UNIT I

CHAPTER 1

Software Engineering Fundamentals

Syllabus

Software Engineering Fundamentals: Nature of Software, Software Engineering Principles, The Software Process, Software Myths.

Syllabus Topic : Software Engineering Fundamentals (Introduction)

1.1 Introduction

Review Questions

Q. Why Computer Software is important ?

Q. Explain the term Computer Software in brief.

- Computer software has become an integral part of our daily lives. It helps in all sorts of businesses and decision makings in business. The application of computer software includes : Telecommunication, transportation, military, medical sciences, online shopping, entertainment industry, office products, education industry, construction business, IT industry, banking sector and many more.
- The software applications have a great impact on our social life and cultural life as well. Due to its widespread use, it is the need of time to develop technologies to produce high quality, user friendly, efficient and economical software.
- Computer software is actually a product developed by software engineers by making use of various software engineering processes and activities.
- Software consists of data, programs and the related documents. All these elements build a configuration that is created as a part of the software engineering process. The main motive behind software engineering is to give a framework for building software with better quality.
- We can say that the software has become the key element in all computer based systems and products.

Syllabus Topic : Nature of Software

1.2 Nature of Software

Review Questions

Q. List and explain the characteristics that describe the nature of software.

Q. Describe the drawback of software.

- The nature of software has great impact on software engineering systems. The general nature of software may describe the important characteristic :
- **Reliability :** If we consider the use of software in air traffic control system and space shuttle, then there should not be any chances of failure of software.

- In these examples, if reliability is not taken into consideration, then the human life is at risk.
- In addition to this, there are four other important characteristics that describe the nature of software :

1. Absence of fundamental theory
2. Ease of change
3. Rapid evolution of technologies
4. Low manufacturing cost

1.2.1 Absence of Fundamental Theory

- If we consider the example of physics, we see there are fundamental laws of physics, but in software there are no such fundamental laws despite the researches done by the computer scientists.
- Because of this drawback, it is very difficult to do any reasoning about the software until it is developed. Thus the developers practice few software engineering standards that are not foolproof. But the codes written for developing software are following the discipline and some solid principles.

1.2.2 Ease of Change

- The software has the provision to be altered at any stage of time. Thus the software development organizations take the advantage of this feature. From the customer's point of view, the change is always required throughout its development cycle and even after its delivery to the customer.
- Since there are no basic rules of software, there is no rule available to accommodate the changes and its impact until it is developed. Thus we can say that the ease of change is a gift of God to the developers and the development organizations.

1.2.3 Rapid Evolution of Technologies

- In the modern age, the software development technologies and development environments are changing rapidly with an extreme speed.
- It becomes the need of the time to keep the software engineers updated and armed with latest technologies and skills.
- The software engineering standards must also be revised with the technology evolutions.

1.2.4 Low Manufacturing Cost

- The cost of software reproduction and installation is less as compared to a new development. Today nearly 80 percent of the software development contains only maintenance and only 20 percent is new development. This reflects that manufacturing a product is considerably involves very low cost.
- The software reusability is an important characteristic that benefits the development organizations a lot in manufacturing the software at low cost.

1.3 Software Engineering : A Layered Technology

SPPU - May 13, May 14, Dec. 14

University Questions

Q. Explain the layered approach to software engineering. **(May 2013, 4 Marks)**

Q. Define Software Engineering. What are the various categories of software? **(May 2014, 4 Marks)**

Q. Define Software Engineering. **(Dec. 2014, 6 Marks)**

Review Questions

Q. Define the term Software Engineering.

Q. Define Software Engineering as Layered Technology.

Q. How Software Characteristics are different from Hardware Characteristics ?

Q. Define the terms Software Crisis.

Q. Define the terms Legacy Software.

Q. Define software engineering. What are the characteristics of software which make its Engineering different from Industrial Manufacturing ?

Q. What are the problems associated with development process ? (Hint : Refer Software Crisis)

- The term software engineering is defined as :
- "By using the principles of sound engineering and its establishment, software is developed that should be economical and should work efficiently on real machines."
- Software engineering is a systematic and disciplined approach towards developments of software. The approach must be systematic for operation of the software and the maintenance of it too.
- Software engineering is considered as a layered technology. These layers includes :

1. Quality focus 2. Process
3. Methods 4. Tools

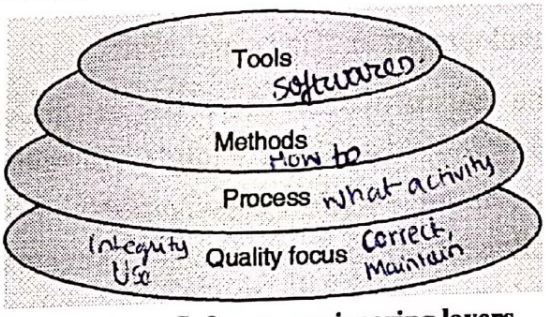

Fig. 1.3.1 : Software engineering layers

1.3.1 Quality Focus

Quality is nothing but **'degree of goodness'**. Software quality cannot be measured directly. Good quality software has following characteristics :

1. **Correctness** is degree to which software performs its required function.
2. **Maintainability** is an ease with which software is maintained. Maintenance of software implies change in software. If software is easy to maintain, then the quality of that software is supposed to be good.
3. **Integrity** is a security provided so that unauthorized user can not access data or information. e.g. password authentication.
4. **Usability** is the efforts required to use or operate the software.

1.3.2 Process

Software process defines a framework that includes different activities and tasks. In short, process defines following **'what'** activities and tasks for software development :

1. What activities are to be carried out ?
2. What actions will be taken ?
3. What tasks are to be carried out in a given action ?

1.3.3 Methods

Method provides technical way to implement the software i.e. **'how to'** implement. Methods consist of collection of tasks that include :

1. **Communication :** Between customer and developer.
2. **Requirement analysis :** To state requirements in detail.
3. **Analysis and design modelling :** To build a prototyping model of software to exhibit the requirements clearly.
4. **Program construction :** Implementation of requirements using conventional programming languages or automated tools.
5. **Testing and support :** Test for errors and expected results.

1.3.4 Tools

Software tool is an automated support for software development.

For example

1. Microsoft front page or Microsoft Publisher can be used as web designing tool.

2. Rational Rose can be used as object oriented analysis and design tool.

1.3.5 The Characteristics of Software

SPPU - May 14, Dec. 16, Apr. 17

University Question

Q. What are the software characteristics? **(May 2014, Dec. 2016, Apr. 2017, 4 Marks)**

- Software is having some characteristics, those are totally different from hardware characteristics or the industrial manufacturing
- Software is developed or engineered and it is not manufactured like other hardware products.
- There exist some similarities between development of software and manufacturing of hardware
- In both the cases quality is achieved by good design but manufacturing phase of hardware can introduce quality problems that are absent in software.
- Both activities depend on people but relationship between people applied and work done is different in both the cases.
- Both the cases require the 'construction of product', but approaches are different.

Software does not wear out.

- Note that, wear out means process of losing the material.
- Hardware components are affected by dust, temperature and other environmental maladies. Stated simply, hardware can wear out. However software does not influenced by such environmental means.
- When hardware component wear out, it can be replaced by spare part. But there are no spare parts for software. Any software error indicates error in design or coding of that software.
- Hence software maintenance involves more complexity than hardware maintenance.
- Mostly software is custom built rather than assembled from existing components.
- Computer hardware can be assembled from existing components as per our requirements. But that is not the case for software. Software is developed according to customer's need.

1.3.6 Software Crisis

SPPU - May 13

University Question

Q. Explain the term : Software crisis. **(May 2013, 2 Marks)**

- Most of software engineers refer the problems associated with software development as "Software crisis".
- The causes of software crisis are linked to all the problems and complexities associated with development process. Following are some most encountered problems associated with development process :

1. Running out of time i.e. deadline crossed
2. Over budget
3. Software inefficient
4. Low quality
5. Not up to expectations of customers
6. No enough development team

1.3.7 Legacy Software

SPPU - May 13

University Question

Q. Explain the term : Legacy software. **(May 2013, 2 Marks)**

- The term legacy software refers to older software developments that were poorly designed and documented. It had supported

for many years. The legacy systems may or may not remain in use.

- Even if it is no longer used, it may continue to impact the organization due to its historical role. Historic data may not have been converted into the new system format and may exist within the new system.

Syllabus Topic : Software Engineering Principles

1.4 Software Engineering Principles

Review Questions

Q. What are seven Software Engineering Principles ? Explain each in detail.

Q. Why continuous validation is required ?

Q. Why it is required to maintain Disciplined Product Control ?

- There are seven Software Engineering Principles that have been determined which are actually supposed to be the independent and complete set.
- Following are the list of seven Software Engineering Principles :
  1. Manage the software using the phrased life cycle.
  2. During the development process, perform continuous validation.
  3. Maintain a disciplined control on the flow of product development.
  4. In development process, use all the innovative and modern programming practices.
  5. Maintain all the accountability of results.
  6. Use the best team.
  7. Always maintain a commitment to improve the overall process.

