Code No. 3153/O / S

FACULTY OF ENGINEERING

B.E. 3/4 (CSE) I-Semester (Old) Examination, May / June 2017

Subject : Software Engineering

Time : 3 hours **Max. Marks : 75**

PART – A (25 Marks)

1 What do you mean by SDLC? List the various phases involved in it. 3

2 What is an iterative process? 2

3 List the objectives of planning and managing a project. 2

4 What is Risk? How do you quantify it? 3

5 List the goals of analysis modeling. 3

6 What is the purpose of design? 2

7 Define software architecture. 2

8 Differentiate cohesion and coupling. 3

9 When is testing complete? 2

10 Differentiate fault-based testing and scenario based testing. 3

PART – B (50 Marks)

11 What are evolutionary models? Explain the spiral model in detail. 10

12 Why is efforst estimation considered as crucial? List the various types of effort estimation with examples of each type. 10

13 Explain the elements involved in the dimensions of design model in detail. 10

14 Explain the different models that come into play when UI is analyzed and designed in detail. 10

15 What is product metrics? Explain the steps involved in calculating functions points in

detail. 10

16 Write short notes on :

a) FDD (Feature Driven Development) 3

b) Scrum 3

c) Object-oriented modeling 4

17 Write short notes on :

a) Art of Negotiation 3

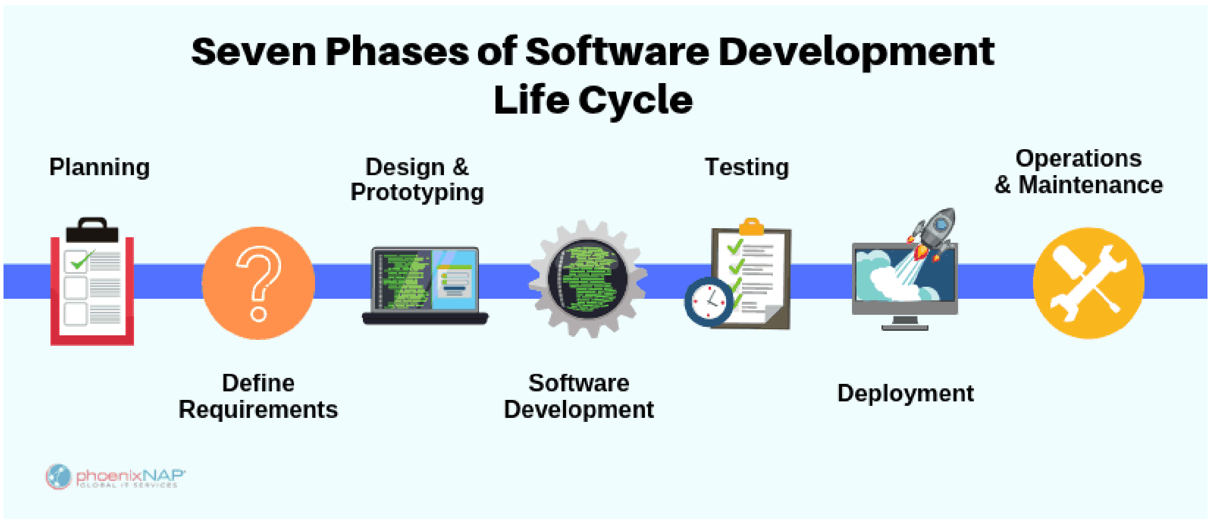
b) Layered Architectural style 4

c) Smoke testing 3

******

1. What do you mean by SDLC? List the various phases involved in it

Ans. Software Development Life Cycle is the application of standard business practices to building software applications. It's typically divided into six to eight steps: Planning, Requirements, Design, Build, Document, Test, Deploy, Maintain. SDLC is a way to measure and improve the development process.



2 What is an iterative process?

Ans. Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. ... During each iteration, the development module goes through the requirements, design, implementation and testing phases.

