time].

- Concurrent Development model is appropriate for system engineering projects where different engineering teams are involved.

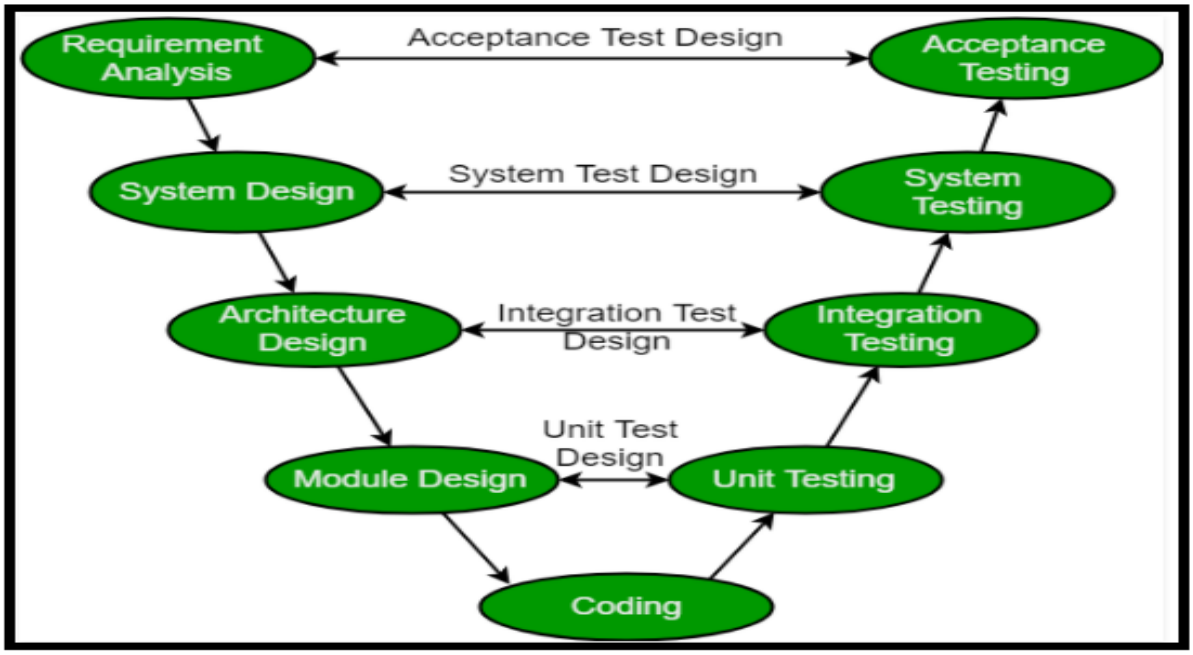
- Diagrammatically it can be represented as series of framework activities, tasks and actions and their associated results.

- Concurrent model is model; **where various activities of software development happen at the same time** for faster development and better outcome.

- **Advantages:**
  - ✓ Concurrent model is used in all types of software development and provides a very clear picture of current state of the system.
  - ✓ It is easy to use and easy to understand.
  - ✓ Provides immediate feedback from testing.

2. V Model

- **The V-model is a type of SDLC model where process executes in a sequential manner in V-shape.**
- **It is also known as Verification and Validation model.**
- **It is based on the association of a testing phase for each corresponding development stage.** Development of each step 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:** It involves static analysis technique (review) done without executing code. It is the process of evaluation of the product development phase to **find whether specified requirements meet.**

- **Validation:** It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after the completion of the development phase to determine **whether software meets the customer expectations and requirements.**

- **So V-Model contains Verification phases on one side** of the **Validation phases on the other side. Verification and Validation phases are joined by coding phase in V-shape. Thus it is called V-Model.**

Design Phase:

- ✓ **Requirement Analysis:** This phase contains detailed communication with the customer to understand their requirements and expectations. This stage is known as Requirement Gathering.
- ✓ **System Design:** This phase contains the system design and the complete hardware and communication setup for developing product.
- ✓ **Architectural Design:** System design is broken down further into modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.
- ✓ **Module Design:** In this phase the system breaks down into small modules. The detailed design of modules is specified, also known as Low-Level Design (LLD).

