Software Engineering and Case Tools (MCA 544)

1

UNIT-I

INTRODUCTION & SRS

2

Introduction

- The term software engineering is composed of two words**, software** and **engineering**.
- **Software** is more than just a program code.
  - A program is an executable code, which serves some computational purpose.
  - Software is considered to be a collection of executable programming code, associated libraries and documentations.
  - Software, when made for a specific requirement is called software product.

3

- **Engineering** is all about developing products, using well-defined, scientific principles and methods.
- *The software engineering can be defined as an engineering branch associated with the development of software product using well-defined scientific principles, methods and procedures.*
- The outcome of software engineering is an efficient and reliable software product.

4

- **IEEE defines software engineering as:**

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

  - Without using software engineering principles it would be difficult to develop large programs.
  - In industry it is usually needed to develop large programs to accommodate multiple functions.

5

- A problem with developing such large commercial programs is that the complexity and difficulty levels of the programs increase exponentially with their sizes. Software engineering helps to reduce this programming complexity.
- Software engineering principles use two important techniques to reduce problem complexity: **abstraction** and **decomposition**.

6

Abstraction

- The principle of abstraction implies that a problem can be simplified by omitting irrelevant details.
- The main purpose of abstraction is to consider only those aspects of the problem that are relevant for certain purpose and suppress other aspects that are not relevant for the given purpose.
- Once the simpler problem is solved, then the omitted details can be taken into consideration to solve the next lower level abstraction, and so on.
- Abstraction is a powerful way of reducing the complexity of the problem.

7

Decomposition

- In decomposition, a complex problem is divided into several smaller problems and then the smaller problems are solved one by one.
- Any random decomposition of a problem into smaller parts will not help.
- The problem has to be decomposed such that each component of the decomposed problem can be solved independently and then the solution of the different components can be combined to get the full solution.
- A good decomposition of a problem should minimize interactions among various components. If the different subcomponents are interrelated, then the different components cannot be solved separately and the desired reduction in complexity will not be realized.

8

Need of Software Engineering

- The need of software engineering arises because of higher rate of change in user requirements and environment on which the software is working.
  - **Large software** - It is easier to build a wall than to a house or building, likewise, as the size of software become large engineering has to step to give it a scientific process.
  - **Scalability-** If the software process were not based on scientific and engineering concepts, it would be easier to re-create new software than to scale an existing one

9

Need of……..

- **Cost-** As hardware industry has shown its skills and huge manufacturing has lower down the price of computer and electronic hardware. But the cost of software remains high if proper process is not adapted.
- **Dynamic Nature-** The always growing and adapting nature of software hugely depends upon the environment in which the user works. If the nature of software is always changing, new enhancements need to be done in the existing one. This is where software engineering plays a good role.
- **Quality Management-** Better process of software development provides better and quality software product.

10

Importance of Software

- Maximizes the Full Potential of Hardware
- Provides an Intuitive Interface to Work With
- Promotes Innovation and Collaboration
- Boosts Efficiency and Productivity

11

Evolving Role of Software

- **As a product**
- **As a medium to deliver a product**
  - Software acts as the basis for the control of the computer (operating systems), the communication of information (networks), and the creation and control of other programs (software tools and environments). Software delivers the most important product of today's time—information
- **Information transformer**
  - producing, managing, acquiring, modifying, displaying, or transmitting information that can be as simple as a single bit or as complex as a multimedia presentation.

12

- Software transforms personal data
- It manages business information to enhance competitiveness
- Provides a gateway to worldwide information networks
- Provides the means for acquiring information in all of its forms

13

Characteristics of Good Software

- A software product can be judged by what it offers and how well it can be used.
- The software must satisfy on the following grounds:
  - Operational
  - Transitional
  - Maintenance

14

Operational

- This tells us how well software works in operations. It can be measured on:
  - Budget
  - Usability
  - Efficiency
  - Correctness
  - Functionality
  - Dependability
  - Security
  - Safety