3 List the objectives of planning and managing a project.

- **Ans.** The successful development and implementation of all project's procedures.
- Productive guidance, efficient communication and apt supervision of the project's team.
- The achievement of the project's main goal within the given constraints.

.

4 What is Risk? How do you quantify it?

Ans. Software risk encompasses the probability of occurrence for uncertain events and their potential for loss within an organization. ... Typically, software risk is viewed as a combination of robustness, performance efficiency, security and transactional risk propagated throughout the system.

The risk of software projects is measured in terms of cost that is needed to abate the risk. Traditional practice to measure the risk of software projects uses risk exposure; however, risk measure cannot quantify the risk beyond the expected value of cost.

5 List the goals of analysis modeling.

Ans. Objectives of Analysis Modelling:

- It must establish a way of creation of **software design**.
- It must describe requirements of customer.
- It must define set of requirements which can be validated, once the **software** is built.

6 What is the purpose of design?

Ans. Software Design is the process to transform the user requirements into some suitable form, which helps the programmer in software coding and implementation. During the software design phase, the design document is produced, based on the customer requirements as documented in the SRS document.

7 Define software architecture.

Ans. software architecture refers to the fundamental structures of a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among them, and properties of both elements and relations. The architecture of a software system is a metaphor, analogous to the architecture of a building. It functions as a blueprint for the system and the developing project, laying out the tasks necessary to be executed by the design teams.

8 Differentiate cohesion and coupling.

Ans. Cohesion is the indication of the relationship within the module. Coupling is the indication of the relationships between modules. Cohesion shows the module's relative functional strength. Coupling shows the relative independence among the modules.

9 When is testing complete?

Ans. Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification.

10 Differentiate fault-based testing and scenario based testing.

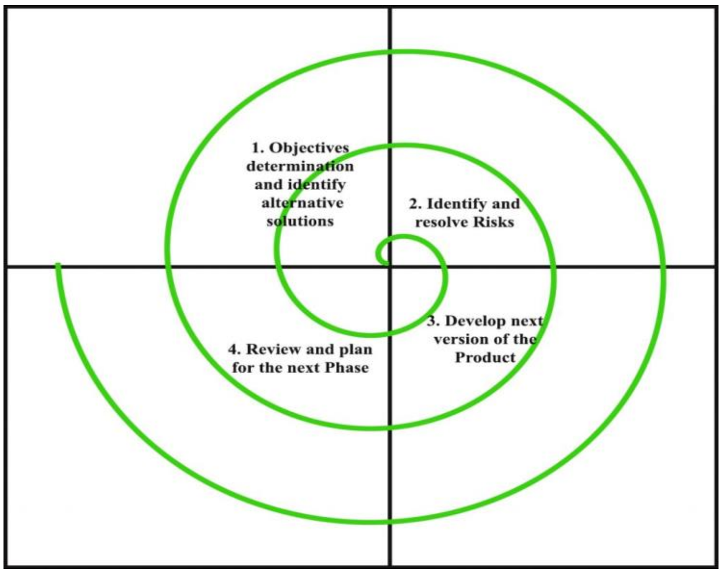
Ans. Fault-based test generation gives a low number of **test** cases with a code coverage lower than that of the other two techniques. Random **testing** may require a long time to find particular failures, but possibly finds failures overseen by **scenario-based testing**

11. What are evolutionary models? Explain the spiral model in detail.

Ans. Evolutionary model is a combination of Iterative and Incremental model of software development life cycle. ... Evolutionary model solves this problem in a different approach. Evolutionary model suggests breaking down of work into smaller chunks, prioritizing them and then delivering those chunks to the customer one by one.

