➜ **7. Artificial intelligence software**

- In these applications non-numerical algorithms are generally used for the purpose of solving complex problems which cannot be handled by computation or straightforward analysis.
- There are various applications in this category such as robotics, games, expert systems, artificial neural networks, proving theorem, pattern matching like image and voice.

1.3 Software Engineering

Q. 1.3.1 Explain the term 'Software Engineering'. (Refer section 1.3) **(5 Marks)**

Software Engineering combines two concepts, Software and Engineering.

☞ Software

- As we have already seen Software is set of executed programs related to specific functionality. Set of executable instructions is called as a program.
- Software is more than just a program code; it also includes data structures which allow programs to manipulate information, and documentation related to software product which defines the functionality and guidance for the user to use this software. Software engineers design and build the software product.

☞ Engineering

Engineering is an application of knowledge and well defined principles to develop products.

☞ Software Engineering

☞ Definition 1

Software Engineering *is the method of applying scientific and technological knowledge, procedures, and rules to design, develop, and maintain the software product.*

OR

☞ Definition 2

Applying technological, scientific, and administrative approach to designing, developing, testing and maintaining the software product in order to meet customers' requirements with best quality of product referred as ***software engineering****.*

1.3.1 Layered Approach

➜ **(MU - May 17)**

Q. 1.3.2 Explain Software Process Framework. (Refer section 1.3.1) **May 17. 5 Marks**

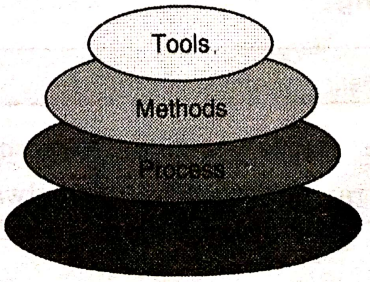

Fig. 1.3.1 : Layered Approach of software Technology

This approach is divided into 4 layers :

1. **A quality Process**
   - Any engineering approach must rest on the quality.
   - The most important aspect in software Engineering is Quality Focus.
2. **Process**
   - Foundation for SE is the Process Layer.
   - SE process is the GLUE that holds all the technology layers together and enables the timely development of computer software.
   - It forms the base for management control of software project.
3. **Methods**
   - SE methods provide the "Technical Questions" for building Software.
   - Methods contain a broad array of tasks that include communication requirement analysis, design modeling, program construction testing and support.
4. **Tools**
   - SE tools provide automated or semi-automated support for the "Process" and the "Methods".
   - Tools are integrated so that information created by one tool can be used by another

1.3.2 Activities of Software Engineering

Q. 1.3.3 Enlist Activities of Software Engineering. (Refer section 1.3.2) (2 Marks)

Software Engineering includes many activities as follows :

- Understanding the user requirements is the first most important activity in software engineering.
- According to the requirements, the process of designing the system and taking decisions is implemented, which further leads to successful completion of development of product.
- Constructing the software product based upon designs and decisions made earlier is the next activity in software engineering.
- To verify the performance and quality of software, testing the software product is essential after it is constructed by software engineers.
- The next step is to provide maintenance for software product which is deployed to the customer.

1.3.3 Need of Software Engineering

Q. 1.3.4 Explain need of Software Engineering. (Refer section 1.3.3) (2 Marks)

- As the technology changes, the user requirements and environment on which software is working also changes. So every organization is ranked based on the software engineering principles used by that organization.

- Implementing and managing large size of software, programmer requires a specific method to modularize the tasks so that size of software can't harm the software quality. Software engineering provides methodology for implementing complex software systems with high quality.
- Without any standard method or management, it is difficult to address defects in the product and correct them as early as possible. Software Engineering provides this functionality.
- Extending the previous software to add new functionality requires more cost in terms of time to develop and efforts taken by people, as compare to the process of developing new software to provide that functionality. Software engineering provides a way in which software system can be able to scale as needed in future.