15

Transitional

- This aspect is important when the software is moved from one platform to another:
  - Portability
  - Interoperability
  - Reusability
  - Adaptability

16

Maintenance

- This aspect briefs about how well a software has the capabilities to maintain itself in the ever-changing environment:
  - Modularity
  - Maintainability
  - Flexibility
  - Scalability

17

Software Crisis

- "Software Crisis" describe the state of the software development industry, where common problems includes:
  - Projects that run over-budget
  - Projects that run over-time
  - Software that make inefficient use of calculations and memory
  - Software of low quality
  - Software that fail to meet the requirements it is developed to meet
  - Projects that become unmanageable and code difficult to maintain
  - Software that never finish development

18

Software Engineering Problems

- The Problem of scale
- Cost
- Schedule
- Quality
  - The quality of a software product as having three dimensions:
    - Product Operation
    - Product Transition
    - Product Revision

19

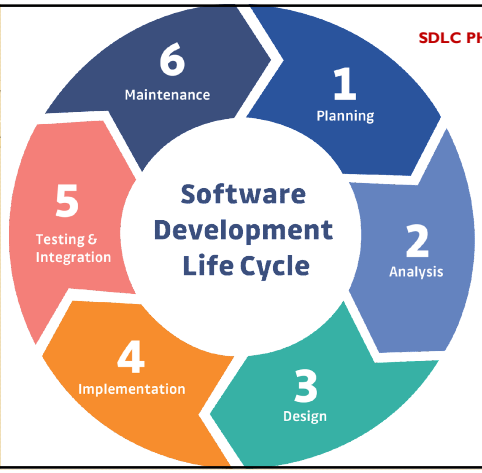
Software Development Life Cycle

- A software life cycle model (also called process model) is a descriptive and diagrammatic representation of the software life cycle.
- A life cycle model represents all the activities required to make a software product transit through its life cycle phases.
- A life cycle model maps the different activities performed on a software product from its inception to retirement.

20

- Different life cycle models may map the basic development activities to phases in different ways.
- The basic activities are included in all life cycle models though the activities may be carried out in different orders in different life cycle models.
- During any life cycle phase, more than one activity may also be carried out.

21



22

Need for Software Development Models

- Without using of a particular life cycle model the development of a software product would not be in a systematic and disciplined manner.
- A software life cycle model defines entry and exit criteria for every phase.
- A phase can start only if its phase-entry criteria have been satisfied. So without software life cycle model the entry and exit criteria for a phase cannot be recognized.
- Without software life cycle models it becomes difficult for software project managers to monitor the progress of the project.

23

Feasibility study

- The main aim of feasibility study is to determine whether it would be financially and technically feasible to develop the product.
- At first project managers or team leaders try to have a rough understanding of what is required to be done by visiting the client side.
- After they have an overall understanding of the problem they investigate the different solutions that are possible.
- Based on this analysis they pick the best solution and determine whether the solution is feasible financially and technically.

24

Requirements analysis and specification

- The aim of the requirements analysis and specification phase is to understand the exact requirements of the customer and to document them properly. This phase consists of two distinct activities, namely
  - Requirements gathering and analysis
    - The goal of the requirements gathering activity is to collect all relevant information from the customer regarding the product to be developed.
  - Requirements specification
    - The requirements analysis activity is begun by collecting all relevant data regarding the product to be developed from the users of the product and from the customer through interviews and discussions

25

Design

- The goal of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language.
- During the design phase the software architecture is derived from the SRS document.
  - Traditional design approach
  - Object-oriented design approach

26

Traditional design approach

- Traditional design consists of two different activities; first a structured analysis of the requirements specification is carried out where the detailed structure of the problem is examined.
- This is followed by a structured design activity. During structured design, the results of structured analysis are transformed into the software design.

27

Object-oriented design approach

- In this technique, various objects that occur in the problem domain and the solution domain are first identified, and the different relationships that exist among these objects are identified.
- The object structure is further refined to obtain the detailed design.