Principle 1 : Manage the software using the phrased life cycle go acc to phases.

- Planned and controlled software projects are conducted for one and only reason because it is the only way known to manage complexity.
- Project failure rate remains higher than it should be even though the success rate for software projects has improved. One of the reasons is that the customers lose interest quickly (because what they have requested was not really as important as they first thought), and the projects are cancelled.
- The software engineers and the software project manager must follow certain guidelines for project plans in order to avoid project failure :
  1. Carefully design a set of common warning signs.
  2. Understand the critical success factors that lead to good project management, and
  3. Develop a common sense approach for planning, monitoring, and controlling the project.
- A description of the products to be developed in each phase, and of the associated development activities and schedules. Major products of each phase should be identified as discrete milestones, in such a way that there can be no ambiguity about whether or not a milestone has been achieved.
- The component activities in each phase should be identified in some detail.
- Following are the six phases in every Software development life cycle model:
  1. Requirement gathering and analysis
  2. Design
  3. Implementation or coding
  4. Testing
  5. Deployment
  6. Maintenance

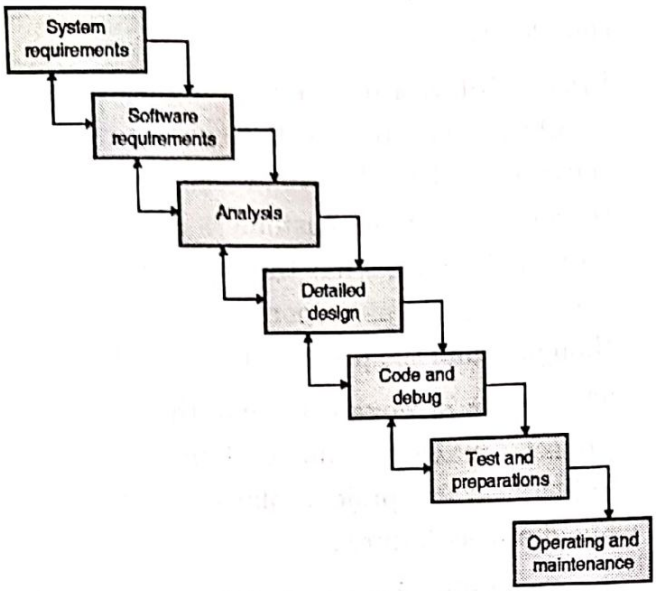

Fig. 1.4.1 : Phases in Software development life cycle

Principle 2 : Continuous validation Check continuously

- One of the most common mistakes (that proves costly later in software projects today) is to defer the activity of detecting and correcting software problems until late in the project, i.e. in the "test and validation" phase after the code has been developed.
- There are two main reasons why this is a mistake:
  1. Most of the errors have already been made before coding begins and
  2. The later an error is detected and corrected, it is a expensive affair.
- Thus there must be continuous validation throughout the development process.

Principle 3 : Maintain disciplined product control

- In fact, any good-sized project must be upto date. accommodate changes in requirements throughout the development cycle.
- Some changes may be due to external environment like :
  1. New government reporting regulations
  2. Improvements in technology
  3. User organizational changes or changes in the overall system like bank, refinery and retail store etc. where this software product is used.
- All these changes have good impact on the system development. Thus it becomes very easy that different versions of the documentation and the code to be reproduced frequently.
- The tester or the user observes that the system is different from the system that was proposed actually.
- Thus, it is necessary to maintain a disciplined product control activity throughout the system life-cycle to avoid such mismatches.

Principle 4 : Use of latest programming practices

- The use of modern programming practices like top-down structured programming or object oriented programming practices help to deal with more visibility into software development process.
- Enhancement in modern practices has good features and functionalities.

Principle 5 : Maintain accountability for results

- Each team member in the project team should have a clear understanding of the results for which he or his group are accountable, and a clear understanding that his future rewards depend on how well he does in producing those results.
- For a project to be able to do this, one additional thing is needed and that is adequate visibility into each team member's individual performance

Principle 6 : Use the best team

- Even if all the principles from 1 to 5 are observed carefully, still there are chances of project failure.
- This failure can be caused due to incompetent individual team member.

- Thus it is very important to recruit a staff of a good repute and a good caliber.

Principle 7 : Always maintain a commitment to improve the overall process

- All the above principles are good enough to produce a good software product. But still it's a good practice to improve the regular processes and practices.
- It is significant in terms of the need for planning and understanding your organization.

Syllabus Topic : The Software Process

1.5 The Software Process

framework ÷ into small activities

Q6

SPPU – May 12, May 15

University Questions

Q. What is a software engineering process ? **(May 2012, 3 Marks)**

Q. What is Software process framework? Explain in detail. **(May 2015, 7 Marks)**

Review Questions

Q. What are various Umbrella Activities associated in the Development Process ?

Q. Describe Common Process Framework with the help of diagram.

- A software process can be illustrated by the following Fig. 1.5.1. In the Fig. 1.5.1, a common process framework is exhibited. This framework is established by dividing overall framework activities into small number of framework activities.

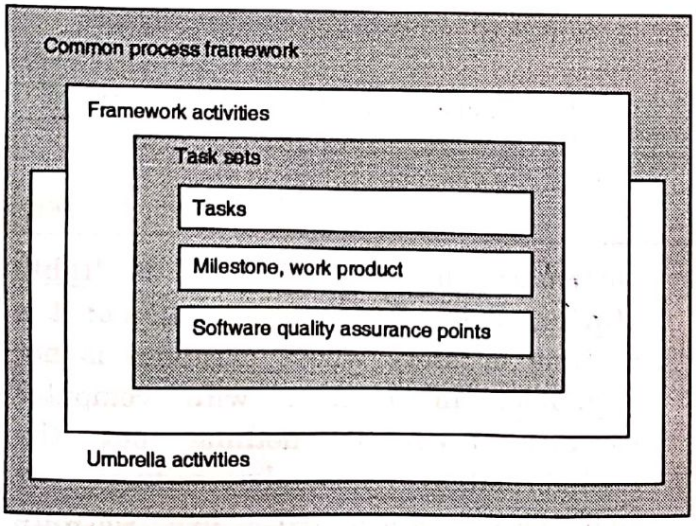

Fig. 1.5.1 : A software process

- And these small activities are applicable to the entire project irrespective of its size and complexity. A framework is a collection of task sets and these task sets consists of :
  - Collection of small work tasks
  - Project milestones, deliverable i.e. actual work product and
  - Software quality assurance points
- There are various activities called **umbrella activities** are also there and these activities are associated throughout the development process. The umbrella activities include :

Q7

1. Software project tracking and control
2. Risk management
3. Software Quality Assurance (SQA)
4. Formal Technical Reviews (FTR)
5. Measurement
6. Software Configuration Management (SCM)
7. Reusability Management
8. Work product preparation and production

- All these umbrella activities actually constitute the process model. Also the umbrella activities are independent and occur throughout the process

1.5.1 Umbrella Activities

Software project tracking and control
Software quality improvement Risk management.

SPPU – Feb. 15, May 16

University Question

Q. What are various umbrella activities applied throughout a software project? **(Feb. 2015, May 2016, 7 Marks)**

The framework described in generic view of software engineering (Fig. 1.5.1 : A software process) is complemented by number of Umbrella activities. Typical **umbrella activities** are :

1. Software project tracking and control

- Developing team has to assess project plan and compare with predefined schedule.

- If project plan doesn't match with predefined schedule, necessary actions are taken to maintain the schedule.

2. Risk management

Risk is event that may or may not occur. But if that event happens, it causes some unwanted outcomes. Hence proper management of risk is required.

3. Software Quality Assurance (SQA)

- SQA is nothing but planned and systematic pattern of activities those are required to give guarantee of software quality.
- For example during software development, meetings are conducted in every stage of development to find out defects and suggest improvement to yield good quality software.

4. Formal Technical Reviews (FTR)

- FTR is a meeting conducted by technical staff. The purpose of meeting is to detect quality problems and suggest improvements.
- The technical staff focuses on quality from customer's point of view, i.e. what customer exactly wants.

5. Measurement

- It includes the efforts required to measure the software.
- Software can not be measured directly. It is measured by some direct measures (e.g. cost, lines of code, size of software etc) and indirect measures (e.g. quality of software, which is measured in terms of other factors. Hence it is an indirect measure of software.)

