CMR INSTITUTE OF TECHNOLOGY	USN										

Internal Assesment Test – II, December 2019

Sub:	SOFTWARE ENGINEERING							Code:	18MCA14
Date:	17.12.2019	Duration:	90 mins	Max Marks:	50	Sem:	I	Branch:	MCA

	Answer **ONE FULL QUESTION** from each part	Marks	OBE	
			CO	RBT
	Part-I			
1	Describe requirements elicitation and analysis process with example	[10]	CO2	L1
	(OR)			
2	Explain requirements validation techniques? Justify why should validate.	[10]	CO2	L2
	Part-II			
3	Write short notes on i. Ethnography ii. Use-case iii. Sequence diagram iv. Feasibility study v. Require change management	[10]	CO2	L1
	(OR)			
4	Describe Components and component models with suitable example	[10]	CO3	L1

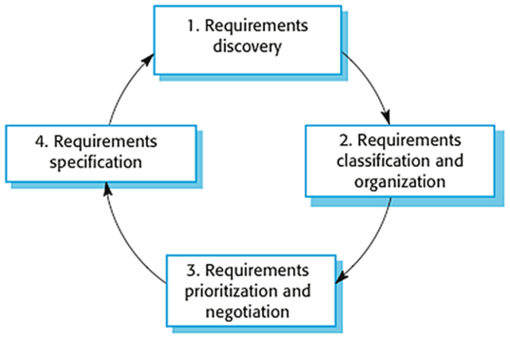
CMR INSTITUTE OF TECHNOLOGY	USN										

Internal Assesment Test – II, December 2019

Sub:	SOFTWARE ENGINEERING							Code:	18MCA14
Date:	17.12.2019	Duration:	90 mins	Max Marks:	50	Sem:	I	Branch:	MCA

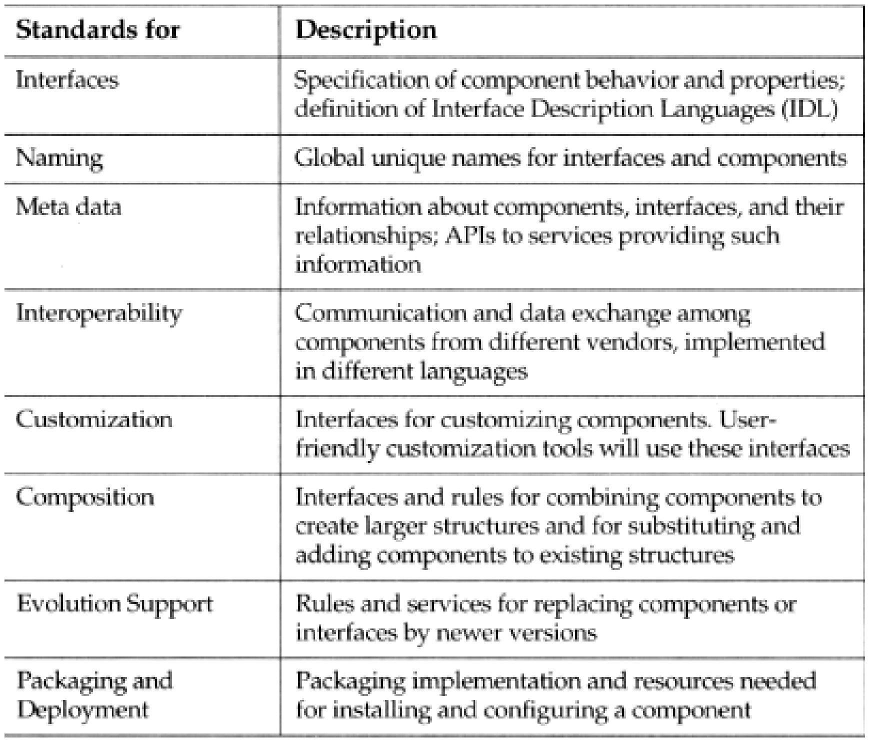
	Answer **ONE FULL QUESTION** from each part	Marks	OBE	
			CO	RBT
	Part-I			
1	Describe requirements elicitation and analysis process with example	[10]	CO2	L1
	(OR)			
2	Explain requirements validation techniques? Justify why should validate.	[10]	CO2	L2
	Part-II			
3	Write short notes on i. Ethnography ii. Use-case iii. Sequence diagram iv. Feasibility study v. Require change management	[10]	CO2	L1
	(OR)			
4	Describe Components and component models with suitable example	[10]	CO3	L1

CMR INSTITUTE OF TECHNOLOGY
Department of Computer Applications-PG Studies
Second Internal Assessment Test- Scheme and Solution
18MCA14-Software Engineering