28

Different software life cycle models

- Many life cycle models have been proposed so far. Each of them has some advantages as well as some disadvantages
  - Classical Waterfall Model
  - Iterative Waterfall Model
  - Prototyping Model
  - Evolutionary Model
  - Spiral Model

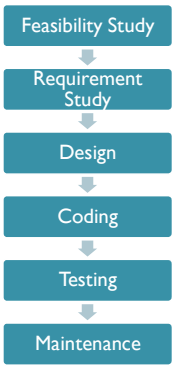
29

Classical Waterfall Model

- The classical waterfall model is intuitively the most obvious way to develop software.
- it is not a practical model in the sense that it cannot be used in actual software development projects.
- Thus, this model can be considered to be a theoretical way of developing software.
- All other life cycle models are essentially derived from the classical waterfall model. So, in order to be able to appreciate other life cycle models it is necessary to learn the classical waterfall model.

30

Classical Waterfall Model



31

When to use Waterfall Model?

- When the requirements are constant and not changed regularly.
- A project is short
- The situation is calm
- Where the tools and technology used is consistent and is not changing
- When resources are well prepared and are available to use.

32

Advantages of Waterfall model

- This model is simple to implement also the number of resources that are required for it is minimal.
- The requirements are simple and explicitly declared; they remain unchanged during the entire project development.
- The start and end points for each phase is fixed, which makes it easy to cover progress.
- The release date for the complete product, as well as its final cost, can be determined before development.
- It gives easy to control and clarity for the customer due to a strict reporting system.

33

Disadvantages of Waterfall model

- In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
- This model cannot accept the changes in requirements during development.
- It becomes tough to go back to the phase. For example, if the application has now shifted to the coding phase, and there is a change in requirement, It becomes tough to go back and change it.
- Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare.

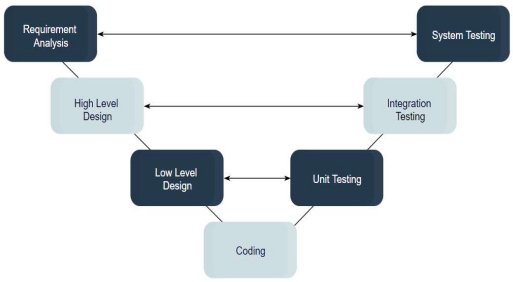
34

V Model

- The V model (Verification and Validation model) is an extension of the waterfall model.
- All the requirements are gathered at the start and cannot be changed.
- We have a corresponding testing activity for each stage.
- For every phase in the development cycle, there is an associated testing phase.
- The V model is highly disciplined, easy to understand, and makes project management easier. But it isn't good for complex projects or projects that have unclear or changing requirements.
- This makes the V model a good choice for software where downtimes and failures are unacceptable.

35

V Model



36

When to use V-Model?

- When the requirement is well defined and not ambiguous.
- The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
- The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.

37

Advantage of V-Model

- Easy to Understand.
- Testing Methods like planning, test designing happens well before coding.
- This saves a lot of time. Hence a higher chance of success over the waterfall model.
- Avoids the downward flow of the defects.
- Works well for small plans where requirements are easily understood.

38

Disadvantage of V-Model

- Very rigid and least flexible.
- Not a good for a complex project.
- Software is developed during the implementation stage, so no early prototypes of the software are produced.
- If any changes happen in the midway, then the test documents along with the required documents, has to be updated.

39

Agile Model

- The meaning of Agile is swift or versatile."**Agile process model**" refers to a software development approach based on iterative development.
- Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning.
- The project scope and requirements are laid down at the beginning of the development process.
- Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

40

Agile Model

Fig. Agile Model

41

- The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements.
- Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

When to use the Agile Model?

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

42

Advantage of Agile Method

- Frequent Delivery
- Face-to-Face Communication with clients.
- Efficient design and fulfils the business requirement.
- Anytime changes are acceptable.
- It reduces total development time.

43

Disadvantages of Agile Model

- Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
- Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

44

Iterative Model

- This model starts with some of the software specifications and develop the first version of the software.
- After the first version if there is a need to change the software, then a new version of the software is created with a new iteration.
- Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.
- The Iterative Model allows the accessing earlier phases, in which the variations made respectively.
- The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.