6. Software Configuration Management (SCM)

It manages the effects of change throughout the software process.

7. Reusability management

- It defines criteria for product reuse.
- If software components developed for certain application can be used in development of other applications, then it's good quality software.

8. Work product preparation and production

It includes the activities required to create documents, logs, forms, lists and user manuals for developed software.

Syllabus Topic : Software Myths

1.6 Software Myths

Q 8

SPPU - May 12, May 13, Dec. 13, Dec. 14, Feb. 15

University Questions

Q. State a software myth each which customers, developers and project manager believe. What is the reality in each case ?
(May 2012, 5 Marks)

Q. Explain the term : Software myth.
(May 2013, 2 Marks)

Q. What do you mean by software myths?
(Dec. 2013, 3 Marks)

Q. What is software myth ? Give an example.
(Dec. 2014, 4 Marks)

Q. What are different software myths that the customers and practitioners believe in and what are their corresponding realities ?
(Feb. 2015, 4 Marks)

– Dictionary meaning of myth is 'fable stories'. It is a fiction, imagination or it is a thing or story whose existence is not verifiable. In relation with computer software myth is nothing but the misinformation, misunderstandings, or confusions propagated in software development field.

- In software development the myths can be considered as three level myths.

1. Management level myths
2. Customer level myths
3. Practitioner level myths

1.6.1 Management Level Myths (or Manager Level Myths) SPPU - May 14

University Question

Q. What are the management myths? **(May 2014, 4 Marks)**

➢ **Myth**

Manager thinks "there is no need to change approach to software development. We can develop same kind of software that we have developed ten years ago".

➢ **Reality**

- Application domain may be same but quality of software need to improve according to customer's demand.
- The customer demands change time to time.

➢ **Myth**

We can buy software tools and use them.

➢ **Reality**

- Software tools are readymade software that helps in creation of other software e.g. Microsoft front page/ Microsoft Publisher. All these software can be used for web development.
- Rational rose used to draw UML diagrams (e.g. use cases, sequential, class diagrams etc).
- Such software tools are available for every stage of software development (Requirement analysis, designing, coding, testing etc). But majority of software developers do not use them. Moreover, these tools also have some limitations.

➢ **Myth**

Manager thinks "when needed, we can add more programmers for faster software development".

➢ **Reality**

- When new peoples are added to the project, the training must be given to such new comers.
- Hence people previously working on that project spend time for training people.
- Hence project can not be completed fast just by adding more people at any time during project development.

➢ **Myth**

If the developer outsource the software project to a third party, he can be tension free and wait for the project to done smoothly.

➢ **Reality**

When an organization is unable to manage and control the software projects internally, it will also be very difficult for them to get the project done by outsourcing it.

➢ **Myth**

There is a book that contains standards and procedures for developing software, it will provide the developer everything that he needs in the development process.

➢ **Reality**

The rule book exists. But the questions arise :

- Is it actually used by the developers ?
- Are software developers aware of this type of book of standards and procedures ?
- Does it contain all the modern engineering practices ?
- Is it foolproof ?
- Does it focus on quality ?

- Does it streamlined to timely delivery ?
- In most of the cases, the simple answer to all these queries is a big "NO".

➢ **Myth**

latest comp → efficiency in work

- The organization has state-of-the-art development tools.
- They have the latest configuration computers for their developers to provide the good platform to produce their work efficiently.

➢ **Reality** CASE → efficiency in work

- In actual scenario the latest hardware configuration computer will not help in producing high-quality software.
- But the Computer-Aided Software Engineering (CASE) tools are very important for achieving good quality software.
- In most of the organizations the software developers do not use these CASE tools effectively and efficiently.

Q10

1.6.2 Customer Level Myths

SPPU – May 14

University Question

Q. State and explain customer's myth. **(May 2014, 4 Marks)**

➢ **Myth** Only general is suffic

Only the general statement is sufficient and no need to mention detail project requirements.

➢ **Reality** Need to mention details

- Only general statement is not sufficient for project development. Other detail project requirements are also essential.
- Customer must provide other information, design details, validation criteria.
- Hence during complete software development process customer and developer communication is essential.

➢ **Myth**

Project requirements continuously change but can be easily accommodated in software.

➢ **Reality** cost ↑s

- If change is requested in early stages of software development, it can be easily carried out. But if change is requested in later stages, cost of change rapidly increases.
- If change is requested in maintenance, modifications are required in design, coding, testing.
- Hence cost of modification rapidly increases. Thus changes can't be easily absorbed. They result in increase in cost.

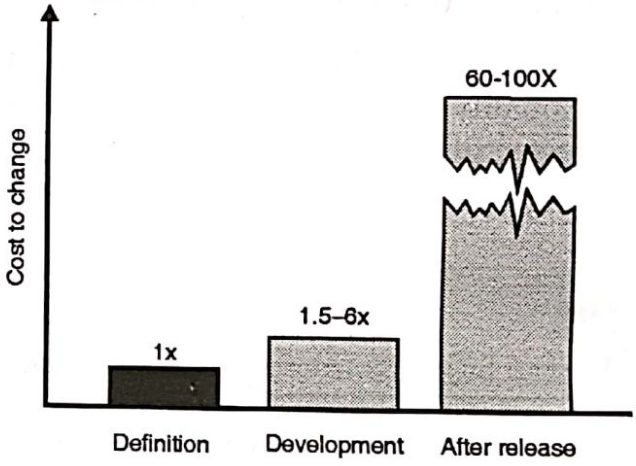

Fig. 1.6.1: The impact of change

Q11

1.6.3 Practitioner Level Myths (or Developer Level Myths)

SPPU – Dec. 12, Dec. 13, Feb.16, Apr. 17

University Questions

Q. State the myths of the practitioner and the realities. **(Dec. 2012, 6 Marks)**

Q. State the myths and facts for software developer. **(Dec. 2013, 3 Marks)**

Q. List and explain practitioner's myths and what are their corresponding realities? **(Feb. 2016, 4 Marks)**

Q. Briefly explain the myths and reality associated with practitioner and management in software project. **(Apr. 2017, 5 Marks)**

➢ **Myth**

Practitioners think "Once we write program and get into work, our job is over".

➢ **Reality**

Almost 60 to 80 percent work is required after delivering the project to the customer for the first time.

➢ **Myth**

Until I get program running, I have no way of assessing its quality.

➢ **Reality** Use FTR

- The most effective quality assurance mechanisms can be applied during project development. The formal Technical Reviews are conducted to assure the quality.
- Formal Technical Review is meeting conducted by technical staff. FTR is applied in any stage of software development (Requirement analysis, designing, coding, testing etc).
- FTR is not problem solving activity but it is applied to find out defects in any stage of software development. These defects can then removed to improve the quality of software.

➢ **Myth**

When project is successful, deliverable product is only working program.

➢ **Reality**

Working program is just part of software product. Actual project delivery includes all documentations, help files and guidance for handling the software by user.

Perescriptive Process Models
1.) Communication.
2.) Planning
3.) Modeling
4.) Construction
5.) Deployment.

➢ **Myth**

The software engineering process creates larger and unnecessary documentation and ultimately it will slow down the process.

➢ **Reality**

- Software engineering the process that creates the quality and it is not the process of only creating the documents.
- The good quality of the product will reduce the extra work involved in rework.
- And finally reducing the overhead of rework will lead to quicker development to complete the desired work on scheduled time.

1.7 Importance of Software Engineering

Q12 **SPPU - Dec. 14**

University Question

Q. Explain the importance of Software Engineering. **(Dec. 2014, 4 Marks)**

- Software engineering is the study and application of engineering to the design, development and maintenance of software that needed in all the aspects of our daily lives.
- In every field of work, specific software is needed. Since software is developed and embedded in machines so that it could meet the requirement of the user. This is possible because of software engineering.
- Using software engineering concepts reduces the complexity in developing the software application.
- It also reduces the software cost.
- Development time is reduced considerably.

□□□

Q2) Categories of Software
a) System Software
b) Application Software
c) Engg software
d) Embedded software
e) Web applications
f) Artificial intelligence software

UNIT I

CHAPTER 2

Process Models

Syllabus

Process Models :A Generic Process Model, Prescriptive Process Models: The Waterfall, Incremental Process(RAD), Evolutionary Process, Unified Process, Concurrent.