Syllabus Topic : Generic View of Process

1.4 Generic View of Process

Q. 1.4.1 Explain in brief Software Process Framework. (Refer section 1.4) **(4 Marks)**

- A software engineering process is the model selected for managing the creation of software from customer initiation to the release of the finished product. The selected process typically involves methods such as
- The software life cycle follows the sequential flow in order to develop software.
- We will see each and every stage of this life cycle in detail in subsequent points :

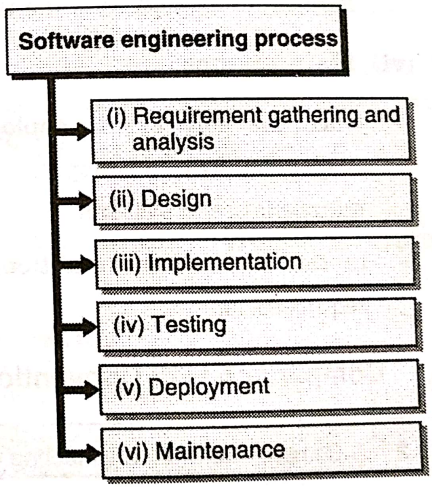

Fig. 1.4.1 : Software engineering process

➜ **(i) Requirement gathering and analysis**

- In this phase stakeholders communicate with customer and system users to gather the business requirements like who will use the system? How user will interact with the system? What should be the system input and output? etc.
- Depending on these requirements, Requirement Specification Document (SRS) is prepared.

➜ **(ii) Design**

- This phase makes use of output of requirement gathering phase i.e. requirement specification document as an input.
- Based on requirement specifications software design is prepared.
- Output of this phase is design specification which specifies hardware and software requirements of system.
- Data Flow Diagram, flowcharts etc. are included by design specification document.
- Design specification acts as an input for implementation phase.

➜ **(iii) Implementation phase**

- Implementation phase of development process decomposes the system work into various smaller parts called **modules.**
- These modules are assigned to development team members and actual coding is started.
- This is longest phase of system development.
- Output of this phase is actual code using specific programming language.

➜ **(iv) Testing**

- Testing phase involves testing of actual code of system against requirements of user in order to ensure that system will satisfy all needs of users.
- Various testing strategies used are unit testing, system testing, integration testing etc.
- Output of testing phase is corrected and modified software code.

➜ **(v) Deployment**

- In this phase the system is deployed at users' site for their use.
- In this phase customer uses the software and give feedback to development team for any changes or modifications in system if any.

➜ **(vi) Maintenance**

- Once the software is deployed actual problems from user's perspective will come up. It is responsibility of development team to solve user's problems time to time. This process is called as maintenance of system.
- In this phase some additional functionality may need to add in the application as per user's requirement.

1.4.1 Comparison with Conventional Engineering Process

Q. 1.4.2 Compare Software engineering with Conventional Engineering Process. (Refer section 1.4.1) **(5 Marks)**

- Software engineering has some similar things as the principles of conventional engineering.
- It is assumed that software engineering is associated to the design of software or data processing products and belongs to its problem solving domain, including the class of problems related to software and data processing.
- This idea is expanded by drawing similarity from the methods that are generally used in engineering.
- It is observed that, just as the famous scientific method is used in the field of scientific research, the steps of engineering design process are used in the process of problem solving in the field of engineering.
- Following steps are mostly repetitive :

1. Problem formulation,
2. Problem analysis,
3. Search for alternatives,
4. Decision,
5. Specification,
6. Implementation

- It is advised that these steps are applicable to the field of software engineering as well.
- Software engineering is considered as a result of integration of hardware and systems engineering, including a set of 3 key elements - methods, tools and procedures which facilitate the manager to control the process of software development.
- The methods give the technical "how to's " for building software; tools offer automated or semi-automated support for methods; and procedures describe the series of applying methods, the deliverables, the controls, and the objectives.
- Compared to traditional engineering disciplines, software engineering demonstrates some remarkable differences.
- In conventional engineering, one moves from an abstract design to a concrete product. On other hand, in software engineering, one moves from design to coding.

Software Engineering	Abstract Design	→	More Abstract Code
Manufacturing Engineering	Abstract Design	→	Concrete Products

- The problem domains of software engineering can be anything, from word processing to real-time control and from games to robotics.
- In contrast to another engineering discipline, it is therefore much larger in scope and thus provides bigger challenges.
- Traditional manufacturing engineering that normally emphasize mass production is loaded with production features. Thus, it is highly production concentrated. Software engineering, in contrast, is inherently design concentrated.
- Product standardization assists in cost reduction in manufacturing, whereas such a possibility is remote in software engineering.
- The possibility of process standardization, however, is very high in the latter.
- Conventional engineering process makes the use of unlimited number of domain and application-specific ideas which are proved. Software engineering, in contrast, makes use of limited, but global concepts such as loops, conditions etc.

Software maintenance reapplies all of the previous life cycle steps to an existing program instead of a new one in order to correct existing or insert new functionality.