Spiral model is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a **Phase of the software development process.** The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using the spiral model.
The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

The below diagram shows the different phases of the Spiral Model: –


1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

1. **Objectives determination and identify alternative solutions**: Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

2. **Identify and resolve Risks**: During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

3. **Develop next version of the Product**: During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

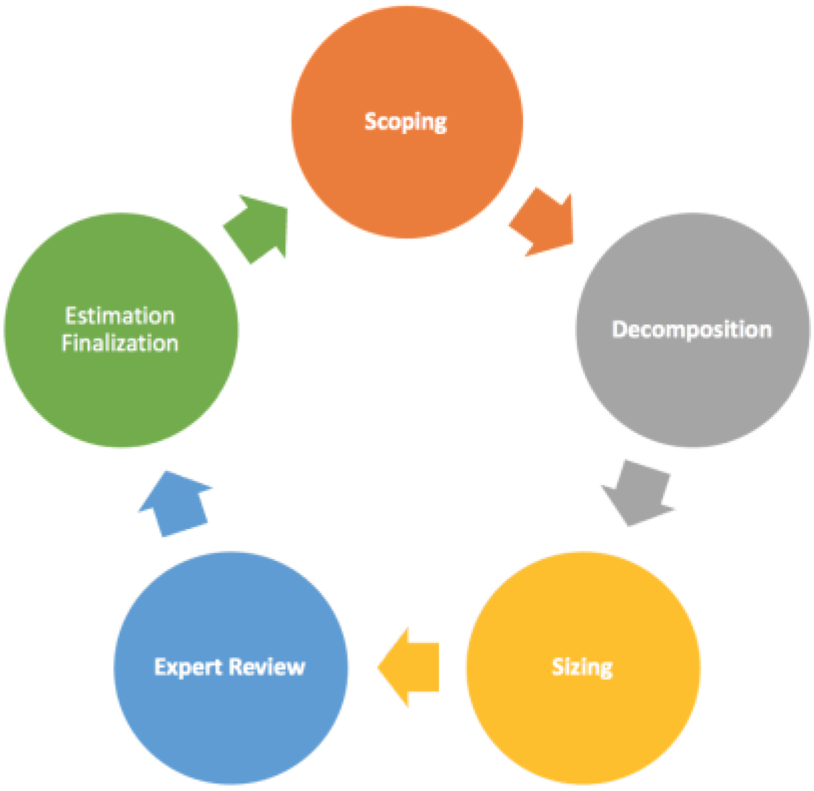
12 Why is efforst estimation considered as crucial? List the various types of effort estimation with examples of each type.

Ans. Effort estimation is the process of forecasting how much effort is required to develop or maintain a software application. This effort is traditionally measured in the hours worked by a person, or the money needed to pay for this work.

Effort estimation is used to help draft project plans and budgets in the early stages of the software development life cycle. This practice enables a project manager or product owner to accurately predict costs and allocate resources accordingly.

The Estimation Process

As mentioned the estimation is a process and this process contains the following steps to reach the estimate, this process is cycling until you reach the final estimate for the project.



1- Scoping

You need first to scope the project even if you do not have the full detailed requirements but you can assume some of them or add margins later. While in most cases you will have a defined scope to start with. You can read this article to understand what is the scope and how it is different from the requirements specifications.
You can always list your assumptions to justify the outcome of the estimation process and its results.

2- Decomposition

In this step, you will need to break your software into smaller components and functions and you can categorize them to a different set of elements, this is similar to work breakdown structure but only for the software components not all the working activities for the software.

You may also collect different data from the project team or the customer to ensure that you have listed all functionalities.

3- Sizing

In this step, the actual estimation will be done for each component alone, and I will illustrate more about how you will do that using the techniques mentioned above, this will be illustrated in 8 steps in details below.

In this step, and for more validation, you can use different estimation techniques to analyze the different estimation outputs and you may take an average of these estimates as well.

4- Expert and Peer Review

After initial estimate, you will need at some point to ask for expert opinion for some new functionalities you may not aware off, or for considering a review from your peers that you have done the correct estimation. Moreover, you may need to do some analogy based techniques for similar components or functions developed before or maybe a similar project to ensure that you are on the correct path.

5- Estimation Finalization

This can be considered the final step as you aggregate all the estimations from all components and functions and have a baseline estimate. You can go another round across the process until reaching the correct estimate which will be approved by the Project team and the Management as well.

