Unit – 1
Software Process and Agile Development

Introduction to Software Engineering, Software Process, Perspective and Specialized Process Models – Introduction to Agility – Agile Process – Extreme Programming – XP process

Software Engineering - Introduction

1.1. INTRODUCTION

"Software engineering is a discipline in which theories, methods and tools are applied to develop professional software product"

In software engineering the systematic and organized approach is adopted. Based on the nature of the problem and development constraints various tools and techniques are applied in order to develop quality software.

The definition of software engineering is based on two terms:

Discipline:

For finding the solution to the problem an Engineer applies appropriate theories, methods and tools. While finding the solutions, Engineers must think of the organizational and financial constraints. Within these constraints only, he/she has to find the solution.

Product:

The software product gets developed after following systematic theories, methods and tools along with the appropriate management activities.

1.2. Defining Software

Software is nothing but a collection of computer programs and related documents that are intended to provide desired features, functionalities and better performance.

Software products may be:

[1] Generic - That means developed to be sold to a range of different customers.

[2] Custom - That means developed for a single customer according to their specification.

1.2.1 Software Characteristics

Software development is a logical activity and therefore it is important to understand basic characteristics of software. Some important characteristics of software are:

- **Software is engineered, not manufactured**

  Software development and hardware development are two different activities. A good design is a backbone for both the activities. Quality problems that occur in hardware manufacturing phase cannot be removed easily. On the other hand, during software development process such problems can be rectified. In both the activities, developers are responsible for producing qualitative product.

- **Software does not ware out**

  In early stage of hardware development process the failure rate is very high because of manufacturing defects. But after correcting such defects the failure rate gets reduced. The failure rate remains constant for some period of time and again it starts increasing because of environmental maladies (extreme temperature, dusts, and vibrations).

On the other hand software does not get affected from such environmental maladies. Hence ideally it should have an *"idealized curve"*. But due to some undiscovered errors the failure rate

is high and drops down as soon as the errors get corrected. Hence in failure rating of software the *"actual curve"* is as shown below:

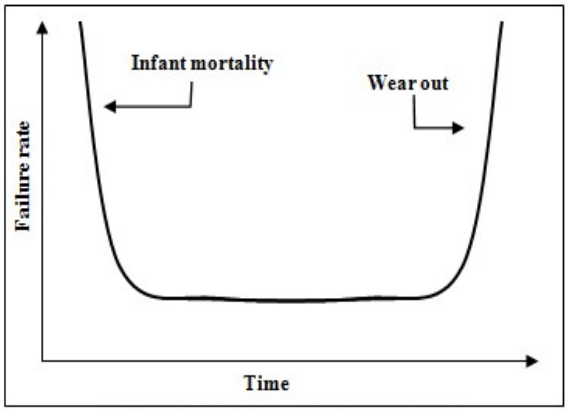


Figure 01: Failure curve for hardware

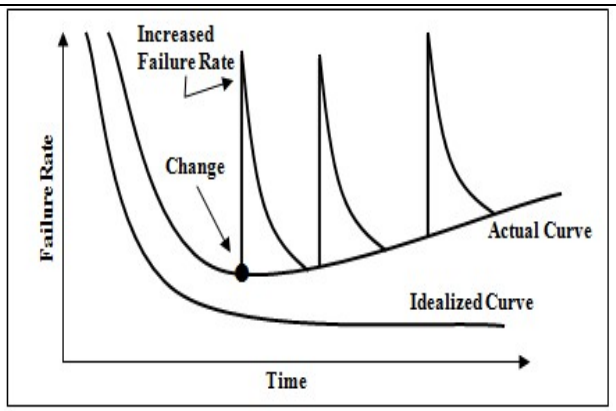


Figure 02: Failure Curves for software

Failure Curve for Hardware and Software

During the life of software if any change is made, some defects may get introduced. This causes failure rate to be high. Before the curve can return to original steady state another change is requested and again the failure rate becomes high. Thus the failure curve looks like a spike. Thus frequent changes in software cause it to deteriorate.

Another issue with software is that there are ***no spare parts for software***. If hardware component wears out it can be replaced by another component but it is not possible in case of software. Therefore software maintenance is more difficult than the hardware maintenance.

- **Most software is custom built rather than being assembled from components**

  While developing any hardware product firstly the circuit design with desired functioning properties is created. Then required hardware components such as ICs, capacitors and registers are assembled according to the design, but this is not done while developing software product. Most of the software is custom built.

However, now the software development approach is getting changed and we look for reusability of software components. It is practiced to reuse algorithms and data structures. Today software industry is trying to make library of reusable components.

For example: in today's software, GUI is built using the reusable components such as message windows, pull down menus and many more such components. The approach is getting developed to use in-built components in the software. This stream of software is popularly known as **component engineering**.

1.2.2 Categories of Software

Software can be applied in a situation for which a predefined set of procedural steps (algorithm) exists. Based on a complex growth of software it can be classified into following categories.

System software

It is collection of programs written to service other programs. Typical programs in this category are compiler, editors, and assemblers. The purpose of the system software is to establish a communication with the hardware.

Application software

It consists of standalone programs that are developed for specific business need. This software may be supported by database systems.

Engineering/scientific software

This software category has a wide range of programs from astronomy to volcanology, from automative stress analysis to space shuttle orbital dynamics and from molecular biology to automated manufacturing. This software is based on complex numeric computations.

Embedded software

This category consists of program that can reside within a product or system. Such software can be used to implement and control features and functions for the end-user and for the system itself.

Web applications

Web application software consists of various web pages that can be retrieved by a browser. The web pages can be developed using programming languages like JAVA, PERL, CGI, HTML, DHTML.

Artificial intelligence software

This kind of software is based on knowledge based expert systems. Typically, this software is useful in robotics, expert systems, image and voice recognition, artificial neural networks, theorem proving and game playing.

1.3 Goals and Objectives of Software.

While developing software following are common objectives.

Satisfy user's requirements

Many programmers simply don't do what the end user wants because they do not understand user requirements. Hence it becomes necessary to understand the demand of end user and accordingly software should be developed.

High reliability

Mistakes or bugs in a program can be expensive in terms of human lives, money, and customer relation. For instance, Microsoft has faced many problems because earlier release of windows has many problems. Thus software should be delivered only if high reliability is achieved.

Low maintenance costs

Maintenance of software is an activity that can be done only after delivering the software to the customer. Any small change in software should not cause restructuring of whole software. This indicates that the design of software has poor quality.

Delivery on time

It is very difficult to predict the exact time on which the software can be completed. But a systematic development of software can lead to meet the given deadline.

Low production costs

The software product should be cost effective.

High performance

The high performance software are expected to achieve optimization in speed and memory usage.

Ease of reuse

Use same software in different systems and software.

Environments reduce development costs and also improve the reliability. Hence reusability of developed software is an important property.

1.4 Challenges in Software Engineering

The key challenges facing software engineering are:

Coping with legacy systems

Old, valuable systems must be maintained and updated. Hardware is evolved faster than software. If original developer have moved on managing, maintaining or integrating of software becomes a critical issue.

Heterogeneity challenge

Sometimes systems are distributed and include a mix of hardware and software. This implies that software systems must cleanly integrate with other different software systems, built by different organizations and teams using different hardware and software platforms.

Delivery time's challenge

There is increasing pressure for faster delivery of software. As the complexity of systems that we develop increases, this challenge becomes harder.

As software is an integral part of computer based systems, it is essential to apply software engineering principles and practices while developing software. Hence the main objective of software engineering is to adopt systematic, disciplined approach while building high quality software.

1.5 Software Myths

There are some misbelieves in the software industry about the software and process of building software. For any software developer it is a must to know such beliefs and reality about them. Here are some typical myths –

Myth: ***Using a collection of standards and procedures one can build software. (Management Myth)***
Reality: Even though we have all standards and procedures with us for helping the developer to build software, it is not possible for software professionals to build desired product. This is because - the collection which we have should be complete, it should reflect modern techniques and more importantly it should be adaptable. It should also help the software professional to bring quality in the product.

