Module 1

Introduction to Software Engineering

SOFTWARE ENGINEERING

Ques 1) What is software engineering?

Or

Define the term "Software Engineering".

Ans: Software Engineering

The goal of software engineering discipline is to ensure the development of fault free software which fulfil the user's requirements and delivered within the time and budget. In 1968, the term Software Engineering is first appeared. There is still much debate on the fact that whether it can be following the traditional definition of engineering or not. Software engineering is a new field compared to other engineering disciplines and there is still a question arises that what really it is.

Software Engineering is a process or discipline whose objective is to develop and maintain the software more systematic and modified on time and within budget.

According to IEEE, "Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software, i.e., the application of engineering to software".

According to Boehm, "Software Engineering is the practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate and maintain them".

According to Sommerville, "Software Engineering is concerned with the theories, methods and tools that are needed to develop the software products in a cost effective way".

Ques 2) What are the main activities of software engineering?

Or

Discuss the phases of software engineering

Or

Discuss the Software Engineering Process?

Ans: Activities of Software Engineering/Software Engineering Process

Regardless of the application area, project size or complexity, the work that is associated with software engineering can be categorized into three generic phases:

1) **Definition Phase:** The main focus is on 'What':
   i) Identify what information is to be processed.
   ii) What function and performance is desired.
   iii) What system behavior is expected?
   iv) What interfaces are to be established.
   v) What design constraints exist?
   vi) Identify the key requirements of system and software.

   Major tasks include information engineering, project planning and requirement analysis.

2) **Development Phase:** The main focus is on 'How':
   i) Define how data are to be structured.
   ii) How a function is to be implemented.
   iii) How are procedural details to be implemented?
   iv) How interfaces are to be characterized.
   v) How will design be transformed into a programming language?
   vi) How testing will be performed.

   Major tasks include software design, code generation and software testing.

3) **Maintenance Phase:** The main focus is on 'change'. It reapplies the definition and the development phases in context of existing software
   i) Error correction.
   ii) Adaptations required as the software's environment evolved.
   iii) Changes due to the enhancements by user are changing needs.

   Four types of changes occur in maintenance phase, which are correction, adaptation, enhancement and prevention.

Ques 3) Write down the objectives of Software Engineering?

Ans: Objectives of Software Engineering

Software engineering objectives are:

1) To understand user conceptual models and development of better specification.
2) To improvement in design language and reusable code.
3) To participatory design and interactive debugging.
4) For Specification of interface and mock-up to confirm specification.
5) To satisfy the user's requirements.
6) To achieve low maintenance and production cost.
7) To provide the software within budget and time (user specified budget and time in SRS).
8) To achieve the high performance.

Ques 4) What is the scope of software engineering?

Or

Discuss software engineering as historical, economic, maintenance, specification & design and team programming aspects.

Ans: Scope of Software Engineering

The scope of software engineering is extremely broad. In general, five aspects are involved:

1) **Historical Aspects:** Software engineering cannot be considered as engineered since an unacceptably large proportion of software products still are being:
   i) Delivered late
   ii) Over budget
   iii) With residual faults.

   Solution: A software engineer has to acquire a broad range of skills, both technical and managerial. These skills have to be applied to:
   i) Programming; and
   ii) Every step of software production, from requirements to post-delivery maintenance.

2) **Economic Aspects:** Applying economic principles to software engineering requires the client to choose techniques that **reduce long-term costs** in terms of the economic sense. Economic aspect includes:
   i) The cost of introducing new technology into an organisation:
      a) Training cost
      b) A steep learning curve
      c) Unable to do productive work when attending the class.
   ii) The maintenance consequence

3) **Maintenance Aspects:** Software engineering is the application of engineering to software. The life cycle paradigm for software includes: requirements, design, construction, testing, and maintenance.

   Software maintenance is an integral part of a software life cycle. However, it has not historically received the same degree of attention as the other phases. Historically, development has had a much higher profile than maintenance in most organisations. This is now changing as organizations strive to obtain the most out of their development investment by keeping software operating as long as possible.

   The following are some aspects of maintenance:
   i) **Classical View of Maintenance:** In the 1970s, software production was viewed as consisting of two distinct activities performed sequentially: development followed by maintenance. The software product was developed and then installed on the client's computer. Any change to the software after installation on the client's computer and acceptance by the client, whether to fix a residual fault or extend the functionality, constituted classical maintenance. Hence, the way that software was developed classically can be described as the **development-then-maintenance model**.

      Two main **reasons** why the development-then-maintenance model is unrealistic are as follows:
      a) During the development, the client's requirements may change. This leads to the changes in the specification and design.
      b) Developers try to reuse parts of existing software products in the software product to be constructed.

   ii) **Modern View of Maintenance:** It is the process that occurs when "software undergoes modifications to code and associated documentation due to a problem or the need for improvement or adaptation".

      That is, maintenance occurs whenever a fault is fixed or the requirements change, irrespective of whether this takes place before or after installation of the product.

   iii) **Classical Post-Delivery Maintenance:** All changes to the product once the product has been delivered and installed on the client's computer and passes its acceptance test.

   iv) **Modern Maintenance (or Just Maintenance):** Corrective, perfective, or adaptive activities performed at any time. Classical post-delivery maintenance is a subset of modern maintenance.

   A software product is a model of the real world, and the real world is perpetually changing. As a consequence, software has to be maintained constantly for it to remain an accurate reflection of the real world. A major aspect of software engineering consists of techniques, tools, and practices that lead to a reduction in post-delivery maintenance cost.

4) **Specification and Design Aspects:** Software professionals are human and therefore sometimes make a mistake while developing a product.

   The earlier we correct a fault, the better. That is, the cost of correcting a fault increases steeply since it is directly related to what has to be done to correct a fault.

