CMR INSTIT UTE OF TECHNLOGY

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

First Internal Test-SEPTEMBER 2019

Sub:	**Software Engineering**					Sub Code:	18CS35	Branch:	**ISE/CSE**
Date:	06/09/2019	Duration:	90 min's	Max Marks:	50	Sem / Sec:	III A ,B,C		OBE

Scheme and Solution

MARKS

1 (a) What are the Attributes of Good software? [04M]

Maintainability

Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.

Dependability and security

Software dependability includes a 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 not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.

Acceptability

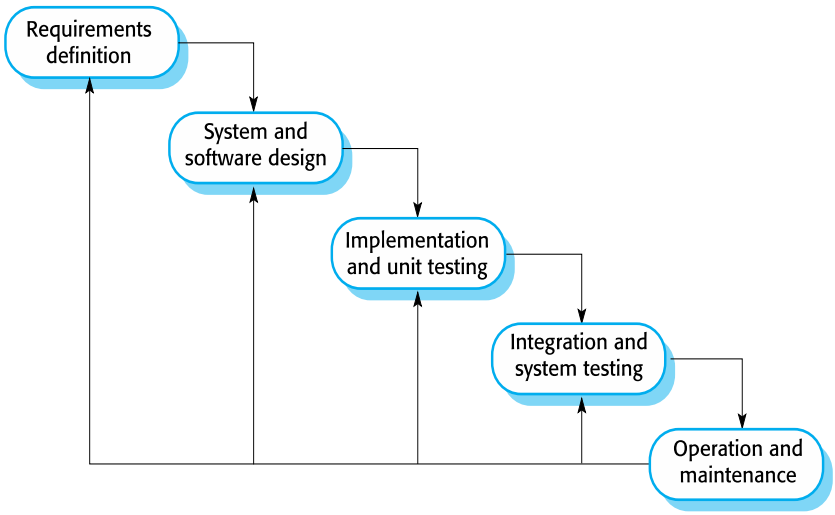
Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

1 (b) Draw the neat diagram, explain the waterfall model of software development process [06 marks]

✧There are separate identified phases in the waterfall model:

- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance

✧The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.



✧Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
- Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
- Few business systems have stable requirements.

✧The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
- In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

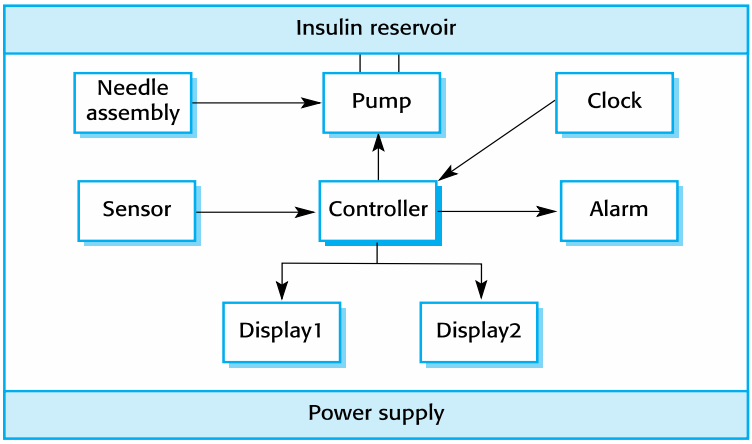
2 (a) Explain an Insulin pump control system with a neat block diagram. [8M]

Collects data from a blood sugar sensor and calculates the amount of insulin required to be injected.

✧Calculation based on the rate of change of blood sugar levels.

✧Sends signals to a micro-pump to deliver the correct dose of insulin.

✧Safety-critical system as low blood sugars can lead to brain malfunctioning, coma and death; high-blood sugar levels have long-term consequences such as eye and kidney damage.