Syllabus Topic : Prescriptive Models

1.7 Prescriptive Models

➔ (MU - Dec. 17)

Q. 1.7.1 What is Prescriptive Model ? (Refer section 1.7) **Dec. 17, 5 Marks**

- The following framework activities are performed irrespective of the process model selected by the organization.

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

- The name **'prescriptive'** is given because the model prescribes a set of activities, actions, tasks, quality assurance and change the mechanism for every project.

Syllabus Topic : Waterfall Model

1.7.1 Waterfall Model

➔ (MU - May 16, Dec. 17)

Q. 1.7.2 Explain waterfall model give its advantage and disadvantages. (Refer section 1.7.1) **May 16, Dec. 17, 10 Marks**

- Waterfall model is the first approach used in software development process.
- It is also called as classical life cycle model or linear sequential model.
- In waterfall model any phase of development process begins only if previous phase is completed.

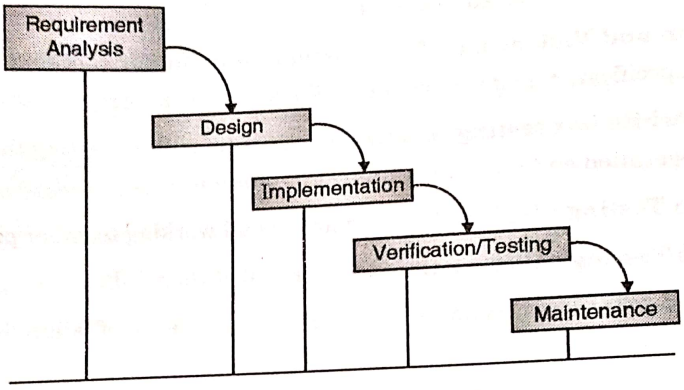

Fig. 1.7.1 : Waterfall Model

1. **Requirement Analysis**
   - In this phase all business requirements of system are gathered and analyzed by communication between stakeholders and customer or user.
   - At the end of this phase Requirement Specification Document (SRS) is created.
2. **Design**
   - Based on requirement, specification document design of system is created called software architecture.
   - It is blue print of system representing system's internal structure and behavior.
3. **Implementation**
   - In implementation phase actual coding is constructed for software architecture using hardware and software requirements of system.
   - It is responsibility of developer.
4. **Verification / Testing**
   - Here coding or job done by developer is verified against requirements of user in order to ensure that software will satisfy all business requirements of user.
   - After the successful verification software is deployed at user's site for their use.
5. **Maintenance**
   - While using software if user faces some problems, then those problems must be solved time to time by development team. This task comes under maintenance of software.
   - Maintenance also includes adding new functionalities in software as per user requirements.

Advantages of waterfall model

(i) It is very simple to understand and easy to use.

(ii) Phases of waterfall model do not overlap with each other.

(iii) It is useful for small projects in which requirements are clear initially.

(iv) Since development is linear it is easy to manage development process.

Disadvantages of waterfall model

(i) It is not useful for large projects.

(ii) Not suitable for projects in which requirements are not clear initially.

(iii) System or product is available only at the end of development process.

(iv) It is very difficult to modify system requirements in the middle of development process.

Practical situations in which Waterfall model can be used

- This model is used only when the requirements are very well known, clear and fixed.
- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements.
- Ample resources with required expertise are available freely.
- The project is short.

1.8 Incremental Process Models

→ **(MU - Dec. 17)**

Q. 1.8.1 Describe incremental model. (Refer section 1.8) **Dec. 17, 5 Marks**

- The incremental model applies the waterfall model incrementally.
- The series of releases is referred to as "increments", with each increment providing more functionality to the customers.
- After the first increment, a core product is delivered, which can already be used by the customer.
- Based on customer feedback, a plan is developed for the next increments, and modifications are made accordingly.
- This process continues with increments being delivered until the complete product is delivered. The incremental philosophy is also used in the agile process model.

Following tasks are common to all the models

1. **Communication :** Helps to understand the objective.
2. **Planning :** Required as many people (software teams) work on the same project but different functions at same time.

3. **Modelling :** Involves business modelling, data modelling, and process modelling.
4. **Construction :** This involves the reuse of software components and automatic code.
5. **Deployment :** Integration of all the increments.

☞ Characteristics of Incremental Model

1. System is broken down into many mini development projects.
2. Partial systems are built to produce the final system.
3. First tackled highest priority requirements.
4. The requirement of a portion is frozen once the incremented portion is developed.