Syllabus Topic : A Generic Process Model

2.1 A Generic Process Model (or Generic Process Framework)

SPPU - Dec. 12

University Question

Q. State the generic process framework activities. **(Dec. 2012, 4 Marks)**

A software process is collection of various activities. There are five generic process framework activities :

1. Communication
2. Planning
3. Modeling
4. Construction
5. Deployment

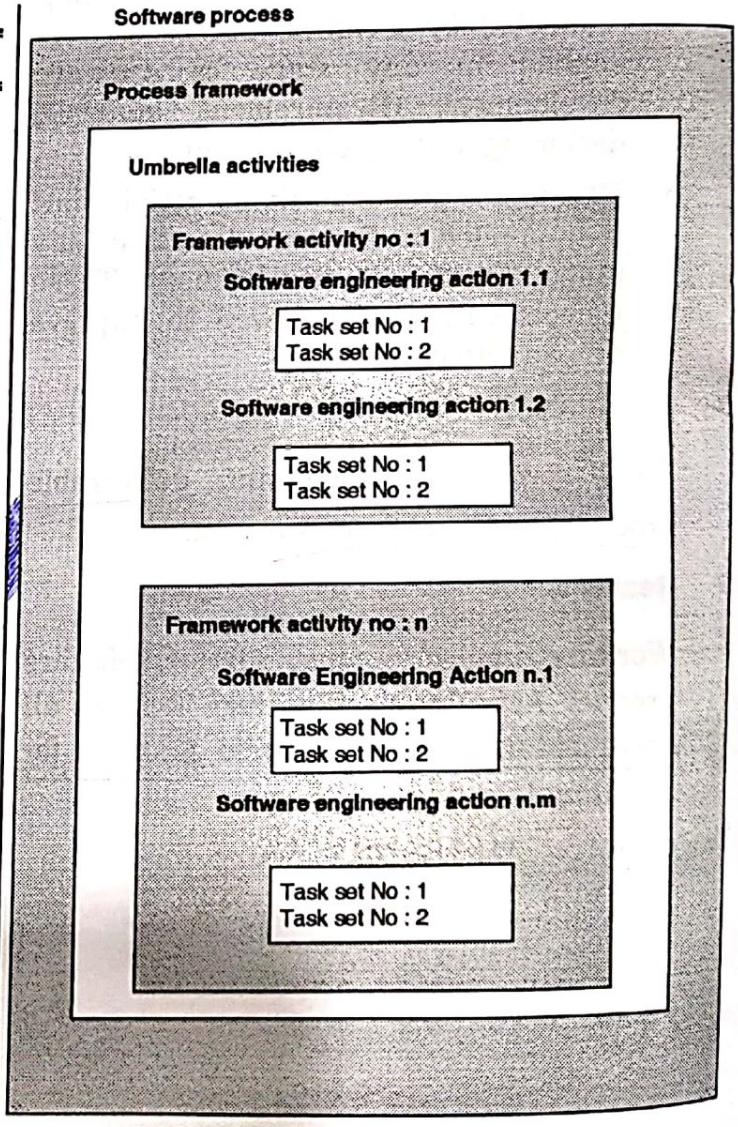

Fig. 2.1.1 : A software process framework

2.1.1 Communication

- The software development process starts with communication between customer and developer.
- According to waterfall model customer must state all requirements at the beginning of project.

2.1.2 Planning

It includes complete estimation (e.g. cost estimation of project) and scheduling (complete timeline chart for project development) and tracking.

2.1.3 Modeling

- It includes detail requirement analysis and project design (algorithm, flowchart etc).
- Flowchart shows complete pictorial flow of program where as algorithm is step-by-step solution of problem.

2.1.4 Construction

It includes coding and testing steps :

(i) **Coding :** Design details are implemented using appropriate programming language.

(ii) **Testing :** Testing is carried out for the following reasons :

- To check whether the flow of coding is correct or not.
- To check out the errors of program e.g. in C program just by pressing F7 key we check step by step execution of program or by using "Add-Watch" we add the variables and watch the values of variables.
- To check whether program is giving expected output as per input specifications.

2.1.5 Deployment

- It includes software delivery, support and feedback from customer.
- If customer suggest some corrections, or demands additional capabilities, then changes are required for such corrections or enhancement.
- These five generic framework activities can be used for :
  - Development of small programs
  - Creation of large programs
  - Development of large system based programs
- In addition to these five generic framework activities, various umbrella activities are also associated throughout the development process.

Syllabus Topic : Prescriptive Process Models

2.2 Prescriptive Process Models

- All the software organizations describe a unique set of framework activities for their own development processes. In general, it should adopt all the generic framework activities and the set of tasks defined in Section 1.5 i.e. generic process model. Whatever process model is chosen by the organization, but it should encompass the following framework activities :

  1. Communication
  2. Planning
  3. Modeling
  4. Construction
  5. Deployment

- The name 'prescriptive' is given since the model prescribes set of activities, actions, tasks, quality assurance and change control mechanism for every project. Each of the prescriptive models also prescribes a workflow.
- Workflow is defined as the flow of process elements and the manner in which they are interrelated. All the generic framework activities are defined earlier, but each of the prescriptive models put different emphasis to these generic framework activities and give different workflow.

- In the following section, we describe some of the prescriptive process models :

1. The waterfall model
2. Incremental process models
3. Evolutionary process models
4. The specialized process models

Syllabus Topic : The Waterfall Model

2.2.1 The Waterfall Model

SPPU – Dec. 12, Dec. 13, Dec. 14

University Questions

Q. Explain waterfall model with its advantages and disadvantages. **(Dec. 2012, 6 Marks)**

Q. Explain the waterfall model with V model. **(Dec. 2013, 6 Marks)**

Q. What are the disadvantages of the waterfall model ? **(Dec. 2014, 2 Marks)**

Review Question

Q. Explain the waterfall model with work products of each activity.

- This model is also called as **'Linear sequential model'** or **'Classic life cycle model'**. Classic life cycle paradigm begin at system level and goes through analysis, design, coding, testing and maintenance.

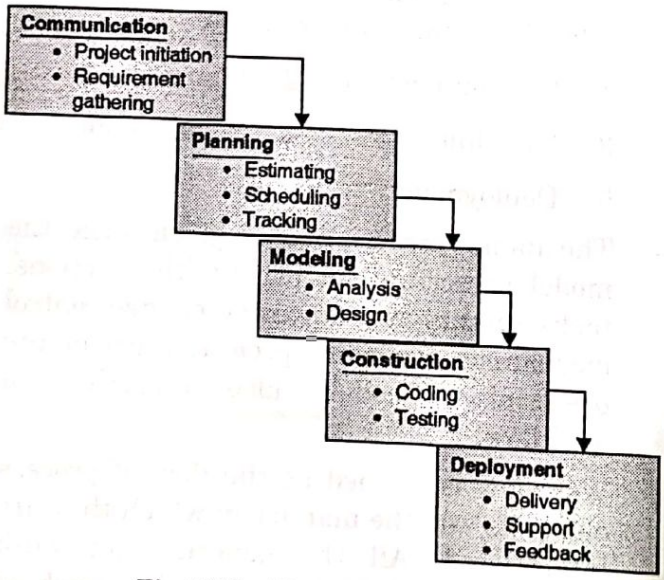

Fig. 2.2.1 : The Waterfall model

- An alternative design for 'Linear Sequential Model' is as shown in Fig. 2.2.2.

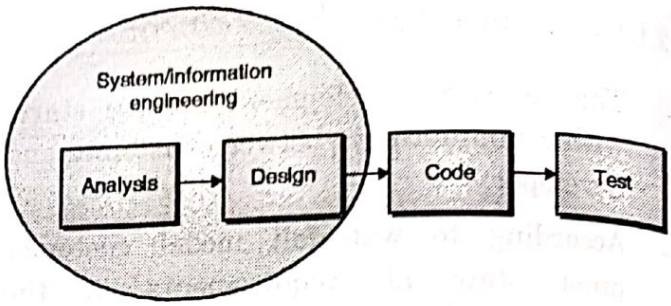

Fig. 2.2.2 : The linear sequential model

1. **Communication**
   - The software development process starts with communication between customer and developer.
   - According to waterfall model customer must state all requirements at the beginning of project.
2. **Planning**

   It includes complete estimation (e.g. cost estimation of project) and scheduling (complete timeline chart for project development) and tracking.
3. **Modeling**
   - It includes detail requirement analysis and project design (algorithm, flowchart etc).
   - Flowchart shows complete pictorial flow of program whereas algorithm is step-by-step solution of problem.
4. **Construction**

   It includes coding and testing steps :

   (i) **Coding** : Design details are implemented using appropriate programming language.

   (ii) **Testing** : Testing is carried out to check whether flow of coding is correct, to check out the errors of program e.g. in C program just by pressing F7 key we check step by step execution of program or by using "Add-Watch" we add the variables and watch the values of variables, to check whether program is giving expected output as per input specifications.

5. Deployment

- It includes software delivery, support and feedback from customer.
- If customer suggest some corrections, or demands additional capabilities then changes are required for such corrections or enhancement.