2 (b) List the fundamental activities of Software Engineering.[2M]

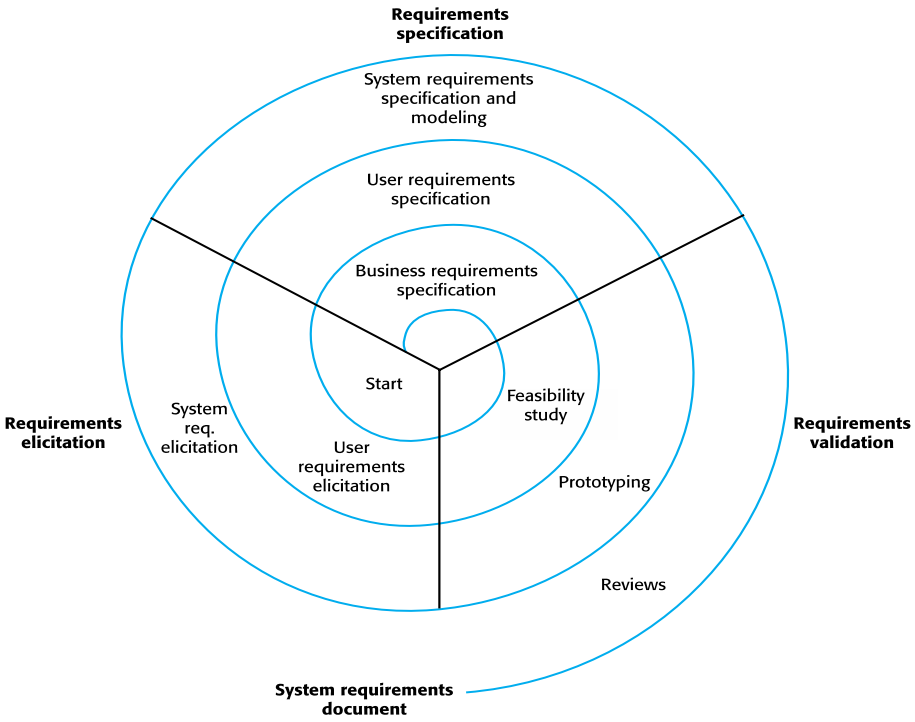
Software specification, software development, software validation and software evolution.

3 (a) With a neat diagram, explain the Requirement Engineering Process. [06M]

✧ The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements.

✧ However, there are a number of generic activities common to all processes
- Requirements elicitation;
- Requirements analysis;
- Requirements validation;
- Requirements management.

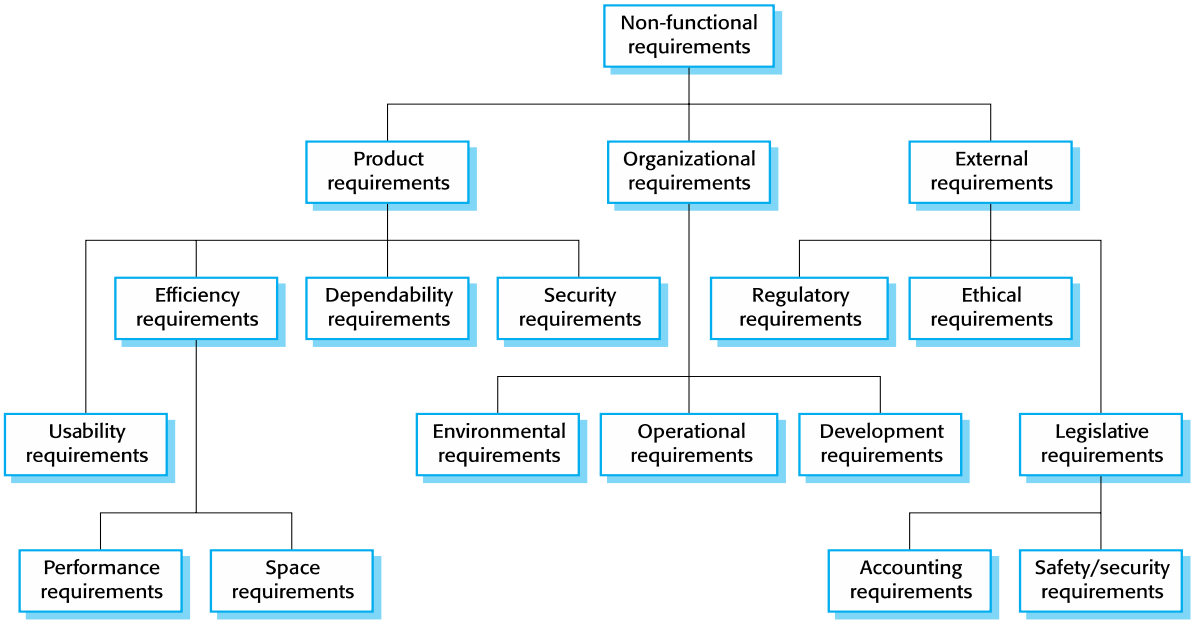
✧ In practice, RE is an iterative activity in which these processes are interleaved.