Myth: ***Add more people to meet deadline of the project. (Management Myth)***
Reality: Adding more people in order to catch the schedule will cause the reverse effect on the software project i.e. software project will get delayed. Because, we have to spend more time on educating people or informing them about the project.

Myth: ***If a project is outsourced to a third party then all the worries of software building are over. (Management Myth)***
Reality: When a company needs to outsource the project then it simply indicates that the company does not know how to manage the projects. Sometimes, the outsourced projects require proper support for development.

Myth: ***Even if the software requirements are changing continuously it is possible to accommodate these changes in the software. (Customer Myth)***
Reality: It is true that software is a flexible entity but if continuous changes in the requirements have to be incorporated then there are chances of introducing more and more errors in the software. Similarly, the additional resources and more design modifications may be demanded by the software.

Myth: ***We can start writing the program by using general problem statements only. Later on using problem description we can add up the required functionalities in the program. (Customer Myth)***
Reality: It is not possible each time to have comprehensive problem statement. We have to start with general problem statements; however by proper communication with customer the software professionals can gather useful information. The most important thing is that the problem statement should be unambiguous to begin with.

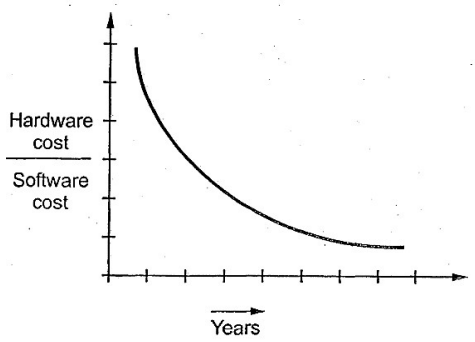
Myth: ***Once the program is running then it's over! (Practitioner's Myth)***
Reality: Even though we obtain that the program is running major part of work is after delivering it to customer.

Myth: ***Working program is the only work product for the software project. (Practitioner's Myth)***
Reality: The working program/software is the major component of any software project but along with it there are many other elements that should be present in the software project such as documentation of software, guideline for software support.

Myth: ***There is no need of documenting the software project; it unnecessarily slows down the development process. (Practitioner's Myth)***
Reality: Documenting the software project helps in establishing ease in use of software. It helps in creating better quality. Hence documentation is not wastage of time but it is a must for any software project.

1.6 Software Crisis

The software crisis means the decisive time or turning point that software developer's encounter during software development. Hence software crisis represent various problems that are faced by the software developers during software development process.
To understand software crisis consider following problem that is often faced by software industry.



Software cost: A software crisis

Software cost is getting increased tremendously day-by-day. The software purchase expenses are higher than the hardware purchase. This is becoming worrying trend over the years. Following graph shows the ratio of h/w and s/w cost vs. years

This graph shows that cost of software is increasing rapidly than the hardware cost. Following are the symptoms of present software crisis -

1) Day-by-day, software purchase cost is getting more than the hardware purchase cost. Hence major part of budget of any software industry is on software purchase.
2) Software products are difficult to alter, maintain, enhance, debug or modify.
3) Software resources are not being used optimally.
4) User requirements are often evolving and cannot be satisfied fully.
5) Software product not being reliable.
6) Many time software products get crashed on occurrence of specific condition.
7) Delivery of software product within specified budget and on scheduled time.

Various factors that have contributed to make software crisis

Following are some factors that bring the software crisis -

1) Increasing size or volume of software.
2) Lowered productivity or quality improvement.
3) Lack of skilled staff.
4) Inadequate software training.
5) Growing demand for more software.

Solutions to present software crisis -

The most effective solutions to present software crisis can be

1) Use and spread of software engineering practices among software engineers and
2) Further enhancement in software engineering disciplines.

1.7 Difference between Software Product and Program

Sl. No	Program	Software product
1.	Programs are developed by individual user and it is used for personal use.	Software product is developed by multiple users and it is used by large number of people or customers.

2.	Programs are small in size and possessing limited functionality.	Software product consists of multiple program codes, related documents such as SRS, design documents user manuals, test cases and so on.
3.	Generally only one person uses the program, hence there is a lack of user interface.	Good graphical user interface is most required by any software product.
4	Program is generally developed by programmers	Software product is developed by software engineers who are large in number and work in a team. Therefore systematic approach of developing software product must be applied.
	Example: Program of Sorting of n elements	**Example:** A word processing software

Software Process

1.8 Layered Technology

Software engineering is a layered technology. Any software can be developed using these layered approaches. Various layers on which the technology is based are quality focus layer, process layer, methods layer, tools layer.

A disciplined **quality management** is a backbone of software engineering technology.
Process layer is a foundation of software engineering. Basically, process defines the framework for timely delivery of software.
In **method layer** the actual method of implementation is carried out with the help of requirement analysis, designing, coding using desired programming constructs and testing.
Software **tools** are used to bring automation in software development process.
Thus software engineering is a **combination** of process, methods, and tools for development of quality software.

1.9 Software Process

Software process can be defined as the structured set of activities that are required to develop the software system.

The fundamental activities are:

- Specification

- Design and implementation
- Validation
- Evolution

A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

1. 9.1 Common Process Framework

The process framework is required for representing the common process activities.
As shown in the figure below, the software process is characterized by process framework activities, task sets and umbrella activities.

Process framework activities

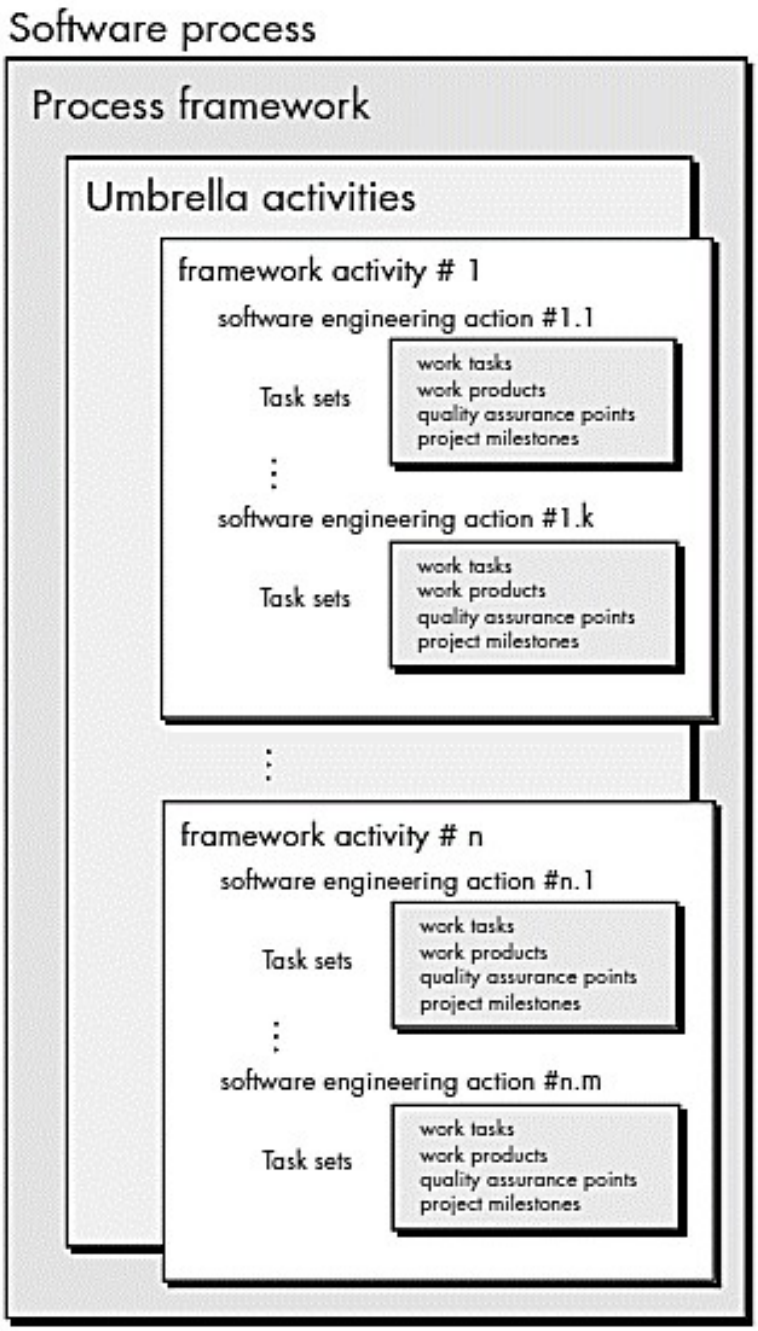
- Communication
  - By communicating customer requirement gathering is done.
- Planning - Establishes engineering work plan, describes technical risks, lists resource requirements, work products produced and defines work schedule.
- Modeling - The software model is prepared by :
  - Analysis of requirements
  - Design
- Construction - The software design is mapped into a code by :
  - Code generation
  - Testing
- Deployment - The software delivered for customer evaluation and feedback is obtained.

