DHANALAKSHMI SRINIVASAN INSTITUTE OF RESEARCH AND TECHNOLOGY

SIRUVACHUR, PERAMBALUR – 611 113

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS 2301 - SOFTWARE ENGINEERING

QUESTION BANK

SIXTEEN MARKS

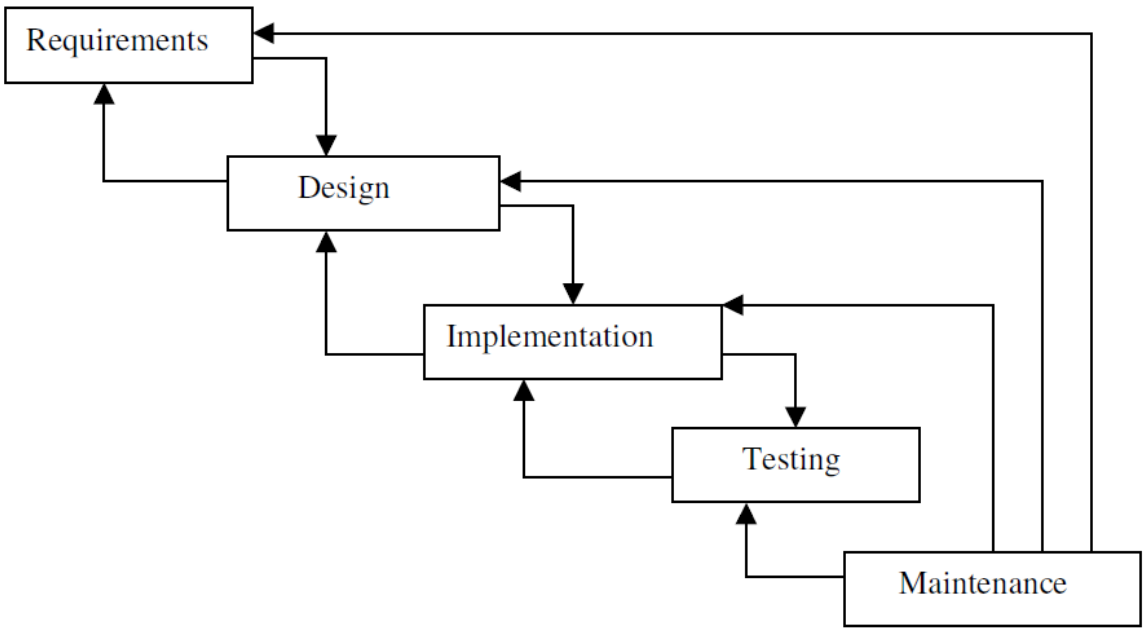
UNIT-1

1. Explain iterative waterfall and spiral model for software life cycle and various activities in each phase.

Answer: Iterative waterfall model

- Requirement gathering phase in which all requirements are identified.
- The design phase is responsible for creating architectural view of the software.
- The implementation phase in which the software design is transformed into coding.
- Testing is a kind of phase in which the developed software component is fully tested.
- Maintenance is an activity by which the software product can be maintained.

The iterative waterfall model is as shown in the following figure:



