
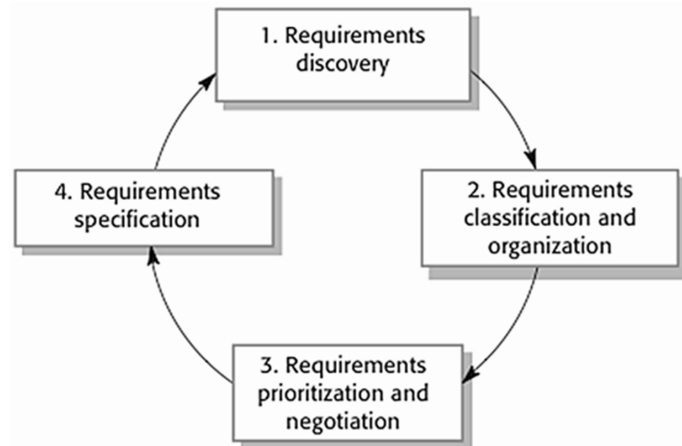


CMR INSTITUTE OF TECHNOLOGY		USN <input type="text"/>							
Internal Assessment Test – II, October 2018									
Sub:	Software Engineering						Code:	17MCA34	
Date:	16-10-2018	Duration:	90 mins	Max Marks:	50	Sem:	III	Branch:	MCA
Answer ONE FULL QUESTION from each part								Marks	OBE
								CO	RBT
Part - I									
1	Explain requirements elicitation and analysis process						10	CO2	L2
(OR)									
2	What are the requirements validation techniques? Explain briefly.						10	CO2	L2
Part – II									
3	Explain system models with suitable examples.						10	CO3	L3
(OR)									
4	What is use-case diagram? Explain the importance of use case modeling.						10	CO3	L3
Part – III									
5	What is architectural design? Explain the repository model and client-server model with an example for each.						10	CO3	L3
(OR)									
6	Explain basic elements of a component model with a neat diagram.						10	CO3	L2
PART - IV									
7 (a)	Briefly explain about the CBSE process.						05	CO2	L2
(b)	Draw a Data flow diagram (DFD) of an ATM						05	CO3	L1
(OR)									
8 (a)	Discuss the importance of behavioral model						05	CO3	L2
(b)	Discuss the importance of the role of software architecture.						05	CO3	L2
Part – V									
9	Discuss the view patient information use-case sequence diagram illustrating basic of notations used.						10	CO3	L2
(OR)									
10	Briefly explain different architecture styles for C and C view.						10	CO3	L2

1. Explain requirements elicitation and analysis process (10)

- After an initial feasibility study, the next stage of the requirements engineering process is requirements elicitation and analysis (Sometimes called requirements discovery)
- The process activities are:
 - i. Requirements discovery: Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage
 - ii. Requirements classification and organization: Groups related requirements and organises them into coherent clusters
 - iii. Requirements prioritization and negotiation: Prioritising requirements and resolving requirements conflicts
 - iv. Requirements specification: Requirements are documented and input into the next round of the spiral



Requirements engineering process

(OR)

2. What are the requirements validation techniques? Explain (10) briefly.

- It is the process of checking that requirements actually define the system that the customer really wants.
- The different types of checks should be carried out on the requirements in the requirements document include:
 - Validity: Does the system provide the functions which best support the customer’s needs?
 - Consistency: Are there any requirements conflicts?
 - Completeness: Are all functions required by the customer included?
 - Realism: Can the requirements be implemented given available budget and technology
 - Verifiability: Can the requirements be checked?
- Requirements validation techniques
 - Requirements reviews: Systematic manual analysis of the requirements.
 - Prototyping: Using an executable model of the system to check requirements.
 - Test-case generation: Developing tests for requirements to check testability

3. Explain system models with suitable examples. (10)

- System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.
- System modeling has generally come to mean representing the system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML).
- System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers.
- You may develop different models to represent the system from different perspectives
 - i. An external perspective: where you model the context or environment of the system.
Example: context diagram
 - ii. An interaction perspective: where you model the interactions between a system and its environment or between the components of a system.
Example: use-case diagram
 - iii. A structural perspective: where you model the organization of a system or the structure of the data that is processed by the system.
Example: class diagram
 - iv. A behavioral perspective: where you model the dynamic behavior of the system and how it responds to events.
Example: activity diagram

(OR)

4. What is use-case diagram? Explain the importance of use case modeling. (10)

- Use case diagrams: which show the interactions between a system and its environment.
- Each use case represents a discrete task that involves external interaction with a system.
- Actors in a use case may be people or other systems.
- Improve communication among team members
- Encourage common agreement about system requirements
- Reveal process alternatives, process exceptions, undefined terms, and outstanding issues
- Expose what belongs outside project scope
- Transform manual processes into automated processes
- They are easy to understand by the client