3 (b) Differentiate between Functional and Non-Functional Requirements with an example for each..[04 M]

- ✧ Functional requirements
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - May state what the system should not do.
  - Describe functionality or system services.
  - Depend on the type of software, expected users and the type of system where the software is used.
  - Functional user requirements may be high-level statements of what the system should do.
  - Functional system requirements should describe the system services in detail.

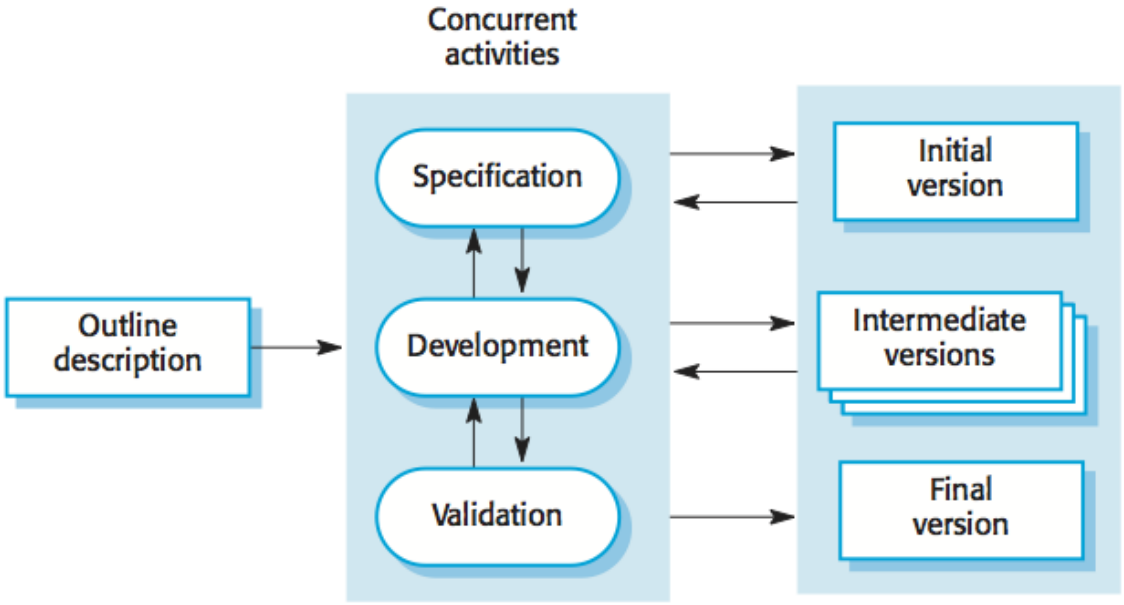
- ✧ Non-functional requirements
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.
  - These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
  - Process requirements may also be specified mandating a particular IDE, programming language or development method.
  - Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.



