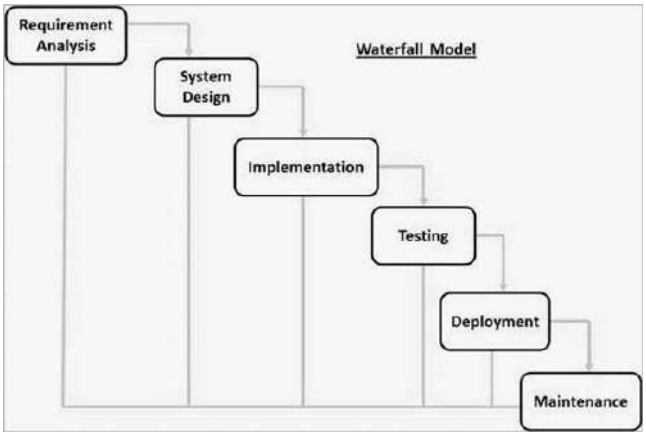
1.	Explain waterfall model with neat diagram? The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The Waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.  **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document. **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture. **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing. **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures. **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market. **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.
2.	Explain incremental model with neat diagram? Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

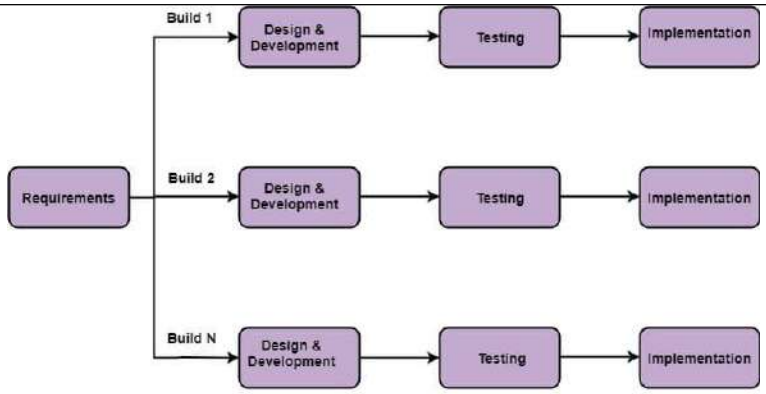


Fig: Incremental Model

1. Requirement analysis: In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

2. Design & Development: In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

3. Testing: In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

4. Implementation: Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

3. Explain with diagram the spiral life cycle model?

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

Spiral Model - Design

Identification

This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

Design

The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent

spirals.

Construct or Build

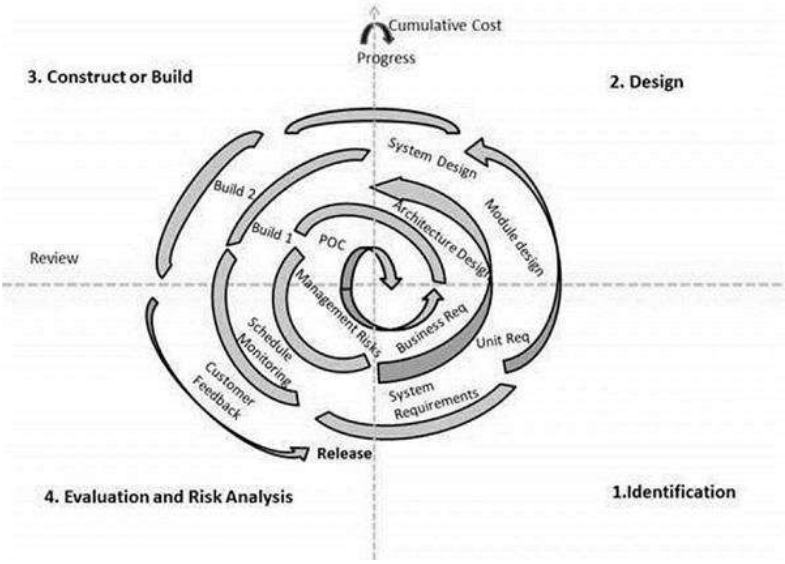
The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

Evaluation and Risk Analysis

Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

The following illustration is a representation of the Spiral Model, listing the activities in each phase.



4. Write short notes on requirement elicitation?

Requirements Elicitation is the process to find out the requirements for an intended software system by communicating with client, end users, system users and others who have a stake in the software system development

Requirement elicitation process can be depicted using the folloiwng diagram:

Requirements gathering - The developers discuss with the client and end users and know their expectations from the software.

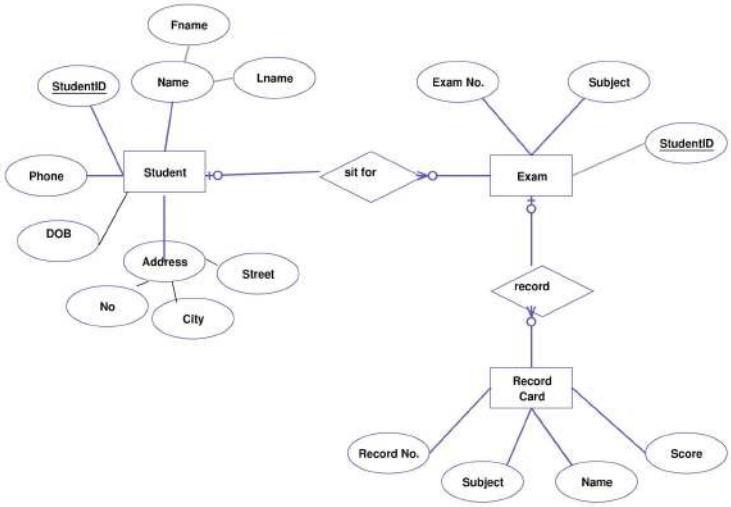
Organizing Requirements - The developers prioritize and arrange the requirements in order of importance, urgency and convenience.