45

Iterative model

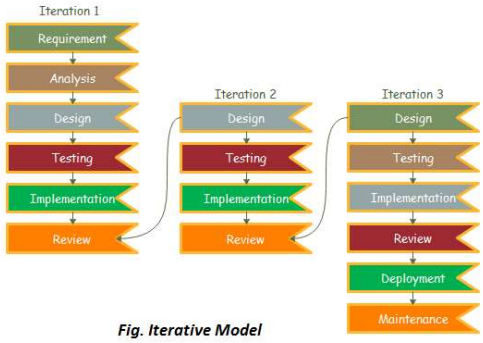


Fig. Iterative Model

46

When to use the Iterative Model?

- When requirements are defined clearly and easy to understand.
- When the software application is large.
- When there is a requirement of changes in future.

47

Advantage of Iterative Model

- Testing and debugging during smaller iteration is easy.
- A Parallel development can plan.
- It is easily acceptable to ever-changing needs of the project.
- Risks are identified and resolved during iteration.
- Limited time spent on documentation and extra time on designing.

48

Disadvantage of Iterative Model

- It is not suitable for smaller projects.
- More Resources may be required.
- Design can be changed again and again because of imperfect requirements.
- Requirement changes can cause over budget.
- Project completion date not confirmed because of changing requirements.

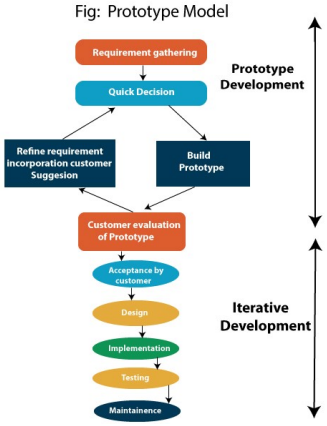
49

Prototyping Model

- **Prototype**
  - A prototype is a toy implementation of the system. A prototype usually exhibits limited functional capabilities, low reliability, and inefficient performance compared to the actual software.
  - A prototype is usually built using several shortcuts. The shortcuts might involve using inefficient, inaccurate, or dummy functions. The shortcut implementation of a function, for example, may produce the desired results by using a table look-up instead of performing the actual computations.
  - A prototype usually turns out to be a very crude version of the actual system.

50

Fig: Prototype Model



51

Need for a prototype in software development

- There are several uses of a prototype. An important purpose is to illustrate the input data formats, messages, reports, and the interactive dialogues to the customer.
- This is a valuable mechanism for gaining better understanding of the customer's needs:
  - how the screens might look like ?
  - how the user interface would behave?
  - how the system would produce outputs ?

52

- A prototyping model can be used when technical solutions are unclear to the development team.
- A developed prototype can help engineers to critically examine the technical issues associated with the product development.
- Often, major design decisions depend on issues like the response time of a hardware controller, or the efficiency of a sorting algorithm, etc. In such circumstances, a prototype may be the best or the only way to resolve the technical issues.
- A prototype of the actual product is preferred in situations such as:
  - User requirements are not complete
  - Technical issues are not clear

53

Advantage of Prototype Model

- Reduce the risk of incorrect user requirement
- Good where requirement are changing/uncommitted
- Regular visible process aids management
- Support early product marketing
- Reduce Maintenance cost.
- Errors can be detected much earlier as the system is made side by side.

54

Disadvantage of Prototype Model

- An unstable/badly implemented prototype often becomes the final product.
- Require extensive customer collaboration
  - Costs customer money
  - Needs committed customer
  - Difficult to finish if customer withdraw
  - May be too customer specific, no broad market
- Difficult to know how long the project will last.
- Easy to fall back into the code and fix without proper requirement analysis, design, customer evaluation, and feedback.
- Prototyping tools are expensive.
- Special tools & techniques are required to build a prototype.
- It is a time-consuming process.

