USN | | | | | | | | | | |

IAT 1 – March 2018

Sub:	Software Engineering					Sub Code:	15CS42	Branch:	CSE	
Date:	12/03/2018	Duration:	90 mins	Max Marks:	50	Sem/Sec:	4 (A,B,C)			OBE

Answer **FOUR** FULL questions selecting AT LEAST ONE question **FROM EACH PART**

		MARKS	CO	RBT

PART A

1 (a) Define Software Engineering. Mention and explain the key challenges or the general issues facing Software Engineering. [4] CO1 L1

[Definition- 1M, Challenges- 3x1M]

3) What is Software Engineering?

It is an engineering discipline that is concerned with all aspects of software production from early stages of system specification to maintaining the system after it has gone into use. Aim is cost effective development of high-quality software systems to be developed by applying engineering principles.

8) Key challenges / issues of software engineering.

- Heterogeneity – Systems are required to operate distributed systems across networks. It is required to integrate new software with older legacy systems written in different programming languages. The challenge is to develop techniques for building dependable software which is flexible enough to cope with this heterogeneity.
- Business and social change – Business and society are changing incredibly quickly. They need to be able to change their existing software and to rapidly develop new software. Many traditional software engineering techniques are time consuming and delivery of new systems often takes longer than planned. They need to evolve so that the time required for software to deliver value to its customers is reduced.
- Security and trust – We should develop techniques that demonstrate that software can be trusted by its users which is especially true for remote software systems accessed through a web page or web services interface. We have to make sure that malicious users cannot attack our software and that information security is maintained.

(b) Explain 8 principles for SE Ethics and Professional practices as specified by IEEE. [8] CO2 L2

[Principles 8x1M]

Eight Principles:

- Public - Act consistently with public interest.
- Client and Employer — Act in favor of client, employer & public.
- Product - Ensure that products and related modifications meet the highest professional standards possible.
- Judgement — Maintain integrity and independence in professional judgement.
- Management - SE managers and leaders shall subscribe to and promote an ethical approach to management of software development and maintenance.
- Profession - Advance integrity and reputation of the profession consistent with public interest.
- Colleagues - Be fair to and supportive of your colleagues.
- Self - Participate in life-long learning regarding the practice of your profession and promote an ethical approach to practice of the profession.

(c) Differentiate between user and system requirements. Also, give examples for the same. [4] CO1 L2

[Definitions- 2x1M, example 2x1M]

User requirements — These are high-level abstract statements, in a natural language plus diagrams, of what services the system is expected to provide to system users & the constraints under which it must operate.

Eg:- The MHC-PMS shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System Requirements — These are more detailed descriptions of the software system's functions, services, and operational constraints. It may be part of the contract b/w the system buyer and the software developers.

Eg— On last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.

If drugs are available in different dose units (eg. 10mg, 20mg) separate reports shall be created for each dose unit.

Access to all cost reports shall be restricted to authorised users listed on a management access control list.

OR

2 (a) Define Software. Explain two different types of software. [3] CO1 L1

Definition of Software- 1M, Types- 2x1M]

1) What is Software?

It is a collection of computer programs, configuration files (used to set up these programs), system documentation (describes the structure of the system), user documentation (explains how to use the system) and web sites for users to download recent product information.

Generic Product	Customised (bespoke) product
• Stand alone	• Software developed for a particular customer
• produced by development organisation.	• developed on order by customer.
• Sold on open market to any customer who is able to buy it.	• Software specification is controlled by customer
• Software specification is controlled by developing organization.	• Eg- Air traffic control system.
• Eg- databases, word processors, etc.	

(b) Write a note on: [9]

(i) Interviews (ii) Scenarios (iii) Ethnography

[Interviews- 3M, Scenarios- 3M, Ethnography- 3M]

Interviewing

formal and informal interviews with system stakeholders.

Interview types

Closed	Open	Hybrid
• predefined set of questions	• no predefined agenda	• mixed

Interviews are good for getting an overall understanding of what stakeholders do and about their interaction with system but not so good for understanding requirements from application domain because :-

- All application specialists use terminology & jargon that is specific to a domain
- Some domain knowledge is so familiar to stakeholders that they either find it difficult to explain or they think it is so fundamental that is isn't worth mentioning.