13. Explain the elements involved in the dimensions of design model in detail.

Ans. There are two dimensions in which a design model can be illustrated: process and abstraction dimension. Process dimension points the changes in the design model as the design tasks are implemented as part of software process. Abstraction dimension represents the level of detail as each element of analysis model is transformed into design equivalent and refined iteratively.

Design model elements are not developed in a sequential fashion. The design model has four major elements:
- Data
- Architecture
- Components
- Interface

Data design focuses on files or databases at an architectural level and on a component level, data design considers the data structures that are required to implement local data objects.

The **architectural model** is derived from the application domain, the analysis model and available styles and patterns.

The **interface design elements** for software tells us how the information flows in and out of the system and how it is communicated among the components. There are three parts to interface design elements: the user interface, interfaces to systems external to application and interfaces to components within the application.
User interface elements include aesthetic elements, ergonomic elements and technical elements. External interface design requires information about entity to which information is sent or received. Internal interface design is closely related to component level design.

The **component level design elements** describes the internal detail of software component. The component level design defines data structures for all local data objects and algorithmic detail processing that occurs within a component and an interface that allows access to all behaviors.

14 Explain the different models that come into play when UI is analyzed and designed in detail.

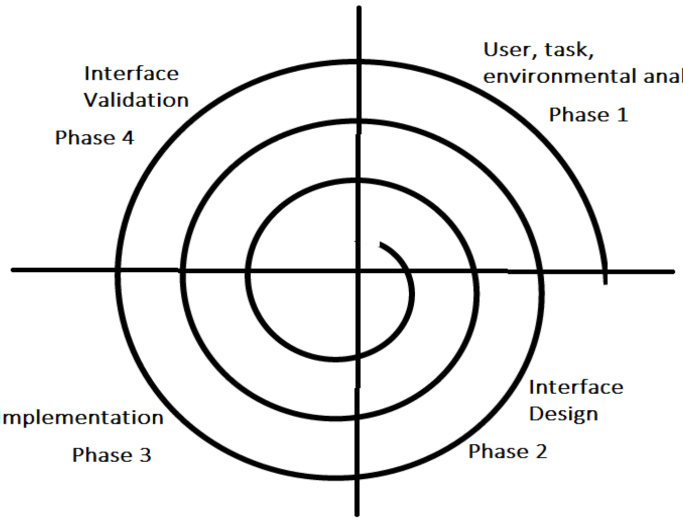
Ans. User interface is the front-end application view to which user interacts in order to use the software. The software becomes more popular if its user interface is:

- Attractive
- Simple to use
- Responsive in short time
- Clear to understand
- Consistent on all interface screens

There are two types of User Interface:

1. **Command Line Interface:** Command Line Interface provides a command prompt, where the user types the command and feeds to the system. The user needs to remember the syntax of the command and its use.
2. **Graphical User Interface:** Graphical User Interface provides the simple interactive interface to interact with the system. GUI can be a combination of both hardware and software. Using GUI, user interprets the software.

User Interface Design Process:



The analysis and design process of a user interface is iterative and can be represented by a spiral model. The analysis and design process of user interface consists of four framework activities.

1. **User, task, environmental analysis, and modeling:** Initially, the focus is based on the profile of users who will interact with the system, i.e. understanding, skill and knowledge, type of user, etc, based on the user's profile users are made into categories. From each category requirements are gathered. Based on the requirements developer understand how to develop the interface. Once all the requirements are gathered a detailed analysis is conducted. In the analysis part, the tasks that the user performs to establish the goals of the system are identified, described and elaborated. The analysis of the user environment focuses on the physical work environment. Among the questions to be asked are:
   - Where will the interface be located physically?
   - Will the user be sitting, standing, or performing other tasks unrelated to the interface?
   - Does the interface hardware accommodate space, light, or noise constraints?
   - Are there special human factors considerations driven by environmental factors?
2. **Interface Design:** The goal of this phase is to define the set of interface objects and actions i.e. Control mechanisms that enable the user to perform desired tasks. Indicate how these control mechanisms affect the system. Specify the action sequence of tasks and subtasks, also called a user scenario. Indicate the state of the system when the user performs a particular task. Always follow the three golden rules stated by Theo Mandel. Design issues such as response time, command and action structure, error handling, and help facilities are considered as the design model is refined. This phase serves as the foundation for the implementation phase.