   If the mistake is made while eliciting the requirements, the resulting fault will probably also appear in the specifications, the design, and the code. Edit the code, recompile and relink the code, and test.

   It is crucial to check that making the change has not created a new problem elsewhere in the product. All the relevant documentation, including manuals, needs to be updated. The corrected product must be delivered and reinstalled.

5) **Team Programming Aspects:** Team development leads to interface problems among code components and communication problems among team members.

Unless the team is properly organized, an inordinate amount of time can be wasted in conferences between team members. Suppose that a product takes a single programmer 1 year to complete. If the same task is assigned to a team of six programmers, the time for completing the task frequently is closer to 1 year than the expected 2 months, and the quality of the resulting code may well be lower than if the entire task had been assigned to one individual.

Because a considerable proportion of today's software is developed and maintained by teams, the scope of software engineering must include techniques for ensuring that teams are properly organised and managed.

It also includes human aspects, such as team organization, economic aspects, and legal aspects, such as copyright law.

These five aspects can be categorised in the fields of Mathematics, Computer Science, Economics, Management, and Psychology.

Ques 5) Discuss the similarities and differences from conventional engineering processes.

Or

Explain the major differences between software engineering and other traditional engineering disciplines.

Ans: Similarity and Differences from Conventional Engineering Processes

Some practitioners believe that they apply concepts of traditional engineering to software design and implementation. They believe this provides a structured, logical approach and subsequently, a stable final product. Other practitioners believe that traditional engineering concepts may not apply, because software is fundamentally different than bridges and roads.

For example, traditional engineers do not use compilers or linkers to build roads.

Software engineers aspire to build low-cost, reliable, safe software, which is much like what traditional engineers do. They borrow many metaphors and techniques from traditional engineering disciplines:

1) Requirements Analysis,
2) Quality Control, and
3) Project Management Techniques.

Traditional engineers now use software tools to design and analyze systems, such as bridges and buildings. These new kinds of design and analysis resemble programming in many respects, because the work exists as electronic documents and goes through analysis, design, implementation, and testing phases, just like software.

Table 1.1: Software Engineering *versus* Traditional Engineering

Basis	Software Engineering	Traditional Engineering
Foundations	Software engineering is based on computer science, information science, and discrete math.	Traditional engineering is based on physics, chemistry, and calculus.
Cost	Compilers and computers are now cheap, so software engineering and consulting often cost more than 50% of a project. Even minor software engineering cost-overruns can cause a project to fail.	Construction and manufacturing costs are high, so traditional engineering may only cost 15% of a project. Even major engineering cost overruns may not affect a project's viability.
Management Status	Few software engineers manage anyone, so they are not viewed as managers, except by themselves.	Many traditional engineers manage construction, manufacturing, or maintenance crews, so they are all treated as managers.
Innovation	Software engineers apply new and untested elements in many software projects.	Though some projects have innovations, traditional engineers apply known and tested principles, and limit the untested innovations that goes into each product.
Replication	Replication is trivial, and most development effort goes into building new (unproven) or changing old designs and features.	Most development effort goes into replicating proven designs.
Number of Practitioners in U.S. in 2000	640,000	1,100,000 total engineers 65,000 computer engineers.

Ques 6) Discuss the layered view of software engineering.

Or

Explain the following statement: "Software Engineering is a layered technology".

Or

Discuss processes, methods and tools in layered technology of software engineering.

Ans: Layered View of Software Engineering

Different practitioners view software engineering differently. Pressman suggested a layered approach to view the software engineering. This approach consists of four layers, *viz.*, quality focus, process model, methods and tools (see **figure 1.1**):

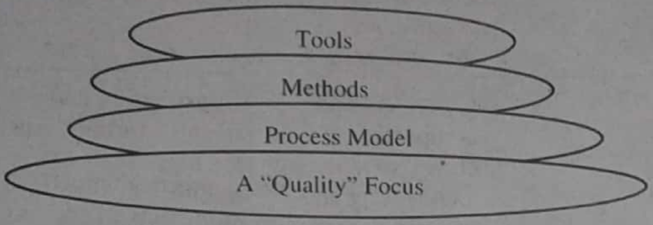


Figure 1.1: Software Engineering Layers

1) **Quality:** The foundation of the layered software engineering technology is quality. This gives rise to concepts such as Total Quality Management (TQM).

2) **Process:** Software engineering process is the glue that holds the technology layers together and enables rational and timely development of computer software.

   The process layer comprises the Key Process Areas (KPAs) which assist the management in technical decision-making. It aids in determining the technical methods which are to be applied, how products are to be produced, maintenance of quality, achievement of milestones and change management.

3) **Methods:** Software engineering methods provide the technical how-to's for building software.Project planning, estimation, system and software requirements analysis, data structure design, program architecture, algorithm procedures, coding, testing and maintenance are the methods of technical planning that help in building the software. They can be referred to as the technical guidelines.

4) **Tools:** Software engineering tools provide automated or semi-automated support for the process and the methods. Software engineering tools, aids the process and methods in automated or semi-automated way. These tools are integrated in a way so that the information created by one can be used by another.

   This system is referred to as CASE which combines software, hardware and Software Engineering database.

Ques 7) Define a software process. What are the characteristics of a software process?

Ans: Software Processes

The process means a particular method of doing something, generally involving a number of steps or operations. In software engineering, software process refers to the method of developing software.