Questions Number	Expected Answer	Marks Break up	Maximum Marks
1	Describe requirements elicitation and analysis process with example **Requirements Elicitation & Analysis** It's a process of interacting with customers and end-users to find out about the domain requirements, what services the system should provide, and the other constrains. *Domain requirements reflect the environment in which the system operates so, when we talk about an application domain we mean environments such as train operation, medical records, e-commerce etc.* It may also involve a different kinds of stockholders; end-users, managers, system engineers, test engineers, maintenance engineers, etc. *A stakeholder is anyone who has direct or indirect influence on the requirements.* **The requirements elicitation and analysis has 4 main process** We typically start by gathering the requirements, this could be done through a general discussion or interviews with your stakeholders, also it may involve some graphical notation. Then you organize the related requirements into sub components and prioritize them, and finally, you refine them by removing any ambiguous requirements that may raise from some conflicts. Here are the 4 main process of requirements elicitation and analysis.  1. Requirements discovery 2. Requirements classification and organization 3. Requirements prioritization and negotiation 4. Requirements specification The process of requirements elicitation and analysis It shows that it's an iterative process with a feedback from each activity to another. The process cycle starts with requirements discovery and ends with the requirements document. The cycle ends when the requirements document is complete.	10	10

2	Explain requirements validation techniques? Justify why should validate. **Requirements Validation** It's a process of ensuring the specified requirements meet the customer needs. It's concerned with finding problems with the requirements. These problems can lead to extensive rework costs when these they are discovered in the later stages, or after the system is in service. The cost of fixing a requirements problem by making a system change is usually much greater than repairing design or code errors. Because a change to the requirements usually means the design and implementation must also be changed, and re-tested. During the requirements validation process, different types of checks should be carried out on the requirements. These checks include: 1. **Validity** *checks*: The functions proposed by stakeholders should be aligned with what the system needs to perform. You may find later that there are additional or different functions are required instead. 2. **Consistency** *checks*: Requirements in the document shouldn't conflict or different description of the same function 3. **Completeness** *checks*: The document should include all the requirements and constrains. 4. **Realism** *checks*: Ensure the requirements can actually be implemented using the knowledge of existing technology, the budget, schedule, etc. 5. **Verifiability**: Requirements should be written so that they can be tested. This means you should be able to write a set of tests that demonstrate that the system meets the specified requirements.	**10**	**10**
3	Write short notes on i. Ethnography ii. Use-case iii. Sequence diagram iv. Feasibility study v. Require change management Ethnography: is the systematic study of people and cultures. It is designed to explore cultural phenomena where the researcher observes society from the point of view of the subject of the study. An ethnography is a means to represent graphically and in writing the culture of a group. Use Case: In software and systems engineering, a use case is a list of actions or event steps typically defining the interactions between a role and a system to achieve a goal. The actor can be a human or other	**10**	**10**

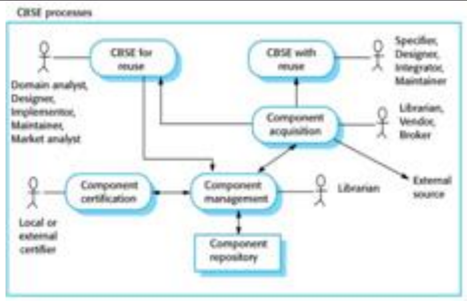
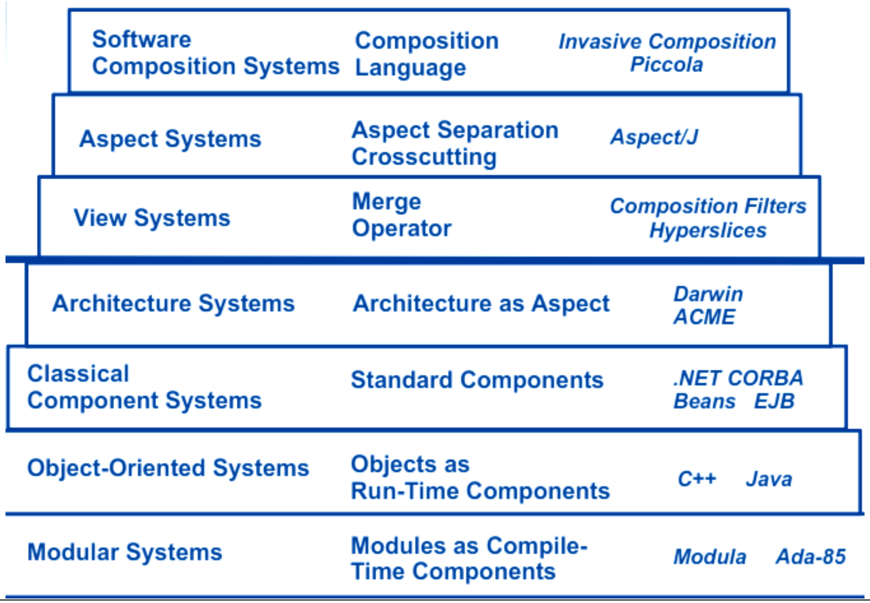
	external system. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Feasibility Study is an assessment of the practicality of a proposed project or system. Change management is the process, tools and techniques to manage the people side of change to achieve the required business outcome. Change management incorporates the organizational tools that can be utilized to help individuals make successful personal transitions resulting in the adoption and realization of change		
4	Describe Components and component models with suitable example	**10**	**10**