CSE/DSIRT

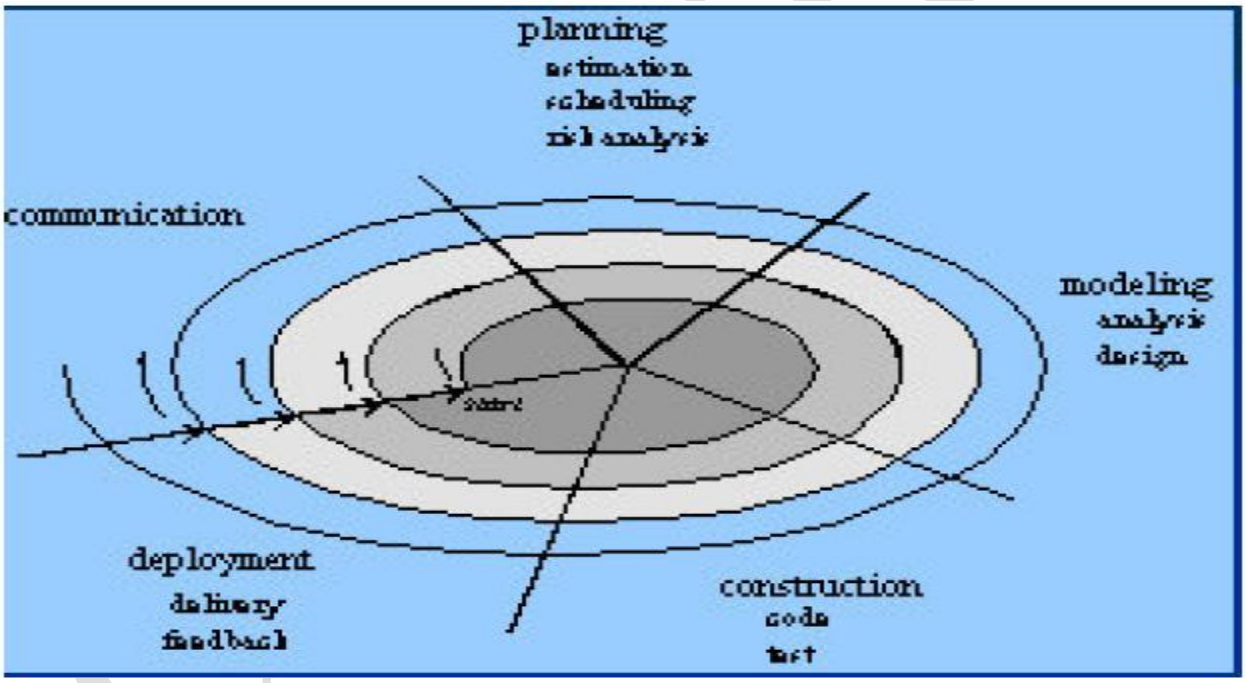
SPIRAL MODEL

- The spiral model is divided into number of frame works. These frameworks are denoted by task regions.
- Usually there are six task regions. In spiral model project entry point axis is defined.
- The task regions are:

Customer communication
Planning
Risk analysis.
Engineering.
Construct and release.
Customer evaluation.

Drawbacks

- It is based on customer communication.
- It demands considerable risk assessment**.**



2. Explain about the incremental model.

- Have same phases as the waterfall model.
- Phases are

Analysis.
Design.
Code.
Test.

- Incremental model delivers series of releases to customers called as increments.

- The first increment is called as core product. Here only the document processing facilities are available.
- Second increment, more sophisticated document producing and processing facilities are available.
- Next increment spelling and grammar checking facilities are given.

Merits

- This model can be adopted when there is less number of people involved in the project.
- Technical risks can be managed with each increment.
- For a very small time span, at least core product can be delivered to the customer.

RAD Model

- Rapid Application Development Model is the type of incremental model.
- Achieves the high speed development using component based construction.

Phases

- Business modeling
- Data modeling
- Process modeling
- Application generation.
- Testing and turnover**.**

3. Explain in detail about the software process.

- It is defined as the structured set of activities that are required to develop the software system.

Fundamental activities

- Specification
- Design and implementation
- Validation
- Evolution

Common Process Framework

- **Process framework activities**

Communication
Planning
Modeling
Construction
Deployment.

- **Task Sets**

CSE/DSIRT

Defines the actual work to achieve the software objective.

- **Umbrella activities**

Software project tracking and control
Risk management
Software quality assurance
Formal technical reviews
Software configuration management
Work product preparation and production
Reusability management.
Measurement.