Task sets - The task set defines the actual work done in order to achieve the software objective. The task set is used to adopt the framework activities and project team requirements using:

- Collection of software engineering work tasks
- Project milestones
- Software quality assurance points

Umbrella activities - The umbrella activities occur throughout the process. They focus on project management, tracking and control. The umbrella activities are

- **Software project tracking and control** - This is an activity in which software team can assess progress and take corrective action to maintain schedule.
- **Risk management** - The risks that may affect project outcomes or quality can be analyzed.
- **Software quality assurance** - These are activities required to maintain software quality.
- **Formal technical reviews** - It is required to assess engineering work products to uncover and remove errors before they propagate to next activity.



1.9.2 Capability maturity Model(CMM)

The software engineering Institute (SEI) has developed a comprehensive process Meta – model emphasizing **process maturity**. It is predicted on a **set of system and software capabilities** that should be present when organizations reach different levels of process capability and maturity.

The capability maturity model (CMM) is used I accessing how well an organization‘s processes allow to complet and manage new software projects.

Various process maturity levels are:

1. Initial- Few processes are defined, and success depends on individual effort.

2. **Repeatable**-Basic process management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3. **Defined -**The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing an maintaining software.
4. **Managed-** Both the software process and products are quantitatively understood and controlled.
5. **Optimizing-** Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

Thus CMM is used for improving the software Project.

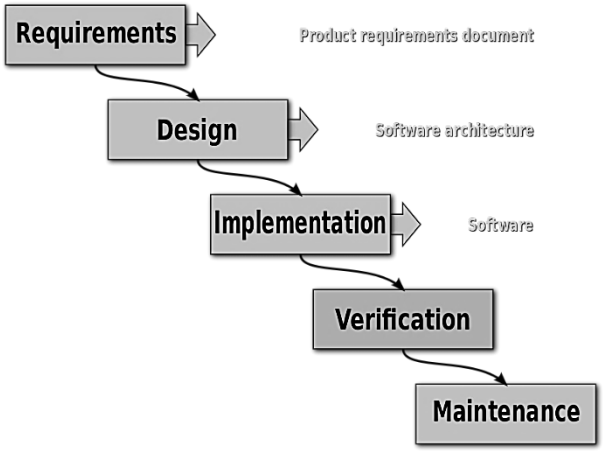
Perspective and Specialized Process Model

1. 10 Perspective and Specialized Process Models

- **Definition:**The **process model** can be defined as the **abstract representation of process**. The appropriate process model can be chosen based on abstract representation of process.
- The software process model is also known as software development life cycle (SDLC) model or software paradigm.
- These models are called perspective process models because they are following some rules for correct usage.
- Various perspective models are as follows

1.10.1 Waterfall Model

The software development starts with requirements gathering phase. Then progresses through analysis, design, coding, testing and maintenance. Following figure illustrates waterfall model.



Waterfall Model

In **REQUIREMENT GATHERING AND ANALYSIS** phase the basic requirements of the system must be understood by software engineer, who is also called Analyst. The information domain,

function, behavioral requirements of the system are understood. All these requirements are then well documented and discussed further with the customer, for reviewing.

The **DESIGN** is an intermediate step between requirements analysis and coding. Design focuses on program attributes such as -

- ✓ Data structure
- ✓ Software architecture
- ✓ Interface representation
- ✓ Algorithmic details.

The requirements are translated in some easy to represent form using which coding can be done effectively and efficiently. The design needs to be documented for further use.

CODING is a step in which design is translated into machine-readable form. If design is done in sufficient detail then coding can be done effectively. Programs are created in this phase.

TESTING begins when coding is done. While performing testing the major focus is on logical internals of the software. The testing ensures execution of all the paths, functional behaviors. The purpose of testing is to uncover errors, fix the bugs and meet the customer requirements.

MAINTENANCE is the longest life cycle phase. When the system is installed and put in practical use then error may get introduced, correcting such errors and putting it in use is the major purpose of maintenance activity. Similarly, enhancing system's services as new requirements are discovered is again maintenance of the system.

This model is widely used model, although it has many drawbacks.

Benefits of waterfall model

The waterfall model is simple to implement.

For implementation of small systems waterfall model is useful.

Drawbacks of waterfall model

There are some problems that are encountered if we apply the waterfall model and those are :

1) It is difficult to follow the sequential flow in software development process. If some changes are made at some phases then it may cause some confusion.

2) The requirement analysis is done initially and sometimes it is not possible to state all the requirements explicitly in the beginning. This causes difficulty in the project.

3) The customer can see the working model of the project only at the end. After reviewing of the working model; if the customer gets dissatisfied then it causes serious problems.

4) Linear nature of waterfall model induces blocking states, because certain tasks may be dependent on some previous tasks. Hence it is necessary to accomplish all the dependent tasks first. It may cause long waiting time.

1. 10.2 Incremental Process Model

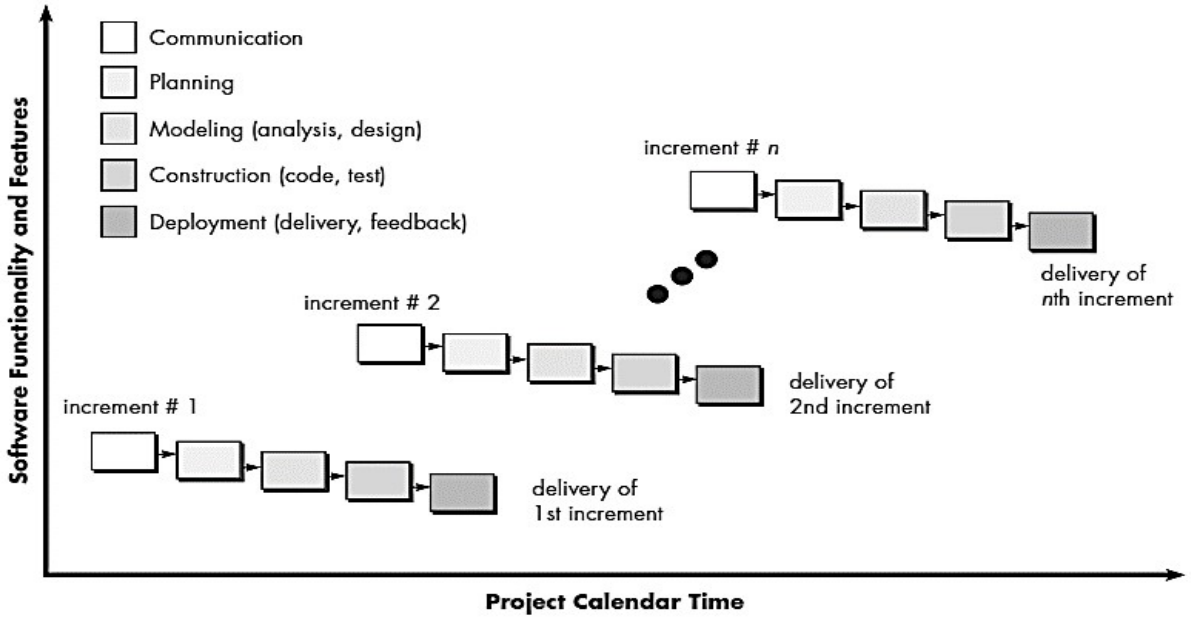
In this model, the initial model with limited functionality is created for user's understanding about the software product and then this model is refined and expanded in later releases.