Standards for	Description
Interfaces	Specification of component behavior and properties; definition of Interface Description Languages (IDL)
Naming	Global unique names for interfaces and components
Meta data	Information about components, interfaces, and their relationships; APIs to services providing such information
Interoperability	Communication and data exchange among components from different vendors, implemented in different languages
Customization	Interfaces for customizing components. User-friendly customization tools will use these interfaces
Composition	Interfaces and rules for combining components to create larger structures and for substituting and adding components to existing structures
Evolution Support	Rules and services for replacing components or interfaces by newer versions
Packaging and Deployment	Packaging implementation and resources needed for installing and configuring a component

5a	Discuss the Component-Based Software Engineering (CBSE) process. Component-based development techniques involve procedures for developing software systems by choosing ideal off-the-shelf components and then assembling them using a well-defined software architecture. Component-based development is also known as component-based software engineering (CBSE)	**5**	**5**

CMR INSTITUTE OF TECHNOLOGY
Department of Computer Applications-PG Studies
Second Internal Assessment Test- Scheme and Solution
18MCA14-Software Engineering

	 CBSE processes CBSE for reuse CBSE with reuse Specifier, Designer, Integrator, Maintainer Domain analyst, Designer, Implementer, Maintainer, Market analyst Component acquisition Librarian, Vendor, Broker Component certification Component management Librarian External source Local or external certifier Component repository		
5b	Explain Component composition techniques. A composition framework based on software architecture. Abstract: Component based software engineering can effectively enhance the software developing quality and efficiency, and the composition technique is one of the key points of the component based software engineering.  Software Composition Systems Composition Language Invasive Composition Piccola Aspect Systems Aspect Separation Crosscutting Aspect/J View Systems Merge Operator Composition Filters Hyperslices Architecture Systems Architecture as Aspect Darwin ACME Classical Component Systems Standard Components .NET CORBA Beans EJB Object-Oriented Systems Objects as Run-Time Components C++ Java Modular Systems Modules as Compile-Time Components Modula Ada-85	**5**	**5**

CMR INSTITUTE OF TECHNOLOGY

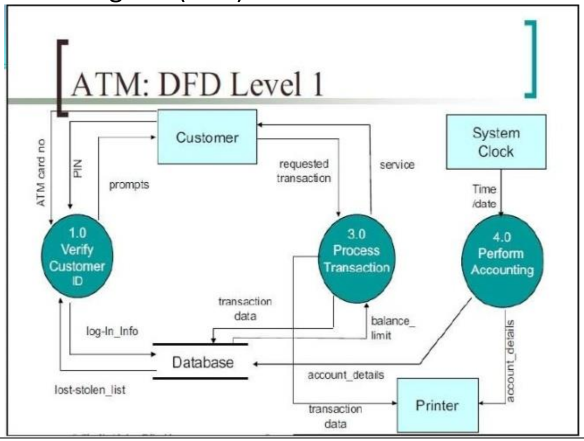
Department of Computer Applications-PG Studies

Second Internal Assessment Test- Scheme and Solution

18MCA14-Software Engineering

6a	Explain system models with suitable examples. 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). However, it is also possible to develop formal (mathematical) models of a system, usually as a detailed system specification. Models are used during the requirements engineering process to help derive the requirements for a system, during the design process to describe the system to engineers implementing the system and after implementation to document the system's structure and operation. You may develop models of both the existing system and the system to be developed: Different models to represent the system from different perspectives .For example: 1. An external perspective, where you model the context or environment of the system. 2. An interaction perspective where you model the interactions between a system and its environment or between the components of a system. 3. A structural perspective, where you model the organization of a system or the structure of the data that is processed by the system. 4. A behavioral perspective, where you model the dynamic behavior of the system and how it responds to events. Four important types of system models, namely, Context models, Interaction models, Structural models(Identification of domain classes, Refining the model) Behavioral models(Data-driven modeling, Event-driven modeling)	**5**	**5**
6b	What is use-case diagram? Explain the importance of use case modeling. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, use case diagrams are modelled to present the outside view. In brief, the purposes of use case diagrams can be said to be as follows − • Used to gather the requirements of a system.	**5**	**5**