Negotiation & discussion - If requirements are ambiguous or there are some conflicts in requirements of various stakeholders, if they are, it is then negotiated and discussed with

	stakeholders. Requirements may then be prioritized and reasonably compromised. The requirements come from various stakeholders. To remove the ambiguity and conflicts, they are discussed for clarity and correctness. Unrealistic requirements are compromised reasonably. **Documentation -** All formal & informal, functional and non-functional requirements are documented and made available for next phase processing.
5.	What is software? Explain the types of software with example? Software is a set of programs, which is designed to perform a well-defined function. A program is a sequence of instructions written to solve a particular problem. There are two types of software − • System Software • Application Software **System Software** The system software is a collection of programs designed to operate, control, and extend the processing capabilities of the computer itself. System software is generally prepared by the computer manufacturers. These software products comprise of programs written in low-level languages, which interact with the hardware at a very basic level. System software serves as the interface between the hardware and the end users. Some examples of system software are Operating System **Application Software** Application software products are designed to satisfy a particular need of a particular environment. All software applications prepared in the computer lab can come under the category of Application software. Application software may consist of a single program, such as Microsoft's notepad for writing and editing a simple text. It may also consist of a collection of programs, often called a software package, which work together to accomplish a task, such as a spreadsheet package.
6.	Explain the attributes of good software? Basically there are FOUR (4) attributes of a good software ... **1. Maintainability** The software should be written in a way that it can be evolve to meet changing needs of the customer . Since the change of needs (requirement change of customer) is unavoidable, this is a CRITICAL ATTRIBUTE . **2. Dependability** A software MUST be Trustworthy (can we trust the software ?) , Reliable , Safe and Secured (are we secured using this software ?) . Use of the software should not be harmful to the customer in anyway (even something goes wrong) . Even at a system failure, a dependable software should NOT cause physical or economical damage . Also malicious users should not be able to access or damage the system . **3. Efficiency**

	A software should be efficient in everyway . The software should not make wasteful of system resources (ex. memory, processing cycles) . Responsiveness should be there, and the memory utilization should be minimum and also the processing time is a consideration . **4. Acceptability** The software must be acceptable to the group of users for which it's designed for . Software should be understandable , reliable and compatible with other systems they use .
7.	Software processes **Software** is the set of instructions in the form of programs to govern the computer system and to process the hardware components. To produce a software product the set of activities is used. This set is called a software process. There are four basic key process activities: 1. **Software Specifications:** In this process, detailed description of a software system to be developed with its functional and non-functional requirements. 2. **Software Development:** In this process, designing, programming, documenting, testing, and bug fixing is done. 3. **Software Validation:** In this process, evaluation software product is done to ensure that the software meets the business requirements as well as the end users needs. 4. **Software Evolution:** It is a process of developing software initially, then timely updating it for various reasons.
8.	Software process model A Process Model describes the sequence of phases for the entire lifetime of a product. Therefore it is sometimes also called Product Life Cycle. This covers everything from the initial commercial idea until the final de-installation or disassembling of the product after its use. Usually there are three main phases: • concept phase • implementation phase • maintenance phase Each of these main phases usually has some sub-phases, like a requirements engineering phase, a design phase, a build phase and a testing phase. The sub-phases may occur in more than one main phase each of them with a specific peculiarity depending on the main phase. Besides the phases a Process Model shall also define at least: • The **activities** that have to be carried out in each of the sub-phases, including the sequence in which these activities have to be carried out. • The **roles** of the executors that have to carry out the activities, including a description of their responsibilities and required skills. • The **work products** that have to be established or updated in each of the activities. Besides the final product there are usually several other items that have to be generated during the development of a product. These are for example requirements and design document, test

	specifications and test reports, etc. Therefore, a Process Model provides a fixed framework that guides a project in: • Development of the product • Planning and organizing the project • Tracking and running the project
9.	Software engineering Software engineering is defined as a process of analyzing user requirements and then designing, building, and testing software application which will satisfy those requirements.
10.	What is SRS? Describe how to prepare a software requirement specification document? A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase. **Qualities of SRS:** Correct Unambiguous Complete Consistent Ranked for importance and/or stability Verifiable Modifiable Traceable  1. Purpose 2. Scope 3. System Overview 4. References 5. Definitions 6. Use Cases

	7. Functional requirements 8. Non-functional requirements
11.	Write short notes on Agile Development methods? Agile is a time-bound, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver all at once. **Why Agile?** Technology in this current era is progressing faster than ever, enforcing the global software companies to work in a fast-paced changing environment. Because these businesses are operating in an ever-changing environment, it is impossible to gather a complete and exhaustive set of software requirements. Without these requirements, it becomes practically hard for any conventional software model to work. The conventional software models such as Waterfall Model that depends on completely specifying the requirements, designing, and testing the system are not geared towards rapid software development. As a consequence, a conventional software development model fails to deliver the required product.
12.	What is ER diagram? Draw an ER diagram for student management system? An **entity–relationship model** (or **ER model**) describes interrelated things of interest in a specific domain of knowledge. A basic ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between entities (instances of those entity types). 
13	Cardinality **Cardinality** refers to the relationship between a row of one table and a row of another table. The only two options for cardinality are one or many.
14.	Modality As cardinality is the maximum number of connections between table rows (either one or many), **modality** is the least number of row connections! Modality also only has two options, 0 being the least or 1 being the least. Another way to think of this is not required or required. If we have a modality of at least zero, there doesn't have to be a connection at all (nullable). If we have a modality of at least one, then we have to have that connection (not null).

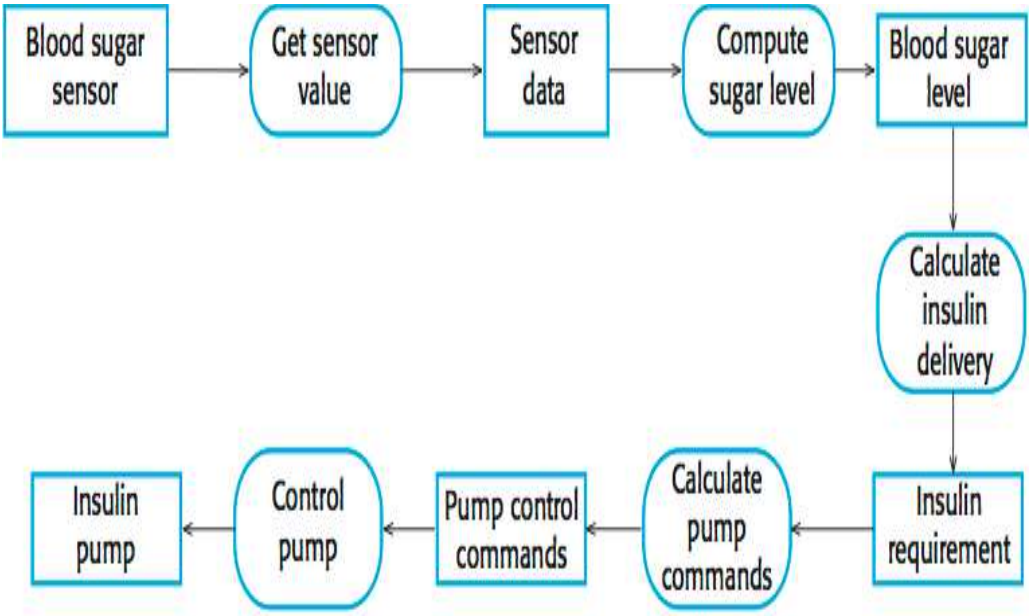
1. write a short note on behavioural models?