1.10.2.1 Incremental Model

The incremental model has same phases that are in waterfall model. But it is iterative in nature. The incremental model has following phases.

1) Analysis
2) Design
3) Code
4) Test

The incremental model delivers series of releases to the customer. These releases are called increments. More and more functionality is associated with each increment.



Incremental Model

- The incremental model delivers series of releases to the customer. These releases are called increments. More and more functionality is associated with each increment.
- The first increment is called core product. In this release the basic requirements are implemented and then in subsequent increments new requirements are added.
- The word processing software package can be considered as an example of incremental model. In the first increment only the document processing facilities are available. In the second increment, more sophisticated document producing and processing facilities, file management functionalities are given. In the next increment spelling and grammar checking facilities can be given. Thus in incremental model progressive functionalities are obtained with each release.

When to choose it?

1) When requirements are reasonably well-defined.
2) When overall scope of the development effort suggests a purely linear effort.
3) When limited set of software functionality needed quickly.

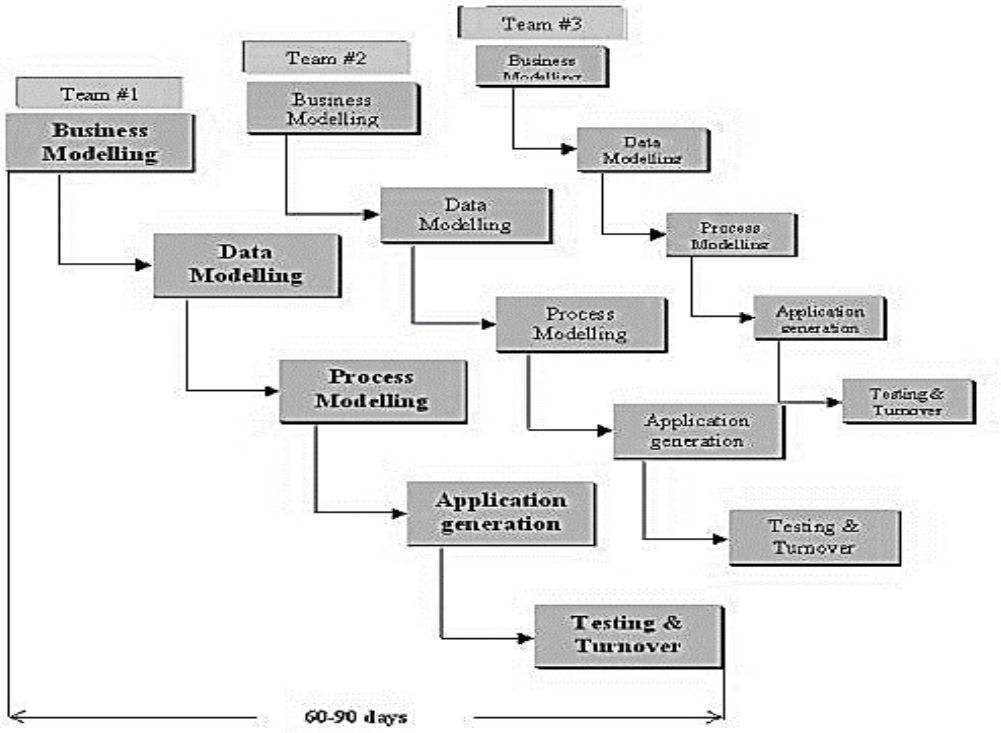
Merits of incremental model

1) The incremental model can be adopted when there are less number of people involved in the project.
2) Technical risks can be managed with each increment.
3) For a very small time span, at least core product can be delivered to the customer.

1.10.2.2 RAD Model

- The RAD Model is a type of incremental process model in which there is **EXTREMELY SHORT DEVELOPMENT CYCLE**.
- When the requirements are fully understood and the **COMPONENT BASED CONSTRUCTION** approach is adopted then the RAD model is used.

- Using the RAD model the fully functional system can be developed **WITHIN 60 TO 90 DAYS**.
- Various phases in RAD are Requirements Gathering, Analysis and Planning, Design, Build or Construction and finally Deployment.
- Multiple teams work on developing the software system using RAD model parallel.
- In the **REQUIREMENTS GATHERING** phase the developers communicate with the users of the system and understand the business process and requirements of the software system.
- During **ANALYSIS AND PLANNING** phase, the analysis on the gathered requirements is made and a planning for various software development activities is done.
- During the design phase various models are created. Those models are Business model, data model and process model.
- The build is an activity in which using the existing software components and automatic code generation tool the implementation code is created for the software system. This code is well tested by its team. The functionalities developed by all the teams are integrated to form a whole.
- Finally the deployment of all the software components (created by various teams working on the project) is carried out.



.

Drawbacks of rapid application development

- It requires multiple teams or large number of people to work on the scalable projects.
- This model requires heavily committed developer and customers. If commitment is lacking then RAD projects will fail.
- The projects using RAD model requires heavy resources.
- If there is no appropriate modularization then RAD projects fail. Performance can be problem to such projects.
- The projects using RAD model find it difficult to adopt new technologies.

Applications of RAD Model

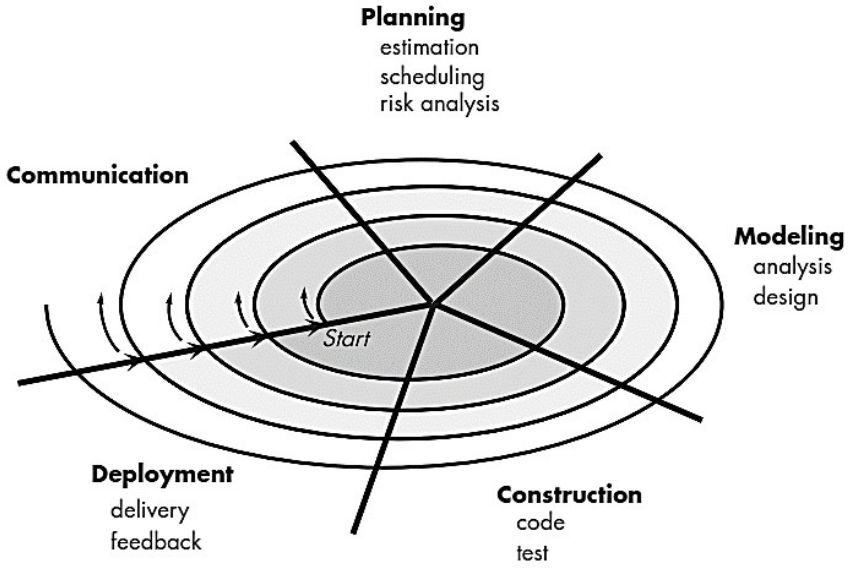
The RAD model is suitable for information system applications, business applications and the for systems that can be modularized because of following reasons -

[1] This model is similar to waterfall model but it uses very short development cycle.
[2] It uses component-based construction and emphasizes reuse and code generation.
[3] This model uses multiple teams on scalable projects.
[4] The RAD model is suitable for the projects where technical risks are not high.
[5] The RAD model requires heavy resources.

1.10.3 Evolutionary Process model

1.10.3 .1Spiral Model

- This model possess the iterative nature of prototyping model and controlled and systematic approaches of the linear sequential model.
- This model gives efficient development of incremental versions of software. In this model, the software is developed in series of increments.
- The spiral model is divided into a number of framework activities. These framework activities are denoted by task regions.
- Usually there are six tasks regions. The spiral model is as shown in the figure below
- Spiral model is realistic approach to development of large-scale systems and software. Because customer and developer better understand the problem statement at each evolutionary level. Also risks can be identified or rectified at each such level.