5. What is architectural design? Explain the repository model and client-server model with an example for each. (5)

- *“the software architecture of a system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them”*
- i. data repositories:
 - Stores shared data (these could be file systems or databases)

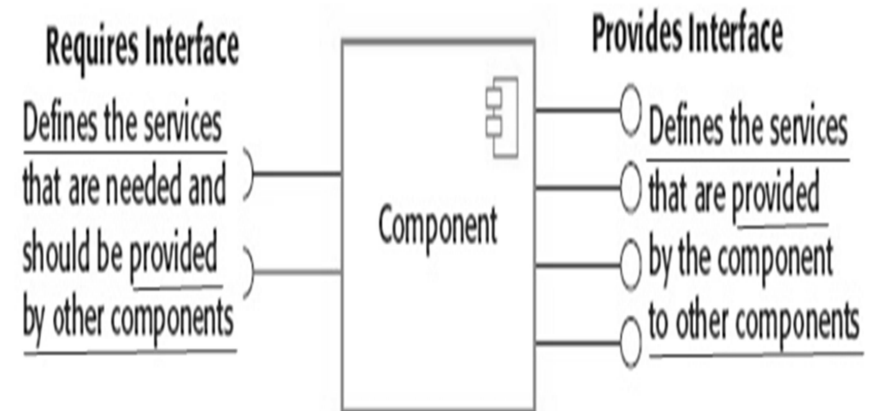
- provide a reliable and permanent storage
 - take care of any synchronization needs for concurrent access
 - provide data access support
- ii. Repository style: passive repository
- The components access the repository as and when they want.
- iii. Client-Server Style
- Another very common style used to build systems today is the client server style
 - there are two component types—clients and servers
 - Clients can only communicate with the server, but not with other clients
 - Communication is initiated by a client which sends request and server responds
 - One connector type – request/reply, which is asymmetric
 - A connector connects a client to a server.
 - Often the client and the servers reside on different machines
 - A general form of this style is the n-tier structure

(OR)

6. Explain basic elements of a component model with a neat diagram. (10)

- Component: is an independent software unit that can be composed with other components to create a software system

- Components have two related interfaces:
 - i. *Requires interface:*
 - specifies what services must be provided by other components in the system
 - ii. *Provides interface*
 - Defines the services that are provided by the component to other components.
 - This interface, essentially, is the component API.
 - It defines the methods that can be called by a user of the component.



▪ Basic elements of component model:

(i) Interface:

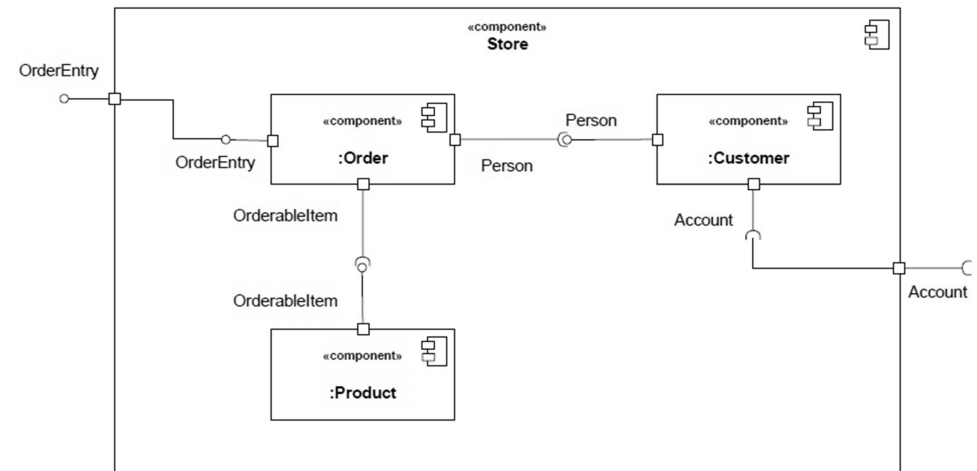
- Components are defined by specifying their interfaces.
- The component model specifies how the interfaces should be defined and the elements, such as operation names, parameters and exceptions, which should be included in the interface definition

(ii) Usage:

- In order for components to be distributed and accessed remotely, they need to have a unique name or handle associated with them.
- This has to be globally unique.

(iii) Deployment

- The component model includes a specification of how components should be packaged for deployment as independent, executable entities.