Behavioral models

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. Two types of stimuli:

- Some **data** arrives that has to be processed by the system.
- Some **event** happens that triggers system processing. Events may have associated data, although this is not always the case.

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. Data-driven models can be created using UML **activity diagrams**:



2.what is software and its application?

Software is a set of instructions, data or programs used to operate computers and execute specific tasks. Opposite of hardware, which describes the physical aspects of a computer, software is a generic term used to refer to applications, scripts and programs that run on a device. Software can be thought of as the variable part of a computer and hardware the invariable part.

Applications of software

An application is any program, or group of programs, that is designed for the end user. Applications software (also called end-user programs) include such things as database programs, word processors, Web browsers and spreadsheets.

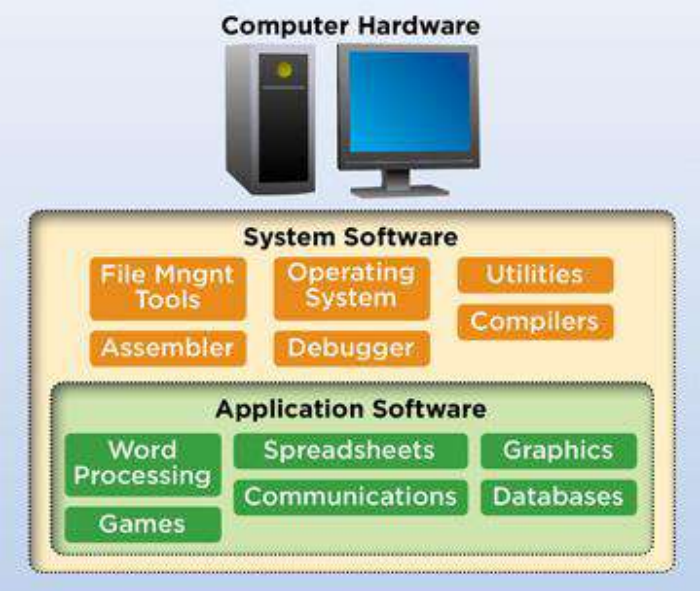


Image: Application Software Diagram

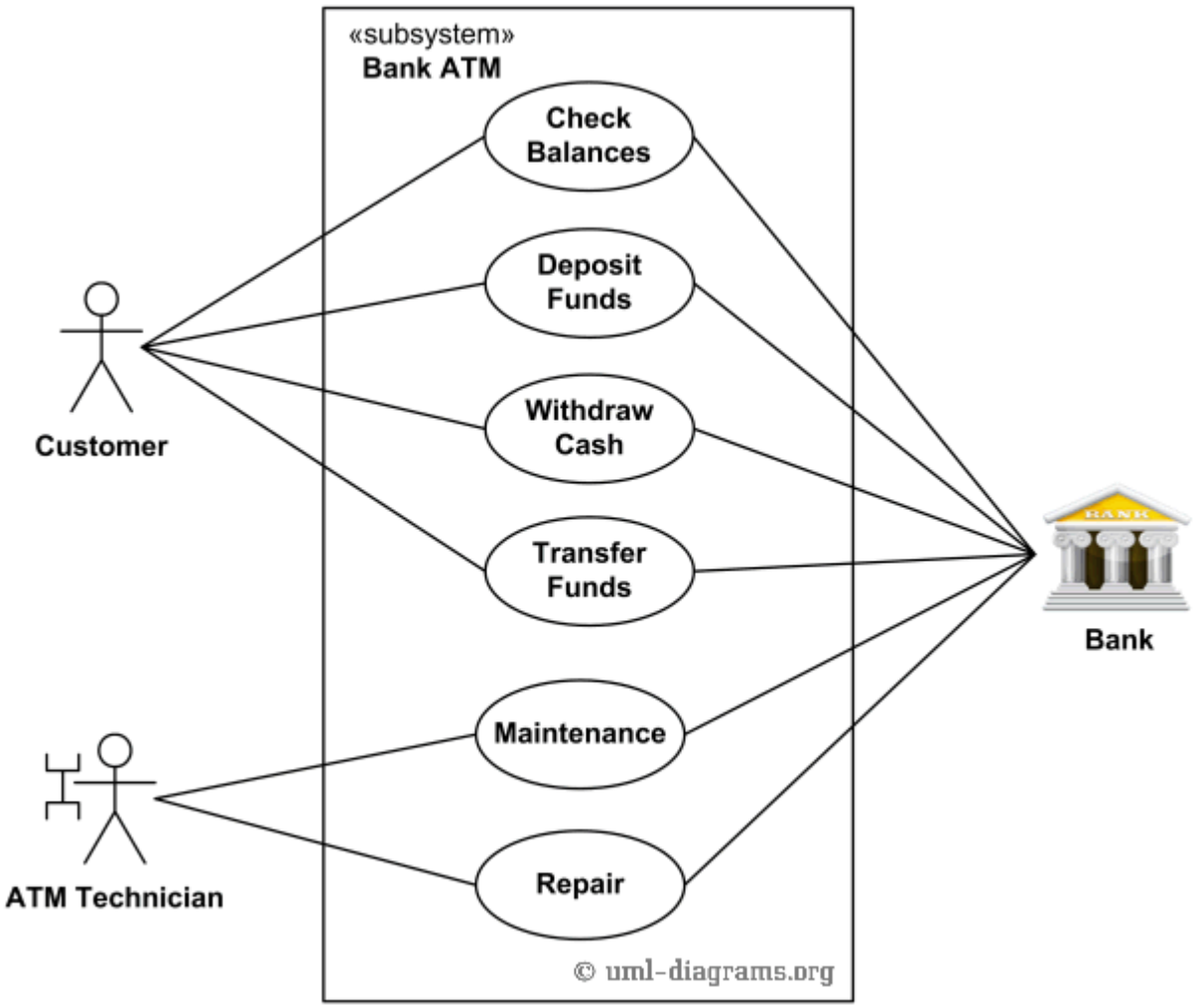
Figuratively speaking, applications sit on top of systems software because they are unable to run without the operating system and system utilities. Systems software consists of low-level programs that interact with the computer at a very basic level. This includes operating systems, compilers, and utilities for managing computer resources

