Course Contents:

1. Introduction

Q1. What is software and programs?

Ans.

Computer Software:

Computer software is the product that software professionals build and then support over the long term.

Software is simply:

1.Instructions

2.Data structure

3.Descriptive information

Programs:

Computer program, a set of instructions that describes how to perform a specific task to a computer.

Computer programming, the act of instructing computers to perform tasks ·

Programming language, an artificial language designed to communicate instructions to a machine ·

OBJ

OBJ

Q2. Characteristics of Software

Ans.

1. developed/engineered

2.Doesn’t wear out, but deteriorates

3.Most software is custom built

Q3. Types of Software

Ans.

S/w is the collection of computer programs, procedures and documentation that performs different tasks on a computer system.

The term ”software” first used by John Tuckey in 1958.

Three Broad Classifications:

1. Application Software

It enables the end users to accomplish certain specific tasks.

Includes programs such as web browsers, office software, games and programming tools.

2. System Software

It helps in running computer hardware and the computer system.

System software refers to the operating systems; device drivers, servers, windowing, systems and utilities(e.g. antivirus software, firewalls, defragmenters)

3. Malicious Software/Malware

Refers to any malicious software to harm and disrupt computers.

E.g Adware, computer viruses, worms, Trojan horses and scareware.

OBJ

OBJ

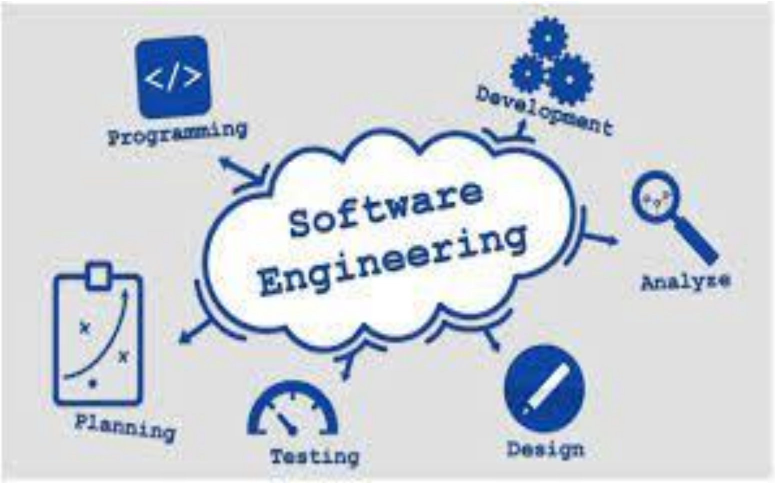
Q4. Define Software Engineering?

Ans.

Software engineering is the systematic application of engineering approaches to the development of software.

Software engineering is the application of engineering concepts to software development.

Its main goal is the creation, improvement, and maintenance of software.



Q5. The Software Process Framework:

Ans.

Process is a collection of activities, actions and tasks that are performed when some work product is to be created.

A process framework establishes the foundation for a complete software engineering process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size or complexity.

In addition, the process framework encompasses a set of umbrella activities that are applicable across the entire software process.

Q6. What are the activities of the Software Process Framework?

Ans.

A generic process framework for s/w engg ecompasses five activities:

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

Q7. What are included in umbrella activities?

Ans.

- Software project tracking and control
- Risk management
- Software quality assurance
- Technical reviews
- Measurement
- Software Configuration management
- Reusability management
- Work product preparation and production

Q8. What do yu understand by the Generic Process Model?

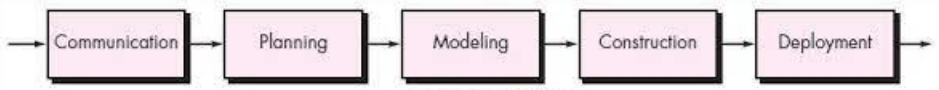
Ans.

Basically, a generic process framework for s/w engg defines five framework activities:- communication, planning, modeling, construction and deployement.

Q9. Linear Process Flow

Ans.

A linear process flow executes each of the five framework activities in sequence, beginning with communication and culminating with deployment.



Linear process flow

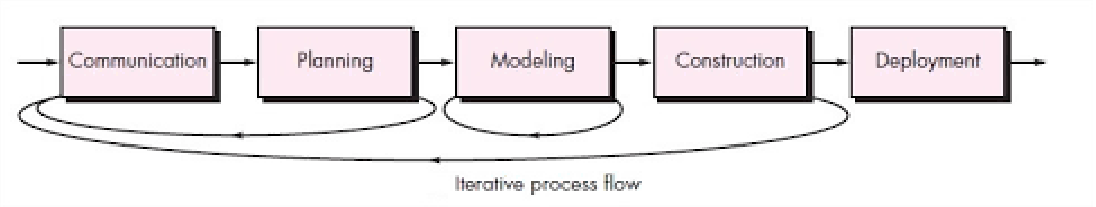
OBJ

Q10. What concept do you have on Iterative Process Flow?

Ans.

Iterative Process Flow

An iterative process flow repeats one or more of the activities before proceeding to the next.



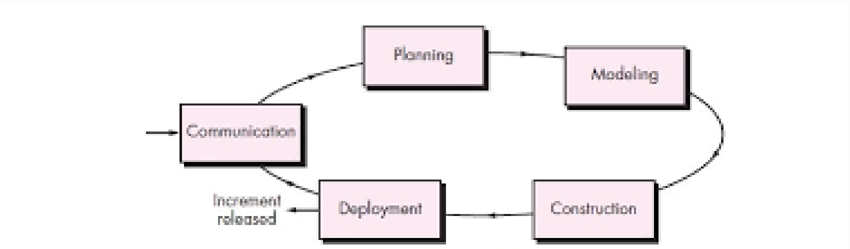
Iterative process flow

Q11. Evolutionary Process Flow

Ans.