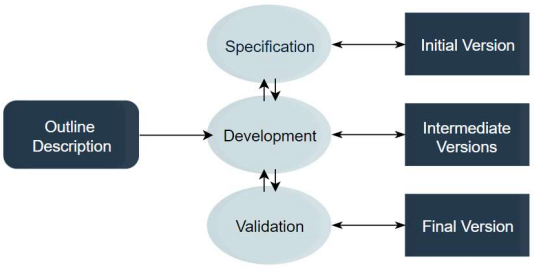
55

Incremental Model

- The incremental model divides the system's functionality into **small increments** that are delivered one after the other in quick succession. The most important functionality is implemented in the initial increments.
- Incremental development is based on developing an initial implementation, exposing it to user feedback, and evolving it through new versions.
- Each iteration passes through the requirements, design, coding, and testing stages.

56

Incremental Model



57

- The incremental model lets customer and developers see results with the first increment.
- If the customer don't like anything, everyone finds out a lot sooner.
- It is efficient as the developers only focus on what is important and bugs are fixed as they arise, but we need a clear and complete definition of the whole system before we start.
- The incremental model is great for projects that have loosely-coupled parts and projects with complete and clear requirements.

58

Spiral Model

- This model appears like a spiral with many loops.
- The exact number of loops in the spiral is not fixed.
- Each loop of the spiral represents a phase of the software process.
- Each phase in this model is split into four sectors (or quadrants)

59

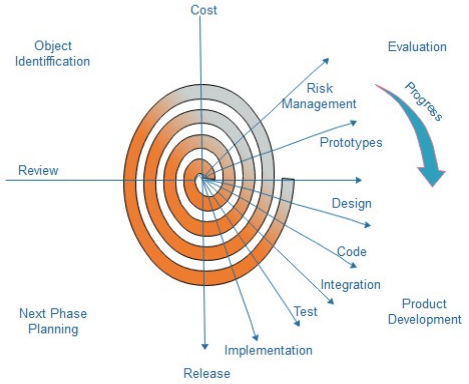


Fig. Spiral Model

60

First quadrant (Objective Setting)

- During the first quadrant, it is needed to identify the objectives of the phase.
- Examine the risks associated with these objectives.

Second Quadrant (Risk Assessment and Reduction)

- A detailed analysis is carried out for each identified project risk.
- Steps are taken to reduce the risks. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.

61

- **Third Quadrant (Development and Validation)**
  - Develop and validate the next level of the product after resolving the identified risks.

Fourth Quadrant (Review and Planning)

- Review the results achieved so far with the customer and plan the next iteration around the spiral.
- Progressively more complete version of the software gets built with each iteration around the spiral.

62

When to use Spiral Model?

- When deliverance is required to be frequent.
- When the project is large
- When requirements are unclear and complex
- When changes may require at any time
- Large and high budget projects

63

- **Advantages**
  - High amount of risk analysis
  - Useful for large and mission-critical projects.
- **Disadvantages**
  - Can be a costly model to use.
  - Risk analysis needed highly particular expertise
  - Doesn't work well for smaller projects.

64

Requirements Analysis and Specification

- The work to understand and document the exact requirement of the customer, Experienced members of the development team carry out. They are called as ***system analysts.***
- The analyst starts requirements gathering and analysis activity by collecting all information from the customer which could be used to develop the requirements of the system. He then analyzes the collected information to obtain a clear and thorough understanding of the product to be developed, with a view to remove all ambiguities and inconsistencies from the initial customer perception of the problem.

65

The questions in analyst mind……?

- What is the problem?
- Why is it important to solve the problem?
- What are the possible solutions to the problem?
- What exactly are the data input to the system and what exactly are the data output by the system?
- What are the likely complexities that might arise while solving the problem?
- If there are external software or hardware with which the developed software has to interface, then what exactly would the data interchange formats with the external system be?

66

Goals of implementation:

- The goals of implementation part documents some general suggestions regarding development. These suggestions guide trade-off among design goals.
- The goals of implementation section might document issues such as revisions to the system functionalities that may be required in the future, new devices to be supported in the future, reusability issues, etc.
- These are the items which the developers might keep in their mind during development so that the developed system may meet some aspects that are not required immediately.