	• Used to get an outside view of a system. • Identify the external and internal factors influencing the system. • Show the interaction among the requirements are actors. Use case diagrams can be used for – • Requirement analysis and high level design. • Model the context of a system. • Reverse engineering. • Forward engineering.		
7	Define architectural design. Explain the taxonomy of architectural model. Architectural design encompasses both the data architecture and the program structure layers of the design model. This section defines the term “software architecture” as a framework made up of the system structures that comprise the software components, their properties, and the relationships among these components. The following are the Taxonomy *(Classification)* of Architectural Style • Data-centered architectures • Data flow architectures • Call and return architectures • Object-oriented architectures • Layered architectures	**10**	**10**
8	Discuss the importance of behavioral model with example Behavioral models are models of the dynamic behavior of a 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. You can think of these stimuli as being of two types: – **Data** Some data arrives that has to be processed by the system. – **Events** Some event happens that triggers system processing. Events may have associated data, although this is not always the case. Data-driven modeling • Many business systems are data-processing systems that are primarily driven by data. They are controlled by the data input to the system, with relatively little external event processing. • Data-driven models show the sequence of actions involved in processing input data and generating an associated output. • They are particularly useful during the analysis of requirements as they can be used to show end-to end processing in a system. Event-driven modeling • Real-time systems are often event-driven, with minimal data processing. For example, a landline phone switching system responds to	**10**	**10**

	events such as 'receiver off hook' by generating a dial tone. Event-driven modeling shows how a system responds to external and internal events. • It is based on the assumption that a system has a finite number of states and that events (stimuli) may cause a transition from one state to another.		
9a	Draw a Data flow diagram (DFD) of an ATM  ATM: DFD Level 1 Customer System Clock ATM card no PIN prompts requested transaction service Time /date 1.0 Verify Customer ID 3.0 Process Transaction 4.0 Perform Accounting transaction data balance_ limit log-In_Info Database account_details lost-stolen_list transaction data Printer account_details	**4**	**4**
9b	Discuss the importance of the role of software architecture. ✧ Understanding and communication ✧ Reuse ✧ Construction and evolution ✧ Analysis ✧ Key technical consultant ✧ *Make decisions* ✧ Coach of development team ✧ Coordinate design ✧ Implement key parts ✧ Advocate software architecture	**6**	**6**

CMR INSTITUTE OF TECHNOLOGY
Department of Computer Applications-PG Studies
Second Internal Assessment Test- Scheme and Solution
18MCA14-Software Engineering

10a	Briefly explain different architecture styles for C and C view. ✧ Two main elements – components and connectors ✧ Components: *Computational elements or data stores* ✧ Connectors: *Means of interaction between components* ✧ A C&C view defines the components, and which components are connected through which connector ✧ The C&C view describes a runtime structure of the system – what components exist at runtime and how they interact during execution ✧ Is a graph; often shown as a box-and-line drawing ✧ Most commonly used structure ✧ Units of computations or data stores ✧ Has a name, which represents its role, and provides it identity ✧ A comp may have a type; diff types rep by diff symbols in C&C view ✧ Comps use ports (or interfaces) to communicate with others ✧ An arch can use any symbols to rep components; some common ones are shown Connectors: ✧ Interaction between components happen through connectors ✧ A connector may be provided by the runtime environment, e.g. procedure call ✧ But there may be complex mechanisms for interaction, e.g http, tcp/ip, ports,…; a lot of sw needed to support them ✧ Important to identify them explicitly; also needed for programming components properly ✧ Connectors need not be binary, e.g. a broadcast bus ✧ Connector has a name (and a type) ✧ Often connectors represented as protocol – i.e. comps need to follow some conventions when using the connector ✧ Best to use diff notation for diff types of connectors; all connectors should not be shown by simple lines	**6**	**6**
10b	Discuss the methods of architecture design documentation ✧ While designing and brainstorming, diagrams are a good means ✧ Diagrams are not sufficient for documenting arch design ✧ An arch design document will need to precisely specify the views, and the relationship between them ✧ An architecture document should contain ✧ *System and architecture context* ✧ *Description of architecture views* ✧ *Across view documentation* ✧ A context diagram that establishes the sys scope, key actors,	**4**	**4**

	and data sources/sinks can provide the overall context ✧ A view description will generally have a pictorial representation, as discussed earlier ✧ Pictures should be supported by ✧ Element catalog: *Info about behavior, interfaces of the elements in the arch* ✧ Architectural rationale: *Reasons for making the choices that were made* ✧ Behavior: *Of the system in different scenarios (e.g. collaboration diagram)* ✧ Other Information: *Decisions which are to be taken, choices still to be made.*		