An evolutionary process flow executes the activities in a “circular” manner.

Each circuit through the five activities leads to a more complete version of the software.



Evolutionary process flow

OBJ

OBJ

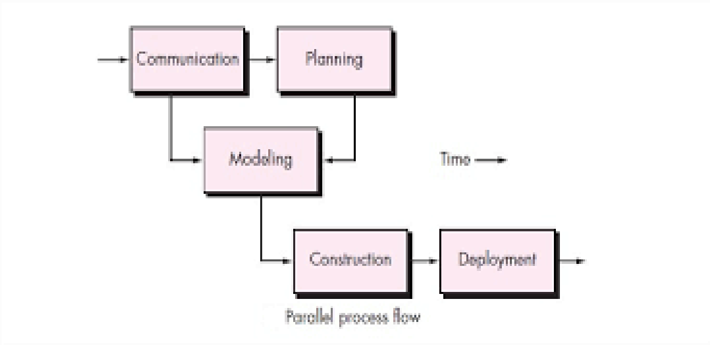
Q12. Parallel Process Flow

Ans.

Parallel Process Flow

A parallel process flow executes one or more activities in parallel with other activities.

(e.g., modeling for one aspect of the software might be executed in parallel with construction of another aspect of the software).



Parallel process flow

Q13. What do you mean by Perspective Process Models?

Ans.

The software development life cycle (SDLC) illustrates the general prescriptive process models for the development of software product.

The generic types of software process models are:

- Linear Sequential Model (Waterfall Model)
- Prototyping Model
- RAD Model
- Evolutionary Model
- Incremental Model
- Spiral Model

OBJ

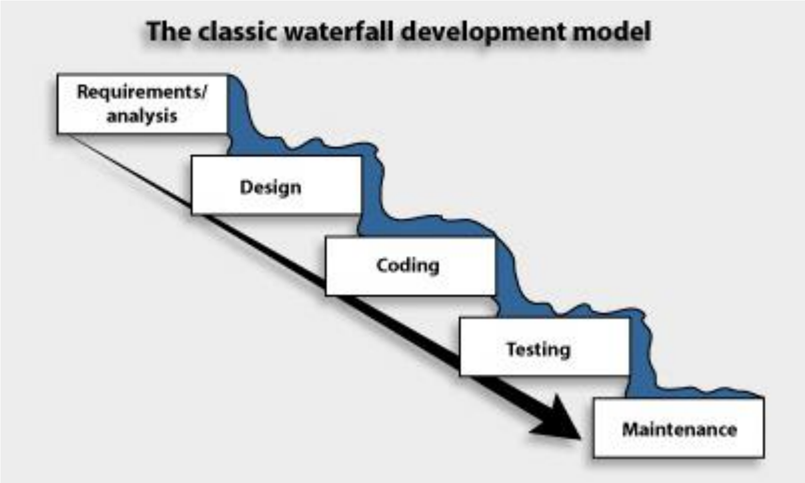
OBJ

Q14. What do you mean by Waterfall Model?

Ans.

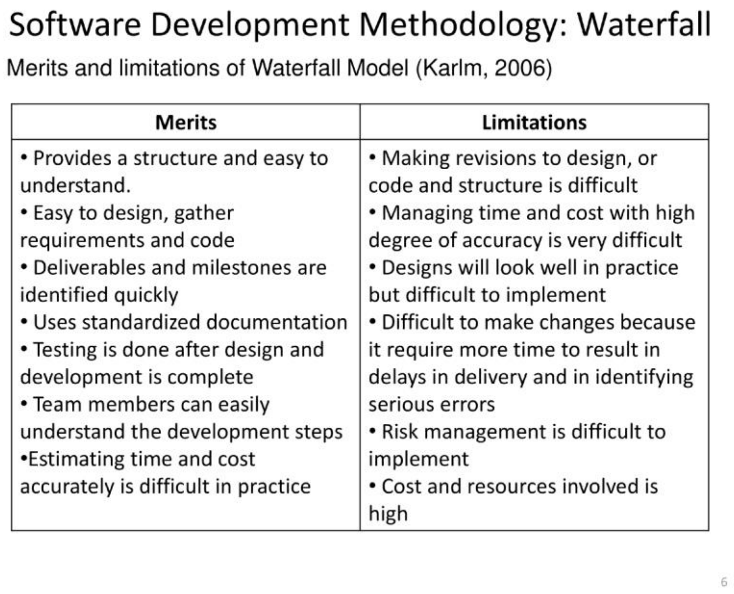
The waterfall model is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one and corresponds to a specialization of tasks.

The approach is typical for certain areas of engineering design.



Q15. Advantages and disadvantages of Waterfall Model

Ans.



Software Development Methodology: Waterfall

Merits and limitations of Waterfall Model (Karlm, 2006)

Merits	Limitations
• Provides a structure and easy to understand. • Easy to design, gather requirements and code • Deliverables and milestones are identified quickly • Uses standardized documentation • Testing is done after design and development is complete • Team members can easily understand the development steps •Estimating time and cost accurately is difficult in practice	• Making revisions to design, or code and structure is difficult • Managing time and cost with high degree of accuracy is very difficult • Designs will look well in practice but difficult to implement • Difficult to make changes because it require more time to result in delays in delivery and in identifying serious errors • Risk management is difficult to implement • Cost and resources involved is high

6

Q16. When to use Waterfall Model

Ans.