Merits / Advantages

1. This model is very simple and easy to understand and use.
2. Waterfall model is systematic sequential approach for software development. Hence it is most widely used paradigm for software development.
3. In this approach, each phase is processed and completed at one time and thus avoids phases overlapping.
4. It is very easy to manage since all the requirements are very well understood in the beginning itself.
5. It establishes the milestones whenever the products are produced or reviews available.

Demerits / Disadvantages

1. Problems in this model remain uncovered until software testing.
2. One of the disadvantages of this approach is 'blocking states'. In blocking state, the members of development team waits for other members to complete the dependent tasks. Due to this, huge amount of time spent in waiting rather than spending time on some productive work. In today's world, the software work is fast paced and thus waterfall model is not useful.
3. According to this model customer must state all his requirements at the beginning stage of development, which is difficult for the customer.
4. This model is step-by-step systematic sequential approach. Ultimately, customer gets the working version of software too late. Hence customer requires patience.
5. This approach is actually not realistic and does not match with real projects.
6. Also it does not incorporate the risk assessment.
7. Finally it is useful for smaller projects only where requirements are well understood in the beginning only.

2.2.1.1 V-Model (Software Development) Q15)

SPPU - Dec. 13

University Question

Q. Explain the waterfall model with V model. **(Dec. 2013, 6 Marks)**

- In software engineering development process, the V-model represents a development process that may be considered as an extension of the waterfall model.
- In more general V-model, instead of moving down in a linear way, the process steps are bent upwards after the coding phase as shown in Fig. 2.2.3.

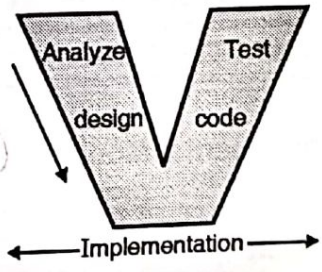

Fig. 2.2.3 : V-model

- The V-model demonstrates the relationships between each phase of the development life cycle.

Syllabus Topic : Incremental Process Models

2.2.2 Incremental Process Models

- Using these models a limited set of customer's requirements are implemented quickly and are delivered to customer.
- Then modified and expanded requirements are implemented step by step.
- Actually in this approach, the overall functionalities are split to smaller modules and these modules are quickly developed and delivered. Then refinements are possible in later increments or versions.
- In this approach, the initial requirements are very well defined. Each increment is passed through the overall development cycle i.e. phases of classic life cycle discussed earlier.
- The customer's feedback is very important after each of the increments (or releases) and next increment is developed. This process continues till the product is finished i.e. all the requirements are satisfied.
- Following are the two types of incremental process models :

1. The incremental model 2. The RAD model

Syllabus Topic : The Incremental Model

2.2.2.1 The Incremental Model (Q16)

SPPU - May 14, Dec. 14

University Question

Q. Explain the incremental model with advantages and disadvantages ?

(May 2014, Dec. 2014, 6 Marks)

- The incremental model combines the elements of waterfall model applied in an interactive fashion. The first increment is generally a core product.
- Each increment produces the product, which is submitted to the customer. The customer suggests some modifications.

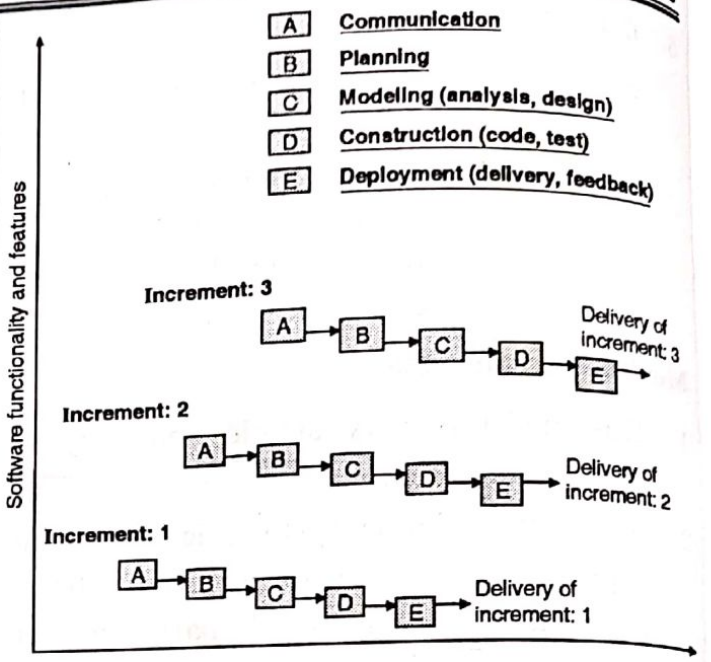

Fig. 2.2.4 : Project calendar time

- The next increment implements customer's suggestions and some additional requirements in previous increment. The process is repeated until the product is finished.
- Consider the development of word processing software (like MS Word or Word Star) using incremental model.

Increment – 1

Basic file management functions like : New, Open, Save, Save as etc. can be implemented in first increment.

Increment – 2

- More sophisticated editing and document production capabilities can be implemented in second increment.
- For example rulers, margins and multiple font selection can be implemented.

Increment – 3

Spelling, grammar checking and auto correction of words is implemented in third increment.

Increment – 4

The final advanced page layout capabilities are developed in fourth increment. And so on till the product is finished.

Merits / Advantages

1. The incremental model is useful when the size of development team is small in the beginning or some team members are not available. Thus early increments can be implemented with small size of team.
2. If the product satisfies the client, then the size of team can be increased.
3. Also increments can be properly planned to handle all types of technical risks. For example some application may require a new hardware that is presently not available. In this situation the increments can be planned to avoid the use of that hardware.
4. The initial product delivery is faster and cost of development is low.
5. The customers can respond to the functionalities after each increment and come up with feedback.

Demerits / Disadvantages

1. The cost of finished product may be increased in the end beyond the cost estimated.
2. After each increment, the customer can demand for additional functionalities that may cause serious problems to the system architecture.
3. It needs a very clear and complete planning and design before the whole system is broken into small increments.

Syllabus Topic : The RAD Model

2.2.2.2 The RAD Model

Review Question

Q. Explain the RAD model with advantages and disadvantages?

- RAD (Rapid Action Development) model is high-speed adaptation of waterfall model. Using RAD model, software product can be developed within a very short period of time i.e. almost within 60 to 90 days.
- The initial activity starts with communication between customer and developer. Depending upon initial requirements the project planning is done. Now the requirements are divided into different groups and each requirement group is assigned to different teams.
- Like other approaches, in RAD approach also all the generic framework activities are carried out. These generic framework activities are already discussed earlier.
- When all teams are ready with their final products, the product of each team is integrated i.e. combined to form a product as a whole.
- In RAD model each team carries out the following steps :

1. **Communication :** It understands the business problem and information for software.
2. **Planning :** Since various teams work together on different modules simultaneously, planning is important part of development process.
3. **Modeling :** It includes :
   - (i) **Business Modeling :** It includes information flow among different functions in the project e.g., what information will be produced by each function, which functions handles that information etc.
   - (ii) **Data Modeling :** It includes different data objects used in software and relationship among different objects.
   - (iii) **Process Modeling :** During process modeling, process descriptions are created. e.g. add, modify, delete etc.
4. **Construction :** It includes :
   - **(i) Component reuse**

     Reusable components means the software components i.e. modules or procedures that are developed for specific application can be reused in development of other application.
   - **(ii) Automatic code generation**

     The software producing the codes after providing proper inputs or

selecting readymade options, e.g. Microsoft FrontPage used in Web development, Rational Rose used for Object Oriented analysis and designing.

(iii) Testing

Testing is carried out to check whether flow of coding is correct, to check out the errors of program e.g. In C program just by pressing F7 key we check step by step execution of program or by using "Add-Watch" we add the variables and watch the values of variables, or check whether program is giving expected output as per input specifications.