Software process teaches us how we can manage on planning, according to the constraints and boundaries. The process that deals with the technical and management issues of software development is called a **software process.**

In other words, a software process specifies the abstract set of activities that should be performed to transform user needs to the final product. One can view the software process as an abstract type, and each project is done using that process as an instance of this type. In other words, there can be many projects for a process (i.e., many projects can be done using a process), and there can be many products produced in a project. This relationship is shown in **figure 1.2:**

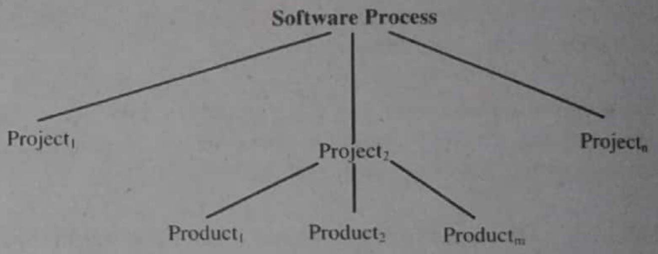


Figure 1.2: Processes, Projects, and Products

The sequence of activities specified by the process is typically at an abstract level because they have to be usable for a wide range of projects.

Characteristics of Software Process

1) **Optimality:** It means that the process should be able to produce high-quality software at low cost, and scalability means that it should also be applicable for large software projects. To achieve these objectives, a process should have some properties.

2) **Predictability:** It of a process determines how accurately the outcome of following a process in a project can be predicted before the project is completed. Predictability can be considered a fundamental property of any process, In fact, if a process is not predictable, it is of limited use.

3) **Maintainability:** One of the important objectives of the development project should be to produce software that is easy to maintain. And the process should be such that it ensures this maintainability.

Ques 8) What is software process model? List out the different types of software process models.

Ans: Software Process Models

The word 'paradigm' has some dictionary meaning, such as – "model", "example" and "pattern".

It can be defined as "an instance that works as a pattern or model".

Alternately, it can be defined as, "a model of something which describes it or demonstrate how it can be produced".

Software engineering sees paradigm as, "a set of associated concepts which are used by an individual to identify the real world or a part of it".

Software engineering includes a developmental approach which comprises the process, methods and tools, layers and is known as the **process model** or **software engineering paradigm**. There are various software processes and models.

A development process is a group of actions, together with as ordering association between actions and its abstract demonstration is known as a process model. The desired product is produced if the ordering association is satisfied by the performance of actions.

The **software process** model identifies that how these activities are involved in the complete software development effort. The selection of the process model is based on the project nature, application, methods, and tools which are used.

It is also based on the controls and deliverables needed in the project.

Raccoon states that software development activities can be divided into four stages:

1) **Status Quo:** This represents the present state of affairs.
2) **Problem Definition:** This refers to the problem identification that is to be solved.
3) **Technical Development:** Using various technologies it provides the solution for the problem.
4) **Solution Integration:** This provides the end result.

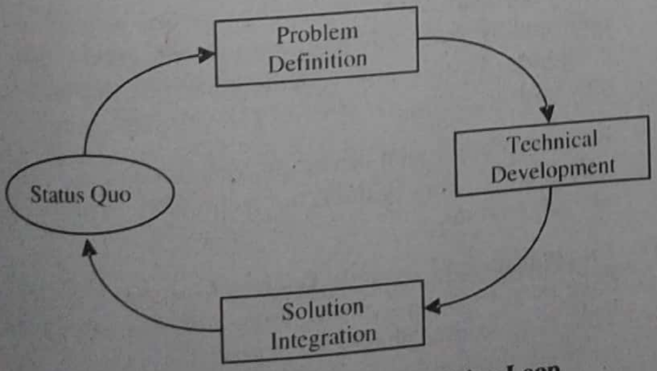
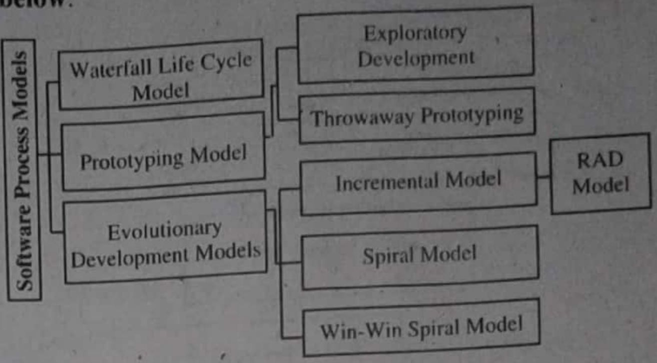


Figure 1.3: Phases of Problem Solving Loop

The problem solving loop is a feature of software engineering work at different levels. Every stage in the loop has a similar problem-solving loop that may have another problem-solving loop, thereby there are many iterations are occurring in a model.

Types of Software Process Models

The various software process models are shown in **figure below**:



Ques 9) What is waterfall model? What are the different phases of waterfall model?

Or

Discuss various activities during SDLC.

Ans: Waterfall Model

The classical waterfall model is not a practical model. It cannot be used to develop the software projects in the real world. It is a theoretical way of developing a software model.

All other lifecycle models are derived from the classical waterfall model. In order to understand the other lifecycle models, it is essential to study the waterfall model as it is well designed and easy to understand.

The waterfall model is also referred as the 'classical life cycle model', 'linear, sequential model' or simply 'waterfall model'. It involves a systematic, sequential approach to software development that starts at the system level and moves forward through analysis, design, coding, testing, and support.

The waterfall model is very simple in understanding. In this model, each phase is completed first and reviewed before moving onto the next phase. This is primarily done to ascertain whether the project is moving in the right direction or not. If the answer is negative, then the project is discarded. Each phase is independent of the other.

Phases of the Waterfall Model/ Activities during SDLC

The classical waterfall model breaks down the lifecycle into the following phases as depicted in **figure 1.4**:

1) **Feasibility Study:** This involves problem analysis and gathering of all data related to the product. Analysis of data is carried out for:
   i) **The requirements of the Customer:** In this, the significant requirements of the customer are taken and left the other details of the requirements.
   ii) **Formulations of the Different Strategies for Solving the Problem:** In this, various methods are identified to solve the problem.
   iii) **Evaluation of Different Solution Strategies:** All the solutions to the problem are studied in detail keeping in mind the benefits and shortcomings, resources required, development cost and time. Each solution is analysed on the basis of these parameters.