Scenarios

These can be useful for adding detail to an outline requirement description. They are descriptions of example interaction sessions. Each scenario covers one or more possible interactions.

A scenario may include :-

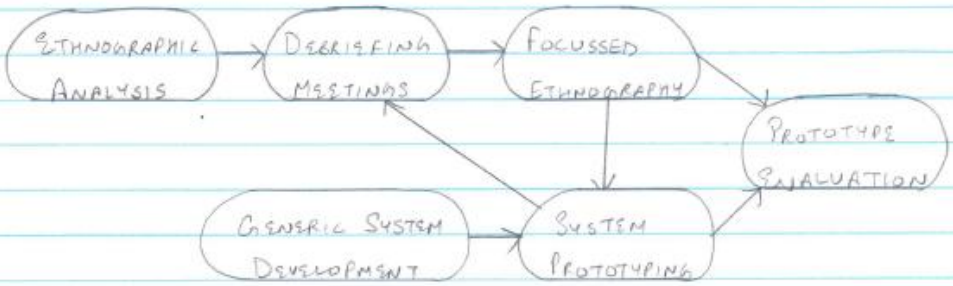
CO1 L2

- A description of what system and users expect when the scenario starts
- a description of normal flow of events in scenario.
- a description of what can go wrong and how this is handled.
- Information about other activities that might be going on at same time.
- a description of system state when the scenario finishes.

Ethnography

It is an observational technique that can be used to understand social and organizational requirements. An analyst ~~enters~~ immerses him/herself in the working environment where the system will be used. He/she observes the day-to-day work and notes made of actual tasks in which participants are involved. It is particularly effective at discovering two types of requirements :-

- Requirements that are derived from the way in which people actually work rather than the way in which process definitions say they ought to work.
- Requirements that are derived from cooperation and awareness of other people's activities.



Ethnography and Prototyping for Req. Analysis

(c) Explain attributes of good software. [4] CO1 L1

[Attributes- 4x1M if defined, 8x0.5M if not defined]

7) What are attributes of a good software?

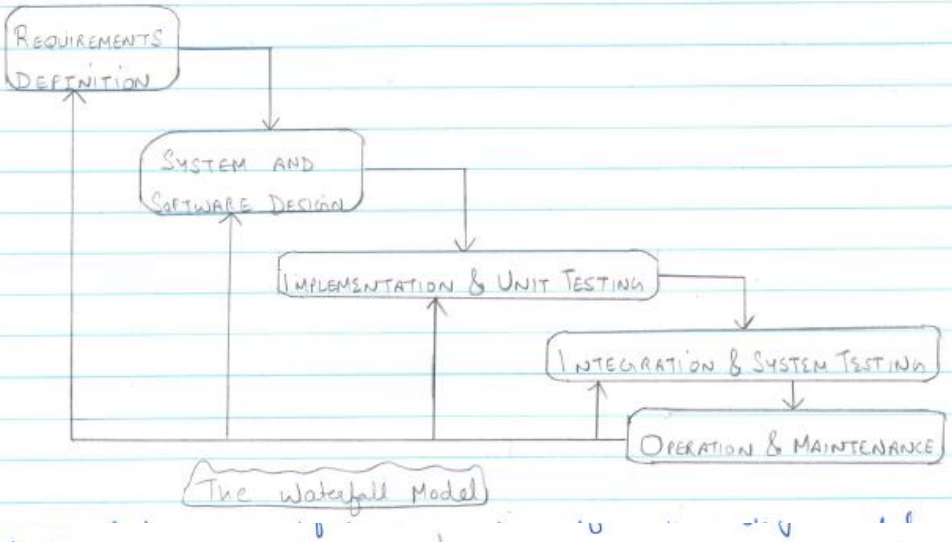
The software should:-

- deliver functionality as specified by customer
- deliver expected performance to the user
- maintainability – easy to evolve in changing requirements / environment.
- dependability and security – it includes range of characteristics including reliability, security, and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
- Efficiency – software should make efficient use of available resources like memory and processor cycles. It includes responsiveness, processing time, memory utilisation, etc.
- acceptability – Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

PART B

3 (a) With neat diagram, explain the waterfall model of software development process. [6]
[Diagram- 4M, explanation- 2M]

Very Imp WATERFALL MODEL



The Waterfall Model

Steps

(i) Requirements analysis and definition — The system's services, constraints and goals are established by consultation with system users. They are then defined in detail & serve as a system specification.