The linear process flow model (i.e., waterfall model) will be appropriate when:

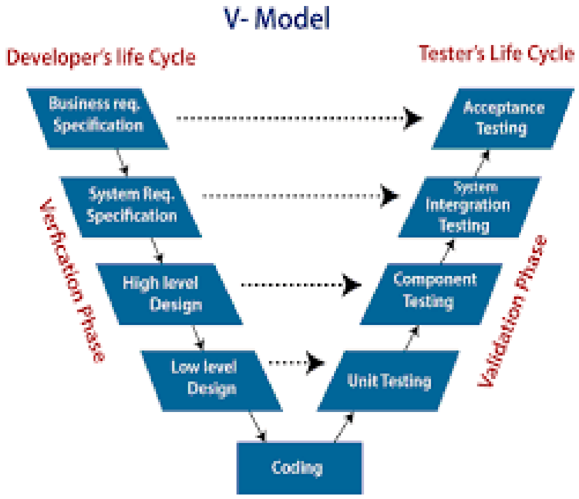
- User requirements are clearly defined
- Less no. Of staffing to do simultaneous work on that project
- No much re-usable components
- Sufficient time for s/w development
- Less communication between customer and developer
- S/w engineers have clear ideas to do the prescribed work

Q17. Define V Model

Ans.

A variation in the representation of the waterfall model is called the V-model.

It provides a way of visualizing how verification & validation are applied to earlier engineering work.



Q18. RAD Model

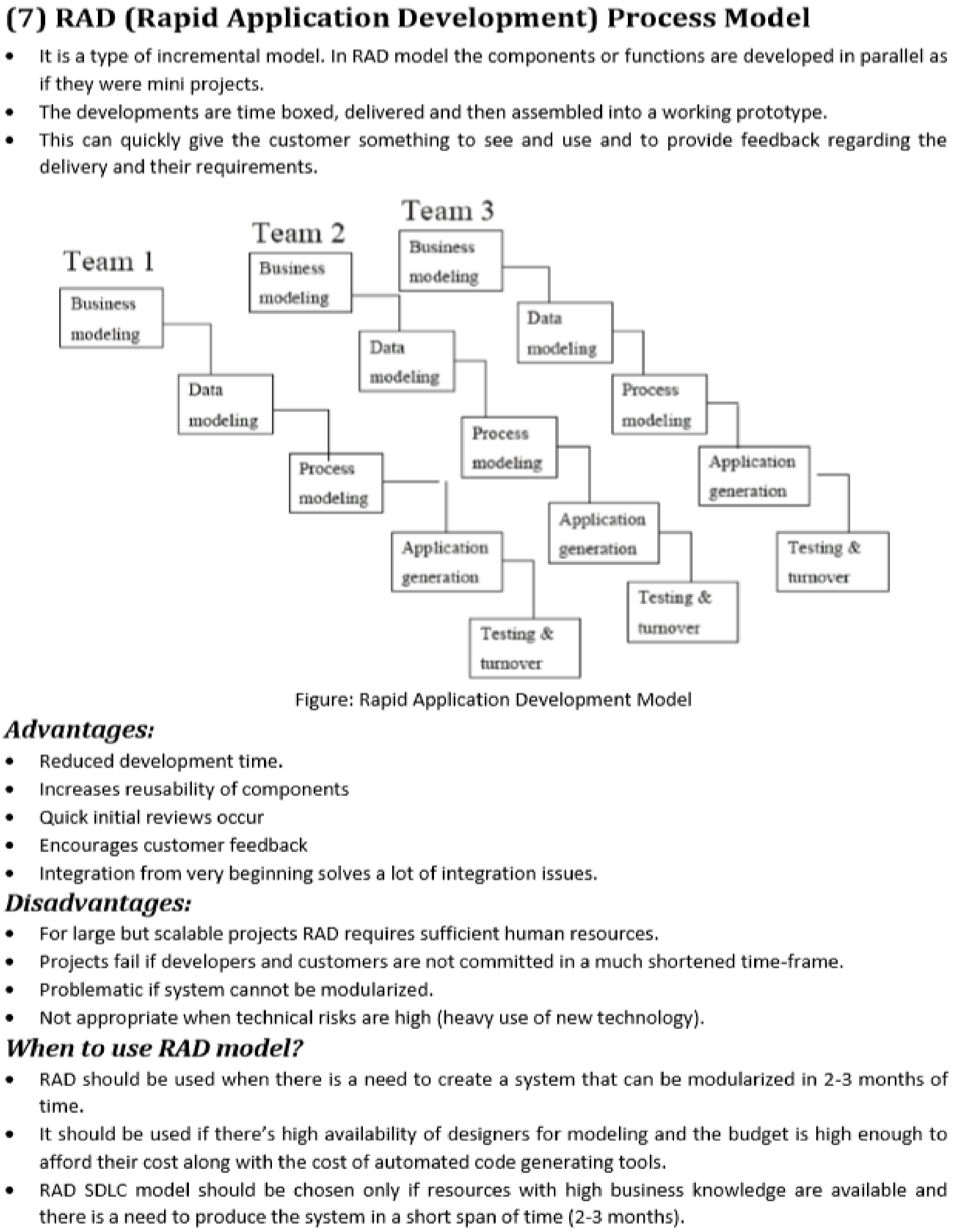
Ans.

Rapid Application Development (RAD) Model

The incremental model or Rapid application development (RAD) model combines elements of linear and parallel process flows discussed.

Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.

If the core product is well received, then additional staff (if required) can be added to implement the next increment.



(7) RAD (Rapid Application Development) Process Model

- It is a type of incremental model. In RAD model the components or functions are developed in parallel as if they were mini projects.
- The developments are time boxed, delivered and then assembled into a working prototype.
- This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

Figure: Rapid Application Development Model

Advantages:

- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.

Disadvantages:

- For large but scalable projects RAD requires sufficient human resources.
- Projects fail if developers and customers are not committed in a much shortened time-frame.
- Problematic if system cannot be modularized.
- Not appropriate when technical risks are high (heavy use of new technology).

When to use RAD model?

- RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
- It should be used if there's high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
- RAD SDLC model should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

Q19. Evolutionary Process Models???

Ans.

Evolutionary models are iterative which is much suitable for new systems where no clear idea of the requirements, inputs and outputs parameters.