3.Draw the use case diagram for ATM system

Bank ATM

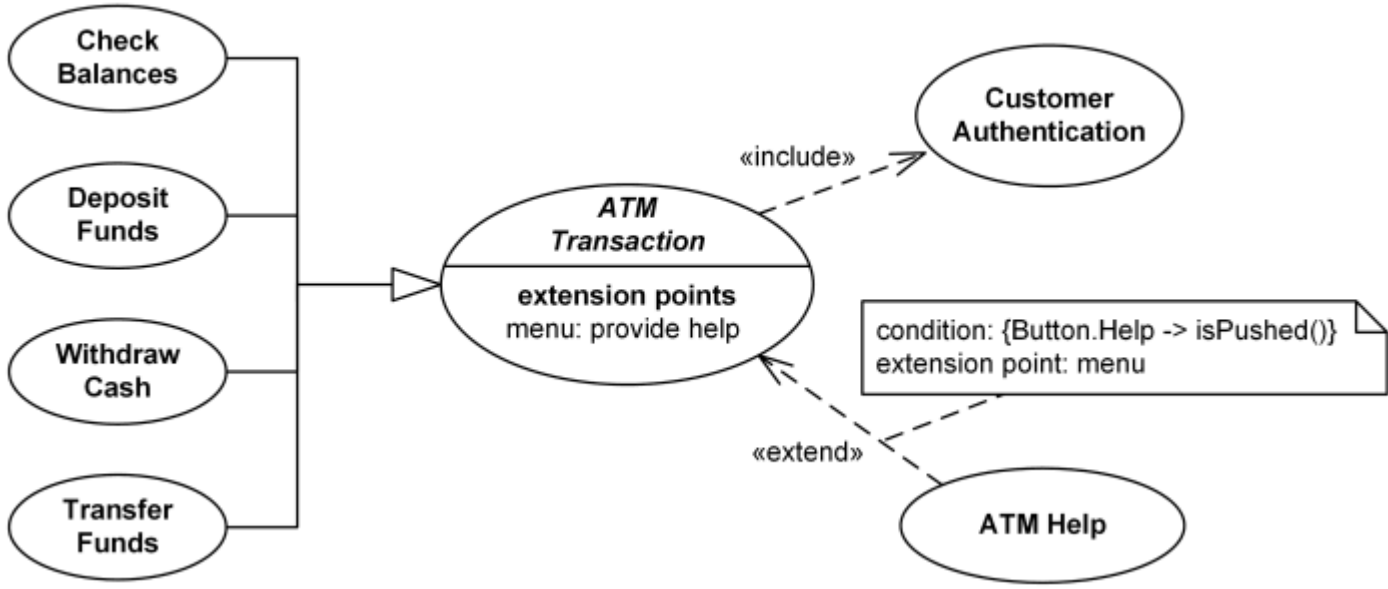
UML Use Case Diagram Examples

An automated teller machine (**ATM**) or the automatic banking machine (**ABM**) is a banking subsystem (**subject**) that provides bank customers with access to financial transactions in a public space without the need for a cashier, clerk, or bank teller.
Customer (**actor**) uses bank ATM to *Check Balances* of his/her bank accounts, *Deposit Funds*, *Withdraw Cash* and/or *Transfer Funds* (**use cases**). *ATM Technician* provides *Maintenance* and *Repairs*. All these use cases also involve *Bank* actor whether it is related to customer transactions or to the ATM servicing.

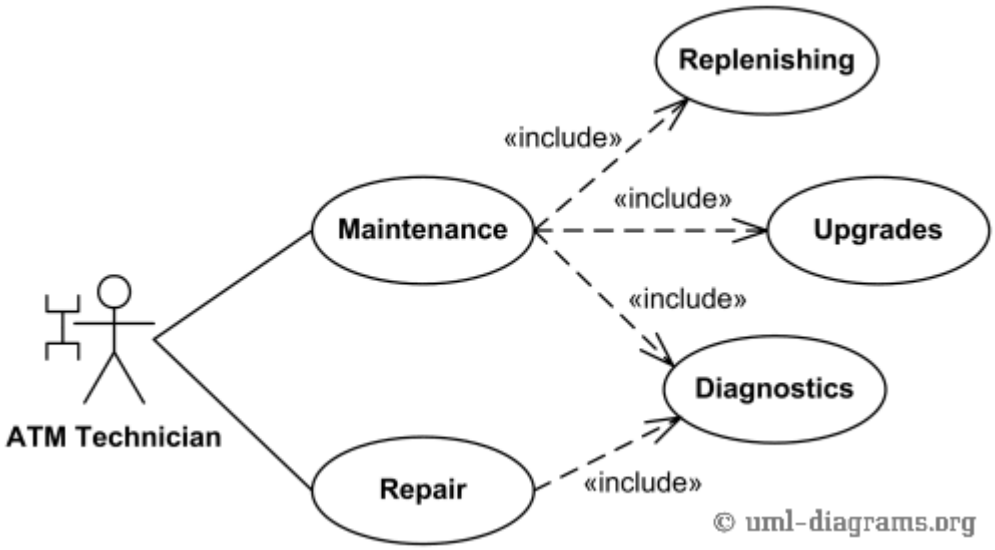


An example of use case diagram for Bank ATM subsystem - top level use cases.

On most bank ATMs, the customer is authenticated by inserting a plastic ATM card and entering a personal identification number (PIN). *Customer Authentication* use case is required for every ATM transaction so we show it as **include** relationship. Including this use case as well as transaction **generalizations** make the *ATM Transaction* an **abstract use case**.



Bank ATM Transactions and Customer Authentication Use Cases Example.
Customer may need some help from the ATM. *ATM Transaction* use case is **extended** via **extension point** called *menu* by the *ATM Help* use case whenever *ATM Transaction* is at the location s



pecified by the *menu* and the bank customer requests help, e.g. by selecting Help menu item.

