Software Engineering Assignment-III

Q.1. what are possible consequences if the software requirement analysis is not done properly? [2 marks]

Ans: If software requirement analysis is not done properly, it is highly likely that the

- Elegantly build software solution may solve the wrong problems.
- Result in wastage of time & money, personal frustration & unhappy customers.

Q.2 what are the steps in software requirement analysis? How is the requirement analysis important for software designers and developers? [4]

Ans: The following are the steps in software requirement analysis:

- Data/function/behavioral requirements are identified by requirement elicitation from the customer.
- Requirements are refined and analyzed to assess their clarity, completeness and consistency.
- Specification incorporating a model of the software is created that is software requirements model.
- Validate by both software engineers/Analyst and customers/users.

The requirement analysis is important for the software designers and developers because it:

- Provides designer with representation of information/function/behavior that can be translated to data/architectural/interface/component-level designs.
- Provides the developer/customer with the means of assess quality once software is built.

Q.3. List the operational analysis principles and guiding principles that an analyst applies for understanding the software requirements more completely and consistently. [6]

Ans:The operational principles include:

- The information domain of a problem must be represented and understood

- The functions that the software is to perform must be defined
- The behavior of the software must be represented.
- The models that depict information, function, and behavior must be partitioned.
- The analysis process should be move from essential information towards the implementation details.

The guiding principles include:

- Understand the problem before an Analyst begins to create the analysis model.
- Develop prototypes that enable a user to understand how human/machine interaction will occur.
- Record the origin of and the reason for every requirement.
- Use multiple views of requirements by building data/functional/behavioral models.
- Rank requirements for analysis and implementation on priority basis.
- Use of formal technical reviews to identify and eliminate any ambiguity.

Q.4 Write down 3 different views of data & control in the information domain. [1.5]

Ans: Three different views of data and control in the information domain are:

- Information Content & Relationship
- Information Flow
- Information Structure

Q.5 Name two dominant methods for the requirement analysis modeling. Which one is the modern modeling method? [1.5]

Ans: The two dominant methods for the requirement analysis modeling are:

- Structure analysis (classical modeling method)
- Object oriented analysis (modern modeling method)

SE Assignment-III

Q.6 Explain briefly the structure analysis model with a suitable diagram.[5]

Ans: The structure analysis model is:

- the classical method of requirement modeling that relies on data modeling and control flow modeling for creating the requirement analysis model
- widely used method of requirement modeling
- initially developed for conventional data processing applications but extensions have made the method applicable to real systems.

Structure analysis model includes the following tools/elements for requirement modeling:

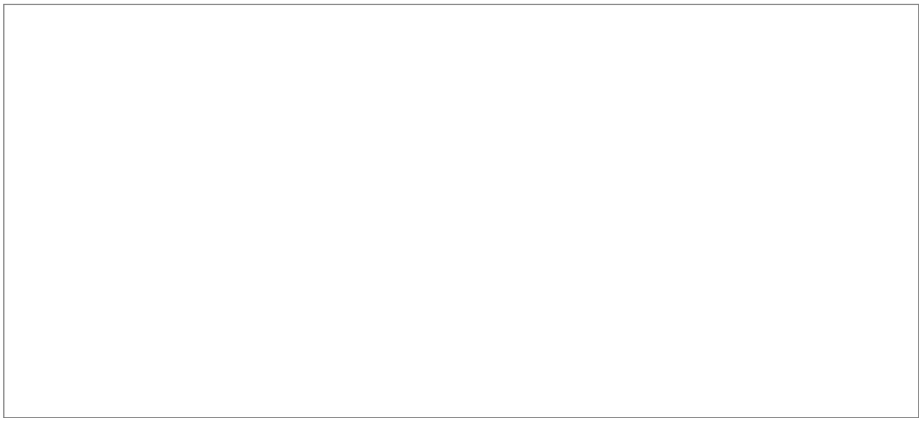
- Data dictionary

  Is the repository that contains descriptions of all the data objects

  consumed or produced by the software.