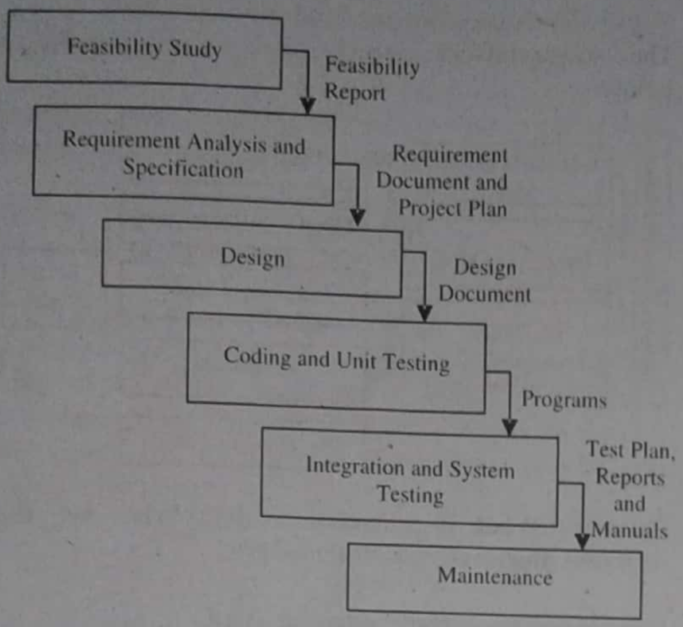

Feasibility Study
Feasibility Report
Requirement Analysis and Specification
Requirement Document and Project Plan
Design
Design Document
Coding and Unit Testing
Programs
Integration and System Testing
Test Plan, Reports and Manuals
Maintenance

Figure 1.4: Phases of the Classical Waterfall

A **feasibility study** is mainly carried out to understand whether developing a product would be financially and technically feasible:

i) Project managers first visit client's side and then try to understand their requirements. After this, they try different input data to come with different output data in the system. By doing this, they study the kind of processing which is required on the data and what are the system constraints that are to be faced.

ii) After getting an idea of the problem, different solutions are investigated. Each solution is then separately analysed on the basis of 'required resources', 'development time' and 'cost'.

iii) Based on this analysis, the best solution is then chosen keeping in mind financial and technical feasibility. This is determined by the customer budget and whether they have adequate technical knowledge to develop the solution.

2) **Requirements Analysis and Specification:** Objective of this phase to recognise the exact requirements of the customer. After requirement analysis the next step is to document the same in detail. So, this phase has two different activities:

i) **Requirements Gathering and Analysis:** Requirement gathering form the customer is done in order to understand the customer needs clearly so as to leave out any ambiguity in the product specifications.

The requirement analysis activity involves in the collection of data regarding the product to be developed. To get the requirements from the user interviews and discussions methods are used.

When a data is collected from the users, then there will be contradictions and ambiguities occur as every user has a partial and incomplete view of the system. These are to be identified and resolved by further customer consultation.

ii) **Requirements Specification:** The requirement specification phase starts once all the ambiguities, inconsistencies have been resolved and then all the customer requirements are documented and this is known as 'Software Requirements Specification (SRS)' document. Functional and non-functional requirements and implementation goals are the main components of SRS.

3) **Design:** In this phase requirements specification mentioned on **SRS** is convert into a structure that can be implemented with some programming language. Technically, software architecture is obtained from the SRS document. Following are the two different design approaches are available:

i) **Traditional Design Approach:** This approach comprises of two different activities. First, a structured analysis of the specifications is carried out to examine the detailed structure of the problem. Structured analysis further breaks down into the functional requirements and the data flow between the functions. Second, a structured design is devised form the structured analysis into a software design wherein two main activities are involved:

a) Architectural Design, and

b) Detailed Design

ii) **Object-Oriented Design Approach:** In this, the different objects that are present in the problem as well as the solution domain are identified along with the relationships among them. This object structure is refined to attain the specific design.

4) **Coding and Unit Testing:** This is also referred to as the implementation phase in software development. In this, the software design is translated into source code.

Every design component is then executed as a program module. All modules are individually tested in isolation to debug the errors. This ensures the correct working of all individual modules.

Every software development organisation develops its own coding standards to come up with good quality programs. Various issues are addressed by this coding standard.

After the completion of coding, each module is tested individually. This is done to ensure smooth working of each module.

5) **Integration and System Testing:** Once each module have been tested and coded, integration is undertaken. This is done in a phased manner as the modules of a software product are never integrated in one instance. At each phase, the partially integrated system is tested for errors and a group of earlier defined modules are added to it. Once it is rectified, the next phase is carried out. When all modules have been integrated, system testing takes place whose aim is to ensure that the system conforms to requirements as defined in the SRS document.

System testing comprises three types of testing:

i) **α-Testing:** The development team carried-out this testing.

ii) **β-Testing:** This testing is carried-out by the group of friendly customers.

iii) **Acceptance Testing:** This is the system testing by the customer after delivery of product to ascertain whether to accept or reject the product.

System testing is planned according to the system test plan document which identifies all testing related activities, testing schedule and resource allocation for testing. Besides, it also includes the test cases and the expected output for each of them.

6) **Maintenance:** Maintenance of software requires a greater effort than development. Many studies have been conducted in the past to ascertain this and indicate that the 'ratio of effort' of development and maintenance stands at 40: 60.

Maintenance involves one or more of the following activities:

i) In the first activities the undetected errors (during the development phase) are corrected. This process is also known as 'corrective maintenance'.

ii) The second activity is to make improvement in the implementation and enhance the functions of the system as per customer requirements. The process of improvement in the implementation is known as Perfective maintenance.

iii) Sometimes porting of software is required and port into a new environment like in a new computer platform or a new operating system. This process is referred as adaptive maintenance.