(ii) System and software design — It partitions the requirements to either hardware or software systems. It establishes an overall system architecture. Software design involves identifying &

CO1 L2

describing the fundamental software system abstractions & their relationships.

(iii) Implementation and Unit Testing — Software design is realised as a set of programs. Unit testing verifies that each unit meets its specification.

(iv) Integration and System Testing — The individual program units / programs are integrated and tested as a complete system to ensure that the software requirements have met.

(v) Operation and maintenance — The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of life cycle, improving & enhancing system's services as new requirements are discovered.

Advantages

- Documentation
- Sequential flow easy to understand

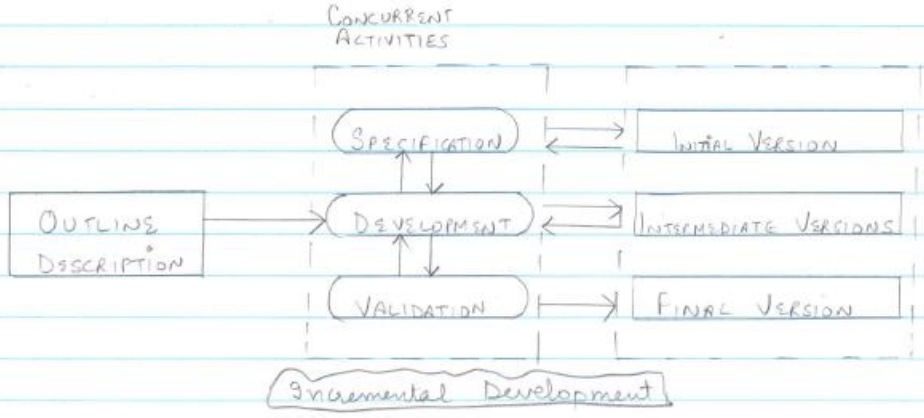
Disadvantages

- Costs of producing & approving documents is high so iterations are costly and involve significant rework.
- Premature freezing may lead to badly structured systems.
- Partioning of project into distinct stages is inflexible
- Difficult to respond to changing customer requirements.

Use

(b) With neat diagram, explain the incremental model of software development process. [6]
Diagram- 4M, explanation- 2M]

Incremental development is based on the idea of developing an initial implementation, getting user feedback on this and evolving it through several versions until an adequate system has been developed.



Incremental development reflects the way we solve problems We move towards a solution in a series of steps, backtracking when we realize that we have made a mistake.

CO1 L2

Each increment or version of the system incorporates some of the functionality that is needed by the customer. Generally, early increments include the most important or most urgently required functionality.

Benefits

- The cost of accommodating changing customer requirements is reduced.
- It is easier to get feedback from customer & easy to make them understand about progress of project.
- Customers are able to use the software even if all of the functionality has not been included.

Problems

- The process is not visible. Managers need regular deliverables to measure progress
- System structure tends to degrade as new increments are added. Regular changes tend to corrupt the structure of system if time & money is not spent on refactoring.

OR

4 (a) Define fundamental activities of software engineering. [4]
[activities- 4x1M]