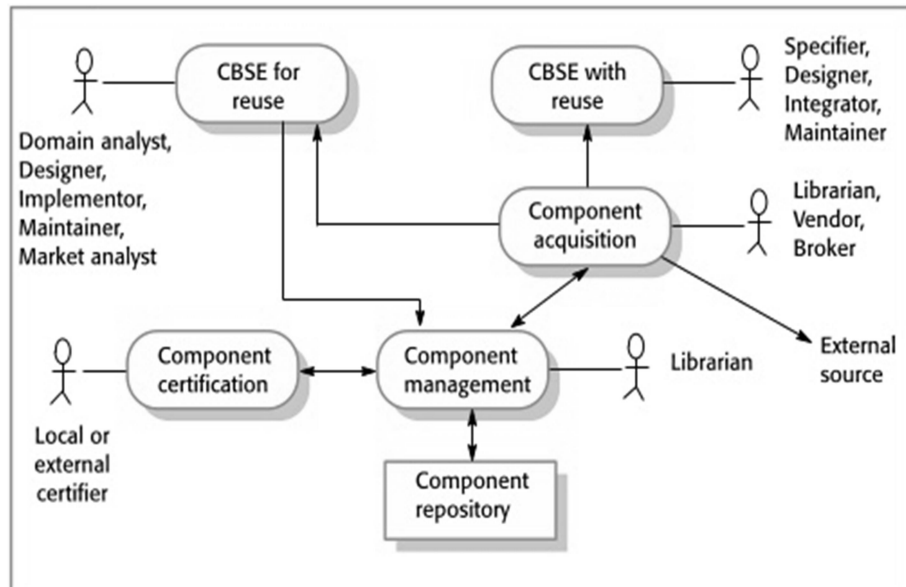


Example of a component model for Order Processing

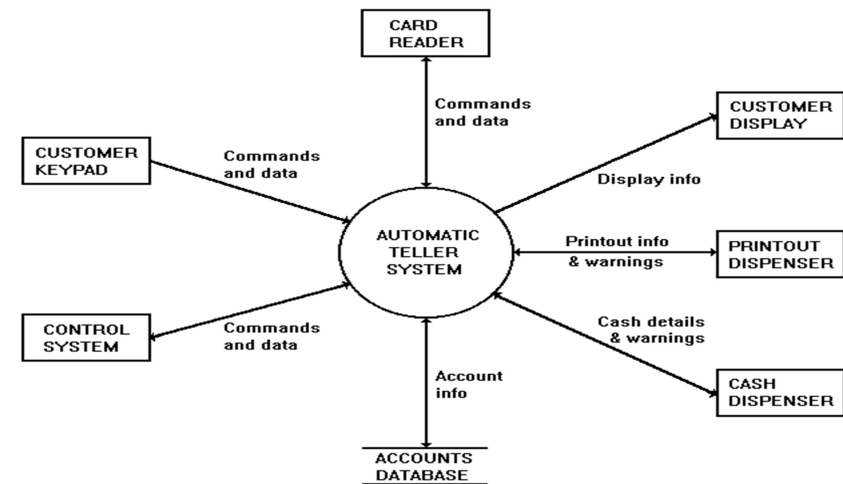
7. (a) Briefly explain about the CBSE process. (5)

- CBSE is the process of defining, implementing, and integrating or composing loosely-coupled, independent components into systems.
- two types of CBSE processes:
 - (i) Development for reuse: concerned with developing components or services that will be reused in other applications. It usually involves generalizing existing components.
 - (ii) Development with reuse: process of developing new applications using existing components and services.

- Three steps:
 - i. Component Acquisition: the process of acquiring components for reuse or development into a reusable component.
 - It may involve accessing locally developed components or services or finding these components from an external source
 - ii. Component management: concerned with managing a company's reusable components, ensuring that they are properly cataloged, stored, and made available for reuse
 - iii. Component certification: the process of checking a component and certifying that it meets its specification



7. (b) Draw a Data flow diagram (DFD) of an ATM (5)



(OR)

8. (a) Discuss the importance of behavioral model (5)

- Behavioral models are models of the dynamic behavior of the system as it is executing.
- They show what happens or what is supposed to happen when a system responds to a stimulus from its environment.
- The sequence diagram depicts the sequence in which the activities are carried out or happened. This enables the stakeholders of the project to understand the sequence of activities.