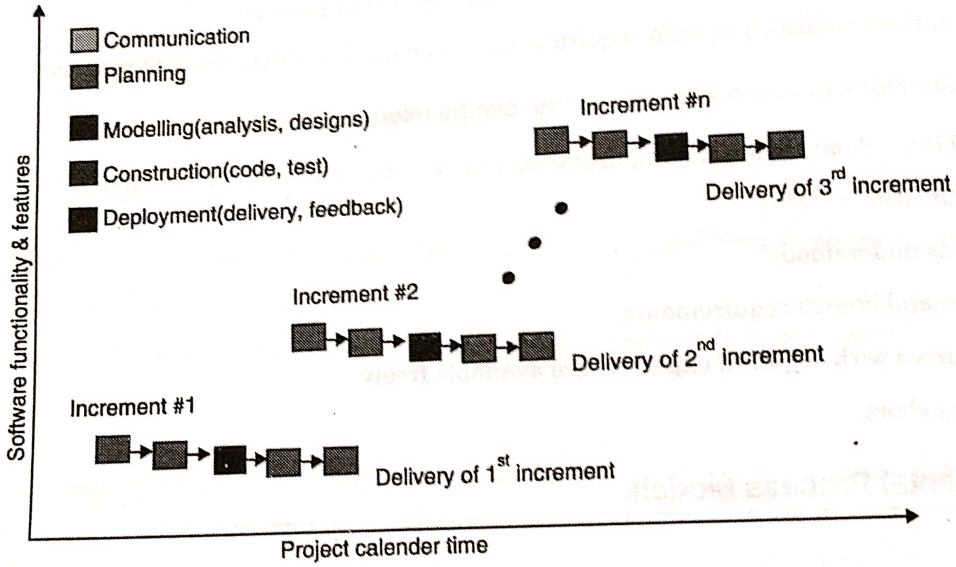

Fig. 1.8.1 : Incremental Model

☞ Advantages of Incremental Model

1. After each iteration, regression testing should be conducted. During this testing, faulty elements of the software can be quickly identified because few changes are made within any single iteration.
2. It is generally easier to test and debug than other methods of software development because relatively smaller changes are made during each iteration. This allows for more targeted and rigorous testing of each element within the overall product.
3. Customer can respond to features and review the product for any needed or useful changes.
4. Initial product delivery is faster and costs less.

☞ Disadvantages of Incremental Model

1. Resulting cost may exceed the cost of the organization.
2. As additional functionality is added to the product, problems may arise related to system architecture which were not evident in earlier prototypes.

1.9 Evolutionary Process Models

➔ (MU - May 18)

Q. 1.9.1 What is evolutionary model? List evolutionary models. (Refer section 1.9) **May 18, 5 Marks**

- Normally the Evolutionary Development Model is based on the principle that "stages consist of growing increments of an operational software product, with the way of evolution being discovered by operational experience".
- According to the business need and the changing nature of the market there are lot of improvements required in the software product over a time.

- Due to this lot of improvement is required in the product hence evolutionary developments model is iterative in nature.
- There are two types of models in Evolutionary process. They are :
  1. Spiral model 2. Prototyping model

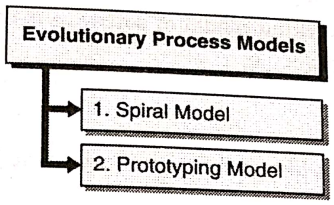

Fig. 1.9.1 : Evolutionary Process Models

Syllabus Topic : Spiral Model

1.9.1 Spiral Model

➔ **(MU - May 15, Dec. 15, May 18)**

Q. 1.9.2	Explain in detail Spiral model. (Refer section 1.9.1)	May 15, Dec. 15, 5 Marks
Q. 1.9.3	Explain any evolutionary model. (Refer section 1.9.1)	May 18, 5 Marks

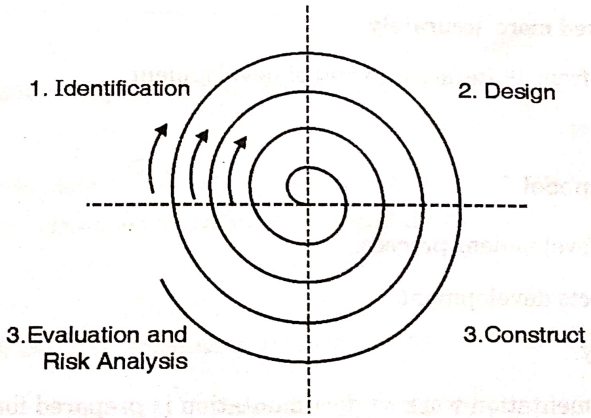

Fig. 1.9.2 : Spiral Model

- Spiral model is a combination of iterative model and waterfall model.
- Spiral model has four phases of development each of these phases is called as spiral.