- lies at the core of the model which is surrounded by three different diagrams such as ERD, DFD and STD.
- Entity Relationship Diagram (ERD)

  Depicts the relationship between the data objects.



Example of ER-diagram.

- Is the notation that is used to conduct the data modeling activity

- Attributes of each data object in ERD can described using a data object description.
- Data flow Diagram(DFD)

  Is the graphical representation that depicts information flow and the transform data as they move through the system.

An example of a data flow diagram.

  - Depict the functions/sub-functions that transform the data flow.
  - Provides mechanism for information flow modeling and functional modeling.
  - serves as basis for the modeling of function.
  - A description of each function presented in the DFD is contained in a **process specification.**
  - **May be used to represent a system or software at any level of abstraction.**
- State-Transition Diagram (STD)
  - Indicates how the system behaves as the consequences of the external events.

- Represents the behavior of the system by depicting its states and the events that cause the system to change state.
- Serves as a basis for behavior modeling.
- Additional information about the control aspects of the software is contained in the control specification.
- Normally used when the state of the entities in the system will change in response to events. As an example of a state diagram, we show the operation of a one-passenger elevator. When a floor button is pushed, the elevator moves in the requested direction. It does not respond to any other request until it reaches its destination.

Example fig. for state-transition diagram

- Data Object Description.
- Control Specification

  It contains the additional information about the control aspects of the software.
- Process Specification

  It contains the descriptions of each functions presented in the Data Flow Diagram.

SE Assignment-III

Q.7 what is the importance of Data Flow Diagram and State Transition Diagram for structure analysis modeling? [4]

Ans:- Data flow diagram is:

- a graphical representation that depicts information flow and the transforms data as they move through the system
- Depict the functions or sub-functions that transform the data flow.
- Provides mechanism for information flow modeling and functional modeling.
- It serves as basis for the modeling of function.
- May be used to represent a system or software at any level of abstraction.

State Transition Diagram (STD):

- Indicates how the system behaves as a consequence of external events.
- Represents the behavior of the system by depicting its states and the events that cause the system to change state.
- Serves as a basis for behavior modeling

Q.8. what is the data modeling? Give a suitable example. [3]

Answer:

Data modeling is the modeling of the data objects that makes use of the Entity Relationship Diagram (ERD) which enables the analyst to identify the data objects and their relationships.

Cardinality and Modality are the most suitable examples of the data modeling:

1. Cardinality

- Defines the maximum number of objects that can participate in a relationship.
- Cardinality can be one to one (1:1), Many to Many (m:n) or one to many(1:n).

2. Modality

- Defines whether an occurrence of relationship is mandatory or not.

- the modality of a relationship is
  - 0 if there is no explicit need for the relationship to occur or the relationship is optional.
  - 1 if an occurrence of the relationship is mandatory.

Q.9. Explain briefly the software design model with a suitable diagram. [5]

Ans: **Software Design Model** provides a holistic view of software that is, it represent at the high level of abstraction.

Fig. Software Design Model

The levels of the software design models are:

- Data design

  Transforms information domain model into data structures.

- Architectural design

  Defines relationship between structural elements of the software

Represents the framework of computer based systems.

- Interface design

  Describes how software communicates within itself, with systems that interoperate with it, with humans.

  Represents the flow of information and type of behavior.

- Component-level design

  Transforms structural elements of the software architecture into procedural descriptions of the software components.

Q.10. what are the software design guidelines and principles? [6]

Ans: Software design guidelines:

- A design should exhibit an architectural structure
- A design should be modular
- A design should contain distinct representations of data, architecture, interfaces, & components
- A design should lead to data structures that are appropriate for the objects to be implemented
- A design should lead to components that exhibit independent functional characteristics
- A design should lead to interfaces that reduce the complexity of connections between modules & external environment.