- The data flow diagram shows the flow and interaction of data from phase to phase. It is used to understand the system's perspective in logical as physical way
- The data flow diagrams are simple and intuitive in nature and hence it is possible to explain them to potential system users who can then participate in validating the model
- The event driven modelling depicts how a system responds to external and internal events

8. (b) Discuss the importance of the role of software architecture. (5)

i. Understanding and communication:

- communicate the architecture to its various stakeholders
- Provides aid in understanding of existing systems
- An architecture description of the proposed system describes how the system will be composed, when it is built.

ii. Reuse:

- A method of reuse is to compose systems from parts and reuse existing parts
- decision about using existing components is made at architectural design time
- Architecture helps specify what is fixed and what is variable in different products

iii. Construction and Evolution:

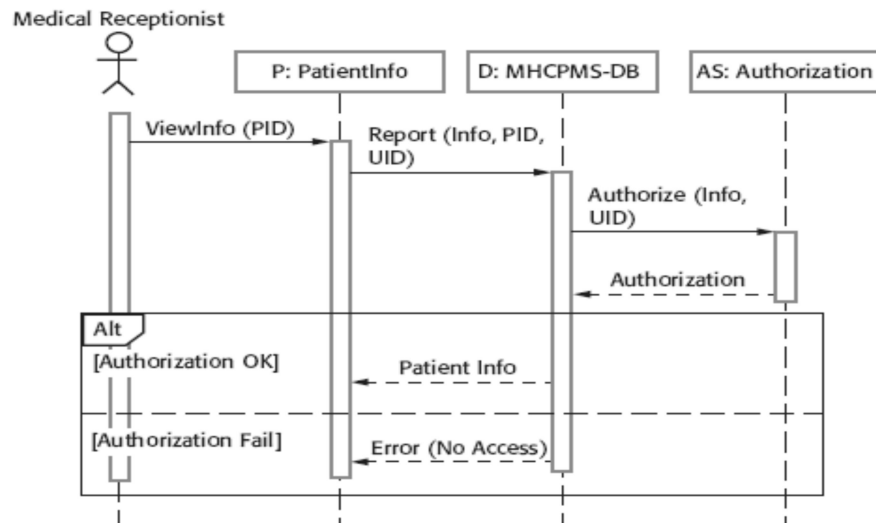
- Architecture can help decide what is the impact of changes to existing components on others
- A suitable partitioning in the architecture can provide the project with the parts that need to be built to build the system.
- During software evolution, architecture helps decide what needs to be changed to incorporate the new changes/features

iv. Analysis:

- It is highly desirable if some important properties about the behavior of the system can be determined before the system is actually built.
- Software architecture provides possibilities for software to consider the alternatives during design to reach the desired performance levels like reliability

9. Discuss the view patient information use-case sequence diagram illustrating basic of notations used. (10)

- Sequence diagrams are part of the UML and are used to model the interactions between the actors and the objects within a system.
- A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.
- The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.
- Interactions between objects are indicated by annotated arrows.



1. The medical receptionist triggers the ViewInfo method in an instance P of the PatientInfo object class, supplying the patient's identifier, PID. P is a user interface object, which is displayed as a form showing patient information.
2. The instance P calls the database to return the information required, supplying the receptionist's identifier to allow security checking (at this stage, we do not care where this UID comes from).
3. The database checks with an authorization system that the user is authorized for this action.
4. If authorized, the patient information is returned and a form on the user's screen is filled in. If authorization fails, then an error message is returned.

(OR)

10. Briefly explain different architecture styles for C and C view. (10)

- an architecture view describes a structure of the system in terms of its elements and relationships among them
- different systems will have different structures, even for the same view
- Architectural styles: some structures and related constraints that have been observed in many systems and that seem to represent general structures that are useful for architecture of a class of problems
- Example: for module views, some of the common styles are decomposition, uses, generalization, and layered. In decomposition style, a module is decomposed into sub-modules, and the system becomes a hierarchy of modules
- In the uses style, modules are not parts of each other, but a module uses services of other modules

(i) Pipe and Filter

- well suited for systems that primarily do data transformation some input data is received and the goal of the system is to produce some output data by suitably transforming the input data
- achieves the desired transformation by applying a network of smaller transformations and composing them in a manner that together the overall desired transformation is achieved

-
- | | |
|--|--|
| <ul style="list-style-type: none">• The pipe and filter style has only one component type called the filter• It also has only one connector type, called the pipe <p>(ii) Shared-Data Style</p> <ul style="list-style-type: none">▪ there are two types of components: <p>i. data repositories:</p> <ul style="list-style-type: none">• Stores shared data (these could be file systems or databases)• provide a reliable and permanent storage• take care of any synchronization needs for concurrent access• provide data access support <p>ii. data accessors</p> <ul style="list-style-type: none">• access data from the repositories• perform computation on the data obtained• if they want to share the results with other components, put the results back in the depository• Communication between data accessors is only through the repository <p>(iii) Client-Server Style</p> <ul style="list-style-type: none">• Another very common style used to build systems today is the client server style• there are two component types—clients and servers• Clients can only communicate with the server, but not with other clients | <ul style="list-style-type: none">• Communication is initiated by a client which sends request and server responds• One connector type – request/reply, which is asymmetric• A connector connects a client to a server.• Often the client and the servers reside on different machines• A general form of this style is the n-tier structure |
|--|--|
-