3. **Interface construction and implementation:** The implementation activity begins with the creation of prototype (model) that enables usage scenarios to be evaluated. As iterative design process continues a User Interface toolkit that allows the creation of windows, menus, device interaction, error messages, commands, and many other elements of an interactive environment can be used for completing the construction of an interface.
4. **Interface Validation:** This phase focuses on testing the interface. The interface should be in such a way that it should be able to perform tasks correctly and it should be able to handle a variety of tasks. It should achieve all the user's requirements. It should be easy to use and easy to learn. Users should accept the interface as a useful one in their work.

Golden Rules:

The following are the golden rules stated by Theo Mandel that must be followed during the design of the interface.

Place the user in control:

- Define the interaction modes in such a way that does not force the user into unnecessary or undesired actions: The user should be able to easily enter and exit the mode with little or no effort.
- Provide for flexible interaction: Different people will use different interaction mechanisms, some might use keyboard commands, some might use mouse, some might use touch screen, etc, Hence all interaction mechanisms should be provided.
- Allow user interaction to be interruptable and undoable: When a user is doing a sequence of actions the user must be able to interrupt the sequence to do some other work without losing the work that had been done. The user should also be able to do undo operation.
- Streamline interaction as skill level advances and allow the interaction to be customized: Advanced or highly skilled user should be provided a chance to customize the interface as user wants which allows different interaction mechanisms so that user doesn't feel bored while using the same interaction mechanism.
- Hide technical internals from casual users: The user should not be aware of the internal technical details of the system. He should interact with the interface just to do his work.
- Design for direct interaction with objects that appear on screen: The user should be able to use the objects and manipulate the objects that are present on the screen to perform a necessary task. By this, the user feels easy to control over the screen.

15. What is product metrics? Explain the steps involved in calculating functions points in detail.

Ans. Product metrics are software product measures at any stage of their development, from requirements to established systems. Product metrics are related to software features only.

Product metrics fall into two classes:

1. Dynamic metrics that are collected by measurements made from a program in execution.
2. Static metrics that are collected by measurements made from system representations such as design, programs, or documentation.

Dynamic metrics help in assessing the efficiency and reliability of a program while static metrics help in understanding, understanding and maintaining the complexity of a software system.

Dynamic metrics are usually quite closely related to software quality attributes. It is relatively easy to measure the execution time required for particular tasks and to estimate the time required to start the system. These are directly related to the efficiency of the system failures and the type of failure can be logged and directly related to the reliability of the software.
On the other hand, static matrices have an indirect relationship with quality attributes. A large number of these matrices have been proposed to try to derive and validate the relationship between the complexity, understand ability, and maintainability. Several static metrics which have been used for assessing quality attributes, given in table of these, program or component length and control complexity seem to be the most reliable predictors of understand ability, system complexity, and maintainability.

16. Write short notes on

a) FDD (Feature Driven Development)

b) Scrum

c) Object-oriented modeling

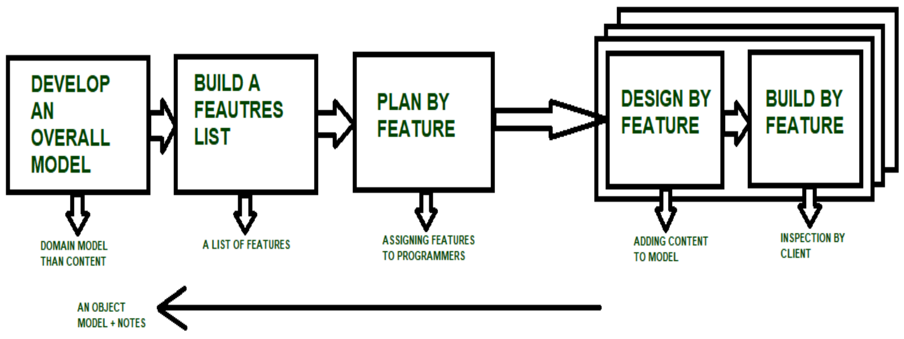
Ans.