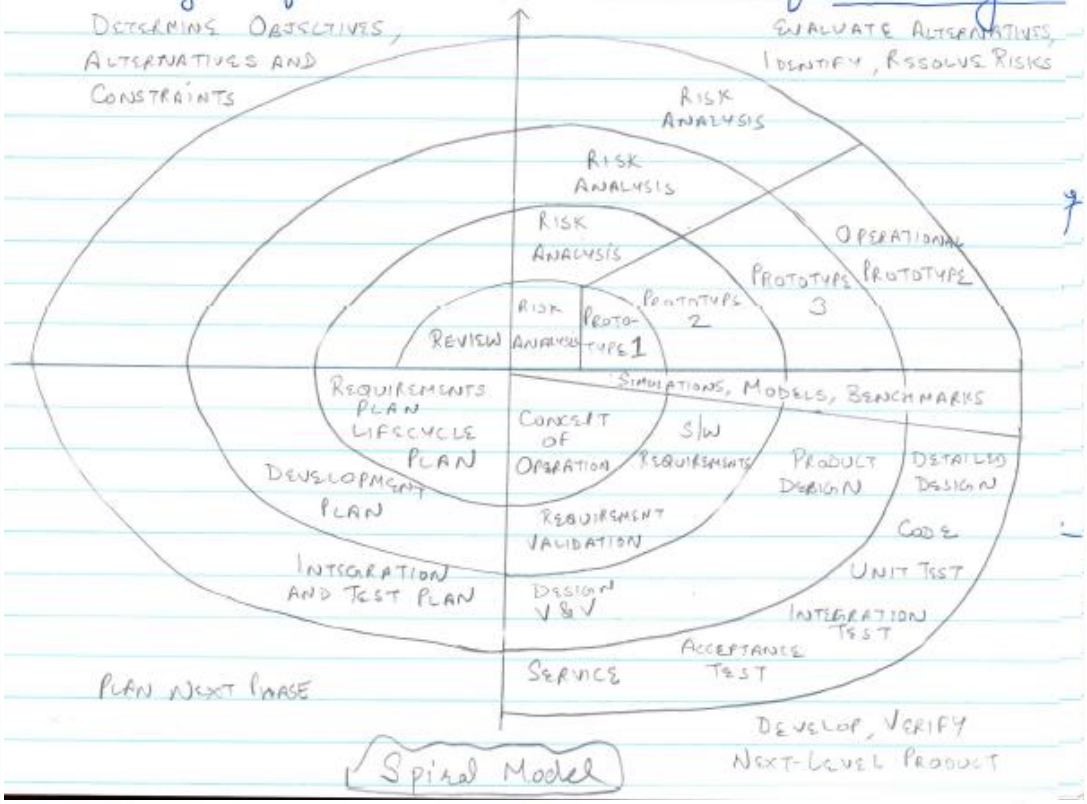
Four fundamental process activities that are common to all software processes are :-

(i) Software specification – define s/w to be produced & constraints on its operation.
(ii) Software development – S/W design and programming.
(iii) Software ~~development~~ validation – check whether software confirms to customer requirements.
(iv) Software evolution – Modifying s/w to adapt it to changing customer and market requirements.

	CO1	L1
(b) Explain Spiral model in detail. [8] [Diagram- 6M, explanation- 2M]	CO1	L2

SPIRAL MODEL

It was proposed by Boehm in 1988. In this, the process is represented as a spiral. Each loop in the spiral represents a phase of the software process. Most important advantage of this model is involvement of risk management



Spiral Model

Each loop in the spiral is split into four sectors:

(i) Objective setting — Specific objectives, constraints, project risks for that phase of project are defined. Alternative strategies depending on these risks may be planned.

(ii) Risk assessment and reduction — for each of the identified project risks, a detailed analysis is carried out and steps are taken to reduce the risk. Risks cause schedule and cost overruns.

(iii) Development and validation — After risk evaluation, a development model for the system is chosen.

(iv) Planning — The project is reviewed and a decision is made whether to continue with a further loop of the spiral. If it is decided to continue, plans are drawn up for the next phase of the project.

		MARKS	CO	RBT
	PART C			
5 (a)	Explain various ways of writing system requirements. [5 ways- 5x1M]	[5]	CO4	L2

System requirements may use following notations :-
Ways of writing a system requirements specification :-

Notation	Description
1. Natural language sentences	Requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
2. Structured natural language	Requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
3. Design description languages	Uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of system. Rarely used except for interface specifications
4. Graphical notations.	Graphical models, supplemented by text annotations. Eg. UML use case, sequence diagrams.
5. Mathematical specifications.	Based on mathematical concepts such as finite-state machines or sets. Difficult to understand by customers.

(b) Explain different checks to be carried out during requirement validation process. [5] CO5 L1
[5 checks- 5x1M]

The **checks** include:-

(i) Validity checks — confirm requirements to actual needs.
(ii) Consistency checks — requirements should not conflict.
(iii) Completeness checks — requirements should define all functions, constraints intended by the system user.
(iv) Realism checks — Ensure existing technology can implement these requirements.
(v) Verifiability — you should be able to write a set of tests that can demonstrate that the delivered system meets each specified requirement.