Ques 10) What are the conditions for using classical waterfall model?

Ans: Conditions for Using Classical Waterfall Model

Necessary conditions for using classical waterfall model are:

1) The requirements are clearly known and fixed.
2) Definition of the product is stable.
3) Technology is well known.
4) Requirements are very clear (unambiguous).
5) Vast amount of resources are available readily with the correct expertise.
6) Time span of project is short.

Ques 11) What are the advantages and disadvantages of waterfall model?

Ans: Advantages of Waterfall Model

Advantages of classical waterfall model are as follows:

1) Easily understandable.
2) Well defined inputs and outputs for each phase.
3) Assists the project manager in efficient project planning.
4) Creates a model into which different methods of analysis, design, code, test and support is tested.

Disadvantages of Waterfall Model

Disadvantages of classical waterfall model are as follows:

1) It is sequential in nature in that one can move forward and not backward in sequence.
2) There is no overlapping and interaction between phases.
3) The project team has very little interaction with the users.
4) Delivery of system is not done in pieces since the model does not allow that.
5) As new projects have vague specifications, it is not suitable for new projects.

Ques 12) Explain the prototyping model with steps?

Ans: Prototyping Model

Whenever a user has incomplete, incoherent and vague requirements for the project then the Prototyping Model is used. The prototype is built on the basis of the current known requirements so as to finalise on the design and coding. This helps in understanding the requirements more accurately to develop a robust system.

The prototype is developed under the design, coding and testing phases but it is not done very formally so that the customer can get an idea (actual feel) of the system.

This interaction with the prototype helps the customer to better understand the design requirements resulting in more stable requirements.

A **prototype** is a dummy of the real system which demonstrates the limited functional capabilities, low reliability and inefficient performance in comparison to the actual system being built.

Creation of prototypes involves high costs. But in few cases, budget of the software development without prototyping is more when compare to with prototyping. The reasons for this are:

1) Experience gained while developing the prototype might actually help in scaling down the costs of real software system.
2) It is also important for the projects where development takes place for a long period of time and requirements constantly change.

Steps in Prototyping Model

Requirements gathering are the first step in prototyping. The various activities are (**figure 1.5**):

Step 1: The customer and developer meet to decide the whole goals for developing the software system. In this meet outline of the uncertainties are also defined.

Step 2: In this step a "quick design" is developed which emphases the input approaches, output formats, etc. and shown to the customer.

Step 3: Then a prototype is constructed.

Step 4: Prototype is used by the customer to understand the refinements required in the requirements of the software.

Step 5: Iteration takes place as the prototype take into consideration changes demanded by the customer as well as enables the developer to better understand the changes which are required.

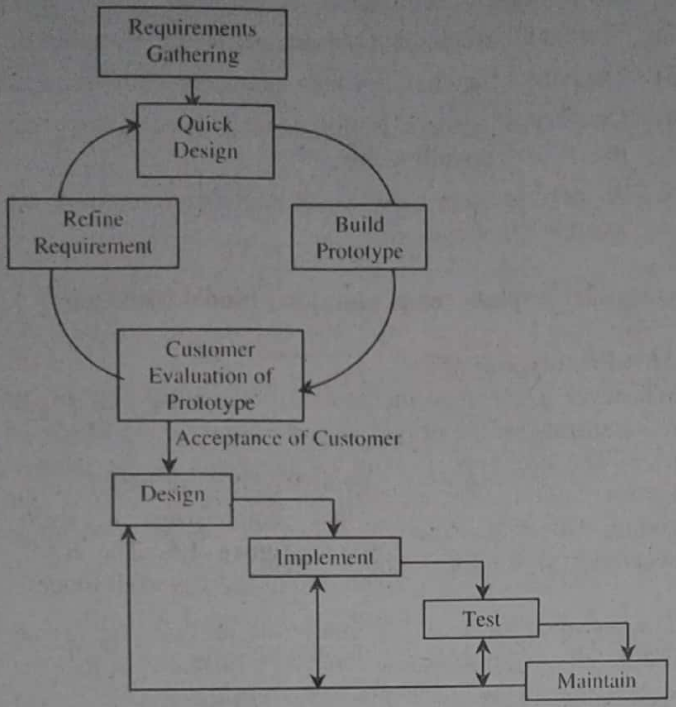


Figure 1.5: Prototyping Model

Ques 13) What are the conditions of using prototyping model? Also discuss the types of prototyping model.

Or

What are the advantages and disadvantages of throwaway prototyping?

Ans: Conditions of Using Prototyping

Various conditions of using prototyping are as follows:

1) **Unclear Requirements:** Prototyping is most effective in projects where requirements are not clearly understood and confidence in gathered information is low. In such cases, if the waterfall model is used then it freezes the requirements so developers cannot continue the development. Any change brings about reworking at the time of development. Once a prototype is built and tested, then requirements become more understandable and stable.

2) **Complicated and Large Systems:** Prototyping also suits large and complex systems for which there are no manual guidelines or existing system for requirement gathering.

3) **Illustration of User Interface:** Prototyping aids in providing a clear picture of the input data formats, messages, reports and the interactive dialogues to the customer. This helps in understanding of user requirements well and later in designing the UID (User Interface Design).

4) **Unclear Solutions:** Sometimes, technical solutions seem ambiguous to the development team. At these times, building of a prototype provides an opportunity to engineers to critically examine all the technical concerns pertaining to the product development.

5) **Throw-away Plans:** In order to develop a good product, one must develop a dummy first to determine what defects can come up. This can be rectified later to come up with a quality product.