Software design principles:

- The design process should not suffer from tunnel vision.
- The design should be traceable to the analysis model.
- The design should be structured to accommodate change.
- The design is not coding & coding is not a design.

- The design should be assessed for quality.
- The design should be reviewed to minimize the conceptual or semantic errors.
- A design should be derived using a repeatable method, driven by information obtained during software requirement analysis.

Q.11. Explain briefly the following software design concepts. [6]

Ans:

11.1 Abstraction

- It consider the problem at the abstract level and hiding its unnecessary details.
- Permits a designer to consider the component/module at the abstract level without worrying about the detail of the implementation of the components.

There are three forms of abstraction used in software design:

- **Data Abstraction -** a named collection of data that describes a data object
- **Procedure abstraction -** a named sequence of instructions that has a specific & limited function
- **Control Abstraction-**implies a program control mechanisms without specifying internal details

11.2 Refinement

- Is a process of elaboration of the description of the module defined at the high level of abstraction to its low level abstraction.
- Unlike the abstraction it reveals the low level details as the design progress

11.3 Modularity

- Is the design concept in which the system is decomposed into several components (modules) which are implemented independently.
- Uses the divide and conquer strategy for solving the large and complex problem.

- It reduces the overall cost of development and maintenance of the software products.

Q.12. what is software architecture? List five different types of models that are used to represent the Architectural design. [4]

Ans:

Software architecture is the structure & organization of the program components that basically depicts the overall structure of the software.

5 different types of models used to represent architectural design are:

1. **Structural models**

- Represent the architecture as an organized collection of software components.

2. **Framework models**

- Indentify repeatable architectural design framework that are similar to types of the applications.

3. **Dynamic models**

- Addresses the behavioral aspect of the software architecture

4. **Process models**

- Focuses on the design of the business or technical process that system must accommodate.

5. **Functionality models**

- Used the functional hierarchy of a system.

Q.13. what are the advantages of modularity in software design? How do the software designers achieve an effective modular design? [4]

Ans:

SE Assignment-III

The advantage of modularity is that it reduces complexity, facilitates change, and results in easier implementation by encouraging parallel development of different parts of a system.

The software designers achieve an effective modular designs with the following criteria:

- **Modular decomposability** - provides a systematic approach for decomposing the problem into subproblems.
- **Modular composability** -: enables existing(reusable) design components to be assembled into a new system
- **Modular understandability**-: module can be understood as a standalone unit (no need to refer to other modules)
- **Modular continuity** -: small changes to the system requirements result in changes to individual modules
- **Modular protection**- :unexpected condition occurs within a module & its effects are constrained within that module

Q.14 what is coupling and cohesion in effective modular design? Are strongly coupled modules more desirable? Justify your answer. [2, 3]

Ans:

- **Cohesion**: is a measure of the relative functional strength of a module.
- **Coupling**: is a measure of the relative interdependence among modules.

Note: A designer should always strive for module with high cohesion and low coupling.

The strongly coupled modules are not desirable because in software design, the designer strives for a lowest possible coupling.

The Low coupling modules results in software which is:

- easier to understand &
- less prone to a "ripple effect“ which is caused when errors occur at one location and propagate through a system.

Q.15. When does the common coupling occur? List other coupling types. [3]

SE Assignment-III

Ans:

The common coupling is a high coupling that occurs when a number of modules reference a global data area. The figure below shows the types of couplings:

- Module ‘a and d’ are subordinate to different modules. Each is unrelated and therefore **no direct coupling** occurs.
- **Data coupling** exist between two modules when one accesses other module via a conventional argument list, through which data are passed.
- **Stamp coupling** is a variation of data coupling which is found when a portion of data structure instead simple argument list is passed via a module interface.
- **Common coupling**: is a high coupling that occurs when a number of modules reference a global data area.
- **Content coupling**: is the highest degree of coupling that occurs when one module makes use of data or control information maintained within the boundary of another module.