It produces an increasingly more complete version of the software with each iteration.

When to use Evolutionary Process Models

The evolutionary process models are appropriate in following situation:

- No clear idea about the product for both customers and developers.
- Good communication between the customers and developers.
- Sufficient time for the software development
- Less chances of outsourcing and availability of re-usable components.

Q20. what are the types of Evolutionary Process Models???

Ans.

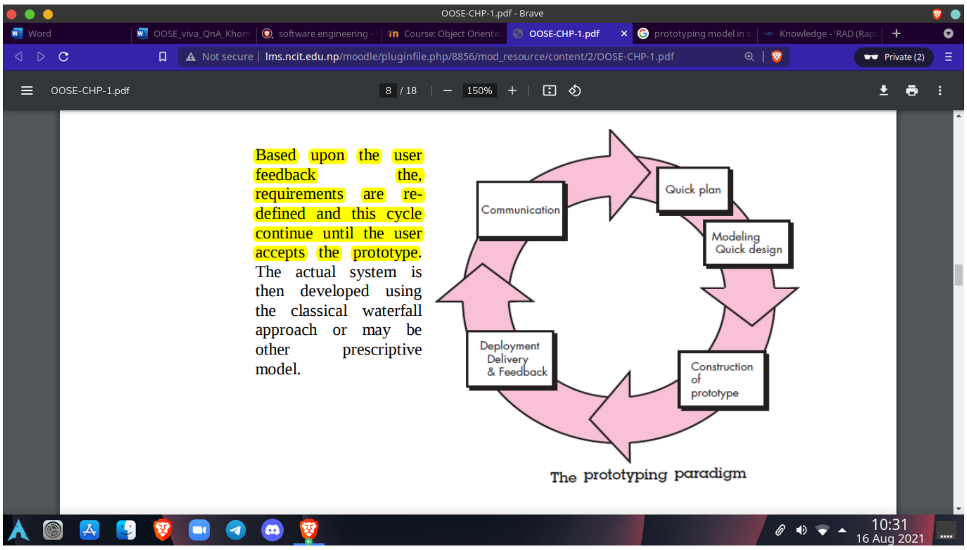
Two common types of Evolutionary Process Models are :

1. Prototyping
2. The Spiral Model

i. Prototyping

Although prototyping can be used as a stand-alone process model, it assists you and other stakeholders to better understand what is to be built when requirements are fuzzy.

Based upon the user feedback the, requirements are redefined and this cycle continue until the user accepts the prototype.



Based upon the user feedback the, requirements are re-defined and this cycle continue until the user accepts the prototype. The actual system is then developed using the classical waterfall approach or may be other prescriptive model.

The prototyping paradigm

ii. The Spiral Model:

The spiral model was originally proposed by Barry Boehm, is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.

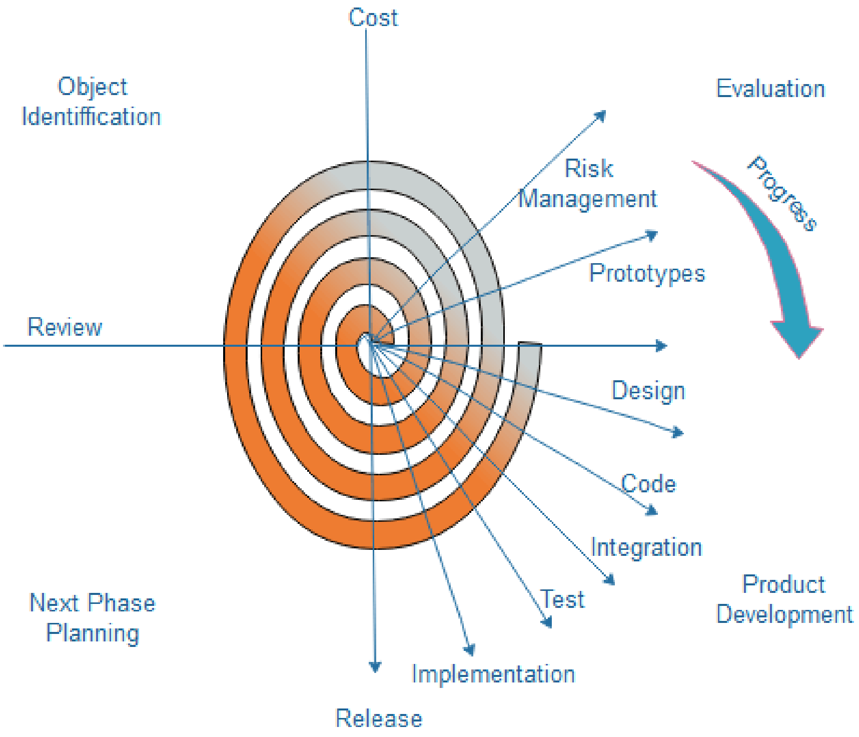


Fig. Spiral Model

The spiral model is a realistic approach to the development of large-scale systems and software.

Developer and customer better understand and react to risks.

Q21. What do you mean by Specialized Process Models?

Ans.

Applied when a specialized or narrowly defined software engineering approach is chosen.

a. **Component-Based Development**

A component is a modular building block for computer software.

The component-based development model incorporates many of the characteristics of the spiral model.

The component-based development model leads to software reuse, and reusability provides software engineers with a number of measurable benefits.

b. **The Formal Methods Model**

The formal methods model providing a mathematically based approach to program modeling and the ability to verify that the model is correct.

Formal methods eliminate many of the problems such as ambiguity, incompleteness, and inconsistency that are difficult to overcome using other software engineering paradigms.

A variation on this approach, called cleanroom software engineering is currently applied by some software development organizations.

c. **Aspect-Oriented Software Development**

Aspect-oriented software development (AOSD), often referred to as aspect-oriented programming (AOP), is a relatively new software engineering paradigm that provides a process and methodological approach for defining, specifying, designing, and constructing aspects.

