G.PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY, Kurnool
RAVINDRA COLLEGE OF ENGINEERING FOR WOMEN, Kurnool
Department of Computer Science and Engineering
Software Engineering
BITS

UNIT I

1. Software is_____ it is not manufactured in the classical sense(developed or engineered)
2. Although the industry is moving toward component-based construction, most software continues to be_____(custom built.)
3. *Layers of software are quality focus, process, methods and _____ (tools)*

4. A generic process framework for software engineering encompasses five activities: Communication, planning, Modeling, Construction, _____ (Deployment)
5. The essence of software engineering practice are *Understand the problem* (communication and analysis), *Plan a solution* (modeling and software design), _____ and *Examine the result for accuracy. (Carry out the plan* (code generation),*)*
6. *Software myths are _____ Practitioners Myth, Managerial Meths(Customer Myth)*
7. CMMI stands for _____ (Capability Maturity Model Integration)
8. The *waterfall model,* sometimes called the _____ (*classic life cycle)*
9. *V*ariation in the representation of the waterfall model is called the _____ (*V-model*.)
10. The *incremental* model combines elements of linear and parallel process flows
11. Spiral model is also called as _____ model (Barry Boehm model)
12. AOSD stands for _____ (Aspect Oriented Software Development)
13. PSP stands from _____(Personal Software Process)
14. _____ is a quality management technique that translates the needs of the customer into technical requirements for software.(Quality function deployment (QFD))
15. System software—a collection of programs written to service other programs.
16. Application software—stand-alone programs that solve a specific business need.
17. Embedded software—resides within a product or system and is used to implement and control features and functions for the end user and for the system itself.
18. Artificial intelligence software—makes use of nonnumerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis.
19. older programs—often referred to as legacy software
20. A process framework establishes the foundation for a complete software engineering process
21. umbrella activities that are applicable across the entire software process
22. Generic framework activities—communication, planning, modeling, construction, and deployment
23. KISS (Keep It Simple, Stupid)
24. RAD – Rappid Application Development
25. The waterfall model, sometimes called the classic life cycle
26. A variation in the representation of the waterfall model is called the V-model.
27. The Spiral Model proposed by Barry Boehm
28. The concurrent development model, sometimes called concurrent engineering,
29. Commercial off-the-shelf (COTS)
30. The formal methods model encompasses a set of activities that leads to formal mathematical specification of computer software
31. UML—a unified modeling language

32. Personal Software Process (PSP)
33. Team Software Process (TSP)
34. Agile software engineering combines a philosophy and a set of development guidelines.
35. Industrial Extreme Programming (IXP)
36. Dynamic Systems Development Method (DSDM)
37. Feature Driven Development (FDD)

UNIT II

1. The work products produced as a consequence of requirements engineering are assessed for quality during a validation step.
2. Quality function deployment (QFD) is a quality management technique that translates the needs of the customer into technical requirements for software.
3. Quality function deployment (QFD) is a quality management technique that translates the needs of the customer into technical requirements for software.
4. Each usage scenario implies a set of objects that are manipulated as an actor interacts with the system
5. Data models depict the information domain for the problem
6. Class-oriented models represents object-oriented classes (attributes and operations) and the manner in which classes collaborate to achieve system requirements
7. Flow-oriented models represents the functional elements of the system and how they transform data as it moves through the system
8. Behavioral models depict how the software behaves as a consequence of external “events”
9. Examples for Scenario-based models are use cases, user stories
10. Examples for Flow models are DFDs data models
11. Example for Behavioral models are state diagrams, sequence diagrams
12. A UML activity diagram represents the actions and decisions that occur as some function is performed.
13. A UML swim lane diagram represents the flow of actions and decisions and indicates which actors perform each.
14. Attributes describe a class
15. Operations define the behavior of an object.
16. Class-responsibility-collaborator (CRC)
17. An association defines a relationship between classes. Multiplicity defines how many of one class are related to how many of another class.
18. A package is used to assemble a collection of related classes.

UNIT III

1. The requirements model, manifested by scenario-based, class-based, flow-oriented, and behavioral elements,
2. The architectural design defines the relationship between major structural elements of the software,
3. The interface design describes how the software communicates with systems that interoperate with it, and with humans who use it.
4. The component-level design transforms structural elements of the software architecture into a procedural description of software components.