In the initial pass, product specification is built and in subsequent passes around the spiral the prototype gets developed and then more improved versions of software gets developed.

During planning phase, the cost and schedule of software can be planned and adjusted based on feedback obtained from customer evaluation.

In spiral model, project entry point axis is defined. This axis represents starting point for different types of projects.

For instance, concept development project will start at core of spiral and will continue along the spiral path. If the concept has to be developed into actual project then at entry point 2 the product

development process starts. Hence entry point 2 is called product development project entry point. The development of the project can be carried out in iterations.
The task regions can be described as :

1) **Customer communication** - In this region, it is suggested to establish customer communication.
2) **Planning** - All planning activities are carried out in order to define resources time line and other project related activities.
3) **Risk analysis** - The tasks required to calculate technical and management risks are carried out.
4) **Engineering** - In this task region, tasks required to build one or more representations of applications are carried out.
5) **Construct and release** - All the necessary tasks required to construct, test, install the application are conducted. Some tasks that are required to provide user support are also carried out in this task region.
6) **Customer evaluation** - Customer's feedback is obtained and based on customer evaluation required tasks are performed and implemented at installation stage.

In each region, numbers of work tasks are carried out depending upon the characteristics of project. For a small project relatively small numbers of work tasks are adopted but for a complex, project large number of work tasks can be carried out.
In spiral model, the software engineering team moves around the spiral in a clockwise direction beginning at the core.

Advantages of spiral model

- Requirement changes can be made at every stage.
- Risks can be identified and rectified before they get problematic.

Drawbacks of spiral mode!

- It is based on customer communication. If the communication is not proper then the software product that gets developed will not be up to the mark.
- It demands considerable risk assessment. If the risk assessment is done properly then only the successful product can be obtained.

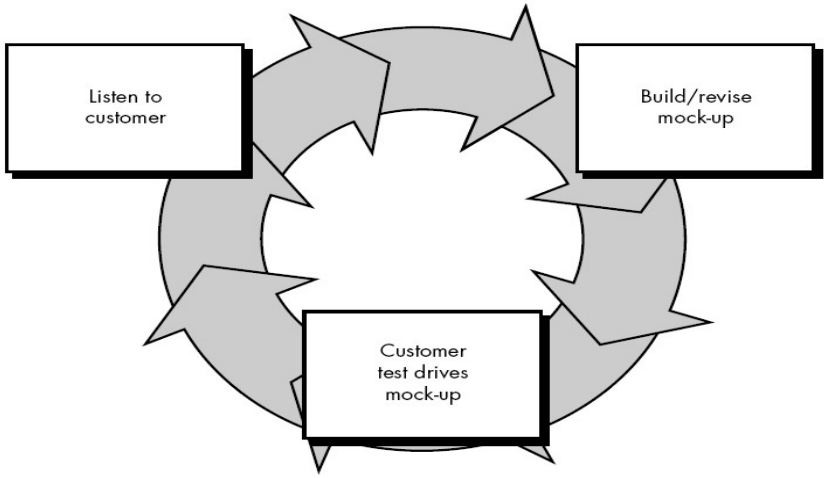
When to choose it?

1) When the prototypes for the software functionality are needed.
2) When requirements are not very clearly defined or complex.
3) When the large or high budget projects need to be developed.
4) When the risk assessment is very critical and essential
5) When project is not expected within a specific limited time span.

1.10.3.2 Prototyping model

- ✓ Gather requirements
- ✓ Developer & customer define overall objectives, identify areas needing more investigation – risky requirements
- ✓ Quick design focusing on what will be visible to user – input & output formats
- ✓ Use existing program fragments, program generators to throw together working version
- ✓ Prototype evaluated and requirements refined
- ✓ Process iterated until customer & developer satisfied
  - ✓ Then throw away prototype and rebuild system to high quality

✓ Alternatively can have evolutionary prototyping – start with well understood requirements.



Prototyping Drawbacks

- Customer may want to hang onto first version, may want a few fixes rather than rebuild. First version will have compromises
- Developer may make implementation compromises to get prototype working quickly. Later on developer may become comfortable with compromises and forget why they are inappropriate

Comparison between Spiral model and Prototyping model

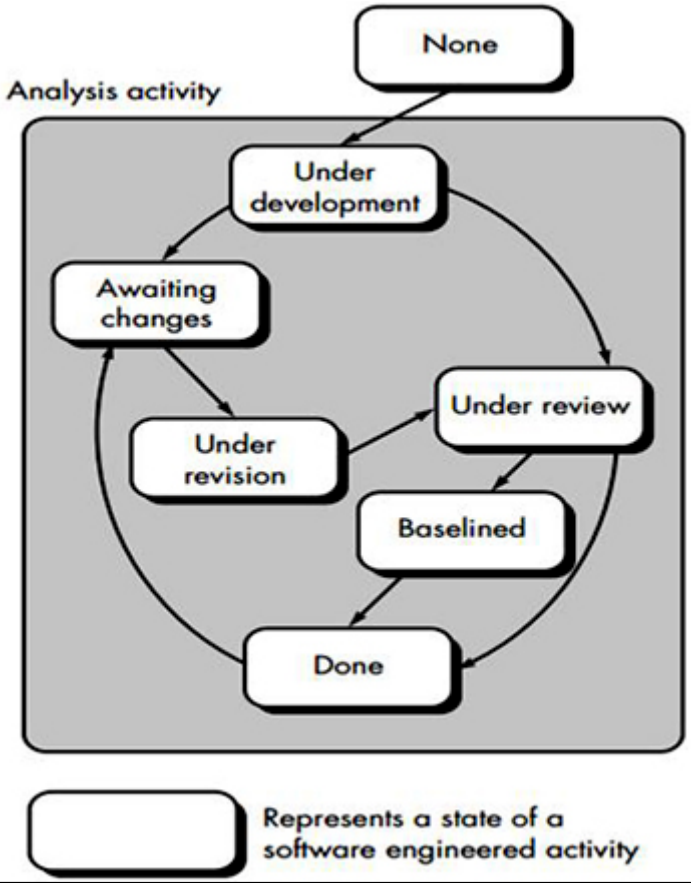
Spiral model	Prototyping model
The development team with less domain knowledge can be accommodated due to iterative nature of this model. The change in technology in the later phase cannot be tolerated.	The development team has adequate domain knowledge. Similarly they can adopt the new technologies if product demands.
All the end-users need not be involved in all the phases of development.	All the end-users are involved in all phases of development.
Funding are not stable for the projects that can be developed using spiral model.	Funding are stable for these type of projects.
The requirements that are gathered and analyzed are high reliability requirements.	Some requirements are gathered initially, but there may be change in requirements when the working prototype is shown to the customer.

1.10.3.3 Concurrent Development Model

- Allow a software team to represent iterative and concurrent elements of any of the process models. For example, the modeling activity defined for the spiral model is accomplished by invoking one or more of the following actions: prototyping, analysis and design.

- The Figure shows modeling may be in any one of the states at any given time. For example, communication activity has completed its first iteration and in the awaiting changes state. The modeling activity was in inactive state, now makes a transition into the under development state. If customer indicates changes in requirements, the modeling activity moves from the under development state into the awaiting changes state.

- Concurrent modeling is applicable to all types of software development and provides an accurate picture of the current state of a project. Rather than confining software engineering activities, actions and tasks to a sequence of events, it defines a process network. Each activity, action or task on the network exists simultaneously with other activities, actions or tasks. Events generated at one point trigger transitions among the states.



1.11 Specialized Process Models

The specialized models are used when only collecting of specialized technique or methods are expected for developing the specific software.
Various types of specialized models are
1. Component based development
2. Formal methods model.
3. Aspect oriented software development.

1.11.1 Component Based Development

Commercial off-the-shelf (COTS) Software components, developed by vendors who offer them as products, can be used when Software is to be built. These components provide targeted functionality with well-defined interfaces that enable the component to be integrated into the Software.

The *component-based development* model incorporates many of the characteristics of the spiral model.