Types of Prototyping

Prototyping can be of two major types:

1) **Exploratory Development:** Exploratory development refers to working with the user to understand the system requirements and deliver a final product. It starts with understanding the parts clearly specified by the customer and developing new features as demanded by customer during the developmental phase. It uses the unfamiliar language and environment with changing client requirements. In this, the final quality of the system gets affected because of design compromises made during the prototyping phase.

   Advantages of Exploratory Development

   Advantages of exploratory development are as follows:

   i) Quick system delivery.

   ii) As the users are involved in the development phase, they feel a sense of commitment towards system success.

2) **Throwaway Prototyping:** Throwaway prototyping tries to understand the user's requirement and come up with a clear definition of the same. It tries to comprehend poorly understood components.

   Advantages of Throwaway Prototyping

   Advantages of throwaway prototyping are as follows:

   i) Brilliant for collecting and improving requirements.

   ii) It is useful for risk assessment and reduction.

   iii) Since a prototype is not the final product and can be discarded at any time, so 'quick and bad' programming technique can be used.

   Disadvantages of Throwaway Prototyping

   Disadvantages of throwaway prototyping are as follows:

   i) Once a working prototype is built then the client and organisation may want that the prototype is delivered as a final product. This is not feasible as a throwaway prototype is built with less emphasis on efficiency, reliability and error checking.

   ii) A prototype is just a dummy of the final system and not the final system in itself. So the user interaction feedback of the prototype may not match with that of the final product.

Ques 14) What are the advantages and disadvantages of prototyping model?

Advantages of Prototyping Model

The advantages of prototyping model are as follows:

1) **Clarity:** The user gets an idea of the functionality of the software and can request modifications to suit his requirement.

2) **Best for Non-IT Literate Users:** This approach is good for dealing with non-IT literate people as they are not able to properly explain their requirements and expectations from the software.

3) **Provide Judgement to Capability of the Developer:** The client can judge the capability of the developer by judging him/her on the basis of the prototype built by the developer.

4) **Risk Identification:** It minimises the failure as risks can be identified at an early stage. After identification of the risk, various risk removal steps are taken to remove the risk.

5) **Good Environment:** Repeated interaction between client and development team during project development stage gives rise to a conducive environment.

6) **Takes Less Time to Complete:** Time for completion of project is considerably reduced since the developer gets a clear idea about the requirements of clients by way of prototyping.

Disadvantages of Prototyping Model

Disadvantages of prototyping model are given below:

1) **High Cost:** The cost of prototyping is borne by the developer and therefore done with minimal resources like the application of Rapid Application Development (RAD) tools. The start-up cost of a development team is also high.

2) **Throwaway:** Once a prototype is shown to the clients, he/she may suggest further changes to the model. Thus, the prototype becomes a waste and is "thrownaway" to build a new one as per newer requirements of client.

3) **Slow Process:** It takes considerable time for building a prototype which makes it a slow process.

4) **More Involvement of Clients:** The client's excessive involvement is sometimes not desired by the developer.

5) **Too Many Changes:** The rhythm of the development team is disturbed by too many changes.

Ques 15) Write short note on evolutionary development models.

Ans: Evolutionary Development Models

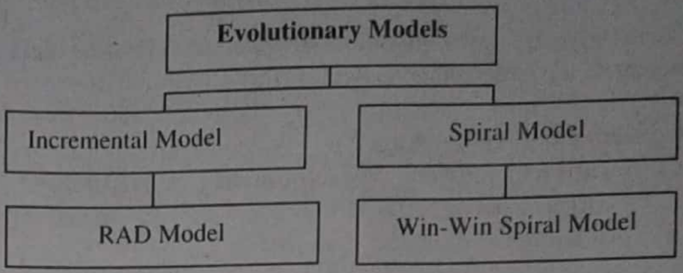
Software engineers need a process model that has been explicitly designed to accommodate a product that evolves over time so that the end product is realistic.

The linear sequential model is designed for straight-line development, i.e., the waterfall approach assumes that a complete system will be delivered after the linear sequence is completed. The Prototyping model is designed to assist the customer or developer in understanding requirements. Generally it is not designed to deliver a production system (throwaway prototyping). The evolutionary nature of software is not considered in either of these classic software engineering paradigms.

Evolutionary models are iterative. They are characterized in a manner that enables software engineers to develop increasingly more complete versions of the software.

Major Evolutionary Models

Two of the important evolutionary models are:



Ques 16) What is RAD model?

Ans: RAD Model

Rapid Application Development (RAD) is an incremental software process model that emphasizes a short development cycle as shown in **figure 1.6**. The RAD model is a "high-speed" adaptation of the waterfall model, in which rapid development is achieved by using a component-based construction approach. If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a "fully functional system" within a very short time period.

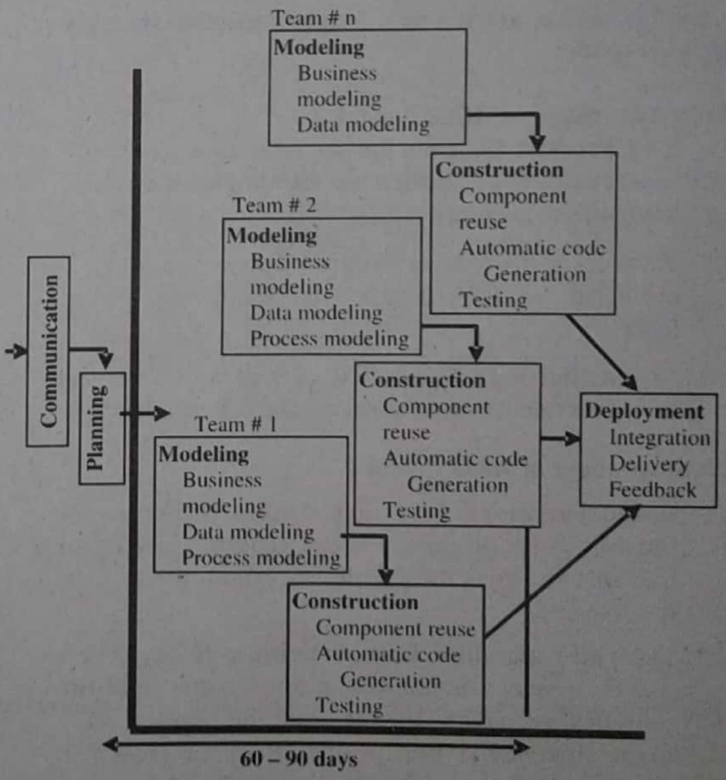


Figure 1.6: RAD Model

Communication works to understand the business problem and the information characteristics that the software must accommodate. Planning is essential because multiple software teams work in parallel on different system functions.