Capability Maturity Model(CMM)
Level 1:Initial – Few processes are defined and individual efforts are taken.
Level 2:Repeatable – To track cost schedule and functionality basic project management processes are established.
Level 3:Defined – The process is standardized, documented and followed.
Level 4:Managed – Both the software process and product are quantitatively understood and controlled using detailed measures.
Level 5:Optimizing – Establish mechanisms to plan and implement change.

4. Explain in detail about the life cycle process.

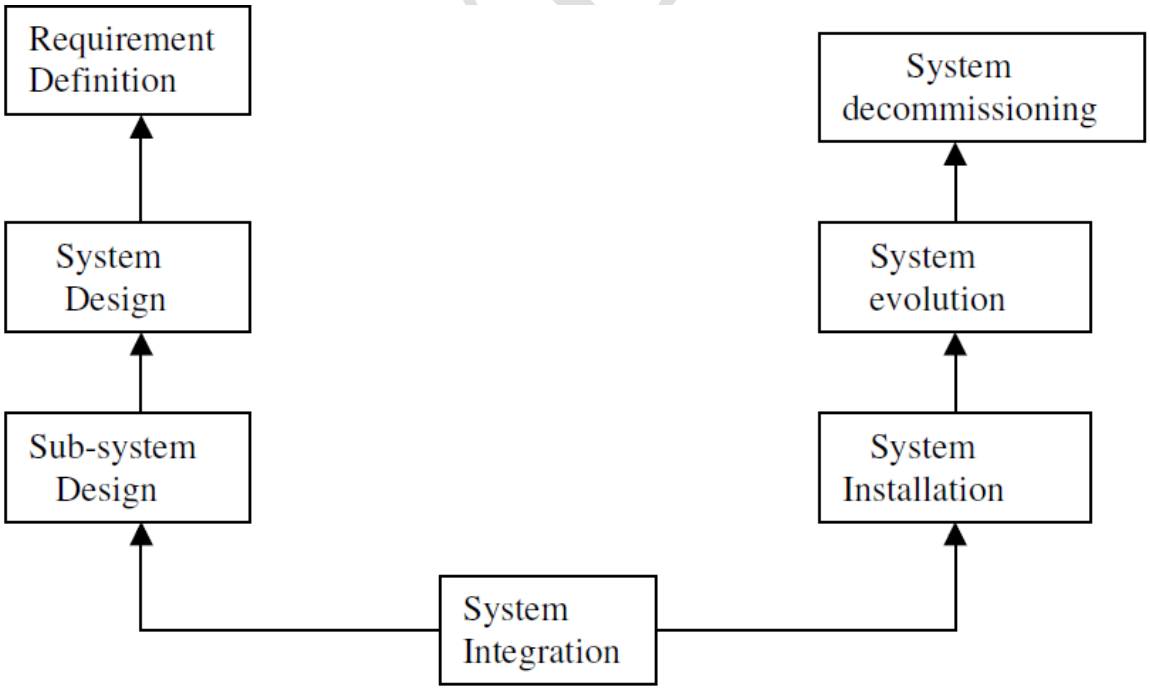


Fig: System engineering process

- System engineering process follows a waterfall model for the

parallel development of different parts of the system.

System requirements definition

- Three types of requirements

Abstract functional requirements.
System properties.
Undesirable Characteristics.

- System objectives
- System requirement problem.

The system design process

Process steps

- Partition requirements
- Identify sub-systems.
- Assign requirements to sub-systems.
- Specify sub-system functionality.
- Define sub-system interfaces.

Requirement Definition
System Design
Sub-system Design
System Integration
System decommissioning
System evolution
System Installation

Sub-System development process

- After system design it starts.
- Involve use of COTS (Commercial-Off-The-Shelf).

System Integration

- It is the process of putting hardware, software and people together to make a system.

System Installation

Issues are

- Environmental assumptions may be incorrect.
- There may be human resistance to the introduction of anew system.
- System may have to coexist with alternative systems for some period.
- There may arise some physical installation problems (e.g. cabling problem).
- Operator training has to be identified.