The *component-based development* model incorporates the following steps:

- Available component-based products are researched and evaluated for the application domain in question.
- Component integration issues are considered.
- Software architecture is designed to accommodate the components.
- Components are integrated into the architecture.
- Comprehensive testing is conducted to ensure proper functionality.

The *component-based development* model leads to Software reuse, and reusability provides Software engineers with a number of measurable benefits.

1.11.2 The Formal Methods Model

The Formal Methods Model encompasses a set of activities that leads to formal mathematical specifications of Software.

Formal methods enable a Software engineer to specify, develop, and verify a computer-based system by applying a rigorous, mathematical notation.

A variation of this approach, called *clean-room* Software *engineering* is currently applied by some software development organizations.

Although not a mainstream approach, the formal methods model offers the promise of defect-free Software. Yet, concern about its applicability in a business environment has been voiced:

- The development of formal models is currently quite time-consuming and expensive.
- B/C few software developers have the necessary background to apply formal methods, extensive training is required.
- It is difficult to use the methods as a communication mechanism for technically unsophisticated customers.

1.11.3 Aspect-Oriented Software Development

Regardless of the software process that is chosen, the builders of complex software invariably implement a set of localized features, functions, and information content.

These localized software characteristics are modeled as components and then constructed within the context of system architecture. As modern computer-based systems become more sophisticated and complex, certain concerns, customer required properties or areas of technical interest, span the entire architecture. Some concerns are high-level properties of a system; others affect functions or are systemic.

When concerns cut across multiple system functions, features, and information they are often referred to as crosscutting concerns.

Aspectual requirements define those crosscutting concerns that have impact across the software architecture. Aspects are mechanisms beyond subroutines and inheritance for localizing the expression of a crosscutting concerns**.**
Aspect-oriented software development (AOSD), often referred to as **aspect-oriented programming (AOP**), is a relatively new software engineering paradigm that provides a process and methodological approach for defining, specifying, designing, and constructing aspects.

1.12 Introduction to Agility

The word 'agile' means −

- Able to move your body quickly and easily.
- Able to think quickly and clearly.

In business, 'agile' is used for describing ways of planning and doing work wherein it is understood that making changes as needed is an important part of the job. Business'agililty' means that a company is always in a position to take account of the market changes.

Ref: Cambridge Dictionaries online.

In software development, the term 'agile' is adapted to mean 'the ability to respond to changes − changes from Requirements, Technology and People.'

1.12.1 Agile Manifesto

A team of software developers published the Agile Manifesto in 2001, highlighting the importance of the development team, accommodating changing requirements and customer involvement.

The Agile Manifesto states that −

We are uncovering better ways of developing software by doing it and helping others do it. Through this work, we have come to value −

- **Individuals and interactions** over processes and tools.
- **Working software** over comprehensive documentation.
- **Customer collaboration** over contract negotiation.
- **Responding to change** over following a plan.

That is, while there is value in the items on the right, we value the items on the left more.

Agile process

- In 1980's the heavy weight, plan based software development development approach was used to develop any software product.
- In this approach too many things are done which were not directly related to software product being produced.
- This approach is rigid, that means if requirements get changed, then rework was essential. Hence new methods were proposed in 1990;s which are known as agile process.

- The agile processes are the light-weight methods are people –based rather than plan based methods.
- The agile process forces the development team to focus on software itself rather than design and documentation.
- The agile process believes in iterative method.
- The aim of agile process is to deliver the working model of software quickly to the customer.

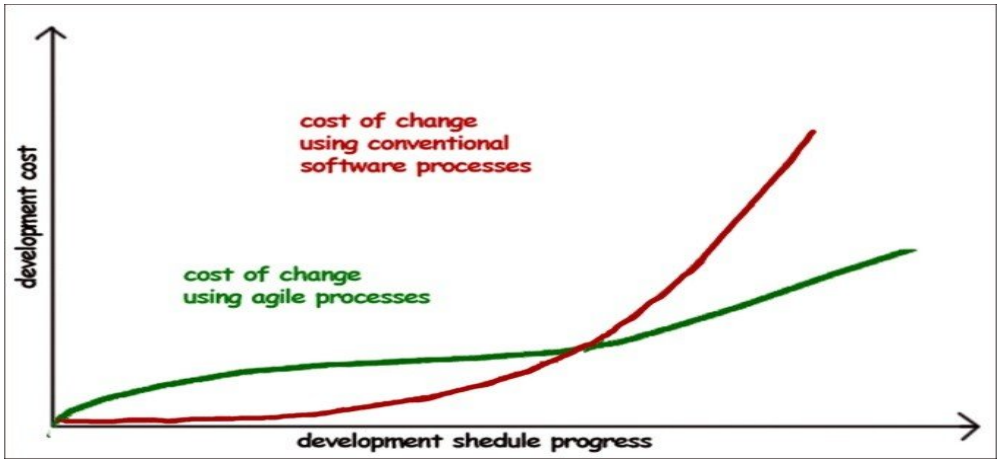
  For Eg: Extreme programming is the best known of agile process

Conventional software development methodology

- As the software project makes the progress ,the cost of the changes increases non linearly.
- It is easy to accommodate changes during the requirement gathering stage. at this stage to accommodate the changes-usage scenarios are modified, list of functions can be extended, or written specification be edited.
- As the progresses and if the customer suggest the changes during the testing phase of the software development life cycle then to accommodate these changes the architectural design needs to be modified and ultimately these changes will affect other phases of software development cycle. These changes are actually costly to execute.

Agile Methodology

- The agile method proponents claim that if the software development is carried out using the agile approach then it will allow the software team to accommodate changes late in a software project without dramatic cost and time impact.
- In other words, if the incremental delivery is combined with agile practices such as continuous unit testing and pair programming then the cost of changes can be controlled.
- The following graph represents how the software development approach has a strong influence on the development cost due to changes suggested.



1.12.2 The Twelve Agile Manifesto Principles

The Twelve Principles are the guiding principles for the methodologies that are included under the title "The Agile Movement." They describe a culture in which change is welcome, and the customer is the focus of the work. They also demonstrate the movement's intent as described by Alistair Cockburn, one of the signatories to the Agile Manifesto, which is to bring development into alignment with business needs.

The twelve principles of agile development include:

- **Customer satisfaction through early and continuous software delivery** – Customers are happier when they receive working software at regular intervals, rather than waiting extended periods of time between releases.
- **Accommodate changing requirements throughout the development process** – The ability to avoid delays when a requirement or feature request changes.
- **Frequent delivery of working software** – Scrum accommodates this principle since the team operates in software sprints or iterations that ensure regular delivery of working software.
- **Collaboration between the business stakeholders and developers throughout the project** – Better decisions are made when the business and technical team are aligned.
- **Support, trust, and motivate the people involved** – Motivated teams are more likely to deliver their best work than unhappy teams.
- **Enable face-to-face interactions** – Communication is more successful when development teams are co-located.
- **Working software is the primary measure of progress** – Delivering functional software to the customer is the ultimate factor that measures progress.
- **Agile processes to support a consistent development pace** – Teams establish a repeatable and maintainable speed at which they can deliver working software, and they repeat it with each release.
- **Attention to technical detail and design enhances agility** – The right skills and good design ensures the team can maintain the pace, constantly improve the product, and sustain change.
- **Simplicity** – Develop just enough to get the job done for right now.
- **Self-organizing teams encourage great architectures, requirements, and designs** – Skilled and motivated team members who have decision-making power, take ownership, communicate regularly with other team members, and share ideas that deliver quality products.
- **Regular reflections on how to become more effective** – Self-improvement, process improvement, advancing skills, and techniques help team members work more efficiently.

The intention of Agile is to align development with business needs, and the success of Agile is apparent. Agile projects are customer focused and encourage customer guidance and participation. As a result, Agile has grown to be an overarching view of software development throughout the software industry and an industry all by itself.

1.13 Extreme Programming (XP)

- The Extreme Programming is commonly used agile process model.
- It uses the concept of object-oriented programming.
- A developer focuses on the framework activities like planning, design, coding and testing. XP has a set of rules and practices.