Bank ATM Maintenance, Repair, Diagnostics Use Cases Example.
ATM Technician maintains or repairs Bank ATM. *Maintenance* use case includes *Replenishing* ATM with cash, ink or printer paper, *Upgrades* of hardware, firmware or software, and remote or on-site *Diagnostics*. *Diagnostics* is also **included** in (shared with) *Repair* use case.

4. .What is the cost of quality? Discuss it

Cost of Quality

DEFINITION

Cost of Quality (COQ) is a measure that quantifies the cost of control/conformance and the cost of failure of control/non-conformance. In other words, it sums up the costs related to prevention and detection of defects and the costs due to occurrences of defects.

- *Definition by ISTQB:* cost of quality: The total costs incurred on quality activities and issues and often split into prevention costs, appraisal costs, internal failure costs and external failure costs.
- *Definition by QAI:* Money spent beyond expected production costs (labor, materials, equipment) to ensure that the product the customer receives is a quality (defect free) product. The Cost of Quality includes prevention, appraisal, and correction or repair costs.

5. What are the factors are required to achieve the software quality?

Correctness - extent to which a program satisfies its specification and fulfills the client's objective.

Reliability - extent to which a program is supposed to perform its function with the required precision.

Efficiency - amount of computing and code required by a program to perform its function.

Integrity - extent to which access to software and data is denied to unauthorized users.

Usability- labor required to understand, operate, prepare input and interpret output of a program

Maintainability- effort required to locate and fix an error in a program.

Flexibility- effort needed to modify an operational program.

Testability- effort required to test the programs for their functionality.

Portability- effort required to run the program from one platform to other or to different hardware.

Reusability- extent to which the program or it's parts can be used as building blocks or as prototypes for other programs.

Interoperability- effort required to couple one system to another.

6. What are the ISO 9126 Quality Factors? Explain.

The purpose of this article is to present an overview of the ISO 9126 standard and to give a detailed description of the software quality model used by this standard.

ISO 9126 is an international standard for the evaluation of software. The standard is divided into four parts which addresses, respectively, the following subjects: quality model; external metrics; internal metrics; and quality in use metrics. ISO 9126 Part one, referred to as ISO 9126–1 is an extension of previous work done by McCall (1977), Boehm (1978), FURPS and others in defining a set of software quality characteristics.

ISO9126–1 represents the latest (and ongoing) research into characterizing software for the purposes of software quality

control, software quality assurance and software process improvement (SPI). This article defines the characteristics identified by ISO 9126–1. The other parts of ISO 9126, concerning metrics or measurements for these characteristics, are essential for SQC, SQA and SPI but the main concern of this article is the definition of the basic ISO 9126 Quality Model.

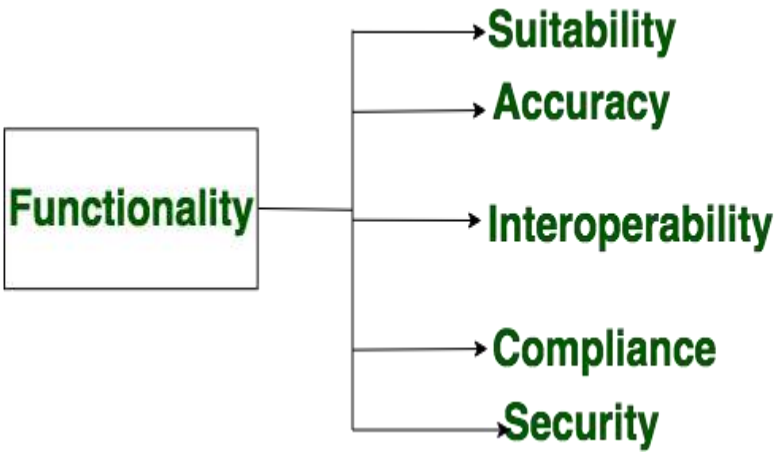
The ISO 9126 documentation itself, from the official ISO 9126 documentation, can only be purchased and is subject to copyright. SQA.net only reproduces the basic structure of the ISO 9126 standard and any descriptions, commentary or guidance are original material based on public domain information as well as our own experience.

The ISO 9126–1 software quality model identifies **6 main quality characteristics**, namely:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

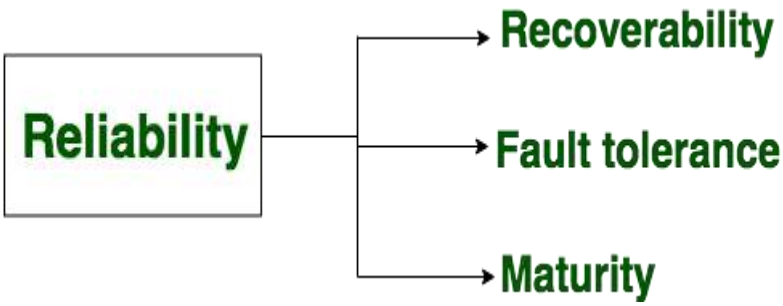
- **Functionality:**
  It refers to the degree of performance of the software against its intended purpose.

Required functions are:



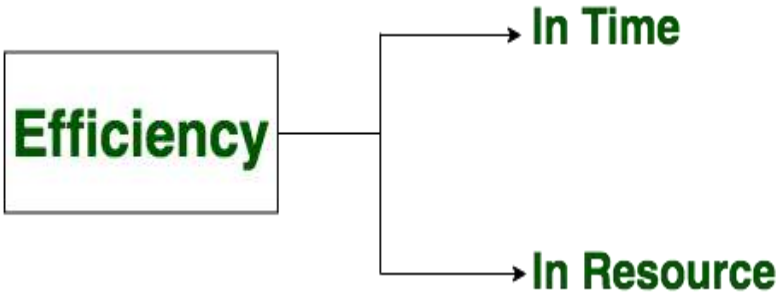
- **Reliability:**
  A set of attribute that bear on capability of software to maintain its level of performance under the given condition for a stated period of time.

Required functions are:



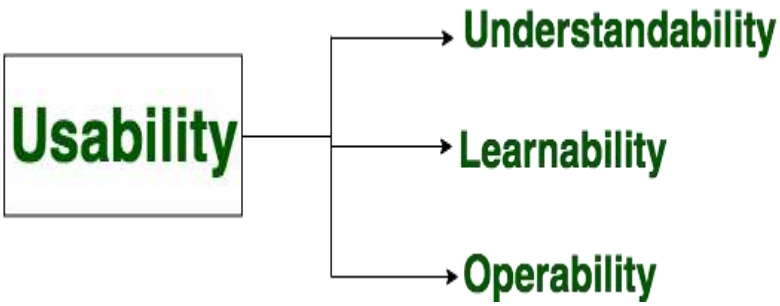
- **Efficiency:**
  It refers to the ability of the software to use system resources in the most effective and efficient manner.the software should make effective use of storage space and executive command as per desired timing requirement.