System evolution

- The lifetime of large systems is too long. They must evolve to meet change requirements.

CSE/DSIRT

- The evolution may be costly.
- Existing systems that must be maintained are sometimes called as legacy systems.

System Decommissioning

- Taking the system out of service after its useful lifetime is called as System Decommissioning.

5. Write short notes on Business Process Engineering overview and Product Engineering overview?

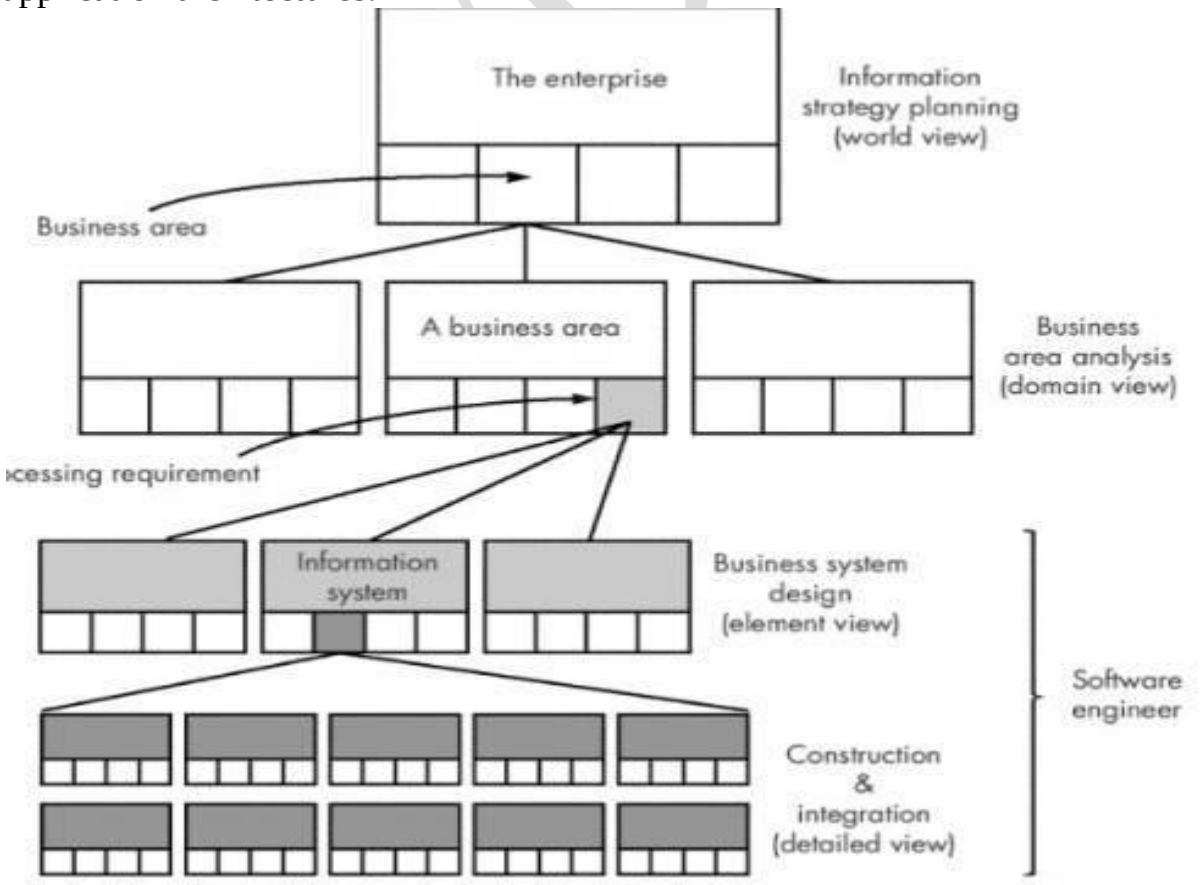
BUSINESS PROCESS ENGINEERING:

"Business process" engineering defines architectures that will enable a business to use information effectively. It involves the specification of the appropriate computing architecture and the development of the software architecture for the organization's computing resources. There are three different architectures must be analyzed and designed within the context of business objectives and goals,

✓ The data architecture provides a framework for the information needs of a business (e.g., ERD)

✓ The application architecture encompasses those elements of a system that transform objects within the data architecture for some business purpose

✓ The technology infrastructure provides the foundation for the data and application architectures.