a) FDD (Feature Driven Development)

FDD stands for Feature-Driven Development. It is an agile iterative and incremental model that focuses on progressing the features of the developing software. The main motive os feature-driven development is to provide timely updated and working software to the client. In FDD, reporting and progress tracking is necessary at all levels.

FDD Lifecycle

- Build overall model
- Build feature list
- Plan by feature
- Design by feature
- Build by feature



Characteristics of FDD

- **Short iterative:** FDD lifecycle works in simple and short iterations to efficiently finish the work on time and gives good pace for large projects.
- **Customer focused:** This agile practice is totally based on inspection of each feature by client and then pushed to main build code.
- **Structured and feature focused:** Initial activities in lifecycle builds the domain model and features list in the beginning of timeline and more than 70% of efforts are given to last 2 activities.
- **Frequent releases:** Feature-driven development provides continuous releases of features in the software and retaining continuous success of the project.

Advantages of FDD

- Reporting at all levels leads to easier progress tracking.
- FDD provides continuous success for larger size of teams and projects.
- Reduction in risks is observed as whole model and design is build in smaller segments.
- FDD provides greater accuracy in cost estimation of the project due to feature segmentation.

Disadvantages of FDD

- This agile practice is not good for smaller projects.
- There is high dependency on lead programmers, designers and mentors.
- There is lack of documentation which can create an issue afterwards.

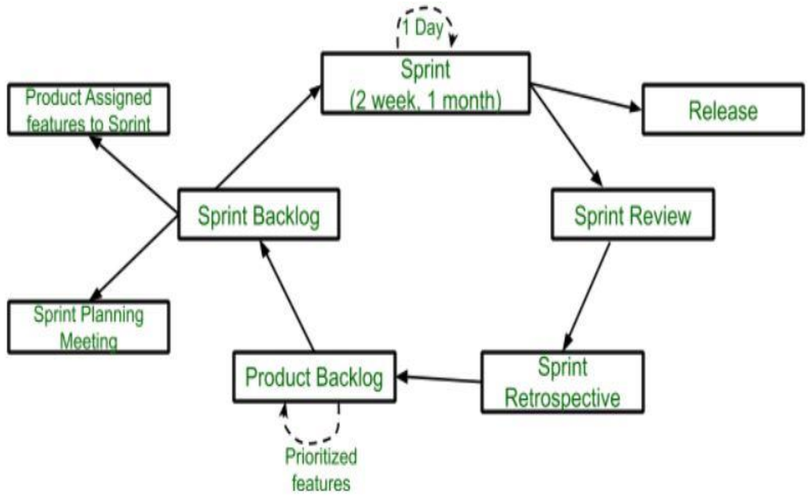
b) Scrum

Scrum is the type of **Agile framework**. It is a framework within which people can address complex adaptive problem while productivity and creativity of delivering product is at highest possible values. Scrum uses **Iterative process**.

Silent features of Scrum are:

- Scrum is light-weighted framework
- Scrum emphasizes self-organization
- Scrum is simple to understand
- Scrum framework help the team to work together

Lifecycle of Scrum:



Sprint:
A Sprint is a time-box of one month or less. A new Sprint starts immediately after the completion of the previous Sprint.
Release:
When the product is completed then it goes to the Release stage.
Sprint Review:
If the product still have some non-achievable features then it will be checked in this stage and then the product is passed to the Sprint Retrospective stage.
Sprint Retrospective:
In this stage quality or status of the product is checked.
Product Backlog:
According to the prioritize features the product is organized.
Sprint Backlog:
Sprint Backlog is divided into two parts Product assigned features to sprint and Sprint planning meeting.
Advantage of using Scrum framework:

- Scrum framework is fast moving and money efficient.
- Scrum framework works by dividing the large product into small sub-products. It's like a divide and conquer strategy
- In Scrum customer satisfaction is very important.
- Scrum is adaptive in nature because it have short sprint.
- As Scrum framework rely on constant feedback therefore the quality of product increases in less amount of time

Disadvantage of using Scrum framework:

- Scrum framework do not allow changes into their sprint.
- Scrum framework is not fully described model. If you wanna adopt it you need to fill in the framework with your own details like Extreme Programming(XP), Kanban, DSDM.
- It can be difficult for the Scrum to plan, structure and organize a project that lacks a clear definition.
- The daily Scrum meetings and frequent reviews require substantial resources.

c) Object-oriented modeling

Intention of object oriented modeling and design is to learn how to apply object -oriented concepts to all the stages of the software development life cycle.Object-oriented modeling and design is a way of thinking about problems using models organized around real world concepts. The fundamental construct is the object, which combines both data structure and behavior.

Purpose of Models:

1. Testing a physical entity before building it
2. Communication with customers
3. Visualization
4. Reduction of complexity

Types of Models:
There are 3 types of models in the object oriented modeling and design are: Class Model, State Model, and Interaction Model. These are explained as following below.

1. **Class Model:**
   The class model shows all the classes present in the system. The class model shows the attributes and the behavior associated with the objects.
   The class diagram is used to show the class model.The class diagram shows the class name followed by the attributes followed by the functions or the methods that are associated with the object of the class.Goal in constructing class model is to capture those concepts from the real world that are important to an application.

1. **State Model:**
   State model describes those aspects of objects concerned with time and the sequencing of operations – events that mark changes, states that define the context for events, and the organization of events and states.Actions and events in a state diagram become operations on objects in the class model. State diagram describes the state model.

2. **Interaction Model:**
   Interaction model is used to show the various interactions between objects, how the objects collaborate to achieve the behavior of the system as a whole.
   The following diagrams are used to show the interaction model:
   - Use Case Diagram
   - Sequence Diagram
   - Activity Diagram

17 Write short notes on :

a) Art of Negotiation

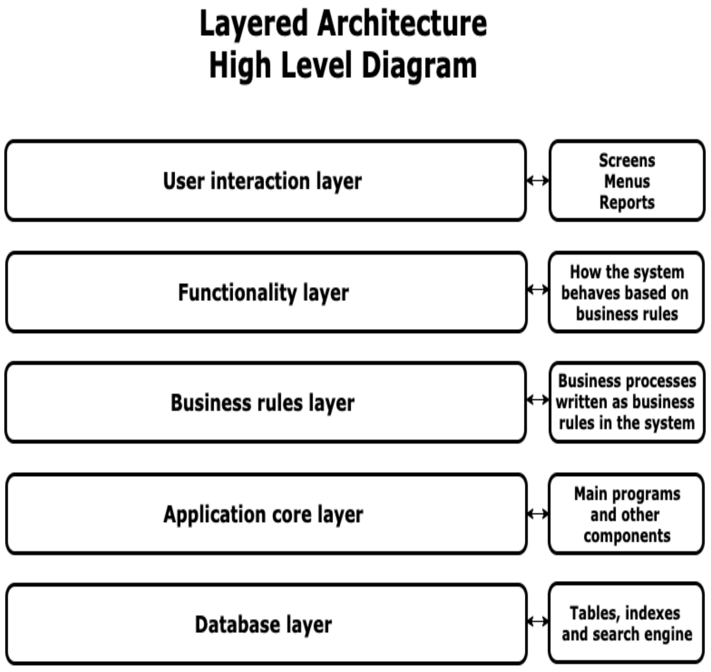
b) Layered Architectural style

c) Smoke testing

a) Art of Negotiation

Negotiation happens at every level in software engineering. The acceptance of requirements proposed by a client is negotiated in terms of timeline, cost, and need. The technical manager must negotiate who handles what requirements based on expertise, interest, and other responsibilities. The implementation is a negotiation between the software architect and the other engineers in a development team. Access policies and resource allocation/acquisitions must be negotiated between the architect, the manager, and the sys admits. Additional features (e.g. scope creep) must be negotiated by the manager, preventing his software engineers from being overworked. The design must be negotiated between the web designers, user experience designers and the front end developers, in terms of timeline, difficulty, and usability.