Required functions are:



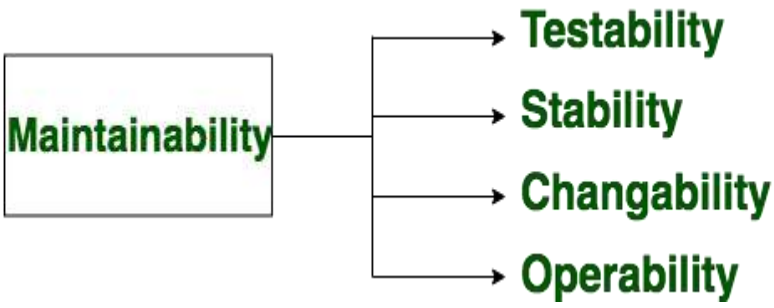
- **Usability:**
  It refers to the extent to which the software can be used with ease.the amount of effort or time required to learn how to use the software.

Required functions are:



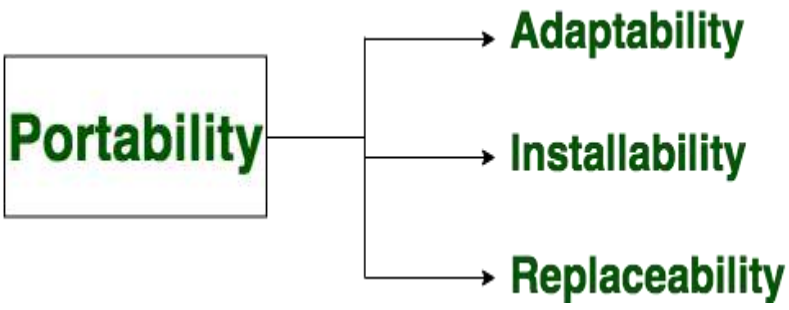
- **Maintainability:**
  It refers to the ease with which the modifications can be made in a software system to extend its functionality, improve its performance, or correct errors.

Required functions are:



- **Portability:**
  A set of attribute that bear on the ability of software to be transferred from one environment to another, without or minimum changes.

Required functions are:



7. What is SQA? What are the goals of SQA?

Software quality assurance (SQA) is a process which assures that all software engineering processes, methods, activities and work items are monitored and comply against the defined standards. These defined standards could be one or a combination of any like ISO 9000, CMMI model, ISO15504, etc.
SQA incorporates all software development processes starting from defining requirements to coding until release. Its prime goal is to ensure quality.

SQA Objectives/Goals

The various objectives of SQA are as follows:

- Quality management approach.
- Measurement and reporting mechanisms.
- Effective software-engineering technology.
- A procedure to assure compliance with software-development standards where applicable.
- A multi-testing strategy is drawn.
- Formal technical reviews that are applied throughout the software process.

The major goals of SQA are as follows:

SQA activities are planned.

- Non-compliance issues that cannot be resolved within the software project are addressed by senior management.
- Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively.
- Affected groups and individuals are informed of SQA activities and results.

8. What is the reference model for technical reviews? Explain it.

The **technical reference** model. The TRM is positioned as a foundation **architecture** within the **architecture** continuum. It defines the components of an information system infrastructure by providing terminology, structure, and rules for interconnection between different components.

What is Technical Reference Architecture?

TOGAF calls it a 'Technical Reference Model', though I've always just heard it called 'technical reference architecture'. What is it? A model or template that describes how the technology standards come together to enable the delivery of solutions. There may be many reference architectures defined, depending on the patter within the enterprise. It's a way to generalize the definition of a 'stack' of resources (hardware, software, net meet specific capabilities. These models can then be used as a basis for the design and delivery of a solution.

As discussed in this week's readings, there are patterns that will emerge as the technical reference architectur together. Transact, publish, collaborate and database patterns are just a few as described in one of the Gartne week. Additional patterns are coming to light around big data, event processing, and mobility. As the patter emerge and technologies shift / mature, additional reference architects will be documented and used as inputs solution's technical/infrastructure architecture. Which patterns will be included in an enterprise' catalog will on its strategic objectives and intended use of technology overall. These 'reference architectures' can be designed/documented in parallel with other architecture efforts, but as a solution architecture begins to come additional patterns may also emerge

While researching, I found this slide deck that outlines the benefits of reference architectures. It's a little dat believe that the benefits still hold true. If you're looking for reasons to establish reference architecture model place to start.

9. Write short notes on class models.

Class Diagram gives the static view of an application. A class diagram describes the types of objects in the system and the different types of relationships that exist among them. This modeling method can run with almost all Object-Oriented Methods. A class can refer to another class. A class can have its objects or may inherit from other classes.

UML Class Diagram gives an overview of a software system by displaying classes, attributes, operations, and their relationships. This Diagram includes the class name, attributes, and operation in separate designated compartments.

Class Diagram helps construct the code for the software application development.

10. Discuss about the evolving role of software.

THE EVOLVING ROLE OF SOFTWARE

Today, software takes on a dual role. It is a product and, at the same time, the vehicle for delivering a product. As a product, it delivers the computing potential embodied by computer hardware or, more broadly, a network of computers that are accessible by local hardware. Whether it resides within a cellular phone or operates inside a mainframe computer, software is an information transformer—producing, managing, acquiring, modifying, displaying, or transmitting information that can be as simple as a single bit or as complex as a multimedia presentation. As the vehicle used to deliver the product, software acts as the basis for the control of the computer (operating systems), the communication of information (networks), and the creation and control of other programs (software tools and environments). Software delivers the most important product of our time—information.

Software transforms personal data (e.g., an individual's financial transactions) so that the data can be more useful in a local context; it manages business information to enhance competitiveness; it provides a gateway to worldwide information networks (e.g., Internet) and provides the means for acquiring information in all of its forms.

11. Explain the software characteristic.

(ANSWER IS SAME AS 6TH QUESTION ANSWER)

12. Defining attributes

(SAME QUESTION REPEATED AS 15TH QUESTION)

END

15) Attributes

* Data Attributes is defined as properties of data objects.

→ They can be use to.
* Name and instance of data object
* Describe the instale
* make a reference to another instance in another table.