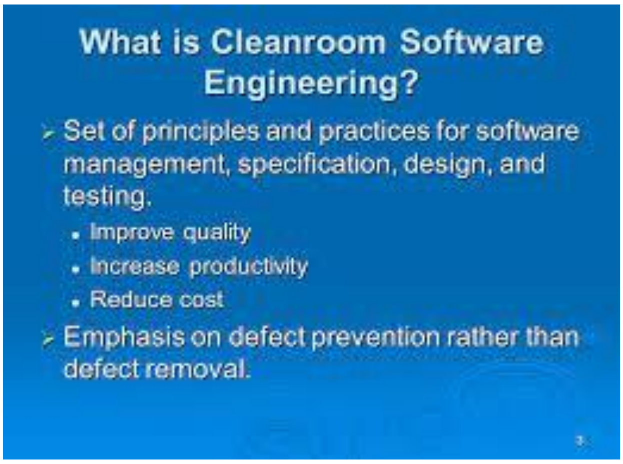
The aspect is customer requirements solely defined for their software system.

d. **Cleanroom Software Engineering**

Cleanroom Software Engineering is a set of principles and practices for software management, specification, design, and testing.

Q22. Define Cleanroom Engineering

Ans.



What is Cleanroom Software Engineering?

- Set of principles and practices for software management, specification, design, and testing.
  - Improve quality
  - Increase productivity
  - Reduce cost
- Emphasis on defect prevention rather than defect removal.

Q23. Formal vs Cleanroom s/w engineering

Ans.

Q24. Define Unified Process and UML.

Ans.

Unified Process:

Jacobson, Rumbaugh, and Booch developed the Unified Process, a framework for object-oriented software engineering using UML.

The phases of unified process are similar to the generic framework activities as shown in figure.

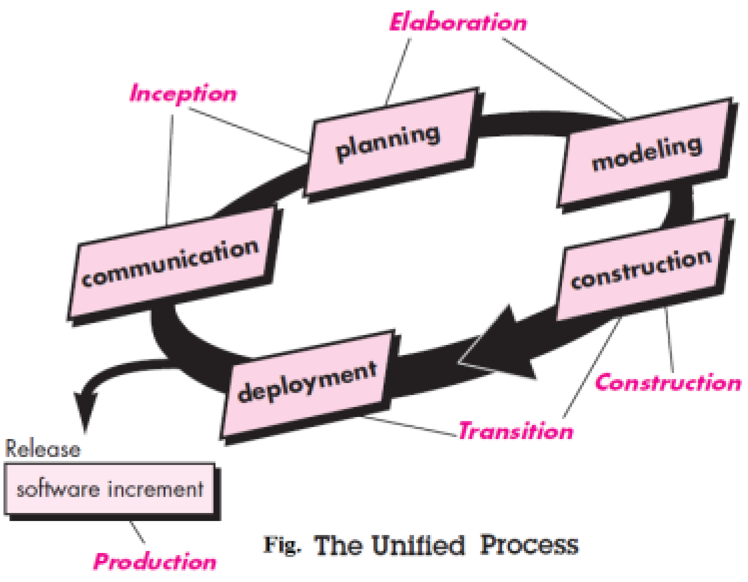


Fig. The Unified Process

The Unified Modeling Language (UML) is a standard language for writing software blueprints.

The UML may be used to visualize, specify, construct, and document the artifacts of a software intensive system, as well as for business modeling and other non-software systems.

The UML is called a modeling language, not a method.

- **Unified:** Combines main preceding OO methods (Booch by Grady Booch, OMT by James Rumbaugh and OOSE by Ivar Jacobson).

- **Modeling:** Primarily used for visually modeling systems. Many system views are supported by appropriate model.

- **Language:** It offers syntax through which to express modeled knowledge.

Q25. What is UML used for?

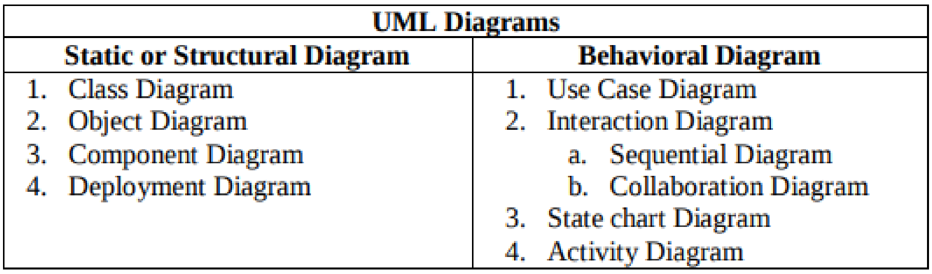
Ans.

UML is a language for:

- ✓ Visualizing
- ✓ Specifying
- ✓ Constructing
- ✓ Documenting

Q26. UML Diagrams

Ans.



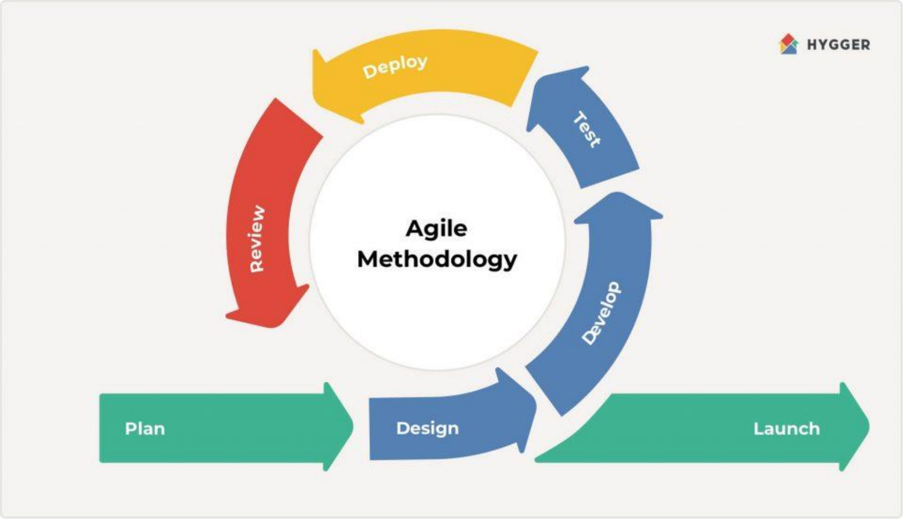
UML Diagrams	
Static or Structural Diagram	**Behavioral Diagram**
1. Class Diagram	1. Use Case Diagram
2. Object Diagram	2. Interaction Diagram
3. Component Diagram	a. Sequential Diagram
4. Deployment Diagram	b. Collaboration Diagram
	3. State chart Diagram
	4. Activity Diagram