Testing Phases:

- ✓ **Unit Testing:** Unit Test Plans are developed during module design phase. These Unit Test Plans are executed to eliminate bugs at code or unit level.
- ✓ **Integration testing:** After completion of unit testing Integration testing is performed. In integration testing, the modules are integrated and the system is tested. Integration testing is performed on the Architecture design phase. This test verifies the communication of modules among themselves.
- ✓ **System Testing:** System testing test the complete application with its functionality, inter dependency, and communication. It tests the functional and non-functional requirements of the developed application.
- ✓ **User Acceptance Testing (UAT):** UAT is performed in a user environment that resembles the production environment. UAT verifies that the delivered system meets user's requirement and system is ready for use in real world.

When to use?

- ✓ **Where requirements are clearly defined and fixed.**
- ✓ The V-Model is used **when ample technical resources are available with technical expertise.**

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.
- ✓ This model focuses on verification and validation activities early in the life cycle thereby enhancing the probability of building an error-free and good quality product.
- ✓ It enables project management to track progress accurately.

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.**

1.3 Agile Methodology:

- It is new approach to software development. It involves **iterative development** and **incremental release of the product.**
- Agile is dynamic, content specific, aggressively change embracing and growth oriented.

The Agile Alliance [AGI03] defines 12 principles for those who want to achieve agility:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Methodology:

- Refers to group of s/w development methodologies and practices.
- New approach to s/w development.
- Involves:

  Iterative development

  Incremental release of product
- **Overall s/w/development cycle is divided into small periods or iterations; <u>each iteration last for about 2 weeks.</u>**
- Overall system development cycle which includes all phases of development (ADCTS) is repeated for every iteration phase.
- **Helps in reducing risk and allows product to adapt changes quickly.**
- Set of guidelines and requirements to be covered in next release are specified before the iterations.
- Programming team works on subset of overall requirements at a time.

- At the end of iteration product is presented to customer who reviews the product and gives response.
- Feedback from customer is available before final release and changes suggested by customer are incorporated in product. This helps in redefining the product if required.

Need of Agile Methodology:

- Existing SDLC phases are predefined and preplanned. Output of one phase is input to another phase. It is necessary to specify all req explicitly at start of project but this is not always feasible. Customers have no interaction with the team.
- But Agile methodology provide adaptive approach. Customer reviews prototype which covers only subset of req. at end of each iteration.

Agile methods:

1. Scrum:

- Each iteration phase is termed as Sprint and has duration that is decided by the team.
- Deliverable product is created during each sprint.
- A product backlog which includes set of req. is prepared at beginning of development process.
- Duration of each sprint is fixed and cannot be changed.
- Expected requirements of product which are not fulfilled in first sprint cycle are incorporated in next sprint cycle.
- Teams are self-organised and cross-functional.
- People are divided according to their roles:
  - ✓ Scrum Master
  - ✓ Product Owner
  - ✓ Team