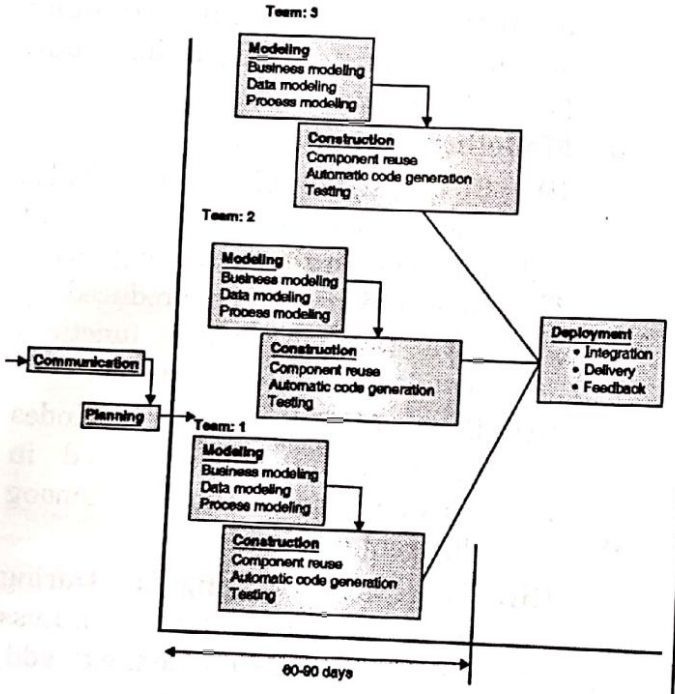

Fig. 2.2.5 : The RAD Model

Merits / Advantages

1. Using the RAD approach, a product can be developed within very short period of time.
2. It supports increased reusability of the components.
3. It results in minimal code writing as it supports automatic code generation.
4. It encourages the customer feedback.
5. In this approach, quick initial reviews are possible.
6. In this approach, modules integration is done from the beginning thus resolving number of integration issues.

Demerits / Disadvantages

1. It can be used, if sufficient number of staff is available. Thus, it requires sufficient man power to create number of teams.
2. In RAD approach, many teams work parallel to implement different functions of same project. Hence all teams must work with equal speed. If one team lags behind the schedule, overall project delivery will be late.
3. If system can not be modularized properly, then building the components for RAD model will be problematic.
4. The RAD model is not appropriate when technical risks are high. (e.g. a new application makes heavy use of new technology).
5. This approach is useful for only larger projects.
6. The RAD model had two primary disadvantages as follows :
   - **Reduced scalability :** It occurs because a RAD application evolves from a prototype to a finished application. Thus there is less scope for scalability.
   - **Reduced features :** It occurs due to time boxing. Time boxing is a time management technique in which the task is to be completed in a given frame of time called time boxes. Thus the features are pushed to be completed to the later releases due to short amount of time.

Syllabus Topic : Evolutionary Process Models

2.2.3 Evolutionary Process Models

SPPU - May 12

University Question

Q. Explain evolutionary process models mentioning the types of projects for which they are suitable. **(May 2012, 6 Marks)**

Review Question

Q. What do you mean by evolutionary process models ? Explain spiral model as an evolutionary process model ?

- Evolutionary process models are iterative type models. Using these models the developer can develop increasingly more complete versions of the software.
- As the requirements change very often, the end product may be unrealistic and a complete version of the product is not possible. To overcome this drawback, the concept of limited version product is introduced and this version is gradually completed over the period of time.
- Each version is refined based on the feedbacks till it is completed.
- Following are the examples of evolutionary process models :

1. The Prototyping paradigm
2. The Spiral model
3. The Concurrent development model

2.2.3.1 The Prototyping Paradigm

SPPU – Dec. 13

University Question

Q. Explain the prototyping model with its advantages and disadvantages. **(Dec. 2013, 6 Marks)**

Review Questions

Q. What do you mean by working and throwaway prototypes? Explain how they are used in prototyping model ?

Q. Explain advantages and disadvantages of prototyping model ?

- The **prototyping paradigm** offers best approach for human machine interaction. Ideally prototype serves as a mechanism for identifying software requirements.
- The prototyping approach is often called as **throw-away prototyping.** By using this approach, there is a rough demonstration of the customer's requirements i.e. the prototype used in demonstration is just a throwaway prototype.
- If working prototype is built, developer can use software tools if required (e.g. report generators)

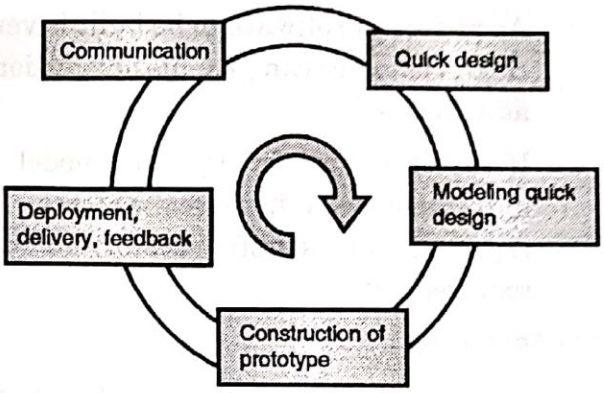

Fig. 2.2.6 : The prototyping model

- This produces working program quickly. Thus prototype can be served as 'first system'.
- Different phases of prototyping model are :

1. Communication

- The software prototyping paradigm starts with the communication between the developer and the customers.
- The software engineer and the customer meet regularly to define the overall objectives of the desired software and to identify it's requirements.

2. Quick design

- Quick design focuses on those aspects of software that are visible to the customer.
- It includes clear input, output formats and human machine interfaces.

3. Modeling quick design

The model of software is now built that gives clear idea of the software to be developed. This enables the developer to better understand the exact requirements.

4. Construction of prototype

The concept of modeling the quick design

leads to the development of prototype. This construction of prototype is deployed by the customer and it is evaluated by the customer itself.

5. **Deployment, Delivery, Feedback**
   - As picture of software to be built is very clear, customer can give his suggestions as a feedback.
   - If result is satisfactory, the model is implemented otherwise process is repeated to satisfy customers all requirements.

Merit / Advantage

Prototyping makes requirements more clear and system more transparent.

Demerits / Disadvantages

- The software developer generally compromises in the quality to get the working prototype quickly.
- Due to this reason, sometimes inappropriate operating system or programming language may be selected. He may use and implement inefficient algorithm.

2.2.3.2 The Spiral Model SPPU - Dec. 13

University Question

Q. State the activities of spiral model. **(Dec. 2013, 4 Marks)**

Review Question

Q. What are the advantages and disadvantages of spiral model ?

- The **Spiral model** is combination of well known waterfall model and iterative prototyping. It yields rapid development of more complete version of software.
- Using spiral model software is developed as series of evolutionary releases. During the initial releases, it may be just paperwork or prototype. But during later releases the version goes towards more completed stage.

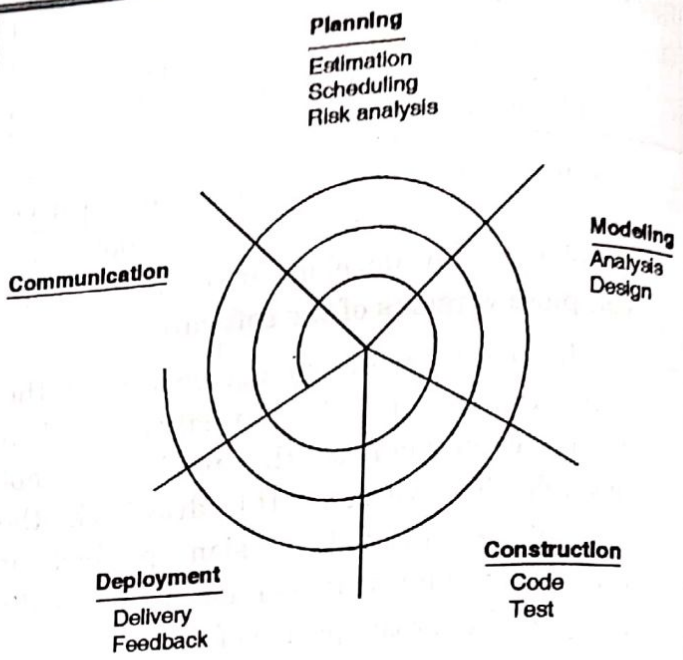

Fig. 2.2.7 : The Spiral Model

- The spiral model can be adopted to apply throughout the entire lifecycle of the application from concept development to maintenance. The spiral model is divided into set of framework activities defined by software engineer team. Each framework activity represents one segment of spiral as shown in Fig. 2.2.7.
- The initial activity is shown from centre of circle and developed in clockwise direction. Each spiral of the model includes following steps :

1. **Communication**

   The software development process starts with communication between customer and developer.

2. **Planning**

   It includes complete estimation (e.g. cost estimation of project) and scheduling (complete timeline chart for project development) and risk analysis.

3. **Modelling**
   - It includes detail requirement analysis and project design (algorithm, flowchart etc).

- Flowchart shows complete pictorial flow of program whereas algorithm is step by step solution of problem.

4. Construction

It includes coding and testing steps :

(i) **Coding :** Design details are implemented using appropriate programming language.

(ii) **Testing :** Testing is carried out.

5. Deployment

- It includes software delivery, support and feedback from customer. If customer suggest some corrections, or demands additional capabilities then changes are required for such corrections or enhancement.
- Note that after customer evaluation, next spiral implements, 'customer's suggestions' plus 'enhancement plan'. Thus, each of iteration around the spiral leads to more completed version of software.