OR

6 (a) Write the structure of requirements document as specified by IEEE. [10] CO4 L2
[Structure- 10x1M]

Structure of requirements document based on IEEE standard :-

Chapter	Description
• Preface	Expected readership, version history, rationale for creation of new version and summary of changes in each version.
• Introduction	Need of the system, brief functions, working, alignment of system with business objectives.
• Glossary	Technical terms
• User requirements definition	Functional and non-functional requirements
• System architecture	High-level architecture and modules/components
• System requirements specification	Detailed functional and non-functional requirements
• System models	object models, data flow models and semantic data models.
• System evolution	fundamental assumptions on which system is based and anticipated changes.
• Appendices	Detailed, specific information regarding application being developed. H/w, DB regs. etc
• Index	Alphabetic index, diagram index and index of functions.

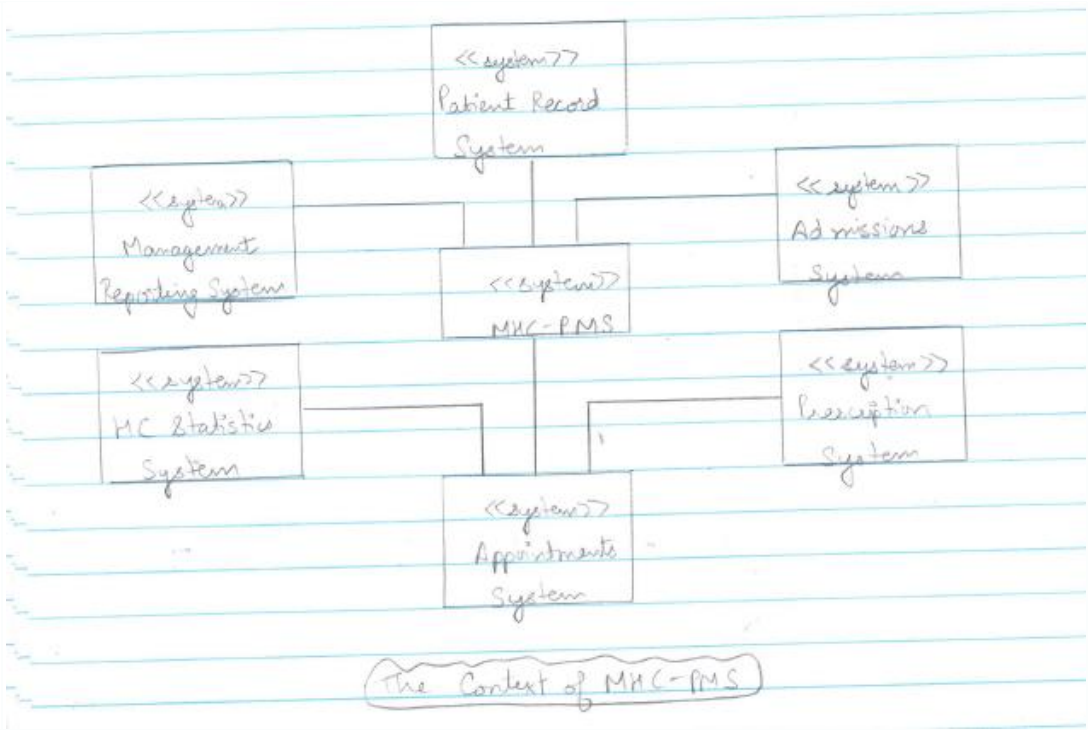
[6] CO5 L3

PART D

7 (a) Explain context models with context diagram for MHC-PMS.
[Diagram-4M, explanation-2M]

CONTEXT MODELS

- Context models are used to illustrate the operational context of a system – they show what lies outside the system boundaries.
- Social and organisational concerns may affect the decision on where to position the system boundaries.
- External systems might produce data for or consume data from the system.
- Architectural models help to define context and dependancies that a system has on its environment.
- Context models are used along with other models, like business process models.