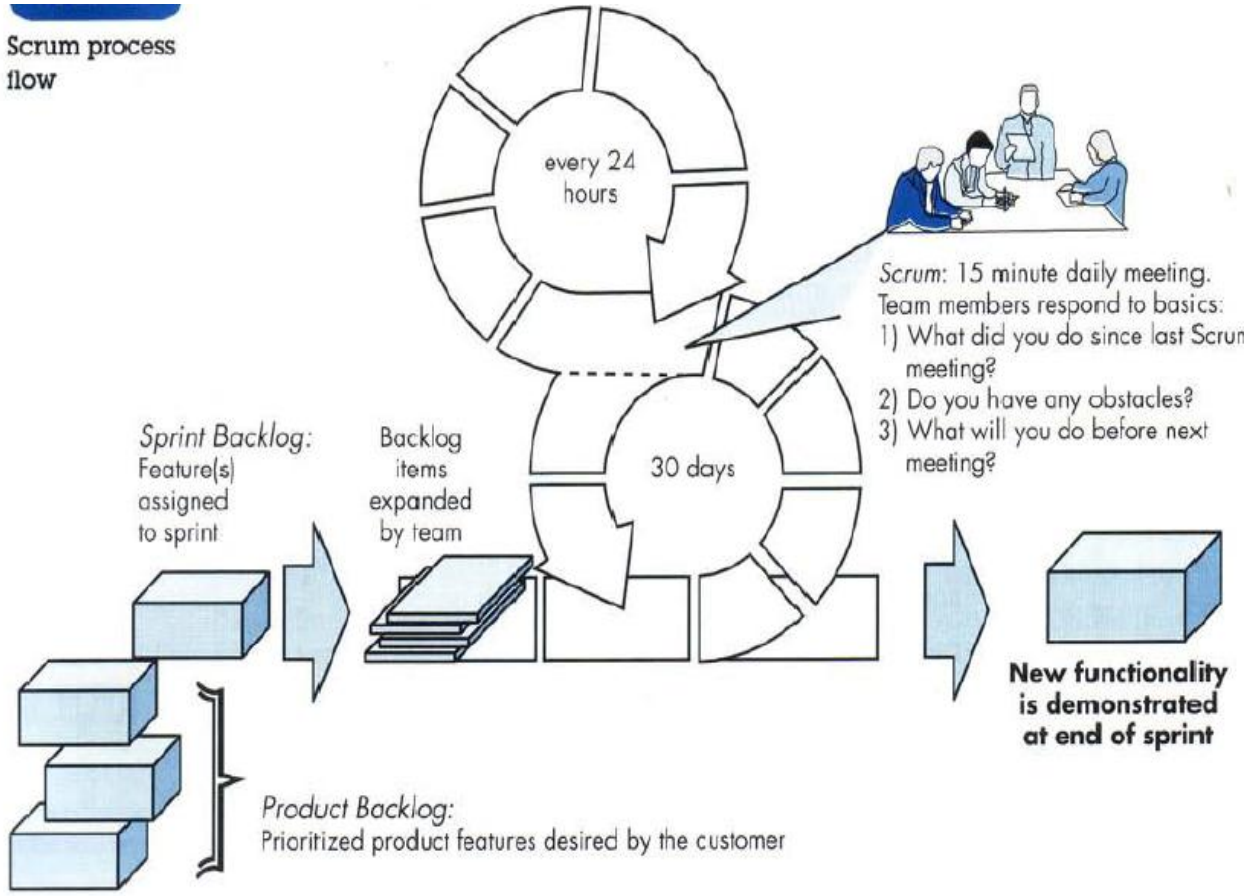


Figure 1.10 Scrum

2. Extreme Programming:

- Duration of each iteration is comparatively less than that of scrum.
- Duration of each iteration is not fixed.
- Duration can be extended if all req have not been covered.