Q27. What do you mean by Agile Development?

Ans.

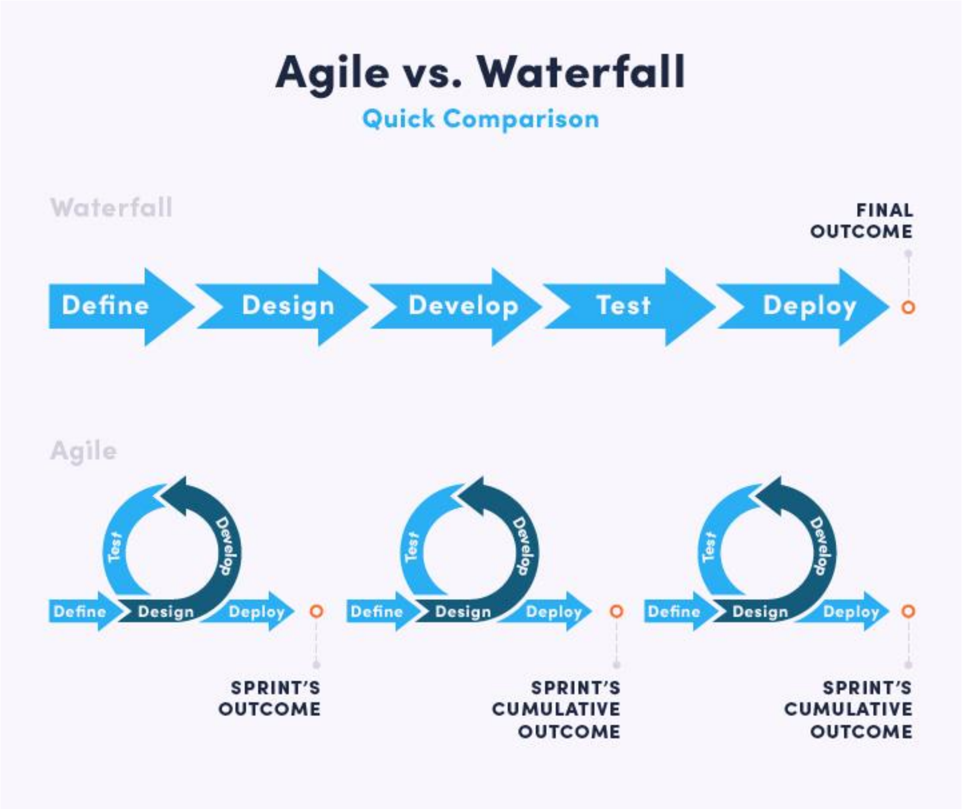
Agile is a term used to describe software development approaches that employ continual planning, learning, improvement, team collaboration, evolutionary development, and early delivery.

It encourages flexible responses to change. The agile software development emphasizes on four core values.



Q28. Agile vs Waterfall

Ans.



Q29. Agile Process

Ans.

Agile Processes

- Are based on three key assumptions
  a. It is difficult to predict in advance which requirements or customer priorities will change and which will not
  b. For many types of software design and construction activities are interleaved (construction is used to prove the design)
  c. Analysis, design, and testing are not as predictable from a planning perspective as one might like them to be

Q30. Agility Principles

Ans.

Q31. Can you enlist the types of Agile Process Models?

Ans.

- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Scrum
- Dynamic Systems Development Method (DSDM)
- Crystal
- Feature Driven Development (FDD)
- Lean Software Development (LSD)
- Agile Modeling (AM) and
- Agile Unified Process (AUP)

Extreme Programming:

- Relies on object-oriented approach
- Key activities:
  - Planning (user stories created and ordered by customer value)
  - Design (simple designs preferred, CRC cards and design prototypes are only work products, encourages use of refactoring)
  - Coding (focuses on unit tests to exercise stories, emphasizes use of pairs programming to create story code, continuous integration and smoke testing is utilized)
  - Testing (unit tests created before coding are implemented using an automated testing framework to encourage use of regression testing, integration and validation testing done on daily basis, acceptance tests focus on system features and functions viewable by the customer)

Adaptive Software Development

- Self-organization arises when independent agents cooperate to create a solution to a problem that is beyond the capability of any individual agent
- Emphasizes self-organizing teams, interpersonal collaboration, and both individual and team learning
- Adaptive cycle characteristics
- Phases
  - Mission-driven
  - Component-based
  - Iterative
  - Time-boxed
  - Risk driven and change-tolerant
  - Speculation (project initiated and adaptive cycle planning takes place)
  - Collaboration (requires teamwork from a jelled team, joint application development is preferred requirements gathering approach, mini specs created)
  - Learning (components implemented and testes, focus groups provide feedback, formal technical reviews, postmortems)