Modeling encompasses three major phases-business modeling, data modeling and process modeling – and establishes design representations that serve as the basis for RAD's construction activity. Construction emphasizes the use of pre-existing software components and the application of automatic code generation. Finally, deployment establishes a basis for subsequent iterations.

Obviously, the time constraints imposed on a RAD project demand "scalable scope".

Contents of RAD Package

1) **Graphical User Development Environment:** various aspects of the application can be created by just using the drag and drop facility.
2) **Reusable Components:** a library of common standard objects such as buttons, dialog boxes. The developer can drag and drop these components.
3) **Code Generator:** after the drag and drop of the components in the design the package automatically writes the code for those components.
4) **Programming Language:** the languages in the package can be like Visual Basic and C++. The package includes an integrated development environment (IDE) for creating, testing and debugging code.

Ques 17) What are the advantages and disadvantages of RAD model?

Ans: Advantages of RAD Model

1) **Fast Product Output:** For appropriate projects, this approach puts an application into production sooner than any other approach,
2) **Efficient Documentation:** Documentation is produced as a by-product of completing project tasks,
3) **Interaction with User:** RAD forces teamwork and lots of interaction between users and stakeholders.

Disadvantages of RAD Model

1) **Requirement of Liveware Support:** For large, but scalable projects, RAD requires sufficient human resources to create the right number of RAD teams.
2) **User may not like Fast Activities:** If developers and customers are not committed to the rapid-fire activities necessary to complete the system in a much abbreviated time frame, RAD projects will fail.
3) **Problematic Outputs:** If a system cannot be properly modularized, building the components necessary for RAD will be problematic.
4) **Lack of Tuning between Interface and System Components:** If high performance is an issue, and performance is to be achieved through tuning the interfaces to system components, the RAD approach may not work.
5) **Not Suitable for Technical Risks:** RAD may not be appropriate when technical risks are high (for example, when a new application makes heavy use of new technology).

Ques 18) Define the Spiral Model.

Ans: Spiral Model

Boehm proposed the Spiral Life Cycle Model which is a software process model that takes into consideration the repetitive characteristics of prototyping along with the controlled and methodical characteristics of the linear sequential model.

This model is a combination of the Waterfall Lifecycle Model and highlighting the use of risk management techniques. Each round of spiral:

i) Recognises the highest risk sub-problem, and
ii) Provides a solution to the problem.

Spiral model is also referred to as **Meta model** as it consists of other models of SDLC like waterfall and prototype models. In this model, the software engineering team moves in a clockwise direction starting from the center where each loop of the spiral represents a phase of the software development phase.

The innermost loop represents the early phases with the last one representing the end completion.

The number of phases is not fixed here as it depends on size of the project and the specifications. Some projects might require three to four loops while in others it might be over in one loop itself.

The spiral model works best for development and enhancement projects as it allows mixture of any type of approach, be it specification, prototype or simulation oriented approach. Each cycle of the spiral is reviewed at the completion of stage for errors including the next cycle plan.

High risk projects adopt the spiral model approach as it uses prototyping as a risk reduction method while conforming to the systematic, sequential waterfall approach.

Quadrants in Spiral Model

There are four quadrants (parts) in the spiral model as shown in **figure 1.7.**

1) **First Quadrant:** The purpose of this quadrant is to identify the objective of the phase and different solutions are considered for the phase.
2) **Second Quadrant:** On the basis of the objectives and constraints it emphasises on the evaluation of various alternatives. It focuses on the risks involved that some of the objectives might not be achieved. This will hinder the completion of a software project.

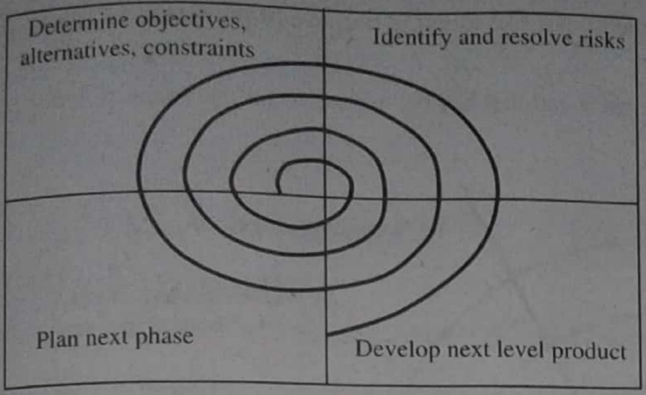

Determine objectives, alternatives, constraints
Identify and resolve risks
Plan next phase
Develop next level product

Figure 1.7: Spiral Model

3) **Third Quadrant:** This involves developing strategies to tackle risks. Such strategies include benchmarking, simulation and prototyping.

4) **Fourth Quadrant:** This stage reviews the results of the previous stage with the customer so that further planning may be done. Iteration is done at every stage of the spiral to come up with a complete software version.

Ques 19) What are the advantages and disadvantages of spiral model?

Ans: Advantages of Spiral Model

Salient advantages of spiral model are as follows:

1) **Risk Identification at Early Stage:** Project risks are identified at the initial stages at a low costs.