5. The design should provide a complete picture of the software, addressing the data, functional, and behavioral domains from an implementation perspective.
6. FURPS—functionality, usability, reliability, performance, and supportability
7. A procedural abstraction refers to a sequence of instructions that have a specific and limited function.
8. A data abstraction is a named collection of data that describes a data object
9. Cohesion is an indication of the relative functional strength of a module.
10. Coupling is an indication of the relative interdependence among modules.
11. Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code [design] yet improves its internal structure.
12. Software architecture must model the structure of a system
13. and the manner in which data and procedural components collaborate with one another.
14. An architectural style is a transformation that is imposed on the design of an entire system.
15. An archetype is an abstraction (similar to a class) that represents one element of system behavior.
16. ACD stands for architectural context diagram
17. Sharing dependencies represent dependence relationships among consumers who use the same resource
18. Flow dependencies represent dependence relationships between producers and consumers of resources
19. Constrained dependencies represent constraints on the relative flow of control among a set of activities.
20. Component is “a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces.”
21. ISP stands for Interface Segregation Principle.
22. Interface is the equivalent of an abstract class that provides a controlled connection between design classes.
23. Example for Tabular Design Notation is Decision tables

UNIT IV

1. User interface design creates an effective communication medium between a human and a computer
2. Knowledgeable, intermittent users. Reasonable semantic knowledge of the application but relatively low recall of syntactic information.
3. Novices. No syntactic knowledge of the system and little semantic knowledge of the application or computer usage in general.
4. The analysis and design process for user interfaces is iterative and can be represented using a spiral model
5. Nouns (objects) and verbs (actions) are isolated to create a list of objects and actions.
6. System response time is measured from the point at which the user performs some control action until the software responds with desired output or action.
7. Consistency. The use of navigation controls, menus, icons, and aesthetics (e.g., color, shape, layout) should be consistent throughout the WebApp
8. Fitt’s law. “The time to acquire a target is a function of the distance to and size of the target”
9. The web design should be moderation and simple.
10. Base for web app design pyramid is Component design
11. Apex of web app design pyramid is interface design
12. Aesthetic design, also called graphic design

13. Architecture design is conducted in parallel with interface design
14. The architectural structures can be combined to form composite structures
15. Hierarchical structures are undoubtedly the most common WebApp architecture.
16. MVC stands for Model-View-Controller architecture
17. OOHDM stands for OBJECT-ORIENTED HYPERMEDIA DESIGN METHOD
18. ADV stands for abstract data view
19. NSU stands for navigation semantic units
20. An interface that uses an interaction metaphor is easier to learn and easier to use

UNIT V

1. Software is tested to uncover errors that were made inadvertently as it was designed and constructed.
2. A Test Specification document is the work product of software testing
3. To perform effective testing, you should conduct effective technical reviews
4. Testing and debugging are different activities
5. Verification: “Are we building the product right?”
6. Validation: “Are we building the right product?”
7. ITG stands for independent test group
8. Software testing strategy may be viewed as Spiral model
9. Initially, tests focus on each component individually, ensuring that it functions properly as a unit. Hence, the test is named as unit testing.
10. System testing is the last high-order testing
11. Driver and/or stub software must often be developed for each unit test.
12. Integration testing is a systematic technique for constructing the software architecture while at the same time conducting tests to uncover errors associated with interfacing.
13. Regression testing may be conducted to ensure that new errors have not been introduced.
14. Smoke testing is an integration testing approach that is commonly used when product software is developed.
15. Functional validity. Tests designed to uncover functional errors are conducted.
16. Cluster testing is one step in the integration testing of OO software.
17. The alpha test is conducted at the developer’s site by a representative group of end users.
18. The beta test is conducted at one or more end-user sites.
19. A variation on beta testing, called customer acceptance testing,
20. Many computer-based systems must recover from faults and resume processing with little or no downtime.
21. Security testing attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration.
22. Stress testing executes a system in a manner that demands resources in abnormal quantity, frequency, or volume
23. A variation of stress testing is a technique called sensitivity testing.
24. Performance testing is designed to test the run-time performance of software within the context of an integrated system.
25. Deployment testing, sometimes called configuration testing
26. Debugging is not testing but often occurs as a consequence of testing