Scrum

- Scrum principles
  - Small working teamed used to maximize communication, minimize overhead, and maximize sharing of informal knowledge
  - Process must be adaptable to both technical and business challenges to ensure bets product produced
  - Process yields frequent increments that can be inspected, adjusted, tested, documented and built on
  - Development work and people performing it are partitioned into clean, low coupling partitions
  - Testing and documentation is performed as the product is built
  - Provides the ability to declare the product done whenever required
- Process patterns defining development activities
  - Backlog (prioritized list of requirements or features the provide business value to customer, items can be added at any time)
  - Sprints (work units required to achieve one of the backlog items, must fit into a predefined time-box, affected backlog items frozen)
  - Scrum meetings (15-minute daily meetings) addressing these questions: What was done since last meeting? What obstacles were encountered? What will be done by the next meeting?
  - Demos (deliver software increment to customer for evaluation)

Dynamic Systems Development Method

- Provides a framework for building and maintaining systems which meet tight time constraints using incremental prototyping in a controlled environment
- Uses Pareto principle (80% of project can be delivered in 20% required to deliver the entire project)
- Each increment only delivers enough functionality to move to the next increment
- Uses time boxes to fix time and resources to determine how much functionality will be delivered in each increment
- Guiding principles
  - Active user involvement
  - Teams empowered to make decisions
  - Fitness foe business purpose is criterion for deliverable acceptance

    - Iterative and incremental develop needed to converge on accurate business solution
    - All changes made during development are reversible
    - Requirements are baselined at a high level
    - Testing integrates throughout life-cycle
    - Collaborative and cooperative approach between stakeholders
- Life cycle activities
    - Feasibility study (establishes requirements and constraints)
    - Business study (establishes functional and information requirements needed to provide business value)
    - Functional model iteration (produces set of incremental prototypes to demonstrate functionality to customer)
    - Design and build iteration (revisits prototypes to ensure they provide business value for end users, may occur concurrently with functional model iteration)
    - Implementation (latest iteration placed in operational environment)

Crystal

- Development approach that puts a premium on maneuverability during a resource-limited game of invention and communication with the primary goal of delivering useful software and a secondary goal of setting up for the next game
- Crystal principles
    - It's always cheaper and faster to communicate face-to-face
    - As methodologies become more formal teams become weighed down and have trouble adapting to project work vagaries
    - As projects grow in size, teams become larger and methodologies become heavier
    - As projects grow in criticality some degree of formality will need to be introduced in parts of the methodology
    - As feedback and communication become more efficient the need for intermediate work products is reduced
    - Discipline, skills, and understanding counter process, formality, and documentation
    - Team members not on the critical project path can spend their excess time improving the product or helping people who are on the critical path
- Incremental development strategy used with 1-to-3-month time lines
- Reflection workshops conducted before project begins, during increment development activity, and after increment is delivered
- Crystal methodologies
    - Clear (small, low criticality projects)
    - Orange (larger, moderately critical projects)
    - Orange Web (typical e-business applications)

Feature Driven Development

- Practical process model for object-oriented software engineering
- Feature is a client-valued function, can be implemented in two weeks or less
- FDD Philosophy
  - Emphasizes collaboration among team members
  - Manages problem and project complexity using feature-based decomposition followed integration of software increments
  - Technical communication using verbal, graphical, and textual means
  - Software quality encouraged by using incremental development, design and code inspections, SQA audits, metric collection, and use of patterns (analysis, design, construction)
- Framework activities
  - Develop overall model (contains set of classes depicting business model of application to be built)
  - Build features list (features extracted from domain model, features are categorized and prioritized, work is broken up into two-week chunks)
  - Plan by feature (features assessed based on priority, effort, technical issues, schedule dependencies)
  - Design by feature (classes relevant to feature are chosen, class and method prologs are written, preliminary design detail developed, owner assigned to each class, owner responsible for maintaining design document for his or her own work packages)
  - Build by feature (class owner translates design into source code and performs unit testing, integration performed by chief programmer)

Agile Modeling

- Practice-based methodology for effective modeling and documentation of software systems in a light-weight manner
- Modeling principles
  - Model with a purpose
  - Use multiple models
  - Travel light (only keep models with long-term value)
  - Content is more important than representation
  - Know the models and tools you use to create them
  - Adapt locally
- Requirements gathering and analysis modeling
  - Work collaboratively to find out what customer wants to do
  - Once requirements model is built collaborative analysis modeling continues with the customer
- Architectural modeling
  - Derives preliminary architecture from the analysis model

- Architectural model must be realistic for the environment and must be understandable by developers

Q32. Define the terms: Backlog, sprints, scrum meetings, Demos

Ans.

See above Q31. Scrum

2. Planning Software Projects

Q1. What do you understand by Project? How can you manage your project?

Ans.

Q2. Tell me about W5HH principle.

Q3. What do you mean by Project Planning Process?

Q4. Define software scope and feasibility? How can you identify whether your project scope is good and how can you make it feasible?

Q5. Define resources.

Q6. What do you know about Software Project Estimation?

Q7. Define outsourcing. Have you ever done outsourcing for your project? If yes, when and how?

Q8. Explain project scheduling and time line charts.

Q9. How can you elaborate Risk management? What are the possible software risks?

Q10. Tell something about software maintenance.

3. Software Modelling

Q1. Briefly describe about software engineering principles and practice?

Ans.

Q2. What do you understand by Requirement Engineering and Requirement Elicitation?

Q3. How can you define Requirement Gathering and Quality Function Deployment?

Q4. Developing Requirement Model

Q5. Can you explain what Requirement Validation refers to?

Q6. What is Requirement Analysis? What are the things included in it?

Q7. Tell me something about Scenario-based Modeling.

Q8. Use case diagram, activity diagram??

Q9. What are the different types of modeling in software engineering?

Q10. Is behavioral modeling dynamic?

Q11. UML is the building block of

4. Quality Management and Testing

Q1.

Q2.

5. Advanced Topics in Software Engineering

Q1.

Q2.

Viva questions asked for me

Q1. What is software process models? Tell me its types

Ans.