PRODUCT ENGINEERING:

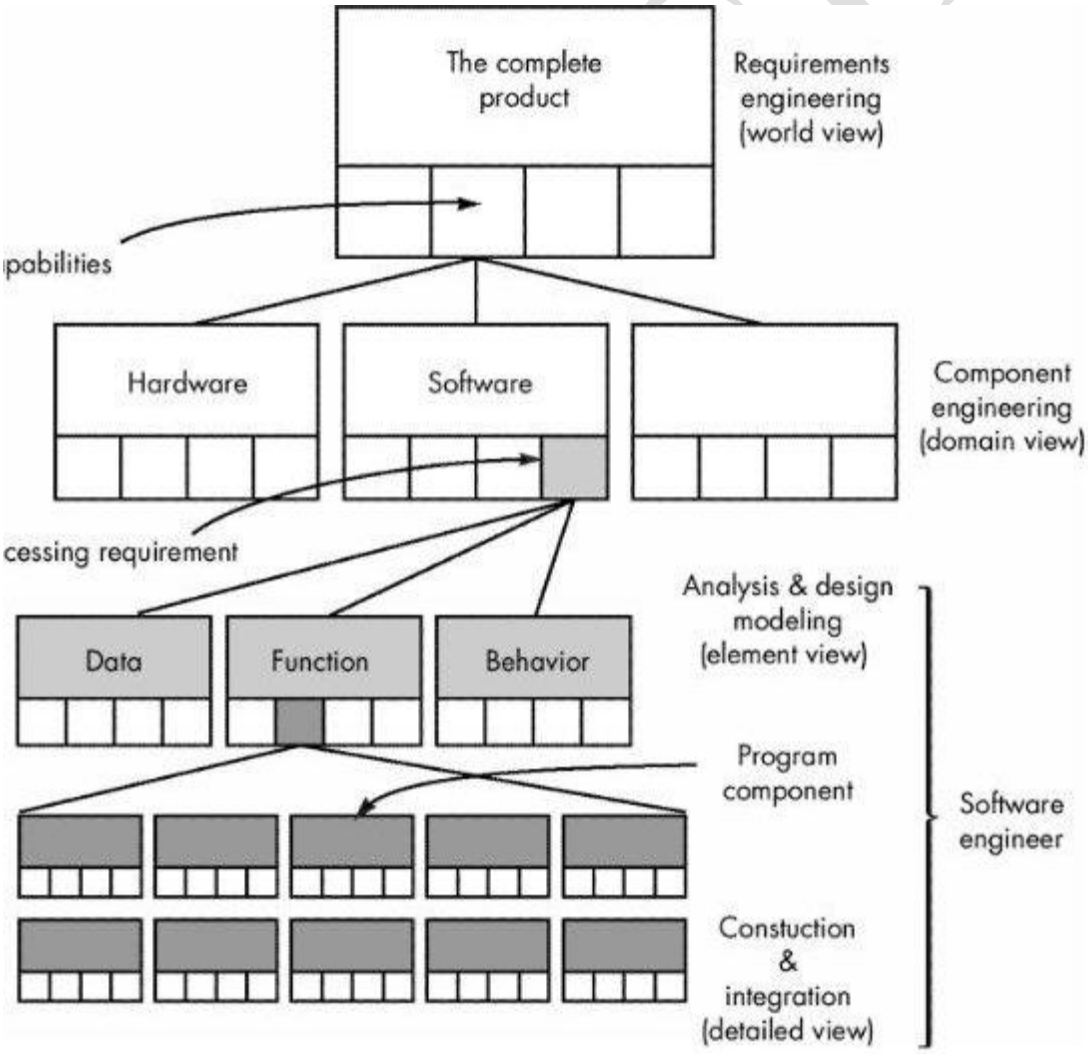
Product engineering translates the customer's desire for a set of defined capabilities into a working product. It achieves this goal by establishing product architecture and a support infrastructure. Product architecture components consist of people, hardware, software, and data. Support infrastructure includes the technology required to tie the components together and the information to support the components.