- ✧ Product requirements
  - ▪ Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- ✧ Organisational requirements
  - ▪ Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- ✧ External requirements
  - ▪ Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

4 (a) Draw block diagram for illustrating incremental development model. State minimum two benefits and problems in the incremental development model. [7M]

Incremental development model



Benefits of incremental development:

Lower cost of changes

The cost of accommodating changing customer requirements is reduced. The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.

Frequent feedback

It is easier to get customer feedback on the development work that has been done. Customers can comment on demonstrations of the software and see how much has been implemented.

Faster delivery

More rapid delivery and deployment of useful software to the customer is possible. Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Problems with incremental development (from the management perspective):

The process is not visible

Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.

System structure tends to degrade as new increments are added

Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

4 (b) Explain the different stages in Software Testing. [3M]

- ✧ Component testing
  - Individual components are tested independently;
  - Components may be functions or objects or coherent groupings of these entities.
- ✧ System testing
  - Testing of the system as a whole. Testing of emergent properties is particularly important.
- ✧ Customer testing
  - Testing with customer data to check that the system meets the customer's needs

5 (a) Elaborate the major themes that are supported in object oriented technology.[06M]

Several themes pervade OO technology. Few are -

1. **Abstraction**

Abstraction lets you focus on essential aspects of an application while ignoring details i.e focusing on what an object is and does, before deciding how to implement it. It's the most important skill required for OO development.

2. **Encapsulation (information hiding)**

It separates the external aspects of an object (that are accessible to other objects) from the internal implementation details (that are hidden from other objects). Encapsulation prevents portions of a program from becoming so interdependent that a small change has massive ripple effects.

3. **Combining data and behaviour**

Caller of an operation need not consider how many implementations exist. In OO system the data structure hierarchy matches the operation inheritance hierarchy (fig).

4. Sharing

OO techniques provide sharing at different levels. Inheritance of both data structure and behaviour lets sub classes share common code. OO development not only lets you share information within an application, but also offers the prospect of reusing designs and code on future projects.

5. **Emphasis on the essence of an object**

OO development places a greater emphasis on data structure and a lesser emphasis on procedure structure than functional-decomposition methodologies.

6. **Synergy**

Identity, classification, polymorphism and inheritance characterize OO languages.

5 (b) Which are the models used to describe a system from different viewpoints? [04M]

THE THREE MODELS

1. **Class Model**: represents the static, structural, "data" aspects of a system. It describes the structure of objects in a system- their identity, their relationships to other objects, their attributes, and their operations.
Goal in constructing class model is to capture those concepts from the real world that are important to an application.
Class diagrams express the class model.

2. **State Model**: represents the temporal, behavioural, "control" aspects of a system.
State model describes those aspects of objects concerned with time and the sequencing of operations – events that mark changes, states that define the context for events, and the organization of events and states.
State diagram express the state model. Each state diagram shows the state and event sequences permitted in a system for one class of objects. State diagram refer to the other models. Actions and events in a state diagram become operations on objects in the class model. References between state diagrams become interactions in the interaction model.

3. **Interaction model** – represents the collaboration of individual objects, the "interaction" aspects of a system. Interaction model describes interactions between objects – how individual objects collaborate to achieve the behaviour of the system as a whole. The state and interaction models describe different aspects of behaviour, and you need both to describe behaviour fully. Use cases, sequence diagrams and activity diagrams document the interaction
model.

6 With the help of a sample class model explain the following; [10M]

i. Association and Association end name
ii. Qualified association
iii. Association classes
iv. Multiplicity

(i)An **association** is a description of a group of links with common structure and common semantics.E.g. a person *WorksFor* a company. An association describes a set of potential links in the same way that a class describes a set of potential objects.

Association end names

Multiplicity implicitly refers to the ends of associations. For E.g. A one-to many association has two ends an end with a multiplicity of "one"
an end with a multiplicity of "many"
You can not only assign a multiplicity to an association end, but you can give it a
name as well.