Software process model is an abstraction of the actual process, which is being described.

It can also be taken as a simplified representation of a software process . …

Types of software process models include

1. Waterfall model
2. V model
3. Incremental model
4. RAD model
5. Iterative model
6. Prototype model
7. Spiral model

Q2. Class diagram vs object diagram? Are they same?

Ans.

No

Class diagram is the main building block of object-oriented modeling.

It’s simply the collection of classes.

Object diagrams provide a snapshot of the instances in a system.

Object is an instance of classifiers in models.

Q3. What do you mean by FP? Tell me its attributes/parameters?

Ans.

FP(Function Points) are a unit of measure to define the value that the end user derives.

Types of FP attributes include:

1. No. Of External Inputs
2. No. Of External outputs
3. No. Of External Output
4. No. Of internal files
5. No. Of external interfaces

Q4. Do you know Coupling and Cohesion?

Ans.

Coupling:

Coupling is the measure of the degree of interdependence/relationships between the modules.

A good sotware has low coupling.

Cohesion:

Cohesion is a measure of the degree of relationships within the module.

A good software has high cohesion.

Q5. What do you understand by SPI?

Ans.

SPI is defined as a sequence of tasks, tools, and techniques to plan and implement improvement activities to achieve specific goals.

It mainly aims at continual improvement.

Q6. Tell me about SQA?

Ans.

SQA stands for Software Quality Assurance, is simply a way to assure quality in the software.

SQA is a process which works parallel to development of a software.

Major Software Quality Assurance activities:

1. SQA management plan
2. Set the check points

3. Multi testing strategy
4. Measure change impact
5. Manage good relations

Maile janeko teti ho sir

Q7. Any idea about Prototyping and Waterfall model.

Ans.

Prototype model: is a software development model in which protoytpe is built, tested and reworked … It's used especially when the project requirements aren't known'

Waterfall model:

Simply it's linear-sequential life cycle model. Used when the project requirements are known.

Q8. In what condition waterfall model is used?

Ans.

Appropriate when

1. Requirements are very well known
2. Product definition is stable.
3. The projec is short

Q9. Do you have any idea about CMMI?

Ans.

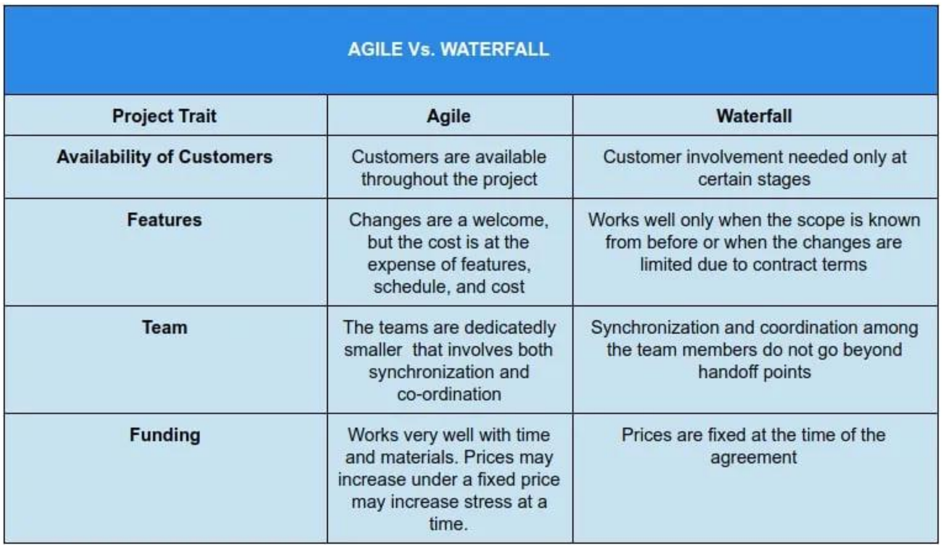
CMMI- Capability Maturity Model Integration is a process and behavioral model that helps organizations streamline process improvement and encourage productive, efficient behaviors that decrease risks in software, product, and service development.

CMMI has 5 levels

1. Level-1 => Initial
2. Level-2 => Managed
3. Level-3 => Defined
4. Level-4 => Quantitatively managed
5. Level-5 => Optimized

Q10. Can you differentiate agile software development and waterfall model?

Ans.



AGILE Vs. WATERFALL

Project Trait	Agile	Waterfall
Availability of Customers	Customers are available throughout the project	Customer involvement needed only at certain stages
Features	Changes are a welcome, but the cost is at the expense of features, schedule, and cost	Works well only when the scope is known from before or when the changes are limited due to contract terms
Team	The teams are dedicatedly smaller that involves both synchronization and co-ordination	Synchronization and coordination among the team members do not go beyond handoff points
Funding	Works very well with time and materials. Prices may increase under a fixed price may increase stress at a time.	Prices are fixed at the time of the agreement

Q11.

Q12.

Q13.

Q14.

Q15.

More to explore

1. **4 P’s**

The Four P's in software engineering is

The People - People of a project includes from manager to developer, from client to finish user.

The Process - A package method provides the framework from that a comprehensive arrange for package development is established.

The Product - Product is any package that needs to be developed.

The Project - The project contains all and everything of the entire development method and to avoid project failure the manager needs to take some steps, needs to fret concerning some common warnings etc.

1. **Verification building model right**
2. **Validation building right model**

What Is a Make-or-Buy Decision?

A make-or-buy decision is an act of choosing between manufacturing a product in-house or purchasing it from an external supplier.

Make-or-buy decisions, like outsourcing decisions, speak to a comparison of the costs and advantages of producing in-house versus buying it elsewhere.

Lean software development is a translation of lean manufacturing principles and practices to the software development domain.

Formal Technical Review (FTR) is a software quality control activity performed by software engineers. Objectives of formal technical review (FTR):