✓ Requirements engineering elicits the requirements from the customer and allocates function and behavior to each of the four components.

✓ System component engineering happens next as a set of concurrent activities that address each of the components separately. Each component takes a domain-specific view but maintains communication with the other domains. The actual activities of the engineering discipline take on an element view.

✓ Analysis modeling allocates requirements into function, data, and behavior.

✓ Design modeling maps the analysis model into data/class, architectural, interface, and component design.



UNIT-2

1. Explain the prototyping approaches in software process.
Two approaches:

i. Evolutionary prototyping – In this approach of system development, the initial prototype is prepared and it is then refined through number of stages to final stage.

ii. **Throw-away prototyping –** Using this approach a rough practical implementation of the system is produced. The requirement problems can be identified from this implementation. It is then discarded. System is then developed using some different engineering paradigm.

EVOLUTIONARY PROTOTYPING
Objective:
· The principal objective of this model is to deliver the working system to the end-user.
· Example-AI systems.

Advantages
· Fast delivery of the working system.
· User is involved while developing the system.
· More useful system can be delivered.
· Specification, design and implementation work is co-ordinated manner.

Problems
· Management problems
· Maintenance problem
· Verification

Incremental Development
· After designing the overall architecture the system is developed and delivered in series of increments.

THROW-AWAY PROTOTYPING
Objective:
· The principal objective of this model is to validate or to derive the system requirements.
· It is developed to reduce requirement risks.

Advantages
· Requirement risks are very less.

Problems

CSE/DSIRT

· It can be undocumented.
· Changes made during the software development proceed may degrade the system structure.
· Sometimes organizational quality standard may not be strictly applied.

2. **Explain about rapid prototyping techniques.**

Executable specification languages.
· Used to animate the system specification.
· It is expressed in a formal, mathematical language to provide a system prototype.

Very high level languages.
· These are programming languages which include powerful data management facilities.
· They simplify program development.

Application generators and fourth-generation languages.
· These are successful languages because there is a great deal of communality across data processing applications.

3. Explain in detail about data modeling.
· Data modeling makes use of the ERD.
· Consists of 3 interrelated information.
The data object.
Attributes.
Relationships.

Cardinality and Modality
· Cardinality is the specification of the number of occurrences of one object that can be related to the number of occurrences of another object.
· One-to-one cardinality.
· One-to-many cardinality.
· Many-to-Many cardinality.
· Modality of a relation is 0 if there is no explicit relationship or relation is optional.
· Modality is 1 if an occurrence of relationship is mandatory.

Entity/Relationship Diagrams
· Components are
Data Objects.
Attributes.
Relationships.
Various type indicators.

CSE/DSIRT

4. Explain in detail about Functional Modeling.

· This model describes the computations that take place within a system.
· This model is useful when the transformation from the inputs to outputs is complex.
· The functional model of a system can be represented by a data Flow Diagram(DFD).

Data Flow Diagrams/Data Flow Graph/Bubble chart

· A DFD is a graphical representation that depicts the information flow and the transforms that are applied as the data move from input to output.
· Level 0 DFD also called as fundamental system model or context model represents the entire software as a single bubble with input and output data indicated by incoming and outgoing arrows.
· Level 1 DFD contains 5 or 6 bubbles. Each bubbles can be refined at Layers to depict more details.