Multiplicity vs Cardinality. Multiplicity is a constraint on the size of a collection.Cardinality is a count of elements that are actually in a collection.
Therefore, multiplicity is a constraint on cardinality.

Association classes

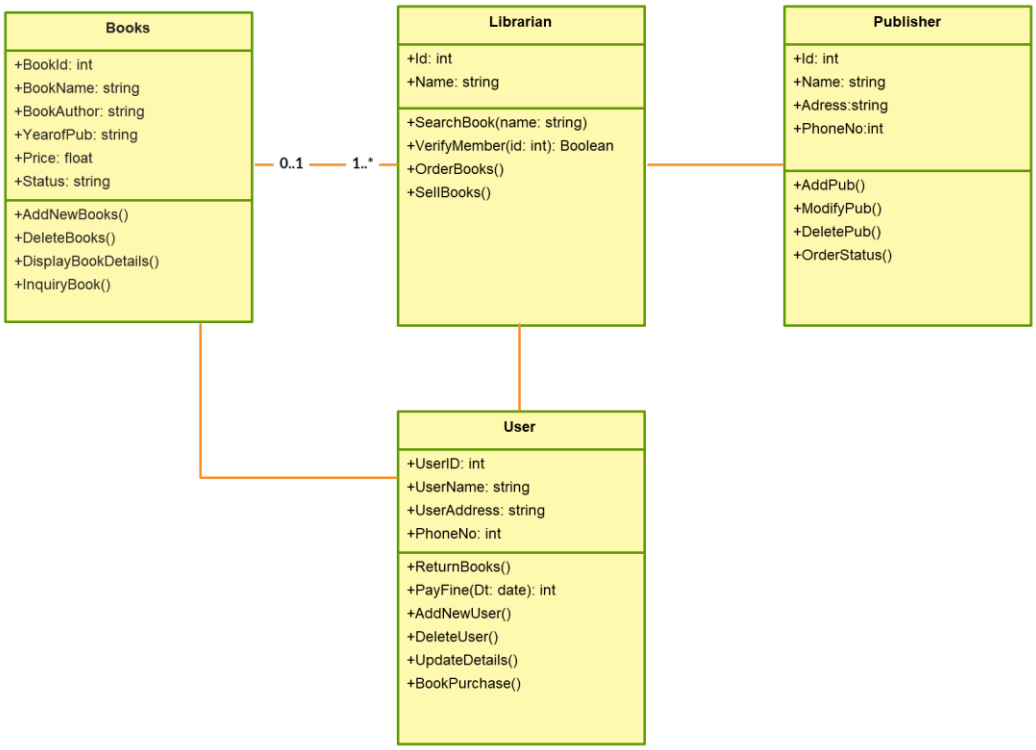
An **association class** is an association that is also a class. Like the links of an association, the instances of an association class derive identity from instances of the constituent classes. Like a class, an association class can have attributes and operations and participate in associations.

Qualified associations

A **Qualified Association** is an association in which an attribute called the **qualifier** disambiguates the objects for a "many" association ends. It is possible to define qualifiers for one-to-many and many-to-many associations. A qualifier selects among the target objects, reducing the effective multiplicity
from "many" to "one".

Ex 1: qualifier for associations with one to many multiplicity. A bank services multiple accounts. An account belongs to single bank. Within the context of a bank, the Account Number specifies a unique account. Bank and account are classes, and Account Number is a qualifier. Qualification reduces effective multiplicity of this association from one-to-many to one-to-one.

7 (a) Design a class diagram for library management system.[03M]



7 (b) Design a class diagram for the following group of classes;School, playground, principal, book, student, teacher, cafeteria, class room,rest room, and computer.
In the class diagram, add minimum three relationships (associations, generalization). Use association names where ever needed and show multiplicity.

http://www.gtucampus.com/uploads/studymaterials/Degree%20Engineeringhiren_patelOODM_CE_Final.pdf