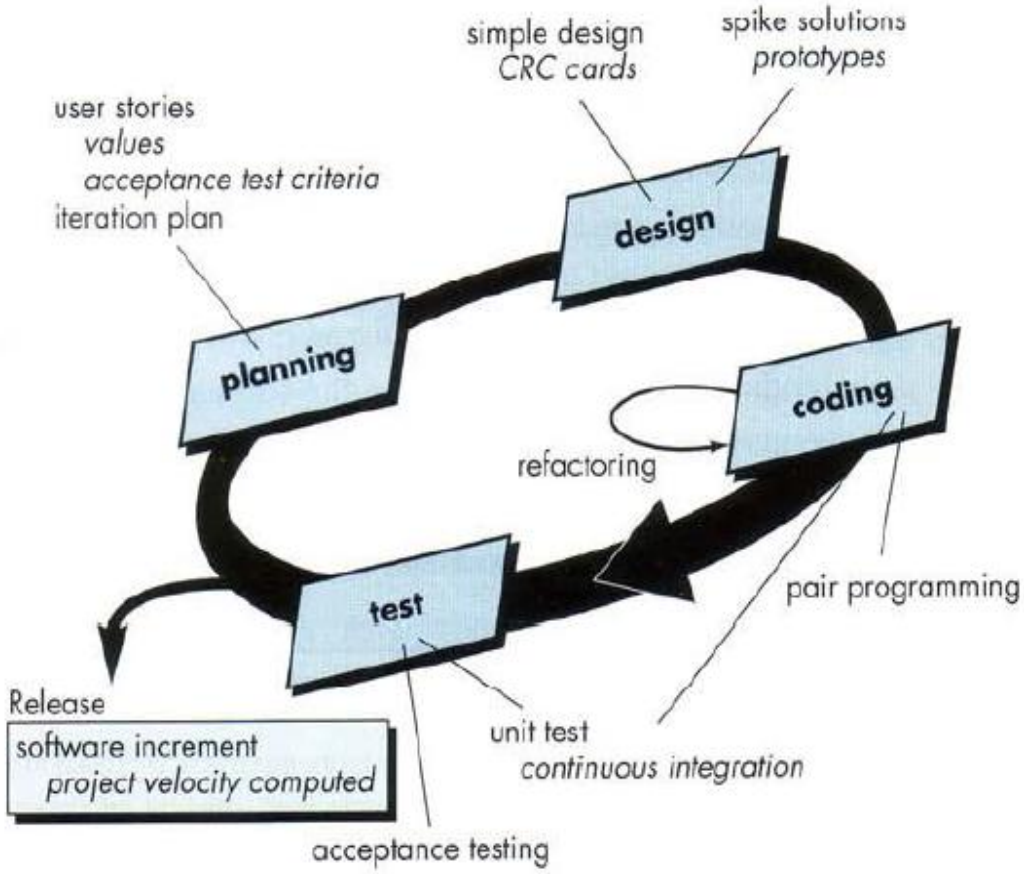


Figure 1.11 Extreme Programming

Benefits of Agile Methodology:

Faster delivery of high quality and more reliable product.
Early identification of defects.
Better customer satisfaction.
Collaborative approach.

Capability Maturity Model (CMM Model)

CMM was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University in 1987.

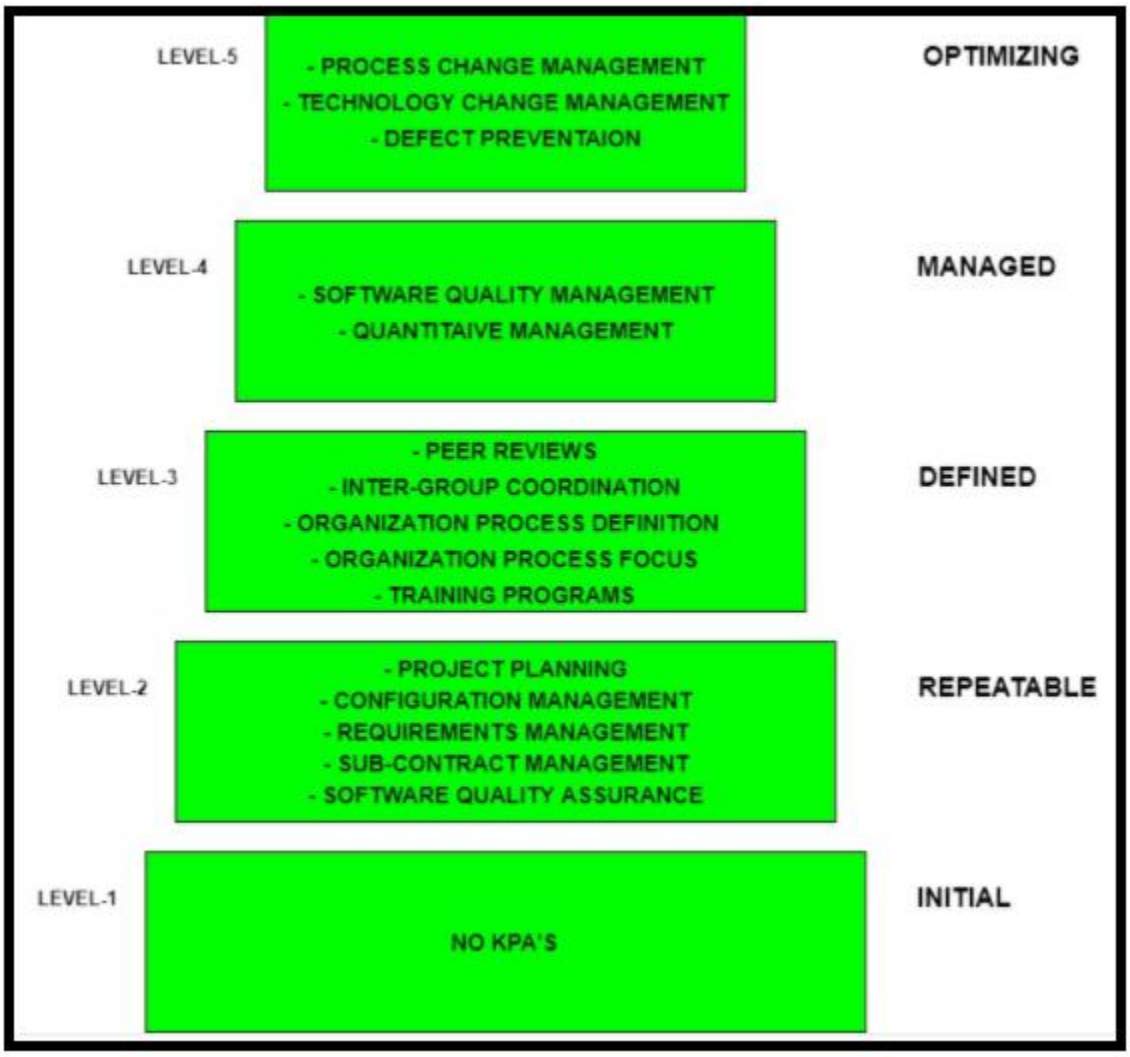
- It is not a software process model. It is a framework which is used to analyze the **approach and techniques** followed by any organization to develop software products.
- **It also provides guidelines** to further enhance the maturity of the process used to develop those software products.
- It is based on profound feedback and development practices adopted by the most successful organizations worldwide.
- This model describes a strategy for software process improvement that should be followed by moving through 5 different levels.
- **Each level of maturity shows a process capability level**. All the levels except level-1 are further **described by** Key Process Areas (KPA's).