Extensions to Real Time Systems

· Ward and Meller extensions
· Hatley and Pirbhai extension.

5. Explain in detail about Structural Modeling.

· Structural model includes a detail refinement of ERD,data flow model and control flow model.
· Creating an ERD.
· Example: Safe Home Security System.
· Developing relationships and cardinality/Modality.
· Creating a data flow model using the guidelines.
· Creating a control flow model which describes the structural connection of Processes
Control flows
Control stores.
· State automation
· Process activation table.

UNIT-3

1. Explain in detail the design concepts.

Abstraction

· Functional abstraction
· Data abstraction
· Control abstraction

Information hiding

· Each module in the system hides the internal details of its processing activities and modules communicate only through over

CSE/DSIRT

defined interfaces.

Structure
· It permits the decomposition of a large system into smaller, more manageable units with well defined relationships to the other units in a system.
· Network is the most general form of structure.

Hierarchical Structures/Structure Charts
· It depicts the structure of subroutines in a system, the data passed between routines, can be indicated on the arcs connecting routines.

Modularity
· Modular system consists of well-defined, manageable units with well defined interfaces among units.

Concurrency
· Independent processes that can be activated simultaneously if multiple processors are available.

Coupling and Cohesion
· **Data coupling –** The data coupling is possible by parameter passing or data interaction.
· **Control coupling –** The modules share related control data in control coupling.
· **Common coupling –** The common data or a global data is shared among modules.
· **Content coupling** – Content coupling occurs when one module makes use of data or control information maintained in another module.

2. Explain the design principles.
· The design process should not suffer from tunnel vision.
· The design should be traceable to the analysis model.
· Design should not reinvent the wheel.
· The design should minimize the intellectual distance between the software and problem as it exists in the real world.
· The design should be structured to degrade gently, even when aberrant data, events or operating conditions are encountered.
· Design is not coding, coding is not design.
· The design should be assessed for quality as it is being created, not after the fact.
· The design should be reviewed to minimize conceptual (semantic) errors.

CSE/DSIRT

3. Explain the design steps of the transform mapping.

· Review the fundamental model.
· Review and refine the DFD for the software.
· Determine whether the DFD has the transform or transaction mapping.
· Isolate the transform center by specifying incoming and outgoing flow boundaries.
· Perform first-level factoring.
· Perform second-level factoring.
· Refine the first iteration architecture using design heuristics for improved software quality.

4. Explain the design steps in transaction mapping.

· Review the fundamental model.
· Review and refine the DFD for the software.
· Determine whether the DFD has the transform or transaction mapping.
· Identify transaction center and the flow characteristics along each of the action paths.
· Factor and refine the transaction structure and the structure of each action path.
· Refine the first iteration architecture using design heuristics for improved software quality.

5. Explain in detail about the real time systems.

· Hard and soft real time systems.
· Real time and high performance.
· Real-Time control.
· Real time software design
Periodic Stimuli – Occur at predictable time intervals.
Aperiodic Stimuli – Occur regularly

UNIT-4

1. Explain the types of software testing.

· Unit testing
· System testing
· Integration testing
· User-acceptance testing
· End-to-End testing
· Regression testing
· Exception testing
· Destructive testing

2. Explain in detail about Black box testing.

· Black box or behavioral testing focuses on the functional requirements of the software.
· It is applied during the last stage of testing.

CSE/DSIRT

· Syntax driven testing is suitable for the specification which are described by a certain grammar.
· Decision table based testing is implemented when the original software requirement have been formulated in the format of " if-then" statements.
· Liquid level control is the study of a simple control problem which is designed to check the liquid level. It has 2 sensors.
· Cause effect graphs in functional testing.