16) Data objects

→ The data object is a representation of info that must be understood by the s/w.

→ A data object can be any external entity, a thing and event, a place, or a structure that can be sensed and defined interms of attributes

17) people involved in s/w process?

→ The software teams
* organizations/structure of teams
→ Democratic decentralized
*No permanent leader * communication is horizontal
→ controlled decentralized.
* defined leader * Horizontal communication
* problem slung is group activity
→ controlled centralized
* Def^n team leader * communication is vertical

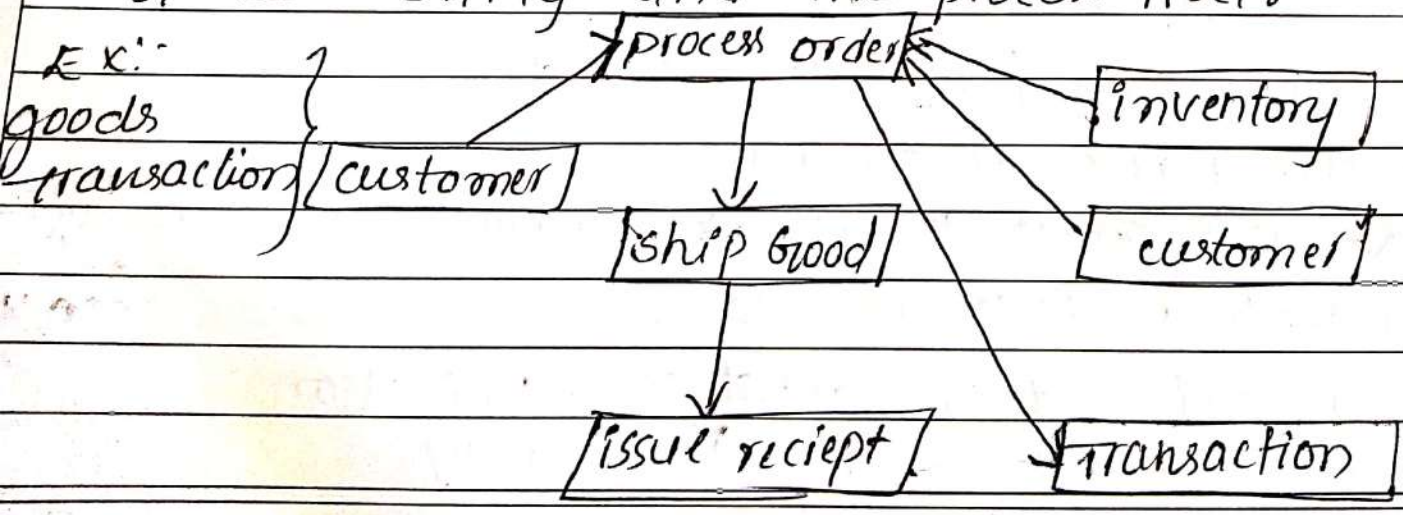
Teacher's Signature : _____

18) d/f approches for collaborative requirement gathering?

-> meetings are conducted and attended by both sw engineers and other stake holders
-> Rules for preperation and participation or establish.
-> An agenda is suggested that is formal enough to cover all the important points
-> Facilitator it can be a customer, developer or a outsider controls the meting
-> A definition mechanism should be used, it can be a ppt, workshops. flipcharts.

19) Data flow dia

* A data flow diagram is a way of representing a flow of a data or process or a system
* The DF-D also provide info about the i/p & o/p of each entity and the process itself



Page No. _____

Expt. No. _____

coupling

It is the measurement of the degree of the independence b/w modules. A good software must have low coupling b/w modules.

Types

1. Data coupling
2. stamp —||—
3. control —|—
4. external —|—
5. common
6. content -|—|-

Cohesol

* It is the measure of the degree to which the elements of module or functionally related.
* It is the degree to which all elements are directed towards performing a single task. in a contained component
* cohesion is a internal connection that keeps the module together
* good s/w will have high efficient

Types

* functional
* Sequential
* Communication
* procedural
* Temporary
* logical
* co-incident

Teacher's Signature :

2. Data centered Arch.

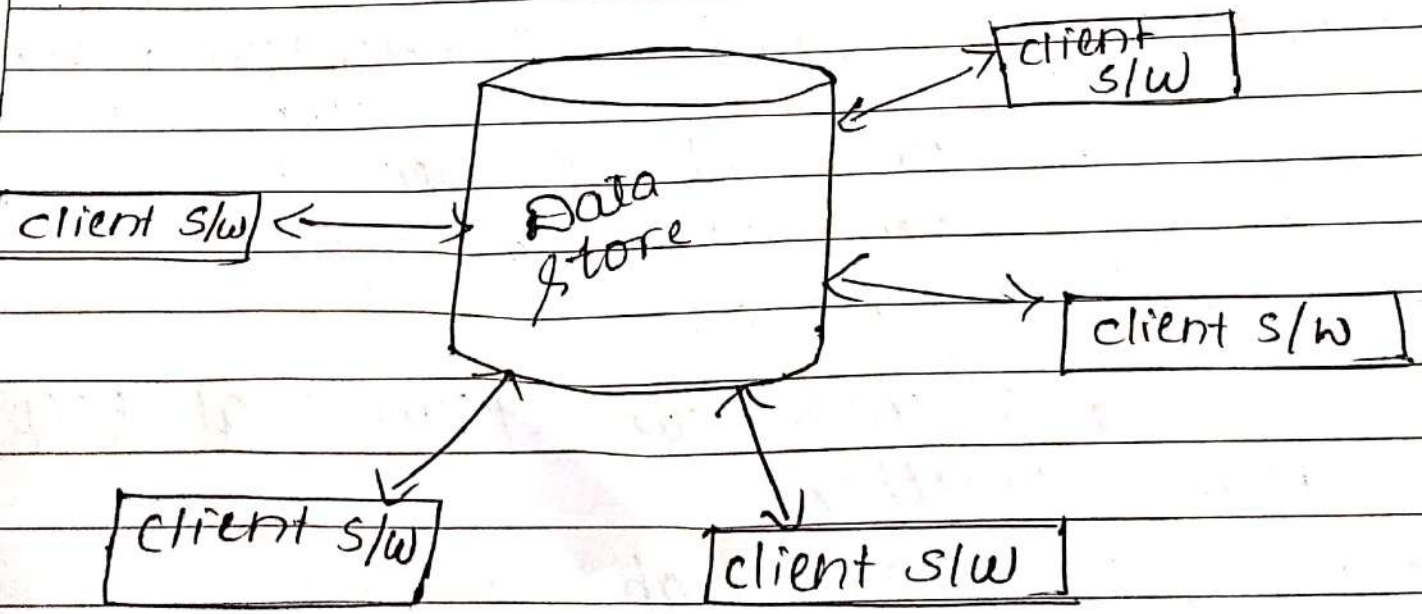
-> In DCA, the data is centralized and accessed frequently by other components, which modify data.