Merits / Advantages

1. In this approach, project monitoring is very easy and more effective compared to other models.
2. It reduces the number of risk in software development before they become serious problems.
3. It is suitable for very high risk projects.
4. Project estimates i.e. schedule and cost is more realistic.
5. Risk management is in-built feature of spiral model.
6. Changes can be accommodated in the later stages of development.

Demerits / Disadvantages

1. If major risk is not discovered in early iteration of spiral, it may become a major risk in later stages.
2. Each iteration around the spiral leads to more completed version of software. But it's difficult to convince (especially in contract situation) to the customer that the model is controllable.
3. Cost of this approach is usually high.
4. It is not suitable for low risk projects.
5. Rules and protocols must be followed very strictly to implement the approach.

Syllabus Topic : Concurrent Model

2.2.3.3 The Concurrent Development Model

Review Question

Q. Explain concurrent development model with advantages and disadvantages ?

- The **concurrent development** model is also called as **concurrent model**. This model is more appropriate for system engineering projects where different engineering teams are involved. In system engineering stage, the project requirements are considered at system level.
- Diagrammatically it can be represented as a series of framework activities, task, actions and their associated states.
- Fig. 2.2.8 exhibits one element of the concurrent process model :

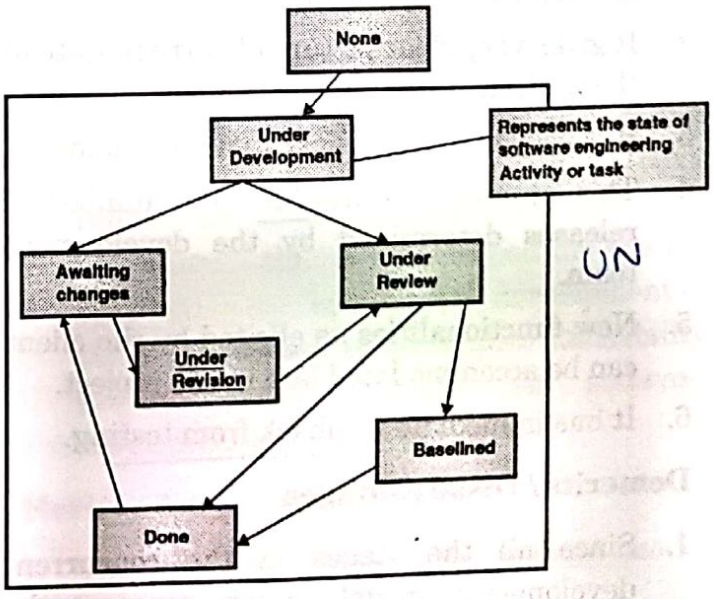

Fig. 2.2.8 : One element of concurrent process model (i.e. Modeling activity)

- The **modelling activity** is one of the states and other activities like **communication** or **construction** can also be represented in an analogous manner.
- Let the **communication activity** has completed its first iteration and available in **awaiting changes state.** The **modelling activity** has completed its initial communication and ready to move to **under development state** from none state.
- During these transitions, if customer indicates some modification in the requirement, then the **modelling activity** transits to **awaiting changes state** from **under development state**.
- Instead of considering activities, actions and tasks as sequence of events, it defines network of activities.
- The concurrent process model activities transits from one state to another state and this model is used in all types of software development and it provides very clear idea of the current state of the project.

Merits / Advantages

1. The concurrent development model is applicable to all types of software development processes.
2. It gives very clear picture of current state of the project.
3. It is easy to use and easy to understand.
4. This approach is flexible and number of releases determined by the development team.
5. New functionalities as elicited by the client can be accommodated late in the project.
6. It has immediate feedback from testing.

Demerits / Disadvantages

1. Since all the stages in the concurrent development model works concurrently, any change in the requirement from the client may halt the progress. This may happen due to dependent components among different stages and it lead to more longer development cycles as compared the planned cycles.
2. It requires excellent and updated communication between the team members. This may not be achieved all the time.
3. The SRS must be updated at regular intervals to reflect the changes.

2.2.3.4 Differentiation between Prescriptive and Evolutionary Process Models

Review Question

Q. Differentiate between prescriptive and evolutionary process models.

Sr. No.	Prescriptive process models	Evolutionary process models
1.	Developed to bring order and structure to the software development process.	Evolutionary software processes do not establish the maximum speed of the evolution. Due to this development process becomes slow.
2.	Defines a distinct set of activities, actions, tasks, milestones, and work products that are required to engineer high-quality software	Evolutionary process models lack flexibility, extensibility, and high quality.
3.	It is more popular.	It is less popular.
4.	It provides complete and full developed systems.	Time does not allow a full and complete system to be developed.
5.	**Example :** Water fall model, Incremental models.	**Example :** Prototyping, Spiral and Concurrent models.

2.2.4 The Specialized Process Models

- Specialized process models have many characteristics of one or more of the conventional models described in earlier section.

- Specialized models are applied when a narrowly defined engineering approach is chosen.
- Following are some specialized process models :

1. Component-based development models
2. The formal methods model
3. Aspect-oriented software development

2.2.4.1 Component-Based Development Models

SPPU - Dec. 13, Dec. 14

University Questions

Q. Explain the component development model with activities. **(Dec. 2013, 6 Marks)**

Q. Explain how component based development reduces the time of software development ? **(Dec. 2014, 6 Marks)**

- Component Based Development has many characteristics of spiral model. It is evolutionary in nature and includes iterative approach for software development.
- This model composes the applications from pre-packaged software components i.e. from existing software modules.
- Modeling and construction activity begin with identification of candidate components.
- These components can designed as either conventional software modules or object oriented classes or packages. The component is nearly independent and replaceable part of system.
- A package of classes is a collection of objects that work together to achieve some result.

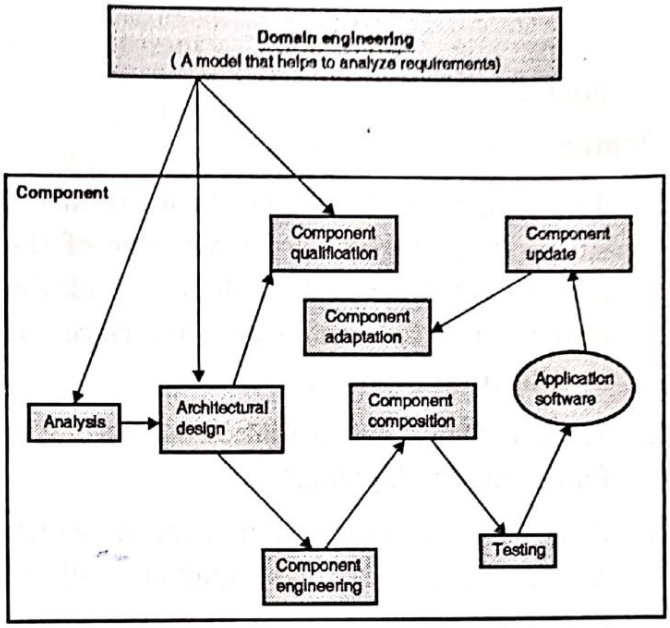

Fig. 2.2.9 : A component-based development model

- Component based development includes following steps :
  1. Available components-based products are researched and evaluated for application domain.
  2. Component integration issues are also considered.
  3. Software architecture is designed to accommodate the components.
  4. Components are integrated i.e. combined into architecture.
  5. Testing is carried out to check the functionality.
- Important feature of Component Based Development is that, it leads to 'software reuse'. Reusable components means the components that are developed for specific application can be reused in development of other application projects also. Reusability provides many benefits for software development.

Merits / Advantages

1. The component based development model supports reusability.
2. Due to reusability, there is reduction in development cycle time.

3. Ultimately the cost of overall development reduces substantially.
4. There is significant increase in productivity.

Demerits / Disadvantages

1. The component based development model suffers from several problems. One of the serious problems is late discovery of the errors due to component interface or architectural mismatches.
2. The problem in encapsulation occurs due to functional overlapping.
3. It is very difficult to find that in which component a particular function will be implemented.
4. It is very difficult to achieve a complete separation of process from component.
5 This approach can not be fully utilized if a development organization does not adopt the principles of component based software engineering.

2.2.4.2 The Formal Methods Model

- The formal methods model includes set of activities that leads to formal mathematical specification of computer software.
- Using formal methods, a software engineer can specify, develop and verify computer based system by applying mathematical notations.

Following are some examples of formal methods model :

A Symbol table

- A program is used to maintain a symbol table. Such tables can be used in many different types of applications. For example, a table that includes a collection of items without any duplication.
- A typical example is names of users of a system without duplicate entries. If such table is examined during execution of the system any time, it will never show duplicate names for use 'Raina'.