b. Layered Architectural style



Layered Architecture
High Level Diagram
User interaction layer
Screens
Menus
Reports
Functionality layer
How the system behaves based on business rules
Business rules layer
Business processes written as business rules in the system
Application core layer
Main programs and other components
Database layer
Tables, indexes and search engine

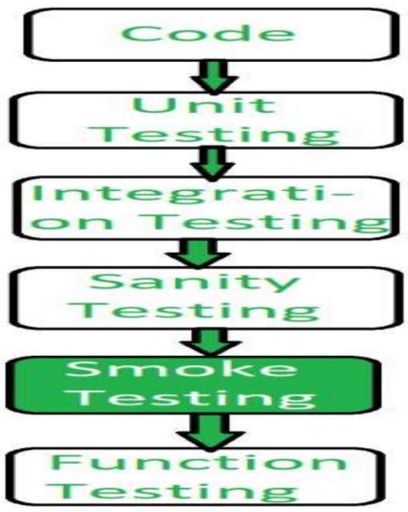
As you can see in the diagram above, a standard layered architecture has five parts:

- **User interaction layer:** This is the layer that interacts with users through screens, forms, menus, reports, etc. It is the most visible layer of the application. It defines how the application looks.
- **Functionality layer:** This is the layer that presents the functions, methods, and procedures of the system based on the business rules layer. It determines how the pull-down menus work, how the buttons work, and how the system navigates through screens.
- **Business rules layer:** This layer contains rules that determine the behavior of the whole application, such as, “If an invoice is printed, then send an email to the customer, select all items sold, and decrease their stock in the stock management module.”
- **Application core layer:** This server contains the main programs, code definitions, and basic functions of the application. Programmers work in this layer most of the time.
- **Database layer:** This layer contains the tables, indexes, and data managed by the application. Searches and insert/delete/update operations are executed here.

c) Smoke testing

Smoke Testing is a software testing method that determines whether the employed build is stable or not. It acts as a confirmation whether the quality assurance team can proceed with further testing. Smoke tests are a minimum set of tests run on each build.
Smoke testing is a process where the software build is deployed to quality assurance environment and is verified to ensure the stability of the application. Smoke Testing is also known as **Confidence Testing** or **Build Verification Testing**.

In other words, we verify whether the important features are working and there are no showstoppers in the build that is under testing.
It is a mini and quick regression test of major functionality. Smoke testing shows that the product is ready for testing. This helps in determining if the build is flawed as to make any further testing a waste of time and resources.



Characteristics of Smoke Testing:
Following are the characteristics of the smoke testing:

- Smoke testing is documented.
- Smoke testing may be stable as well as unstable.
- Smoke testing is scripted.
- Smoke testing is type of regression testing.

Smoke Testing is usually carried out by the quality assurance engineers.

Goal of Smoke Testing:
The aim of Smoke Testing is:

1. To detect any early defects in software product.
2. To demonstrate system stability.
3. To demonstrate conformance to requirements.
4. To assure that the acute functionalities of program is working fine.
5. To measures the stability of the software product by performing testing.
6. To test all over function of the software product.

Types of Smoke Testing:
There are 2 types of Smoke Testing: **Manual**, and **Automation**.
Advantages of Smoke Testing:

1. Smoke testing is easy to perform.
2. It helps in identifying defects in early stages.
3. It improves the quality of the system.
4. Smoke testing reduces the risk of failure.
5. Smoke testing makes progress easier to access.
6. It saves test effort and time.
7. It makes easy to detect critical errors and helps in correction of errors.
8. It runs quickly.
9. It minimizes integration risks.