(i) Identification

- This phase identifies all business requirements of system at the beginning. It involves clear understanding of requirements by communication between stakeholders and customer.
- In the subsequent iterations all subsystem requirements and unit requirements are identified.

(ii) Design

- In first iteration, design phase develops conceptual design of system based on initially gathered requirements.

- In further spirals or iterations, it develops logical design, physical design, architectural design and final design of system.

(iii) Construct

- Initially construct phase develops a code for conceptual design to get user feedback. In next subsequent spirals, detailed working model of software is constructed with increment number and are delivered to customer for feedback.

(iv) Evaluation and risk analysis

- In this phase management risks like cost overrun are identified and monitored, technical feasibility of system is also done.
- At end of each spiral customer evaluates software for potential risks in system and provides feedback.
- Spiral model can be used for projects where budget and risk evaluation are important factors.
- If customers are not sure about their requirements then spiral model is useful than waterfall model.

☞ **Advantages of spiral model**

(i) It is more flexible to changing requirements.

(ii) Requirements are achieved more accurately.

(iii) User can see the system from 1st iteration to end of development.

(iv) Risk management is easier.

☞ **Disadvantages of spiral model**

(i) It is difficult to manage development process.

(ii) Not useful for small projects development.

(iii) Spiral can run indefinitely.

(iv) It requires excessive documentation work as documentation is prepared for each iteration.

Syllabus Topic : Prototyping Model

1.9.2 Prototyping Model

Q. 1.9.4 Describe Prototype model. (Refer section 1.9.2) (5 Marks)

- Prototyping model refers to developing software application prototypes (early approximation / version) which displays the behaviour of product under development but may not actually contain the exact logic of the original application.
- Prototyping allows user to evaluate developers' proposal and try it before actual implementation.
- Prototyping model is widely used popular software development model as it helps to understand user requirements in early stage of development process.

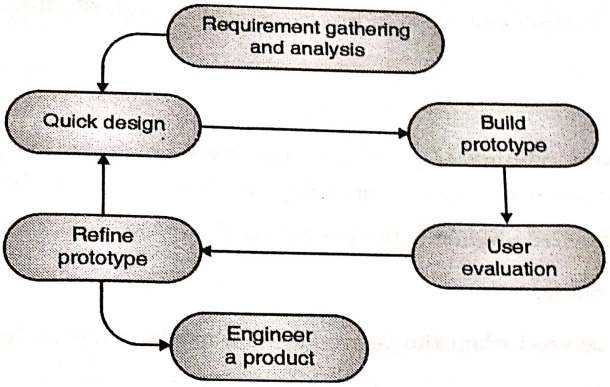

Fig. 1.9.3 : Prototyping Model

Phases of prototyping model are as follow :

(i) Requirement gathering and analysis

- Building of prototype begins with requirement gathering and analysis.
- In this phase various users of system are interviewed in order to know their system requirements or expectations from system.
- Requirement specification report is generated as an output of this phase.

(ii) Quick Design

- After gathering and analysis of user requirements quick design (prototype design) of system is developed.
- Quick design is not detailed design of system; it contains only necessary characteristics of the system which gives basis idea of system to the end users.

(iii) Build prototype

- Based on feedback received from user about quick design of system, the first prototype of system is created.
- Prototype is working model of system under development.

(iv) Use Evaluation

- Once prototype is built, proposed system is presented to end user of system for its evaluation.
- User determines strengths and weaknesses of prototype i.e. what needs to be added or what is to be removed.
- All the comments and suggestions given by user in feedback are passed to developer.

(v) Refining prototype

- Depending on comments and suggestions came from user, developers refine previous prototype to form new prototype of system.
- New prototype is again evaluated just like previous prototype.

- The process of evaluation and refining prototype continues until all user requirements are met by prototype.

(vi) Engineer a product

- Once evaluation and refining of prototype completes i.e. when user accepts final prototype of system the final system is evaluated thoroughly and deployed at user's site.
- Deployment of engineered product is further followed by regular maintenance of system.