The Context of MHC-PMS

(b)	Explain the term class diagram, generalization and aggregation. [Class diagram- 2M, Generalization- 2M, Aggregation- 2M]	[6]	CO5 L2

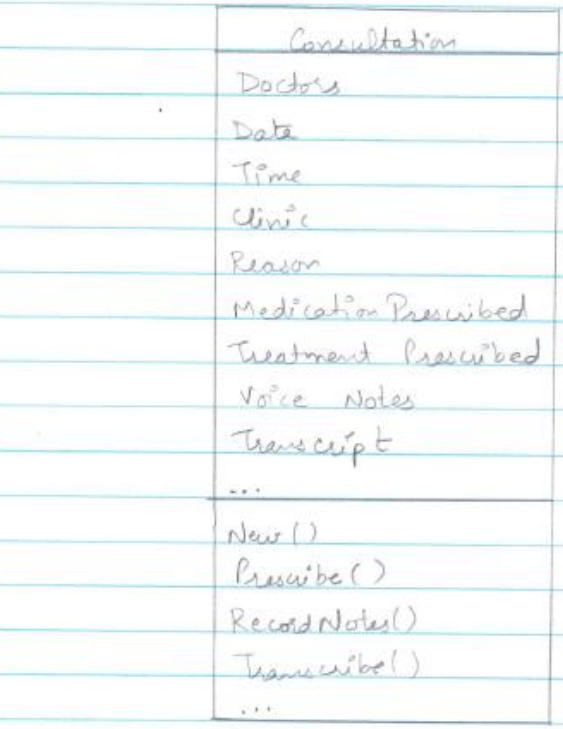
(i) Class diagrams

Class diagrams are used for object-oriented system model. to show the classes in a system and the associations between these classes.

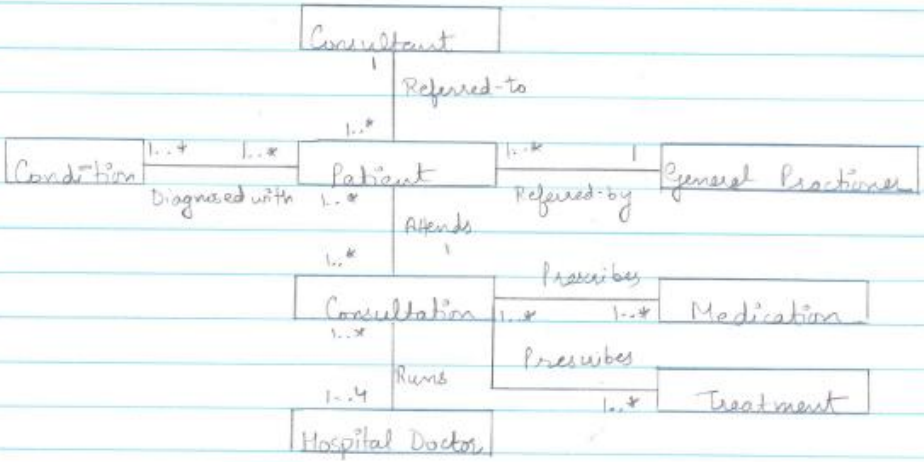
Object class is a general definition of one kind of system object.

An association is a link between classes to indicate relationships.

Class diagrams may be expressed at different levels of detail.



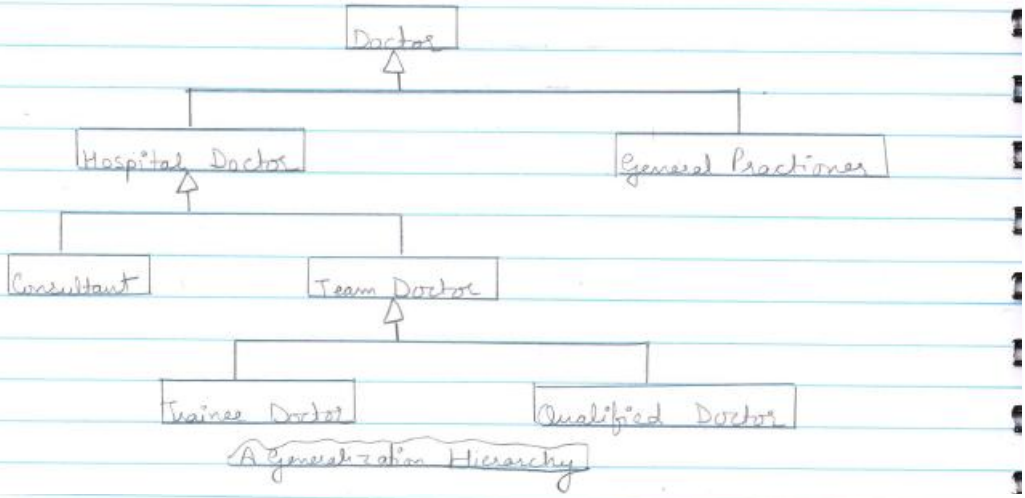
The Consultation class



Class Diagram MHC-PMS

(ii) Generalization

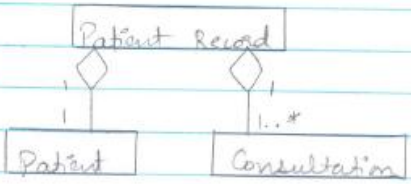
Features that are common to a set of classes are put in a general class which acts as a parent of specialised classes. This ensures that common information is maintained in only one place and helps to manage complexity.