1. Virat
2. Dhoni
3. Shikhar
4. Yuvraj
5. Raina

(An example symbol table of users used in formal methods model)

Merits / Advantages

1. When formal methods are used, many problems get eliminated those are difficult to overcome using other software engineering paradigms. For example Ambiguity, incompleteness and inconsistency can be discovered and corrected more easily using mathematical analysis.
2. The formal methods used in the design process provide the basis for program verification. They also enable the software developer to uncover the undetected errors and then fix them.
3. Using this method it's possible to develop accurate and defect free software.

Demerits / Disadvantages

1. This model is just time consuming and more expensive with compared to other methods.
2. As formal method models requires that software developers must have knowledge about formal methods. Otherwise training about formal methods must be given to developers.
3. It is difficult to use the model as a communication mechanism for technically unsophisticated customers.

2.2.4.3 Aspect-Oriented Software Development

- Irrespective of the software process that is chosen, the builder of complex software implements a set of localized features, functions and information content.

- These localized software characteristics are models as components (e.g. Object Oriented Classes) and then constructed within the context of system architecture.
- As modern computer based systems become more sophisticated, certain "concerns" (i.e. customer required properties of technical interest-span the entire architecture). Some concerns are high-level properties of the system (e.g. security). Other concern affects functions (e.g. application of business rules), while others are systematic (e.g. task synchronization or memory management).
- In most of the large systems the relationship between different program components and their requirements are complex. One single requirement can be implemented by various components and vice versa that means one single component may be involved in various requirements.
- To address this particular problem and make programs easier and maintain good reusability, Aspect Oriented Software Engineering (AOSE) is used.
- AOSE implements system functionality at various places in the program. An important characteristic of AOSE is that it includes a definition of an aspect for its use in a proper place in a program.
- The important benefit of this approach is that it supports the separation of concerns into independent elements. For example, user authentication can be represented as an aspect that requests a login name and a password. This login name and password can be used in a program wherever authentication is required.
- The aspect oriented process is likely to adopt characteristics of spiral models and concurrent process models. The spiral model's evolutionary nature is best the aspects since in these models, aspects are identified and after that they are constructed.
- The concurrent development model has parallel nature and it is necessary, because aspects are developed independently by using local software components. Still aspects have a direct impact on these software components.

2.2.4.4 Need of Process Models

Workflow is defined as the flow of process elements and the manner in which they are interrelated. All the generic framework activities are defined earlier, but each of the prescriptive models put different emphasis to these generic framework activities and gives different workflow. This is the reason why different process models are required in software development. Each model has its own importance and significance.

Syllabus Topic : The Unified Process

2.3 The Unified Process

SPPU - Dec 16

University Question

Q. Explain in detail the Unified process indicating workflows and process phases. What are the advantages of iterative development ? **(Dec. 2016, 5 Marks)**

- The object oriented methods and programming languages are widely used in software development. Hence Object Oriented Analysis (OOA) and Object Oriented Design (OOD) methods are proposed for Object Oriented software development.
- Object Oriented Process Model similar to that of evolutionary models are used as new paradigm for software development using object oriented Languages.
- During 1990 Rumbaugh, Booch and Jacobson combined, presented a Unified Modeling Language (UML) that includes

notations for modeling and development of OO systems. By 1997 UML became industry standard for Object Oriented software development and Rational corporation developed automated tools to support UML methods (i.e. Rational Rose software).

- UML provides the necessary technology to support object oriented software engineering practice, but doesn't provide the process framework to guide project teams in their application of the technology. Hence, Rumbaugh, Booch and Jacobson developed the Unified Process, a framework for object oriented software engineering using UML.

2.3.1 The Phases of Unified Process

SPPU – Dec. 12

University Question

Q. Explain the phases of unified process model. **(Dec. 2012, 6 Marks)**

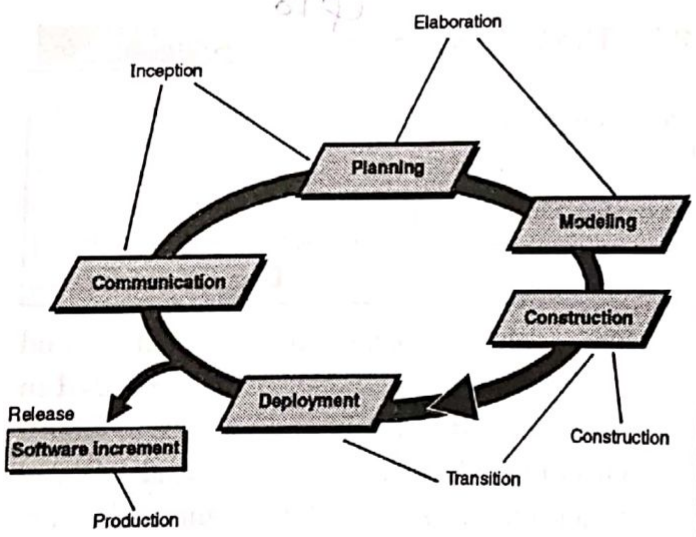

Fig. 2.3.1 : The unified process

The unified process includes following phases :

1. Inception phase

- It includes customer communication and planning activities.
- Here with customer communication requirements are identifies and architecture for the system is planned.
- To understand the requirements better, the Use case diagram is developed.
- The Use case diagram shows the relationship between actors (e.g. person, machine, another system, etc.) and use cases (sequence of actions i.e. procedures /functions). Fig. 2.3.2 shows a use case diagram for sales management system.

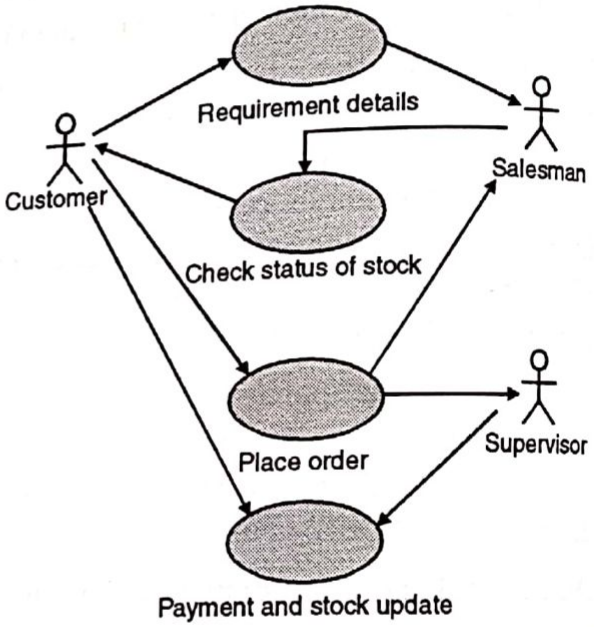

Fig. 2.3.2 : Use Case diagram for Sales Management System

2. Elaboration phase

It includes customer communication and modeling activities of generic process model. Elaboration refines and expands the use cases developed in inception phase and expands architectural representation to include five different views of software.

- Use case model
- Analysis model
- Design model
- Implementation model
- Deployment model

3. Construction phase

- This phase is equivalent to construction activity for generic software process.
- Using the architectural model as input the construction phase develops software components that will make use of each use case diagram.
- To acquire this, analysis and design models that were started during elaboration phase are completed to reflect the final version of software increment.

4. Transition phase

- During this step the software is given to end user for beta testing.
- Now user can report the defects and necessary changes.
- Developing team also creates all necessary information regarding developed software.
- For example user manuals, troubleshooting guides (Guidelines if some trouble occur during handling of the software), and installation procedures. As a conclusion of this phase software is now ready to release.

5. Production phase

It is equivalent to deployment activity of generic process.

During this phase

- Ongoing use of software is monitored.
- Support for operating environment is provided.
- Defect reports and requests for changes are submitted and evaluated.
- During the time of construction, transition and production phases, preparation of next increment is started.

2.3.2 The Iteration among Four Phases

- A phase is said to be the period of time between two processes. As shown in Fig. 2.3.1, a software development life cycle consists of mainly four phases :
  - Inception
  - Elaboration
  - Construction and
  - Transition
- Between all these four phases, one important element is present i.e. iteration. Iteration actually separates all these phases. Iteration consists of different set of activities in the project plan and the criteria for evaluation necessary for future releases.
- This software development life cycle is revolves around all these four phases and software development activities can be divided into these four phases. This development process is highly dependent on continuous stream of releases. This is the reason why all the four phases of the unified process supports incremental and iterative development.

□□□

Unified
→ Inception → communicate for requirement
→ Elaboration → Elaborate Inception
→ construction → Construct what is ellaborated.
→ transition → feedback
→ Production → ~~Sub~~ Delivery.