2) **Visible Prototype to Users:** Rapid prototyping tools enable users to view the prototype of the system at the very initial stages.

3) **Suitable for High Risk Projects:** This model is suitable for projects where business needs are unstable with high risks.

4) **Flexible for Adding Functionality:** It is also flexible where extra functionality can be added later.

Disadvantages of Spiral Model

Some of the disadvantages of spiral model are as follows:

1) **Costly:** Spiral model is expensive to use as it involves high costs.

2) **Risk Dependent:** Good risk analysis ensures the project's success.

3) **Not Suitable for Smaller Projects:** It works well for large projects not smaller ones.

4) **Complicated Approach:** Projects with clear SRS will find this approach complicated.

5) **Difficult to Meeting Budget:** This development process makes meeting budgetary and scheduling requirements difficult.

Ques 20) Describe the Win-Win Spiral model with steps.

Ans: Win-Win Spiral Model

A slight variation in the **Boehm's spiral model** is **Boehm's Win-Win spiral model** which focuses on the reality that customer and developer enter into a process of negotiation where both parties negotiates to balance functionality, performance, and other product or system characteristics against cost and time to market.

The best negotiations strive for a "win-win" result, where the customer wins by getting the system satisfying most of their requirements and developers wins by working on realistic and achievable budgets and deadlines.

Boehm's WIN-WIN spiral model defines a set of negotiation activities at the beginning of each pass around the spiral. The following activities are defined:

1) Identification of the system or subsystem's key "stakeholders."
2) Determinations of the stakeholders "win conditions".
3) Negotiations of the stakeholders win conditions to reconcile them into a set of win-win conditions for all concerned (including the software project team).

Steps of Win-Win Spiral Model

There following are seven steps as shown in the **figure 1.8**:

Step 1) Identify next-level Stakeholders: The next-level stakeholders are identified. These are the people who you need to make happy after the current phase of development.

Step 2) Identify Stake holder's win Conditions: Stakeholders' "goals and concerns" or objectives are documented.

Step 3) Reconcile Win Conditions, Establish Next Level Objectives, Constraints, Alternatives: In this step, it is made sure that the win conditions are consistent and can begin planning for the next level.

Step 4) Evaluate Product and Process Alternatives, Resolve Risks: Current project is examined carefully to consider alternatives and to resolve any risks that may have been found. If they have not been found but later cause problems, blame someone no longer on the project.

Step 5) Define Next Level of Product and Process, Including Partitions: At this step, the next level is further defined and may partition the system into subsystems that can be developed in parallel cycles.

Step 6) Validate Product and Process Definitions: Most people do not like this step because of the repetitive nature but it is an important step.

Step 7) Review Commitment: In this final step, all the work is reviewed that has been done so far and make sure that the product is still feasible and worthwhile.

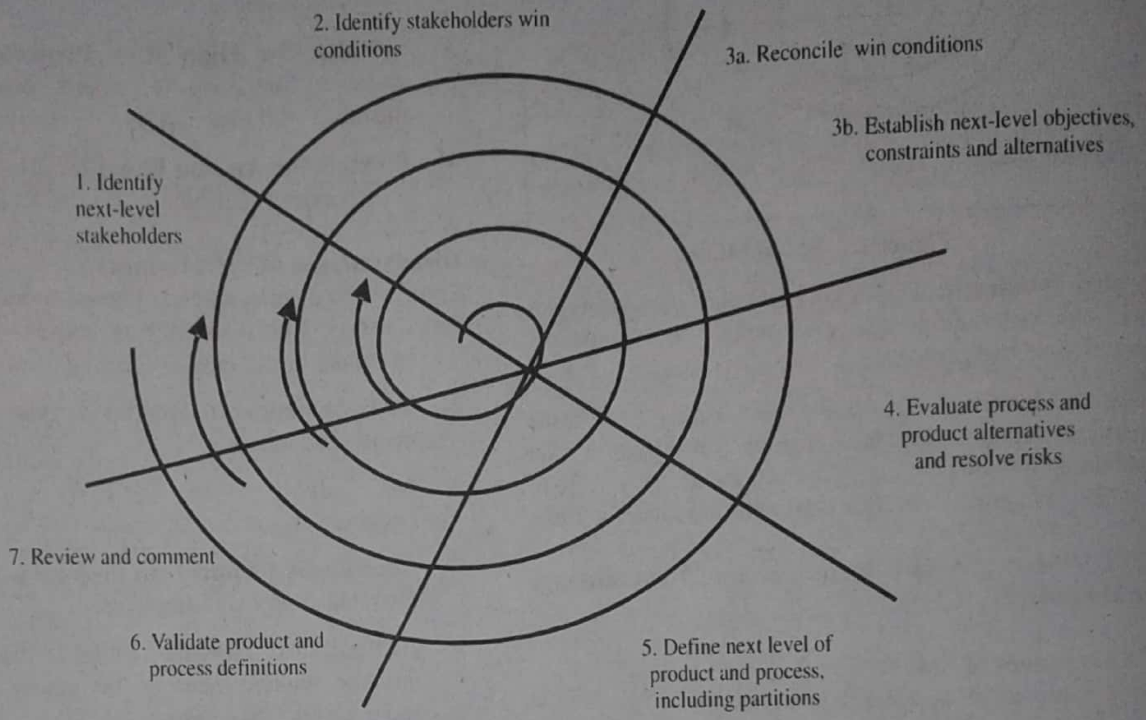

2. Identify stakeholders win conditions
3a. Reconcile win conditions
3b. Establish next-level objectives, constraints and alternatives
1. Identify next-level stakeholders
4. Evaluate process and product alternatives and resolve risks
7. Review and comment
6. Validate product and process definitions
5. Define next level of product and process, including partitions

Figure 1.8: Win-Win Spiral Model