67

Identifying functional requirements from a problem description

- Example: - Consider the case of the library system, where –
  - **F1**: Search Book function
  - **Input**: an author's name
  - **Output**: details of the author's books and the location of these books in the library

68

Documenting functional requirements

- For documenting the functional requirements, we need to specify the set of functionalities supported by the system.
- Example: -
  - Withdraw Cash from ATM
  - R1: withdraw cash

69

R1.1 select withdraw amount option

Input: "withdraw amount" option

Output: user prompted to enter the account type

R1.2: select account type

Input: user option

Output: prompt to enter amount

R1.3: get required amount

Input: amount to be withdrawn in integer values greater than 100 and less than 10,000 in multiples of 100.

Output: The requested cash and printed transaction statement.

Processing: the amount is debited from the user's account if sufficient balance is available, otherwise an error message displayed

70

Properties of a good SRS document

- **Concise:** The SRS document should be concise and at the same time unambiguous, consistent, and complete.
- **Structured:** It should be well-structured. A well-structured document is easy to understand and modify.
- **Black-box view:** The SRS document should specify the external behaviour of the system and not discuss the implementation issues. The SRS document should specify the externally visible behaviour of the system. For this reason, the SRS document is also called the black-box specification of a system.

71

- **Conceptual integrity:** It should show conceptual integrity so that the reader can easily understand it.
- **Response to undesired events**: It should characterize acceptable responses to undesired events. These are called system response to exceptional conditions.
- **Verifiable:** In the SRS document it should be possible to determine whether or not requirements have been met in an implementation.

72

Role of Management in Software Development

- Software Project Management consists of many activities:
  - Project planning and Tracking
  - Project Resource Management
  - Scope Management
  - Estimation Management
  - Project Risk Management
  - Scheduling Management
  - Project Communication Management
  - Configuration Management

73

- **1. Project Planning:** It is a set of multiple processes, or we can say that it a task that performed before the construction of the product starts.
- **2. Scope Management:** It describes the scope of the project. Scope management is important because it clearly defines what would do and what would not. Scope Management create the project to contain restricted and quantitative tasks, which may merely be documented and successively avoids price and time overrun.

74

- **3. Estimation management:** This is not only about cost estimation because whenever we start to develop software. it includes all the elements such as:
  - Size of software
  - Quality
  - Hardware
  - Communication
  - Training
  - Additional Software and tools
  - Skilled manpower

75

- **4. Scheduling Management:** Scheduling Management in software refers to all the activities to complete in the specified order and within time slotted to each activity. Project managers define multiple tasks and arrange them keeping various factors in mind.

For scheduling, it is compulsory –

- Find out multiple tasks and correlate them.
- Divide time into units.
- Assign the respective number of work-units for every job.
- Calculate the total time from start to finish.
- Break down the project into modules.

76

- **5. Project Resource Management:** In software Development, all the elements are referred to as resources for the project. It can be a human resource, productive tools, and libraries.

Resource management includes:

- Create a project team and assign responsibilities to every team member
- Developing a resource plan is derived from the project plan.
- Adjustment of resources.

77

- **6. Project Risk Management:** Risk management consists of all the activities like identification, analyzing and preparing the plan for predictable and unpredictable risk in the project.

Several points show the risks in the project:

- The Experienced team leaves the project, and the new team joins it.
- Changes in requirement.
- Change in technologies and the environment.
- Market competition.

78

7.Project Communication Management :

Communication is an essential factor in the success of the project. It is a bridge between client, organization, team members and as well as other stakeholders of the project such as hardware suppliers.

- From the planning to closure, communication plays a vital role. In all the phases, communication must be clear and understood. Miscommunication can create a big blunder in the project.
- **8. Project Configuration Management:** Configuration management is about to control the changes in software like requirements, design, and development of the product.
- The Primary goal is to increase productivity with fewer errors.

79