When to use Prototype model

- Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
- Typically online systems in which web interfaces have a very high amount of interaction with end users, are best suited for Prototype model.
- It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
- Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system.
- They are excellent for designing good human computer interface systems.

Advantages of prototyping model

(i) It enables early evaluation of system by providing working model to end users at early stage of development.

(ii) Refining of prototype results in better implementation of system requirements.

(iii) Communication between developer and user reduces ambiguity.

(iv) More involvement of user in development process results in meeting user's requirement at greater extent.

Disadvantages of prototyping model

Q. 1.9.5 What are the potential problems of Prototyping Model ? (Refer section 1.9.2) (5 Marks)

(i) Refining of prototype continues until user is completely satisfied, thus it is time consuming and expensive process.

(ii) More involvement of user in development process is not accepted by developer always.

(iii) Refining of prototype again and again may disturb the working of development team.

(iv) Practically prototyping model results in increasing the complexity of system as scope of system extends beyond original plan.

1.11 Specialized Models : Component Based, Aspect Oriented Development

Special process models take on many of the characteristics of one or more of the conventional models. *However, specialized models* tend to be applied when a narrowly defined software engineering approach is chosen.

1.11.1 Component - Based Development

→ (MU - May 16, Dec. 17)

Q. 1.11.1	Write note on component based development. (Refer section 1.11.1)	May 16, 10 Marks
Q. 1.11.2	Explain component based development. (Refer section 1.11.1)	Dec. 17, 5 Marks

- Commercial off-the-shelf (COST) 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. It is evolutionary in nature, demanding an iterative approach to the creation of software.
- However, the model composes applications from prepackaged software components. Modeling and construction activities begin with the identification of candidate components.
- These components can be designed as either conventional software modules or object-oriented classes or packages of classes.
- Regardless of the technology that is used to create the components, the component-based development model incorporates following steps (implemented using an evolutionary approach) :
  - 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. Based on studies of reusability, component-based development can lead to reduction in development cycle time, reduction in project cost and increase in productivity. Although these results are a function of the robustness of the component library, there is little question that whether the component-based development model provides significant advantages for software engineers.

1.11.2 The Formal Methods Model

→ (MU - Dec. 17)

Q. 1.11.3	Explain Formal Methods Model. (Refer section 1.11.2)	Dec. 17, 5 Marks

- The formal methods model encompasses a set of activities that leads to formal mathematical specification of computer software.
- Formal methods enable a software engineer to specify, develop, and verify a computer-based system by applying rigorous, mathematical notation.
- A variation on this approach is called clean-room software engineering.

- When formal methods are used during development, they provide a mechanism for eliminating many of the problems that are difficult to overcome using other software engineering paradigms.
- Ambiguity, incompleteness, and inconsistency can be discovered and corrected more easily, not through ad hoc review, but through the application of mathematical analysis. When formal methods are used during design, they serve as a basis for program verification and therefore enable the software engineer to discover and correct errors that might otherwise go undetected.
- 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 expressed :
  - The development of formal models is currently quite time-consuming and expensive.
  - Because only few software developers have the necessary background to apply formal methods, extensive training is required.
  - It is difficult to use the model as a communication mechanism for technically unsophisticated customers.

1.11.3 Aspect-Oriented Development

➔ (MU - May 15, May 17)

Q. 1.11.4	Write note on - Aspect Oriented Model. (Refer section 1.11.3)	May 15, 10 Marks
Q. 1.11.5	Explain Aspect Oriented Development. (Refer section 1.11.3)	May 17, 5 Marks

- 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 a 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.
- A distinct aspect-oriented process has not yet matured. However, it is likely that such a process will adopt characteristics of both the spiral and concurrent process models.
- The evolutionary nature of the spiral is appropriate as aspects are identified and then constructed. The parallel nature of concurrent development is essential because aspects are engineered independently of localized software components and yet aspects have a direct impact on these components.

1.11.4 The Unified Process

- In some ways the unified process (UP) is an attempt to draw on the best features and characteristics of conventional software process models, but characterize them in a way that implements many of the best principles of agile software development.
- The unified process recognizes the importance of customer communication and streamlined methods for describing the customer's view of a system.
- It emphasizes the important role of software architecture and helps the architect focus on the right goals, such as understandability, reliance to future changes, and reuse.
- It suggests a process flow that is iterative and incremental, providing the evolutionary feel that is essential in modern software development.

Syllabus Topic : Agile Methodology