Key Process Areas (KPA's):

Each of these KPA's defines the basic requirements that **should be met by a software process** in order to satisfy the KPA and achieve that level of maturity.

Conceptually, key process areas form the basis for management control of the software project and establish a context in which technical methods are applied, work products like models, documents, data, reports, etc. are produced, milestones are established, quality is ensured and change is properly managed.

Levels of CMM:



The 5 levels of CMM are as follows:

Level-1: Initial –

- ✓ No KPA's defined.
- ✓ **Processes followed are adhoc** and immature and are not well defined.
- ✓ **Unstable environment** for software development.
- ✓ **No basis for predicting product quality**, time for completion, etc.

Level-2: Repeatable –

- ✓ Focuses on establishing **basic project management policies.**

- ✓ **Experience with earlier projects is used for managing new similar natured projects.**
- ✓ Project Planning- It includes defining resources required, goals, constraints, etc. for the project. It presents a detailed plan to be followed systematically for successful completion of a good quality software.
- ✓ Configuration Management- The focus is on maintaining the performance of the software product, including all its components, for the entire lifecycle.
- ✓ Requirements Management- It includes the management of customer reviews and feedback which result in some changes in the requirement set. It also consists of accommodation of those modified requirements.
- ✓ Subcontract Management- It focuses on the effective management of qualified software contractors i.e. it manages the parts of the software which are developed by third parties.
- ✓ Software Quality Assurance- It guarantees a good quality software product by following certain rules and quality standard guidelines while development.

Level-3: Defined –

- ✓ At this level, documentation of the standard guidelines and procedures takes place.
- ✓ It is a well defined integrated set of project specific software engineering and management processes.

- ✓ **Peer Reviews-** In this method, defects are removed by using a number of review methods like walkthroughs, inspections, buddy checks, etc.
- ✓ **Intergroup Coordination**- It consists of planned interactions between different development teams to ensure efficient and proper fulfilment of customer needs.
- ✓ **Organization Process Definition**- It's key focus is on the development and maintenance of the standard development processes.
- ✓ **Organization Process Focus-** It includes activities and practices that should be followed to improve the process capabilities of an organization.
- ✓ **Training Programs-** It focuses on the enhancement of knowledge and skills of the team members including the developers and ensuring an increase in work efficiency.

Level-4: Managed –

- ✓ At this stage, quantitative quality goals are set for the organization for software products as well as software processes.

- ✓ The measurements made help the organization to predict the product and process quality within some limits defined quantitatively.

- ✓ Software Quality Management- It includes the establishment of plans and strategies to develop a quantitative analysis and understanding of the product's quality.

- ✓ **Quantitative Management**- It focuses on controlling the project performance in a quantitative manner.

Level-5: Optimizing –

- ✓ **This is the highest level of process maturity in CMM** and focuses on **continuous process improvement** in the organization using quantitative feedback.
- ✓ **Use of new tools, techniques and evaluation of software processes** is done to prevent recurrence of known defects.