A Generalization Hierarchy

(iii) Aggregation

Objects in the real world are often composed of different parts. Aggregation is a special type of association between classes ~~set~~ that means that one object (the whole) is composed of other objects (the parts). Diamond shape next to the class represents the whole.

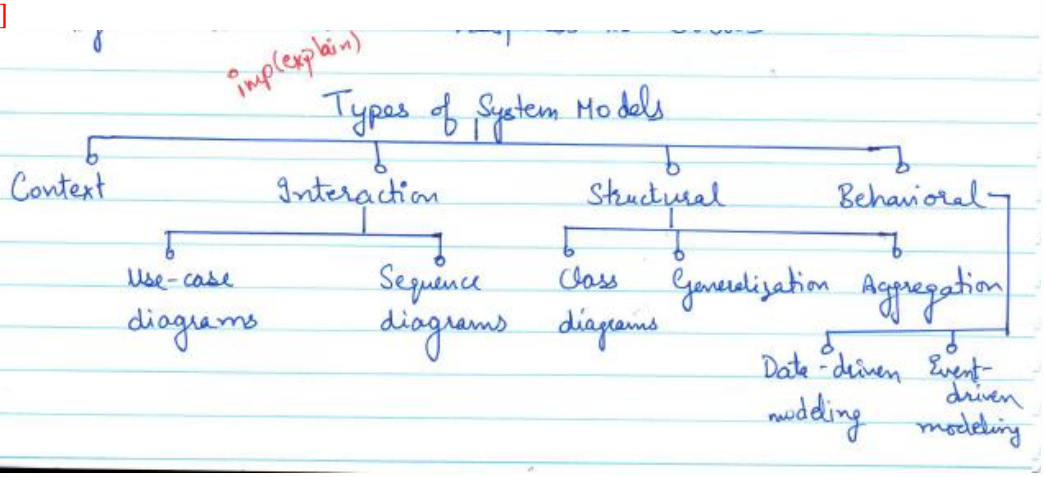


OR

8 (a) Explain types of system models. [8]

[Context models-2M, Interaction models-2M, Structural models-2M, Behavioral models-2M]

imp(explain)



CO4 L2

→ External perspective - model context or environment of system
→ Interaction perspective - model interactions b/w system and its environment or between components of a system.
→ Structural perspective - model organization of system or structure of data processed by system.
→ Behavioral perspective - model dynamic behaviour of system and how it responds to events

Context models: Represent external perspective of the system.
Interaction models: Represent interaction perspective of the system.
Use case diagrams: model user and system interactions
Sequence diagrams: model object/ component interactions.
Structural models: Represent structural perspective of the system.
Class diagram: Represent set of classes and associations.
Generalization: Represent class hierarchy.
Aggregation: Represent whole-part relationship among classes.
Behavioral models: Represent behavioral perspective.
Data-driven modeling: Represent how data is transformed from input to output. Examples include DFD and sequence diagram.
Event driven modeling: Represent event/responses, Example state diagrams

(b) What is model driven engineering? State three types of abstract system models produced. [4]
[Definition- 1M, Models- 3x1M]

MODEL-DRIVEN ENGINEERING

Model-driven engineering (MDE) is an approach to software development where models rather than programs are the principal outputs of the development process. The programs that execute on a hardware/software platform are then generated automatically from the models.

Types of model

(i) Computation independent model (CIM)
These model the important domain abstractions used in a system. They are also called domain models.

(ii) Platform independent model (PIM)
These model the operation of the system without reference to its implementation. It is usually described using models that show the static system structure and how it responds to external and internal events.

(iii) Platform specific models (PSM)
These are transformations of the platform-independent model with a separate PSM for each application platform. There may be layers of PSM, with each layer adding some platform specific detail.

CO4 L1