3. Explain about the software testing strategies.
· A strategic approach to software testing.
· Verification and Validation. ¾_Verification refers to the set of activities that ensure that software
correctly implements a specific function. ¾_Validation refers to a different set of activities that ensure that the
software that has been built is traceable to the customer requirements.
According to Boehm, ¾_Verification:" Are we building the product right?"
¾_Validation:" Are we building the right product?"
· Organization for software testing
· A software testing strategy.
· Criteria for completion of testing.

4. Explain in detail about Integration testing.
· It is a systematic technique for constructing the program structure.
· **Incremental integration –** The program is constructed and tested in small increments.
Top-down integration
· It is an incremental approach.
· Modules are integrated by moving downward through the control hierarchy beginning with the main control module(main program).
· Subordinate modules are incorporated by depth-first or breadth-first manner.
Bottom-up integration
· This testing begins construction and testing with the components at the lowest levels in the program structure.
Regression testing
· It is the re-execution of some subset of tests that have already been conducted to ensure the changes that have not been propagated unintended side effects.
Smoke testing
· It minimizes the integration risk.
· Error diagnosis and correction are simplified.

CSE/DSIRT

5. Explain in detail about system testing.

· System testing
· Stress testing
· Security testing.
· Performance testing

UNIT-5

1. Explain in detail about SCM.

· Software Configuration Management is an umbrella activity that is applied throughout the software process.

SCM Activities

· Identify change.
· Control change.
· Ensure the change is properly implemented.
· Report change to others.

Need for SCM

· When you build computer software change happens, you need to control it effectively.

SCI

· Software Configuration Item is information that is carried as part of the software engineering process.

2. Explain about software cost estimation.

· Major factors are:
Programme ability.
Product Complexity.
Product size.
Available time.
Required reliability.

3. Explain in detail about COCOMO model.

· Constructive Cost Model.
· Software cost estimation gives the estimation of how much months a man take to develop a software product.
· Application Composition Model.
· Early design stage model
· Post-architecture stage model.
· COCOMO II application composition uses object points.
· NOP=(object point)X[100-%reuse)/100]
NOP-New Object Point.
· Productivity Rate, PROD=NOP/person-Month.

CSE/DSIRT

4. Explain in detail about Delphi Method.

Procedure

· The co-ordinator presents a specification and estimation form to each expert.
· Co-ordinator calls a group meeting in which the experts discuss estimation issues with the coordinator and each other.
· Experts fill out forms anonymously.
· Co-ordinator prepares and distributes a summary of the estimates.
· The Co-ordinator then calls a group meeting.In this meeting the experts mainly discuss the points where their estimates vary widely.
· The experts again fill out forms anonymously.
· Again co-ordinator edits and summarizes the forms,repeating steps 5 and 6 until the co-ordinator is satisfied with the overall prediction synthesized from experts.

5. Explain in detail about software Maintenance.

· Software maintenance is an activity in which program is modified after it has been put into use.
· Maintenance is defined as the process in which changes are implemented by either modifying the existing system' s architecture or by adding new components to the system.

Different aspects of maintenance

· The bug-fixing view
· The need-to-adapt view
· The user-support view

Need for software maintenance

· To provide continuity of service
· To support mandatory upgrades.
· To support user requests for improvements.
· To facilitate future maintenance work.

6. Explain about CASE tools.

· Computer Aided Software Engineering.
· Business Process Engineering Tools.
· Process Modeling and Management Tools.
· Project Planning Tools.
· Risk analysis tools.
· Project management tools.
· Requirement tracing tools.
· Metrics and management tools.
· Documentation tools.
· System Software tools.
· Quality Assurance tools.
· Database management tools.

CSE/DSIRT

· SCM tools.
· Analysis and design tools.
· PTO/SIM tools.
· Interface design and development tools.
· Prototyping tools.
· Programming tools.
· Web development tools.
· Integration and testing tools.
· Static analysis tools.
· Dynamic analysis tools.
· Test management tools.
· Client/server tools.
· Re-engineering tools.

CSE/DSIRT