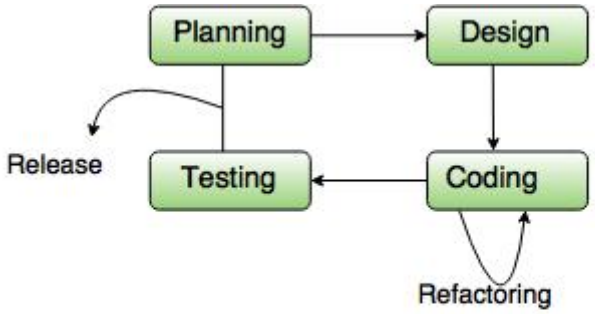


Fig. - The Extreme Programming Process

1.13.1 XP Values

Following are the values for extreme programming:

1. Communication

- Building software development process needs communication between the developer and the customer.
- Communication is important for requirement gathering and discussing the concept.

2) Simplicity

The simple design is easy to implement in code.

3. Feedback

Feedback guides the development process in the right direction.

4. Courage

In every development process there will always be a pressure situation. The courage or the discipline to deal with it surely makes the task easy.

5. Respect

Agile process should inculcate the habit to respect all team members, other stake holders and customer.

1.13.2 XP Process

The XP process comprises four framework activities:

1. Planning

- Planning starts with the requirements gathering which enables XP team to understand the rules for the software.
- The customer and developer work together for the final requirements.

2. Design

- The XP design follows the 'keep it simple' principle.
- A simple design always prefers the more difficult representation.

3. Coding

- The coding is started after the initial design work is over.
- After the initial design work is done, the team creates a set of unit tests which can test each situation that should be a part of the release.
- The developer is focused on what must be implemented to pass the test.
- Two people are assigned to create the code. It is an important concept in coding activity.

4. Testing

- Validation testing of the system occurs on a daily basis. It gives the XP team a regular indication of the progress.
- 'XP acceptance tests' are known as the customer test.

1.13.3 XP Process

Industrial XP Joshua Kerievsky describes Industrial Extreme Programming (IXP) in the following manner: "IXP is an organic evolution of XP. It is imbued with XP's minimalist, customer-centric, test-driven spirit. IXP differs most from the original XP in its greater inclusion of management, its expanded role for customers, and its upgraded technical practices." IXP incorporates six new practices that are designed to help ensure that an XP project works successfully for significant projects within a large organization.

Readiness assessment. Prior to the initiation of an IXP project, the organization should conduct a readiness assessment. The assessment ascertains whether

(1) An appropriate development environment exists to support IXP,

(2) The team will be populated by the proper set of stakeholders,

(3) The organization has a distinct quality program and supports continuous improvement,

(4) The SOFTWARE ENGINEERING RCPIT SHIRPUR 30 organizational culture will support the new values of an agile team, and

(5) The broader project community will be populated appropriately.

Project community. Classic XP suggests that the right people be used to populate the agile team to ensure success. The implication is that people on the team must be well-trained, adaptable and skilled, and have the proper temperament to contribute to a self-organizing team. When XP is to be applied for a significant project in a large organization, the concept of the "team" should morph into that of a community. A community may have a technologist and customers who are central to the success of a project as well as many other stakeholders (e.g., legal staff, quality auditors, manufacturing or sales types) who "are often at the periphery of an IXP project yet they may play important roles on the project". In IXP, the community members and their roles should be explicitly defined and mechanisms for communication and coordination between community members should be established.

Project chartering. The IXP team assesses the project itself to determine whether an appropriate business justification for the project exists and whether the project will further the overall goals and objectives of the organization. Chartering also examines the context of the project to determine how it complements, extends, or replaces existing systems or processes.

Test-driven management. An IXP project requires measurable criteria for assessing the state of the project and the progress that has been made to date. Test-driven management establishes a series of measurable "destinations" and then defines mechanisms for determining whether or not these destinations have been reached.

Retrospectives. An IXP team conducts a specialized technical review after a software increment is delivered. Called a retrospective, the review examines "issues, events, and lessons-learned" across a software increment and/or the entire software release. The intent is to improve the IXP process.

Continuous learning. Because learning is a vital part of continuous process improvement, members of the XP team are encouraged (and possibly, incented) to learn new methods and techniques that can lead to a higher quality product.

Part – A

Software Engineering - Introduction

1. What is Software Engineering? **[ND 2013/AM 19]**

Software engineering is a discipline in which theories, methods and tools are applied to develop professional software product.

2. Write the IEEE Definition Of Software Engineering. **[ND 2017]**

According to IEEE's definition software engineering can be defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

3. Define software.

Software is a collection of computer programs and related documents that are intended to provide desired features, functionalities and performance. A software can be of two types - 1. Generic software and 2. Custom software.

4. What are two type of software products?

There are two types of software products -

Generic - These products are developed and to be sold to the range of different customers.

Custom - These type of products are developed and sold to the specific group of customers and developed as per their requirements.

5. Software doesn't wear out. Justify.

Software does not wear out like hardware, software also similar to hardware shows high failure rate at its initial state. Then it gets modifications and the defects get corrections and thus it comes to the idealized state.

Software Process

6. Define Software Process. **[ND 2019]**

Software Process defines a framework for a set of Key Process Areas (KPAs) that must be established for effective delivery of software engineering technology. This establishes the context in which technical methods are applied, work products such as models, documents, data, reports, forms, etc. are produced, milestones are established, quality is ensured, and change is properly managed.

7. Define Software Process Framework.

A process framework establishes the foundation for a complete software process by identifying a small number of framework activities that are applicable to all software projects, regardless of size or complexity. It also includes a set of umbrella activities that are applicable across the entire software process.

8. What led to the transition from product oriented development to process oriented development? **[MJ 2016]**

A process framework establishes the foundation for a complete software process by identifying a small number of framework activities that are applicable to all software projects, regardless of size or complexity. It also includes a set of umbrella activities that are applicable across the

entire software process. This led to the transition from product oriented development to process oriented development.

9. Write the Process framework and Umbrella activities. [MJ 2015]

A process framework establishes the foundation for a complete software process by identifying a small number of framework activities that are applicable to all software projects, regardless of size or complexity. It also includes a set of umbrella activities that are applicable across the entire software process.

Umbrella Activities are

1. Software project tracking and control
2. Formal technical reviews
3. Software quality assurance
4. Software configuration management
5. Document preparation and production
6. Reusability management
7. Measurement
8. Risk management

10. Why software architecture is important in software process?

The system architecture defines the role of hardware, software, people, database procedures and other system elements. The architectural design of the system help the developer to understand the system as a whole. Hence system architecture must be built before specifications are completed. Thus architectural design is important in software engineering.

11. What is software process model? On what basis it is chosen? [ND 2016]

The software process model can be defined as abstract representation of process. It is based on nature of software project.

12. What is software process?

Software process can be defined as the structured set of activities that are required to develop the software system. The fundamental activities are -

1. Specification
2. Design and Implementation
3. Validation
4. Evolution

13. Distinguish between process and methods. [MJ 2014]

Software process can be defined as the structured set of activities that are required to develop the software system. Various activities under software process are

- Specification
- Design and implementation
- Validation
- Evolution

Method is used mainly in object-oriented programming, the term method refers to a piece of code that is exclusively associated either with a class (called class methods) or with an object (called instance methods).

14. Define the terms product and process in software engineering.

The process in software engineering can be defined as the structured set of activities that are required to develop software system. Various activities under software process are –

- Specification
- Design and implementation
- Validation
- Evolution

Perspective and Specialized Process Models

15. What is Linear Sequential Model?

Linear Sequential Model. It is also called "Classic Life Cycle" or "Waterfall" model or "Software Life Cycle" suggests a systematic and sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing and support.

16. What is meant by 'blocking states' in linear sequential model? [ND 2014]

The linear nature of linear sequential model brings a situation in the project that some project team members have to wait for other members of the team to complete the dependent tasks. This situation is called "blocking state" in linear sequential model. For example, after performing the requirement gathering and analysis step the design process can be started. Hence the team working on design stage has to wait for gathering of all the necessary requirements. Similarly the programmers can not start coding step unless and until the design of the project is completed

17. State the benefits of waterfall life cycle model for software development.

The waterfall model is simple to implement.

For implementation of small systems the waterfall model is used.

18. List the two deficiencies in waterfall life cycle model. Which process model do you suggest to overcome each deficiency? [MJ 2017]

Disadvantages of waterfall model:

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

I would prefer the spiral model in this approach.

19. Define an evolutionary prototype. **[ND 2019]**

Evolutionary Prototyping is a lifecycle model in which the system is developed in increments so that it can readily be modified in response to end-user and customer feedback.

20. Define Prototyping Model.

The Prototyping Model is a systems development method (SDM) in which a prototype (an early approximation of a final system or product) is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed.

21. What are the advantages of prototyping model? **[ND 2014]**

Advantages of prototyping model -

The working model of the system can be quickly designed by construction of prototype. This gives the idea of final system to the user.

The prototype is evaluated by the user and the requirements can be refined during the process of software development.

This method encourages active participation of developer and user.

This type of model is cost effective.

This model helps to refine the potential risks, associated with the delivery of the final system.

The system development speed can be increased with this approach.

22. Define RAD

Rapid application development (RAD) is a software development methodology that uses minimal planning in favor of rapid prototyping. A prototype is a working model that is functionally equivalent to a component of the product.

23. What are the phases encompassed in the RAD model?

Various phases in the RAD model are

1. Business modelling
2. Data modelling
3. Process modelling
4. Application generation
5. Testing and turnover

24. What are the drawbacks of rapid application development life cycle model?

Disadvantages of RAD model:

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

25. Define Agile?

The word 'agile' means −

- Able to move your body quickly and easily.
- Able to think quickly and clearly.

26. What agile manifesto states?

The Agile Manifesto states that −

We are uncovering better ways of developing software by doing it and helping others do it. Through this work, we have come to value −

- **Individuals and interactions** over processes and tools.
- **Working software** over comprehensive documentation.
- **Customer collaboration** over contract negotiation.
- **Responding to change over** following a plan

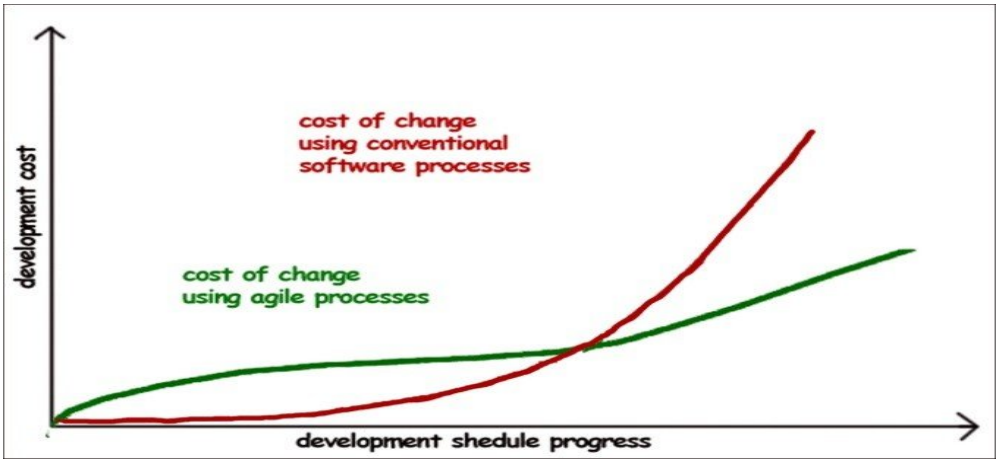
27. Define Agile process.

- This approach is rigid, that means if requirements get changed, then rework was essential. Hence new methods were proposed in 1990;s which are known as agile process.
- The agile processes are the light-weight methods are people –based rather than plan based methods.
- The agile process forces the development team to focus on software itself rather than design and documentation.
- The agile process believes in iterative method.
- The aim of agile process is to deliver the working model of software quickly to the customer.

For Eg: Extreme programming is the best known of agile process

28. What the Agile Methodology states?

- The agile method proponents claim that if the software development is carried out using the agile approach then it will allow the software team to accommodate changes late in a software project without dramatic cost and time impact.
- In other words, if the incremental delivery is combined with agile practices such as continuous unit testing and pair programming then the cost of changes can be controlled.
- The following graph represents how the software development approach has a strong influence on the development cost due to changes suggested.



29. State the Twelve Agile Manifesto Principles

- Customer satisfaction through early and continuous software delivery
- Accommodate changing requirements throughout the development process
- Frequent delivery of working software
- Collaboration between the business stakeholders and developers throughout the project
- Support, trust, and motivate the people involved
- Enable face-to-face interactions
- Working software is the primary measure of progress
- Agile processes to support a consistent development pace
- Attention to technical detail and design enhances agility
- Simplicity
- Self-organizing teams encourage great architectures, requirements, and
- Regular reflections on how to become more effective

30. What is extreme programming?

- The Extreme Programming is commonly used agile process model.
- It uses the concept of object-oriented programming.
- A developer focuses on the framework activities like planning, design, coding and testing.

XP has a set of rules and practices.

31. What are the XP values?

- Communication
- Simplicity
- Feedback
- Courage
- Respect

32. What are the four main framework activities that XP process compromises

- Planning
- Design
- Coding
- Testing

33. List any two agile process model. **[AM 2019]**

- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Dynamic Systems Development Method (DSDM)
- Scrum.
- Crystal.
- Feature Driven Development (FDD)

PART B Questions Bank

Q.No	Questions	Univ. QP
1.	Explain in detail what Software Engineering is and list out various characteristics of a Software	[ND 2017]
2.	Write a short note on various goals and objectives of software being engineered and the challenges encountered while engineering software.	[MJ 2012]
3.	Discuss in detail what does the term "Software Crisis" mean to software developers	
4.	Write a short note on Layered Technology being followed in software engineering by explaining the software process	
5.	Elucidate the key features of the software process models with Suitable a Example	[MJ 2016]
6.	Identify the umbrella activities in software engineering process and explain them in detail	[ND 2017]
7.	Explain the CMMI Model to assess the organization level.	[ND 2017] [ND 2015]
8.	WHAT is a process model? Describe the process model that you would choose to manufacture a car. Explain giving suitable reasons.	[MJ 2017] [ND15/16]
9.	Discuss linear sequential model in detail along with its advantages and disadvantages [OR] Explain the Software Development Life Cycle using simple waterfall model.	[MJ 2012] [MJ 2016]
10.	Describe the situations where iterative enhancement model might lead to difficulties.	[MJ 2016]
11.	Explain in detail how Incremental Model of Software development is carried out.	
12.	Write a short note on RAD model and which type of applications suit	[MJ 2015]

	RAD model? Justify your answer. [OR] Neatly explain the Rapid Application Development process models and write their advantages and disadvantages.	
13.	Discuss the prototyping model. What is the effect of designing a prototype on the overall cost of the software project	[MJ 2016]
14.	With suitable illustration explain about SPIRAL model evolutionary software development. [OR] Neatly explain the spiral process models and write their advantages and disadvantages [OR] Which process model is best suited for risk management? Discuss with an example. Give the advantages and disadvantages.	[MJ 2015] [ND 2017] [ND 2016]
15.	Explain the Component based Software development with a neat a sketch.	[ND 2017]
16.	Explain Agile Process in detail?	
17.	Describe in detail Extreme Programming.	
18.	Explain Industrial Xp.	
19.	Compare the waterfall, prototyping and spiral model. List the features of each model, advantages and disadvantages and a type of application where the model will be acceptable .	[MJ 2019]
20.	Define Agility .List any five principles of agility(5 marks) Explain the phases in extreme programming process(8 Marks)	[MJ 2019]
21.	What is agility? Elaborate the agile principles.	[ND 2019]
22.	Outline the Spiral life cycle model with a diagram.	[ND 2019]