-> The main purpose of this style is to achieve integrality of data

-> It consist of diff components that communicate through shared data

Ex:- Data base Architecture.

In DBA, the common data base schema is created with data defn protocol.



Date.

Page No.

→ Data is only means of communication among the clients the flow has 2 arch.

1) Repository :- The repository, the data store is passive and clients of the data store are active which control logic flow.

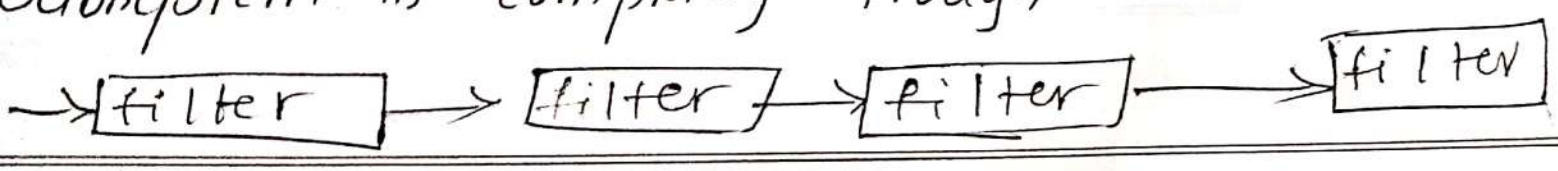
2) Blackboard :- the data store is active and clients of data store are passive which control current data status.

22) Data flow architecture

- In Data flow architecture the whole s/w sys is seen in series of transformations where data and operations are independent of each other.
- There are 3 types of execution sequence in data flow

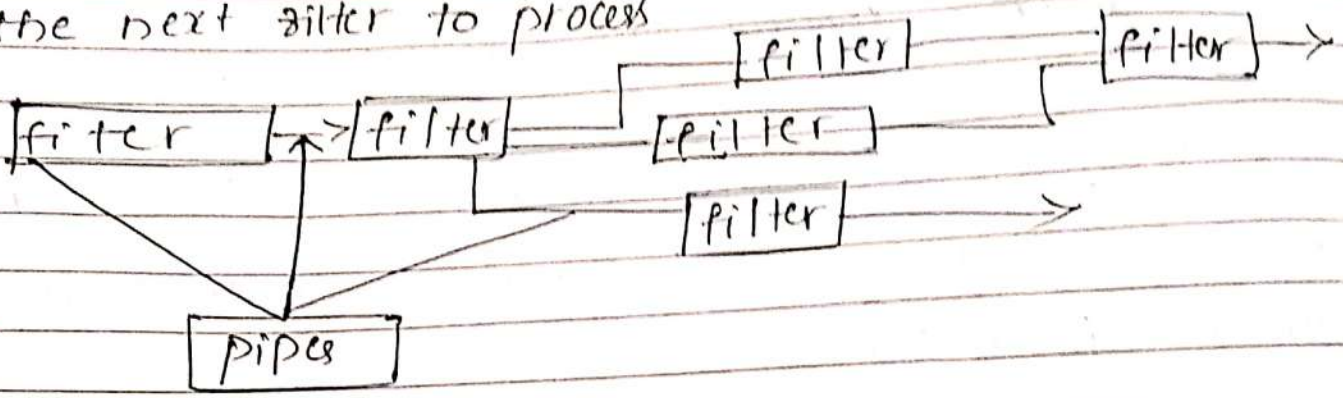
1) Batch sequential

In this, the data transformation subsystem can initiate its process only after its previous subsystem is completely trough



2. pipe and filter architecture.

- A filter is an independent data transformer which transforms data o/p stream writes the transform data stream over a pipe for the next filter to process



3. process control arch.

It is type of architecture where data is neither batched nor pipelined stream, It decomposes the entire system into subsystem or modules and connects them.

S/w Quality and dimensions

S/w Quality is defined as total functionality and features of a s/w product that bear on its ability to satisfy needs dimensions.

* acceptibility

→ The degree to which s/w can be used comfortably by a wide variety of people.

* compatibility

→ The s/w is suitable for use in d/f environments like d/f devices.

* concurrency

→ Ability of s/w to service multiple sequents at some time

* efficiency

→ ability of s/w ~~to~~ service ~~re~~ ~~re~~ to perform well or achieve a result.

* functionality

→ Ability of s/w to carry out the func as specific

* performance

→ The speed at which s/w performs under a particular load

Teacher's Signature :

* portability

- The ability of slw to transfer easily from one location to another

MODULE -2.

System Engineering.

→ What is Requirement Engineering?

Requirement Engineering builds a bridge to design & construction. The task or a technique that leads in understanding of requirements is called as Requirement Engineering.

For a software process perspective, requirement Engineering is a major software Engineering action that begins during the communicati activity & continues into modeling activity.

Requirement engineering provides a mechanism for understanding what customer want, analysing need, assessing feasabilities.

Negotiating reasonable solut^n., specify th problem properly, validation & specification & Managing the requirement.

- Requirement engineering encompasses 7 distinct tasks:-

1. Inception.
2. Elicitation.
3. Elaboration.
4. Negotiation.
5. Specification.
6. Validation.
7. Management.

1. Inception :- It describes on how does a software process gets started. The Inception task doesn't have a definative answers. Here a casual conversation is all that is needed to have a perpective on the product. This perspective on the product can be defined from the potential stake holders.

2. There are number of problems that are encountered as elicitation occurs.
   (1) Problem of scope
   (2) Problem of understanding.
   (3) Problem of volatility.

- Functional requirement & non-functional.

Software system requirements are classified as functional & non-functional requirements

- Functional :-

These are statements of services the system should provide on how the system should react to a particular input & how the system should behave in particular situation.
eg :- Business rules, authentication, transaction correction

- Non-functional.

These are the constraints on the service or function offered by the system. Some of the examples are performace, capability, availability, reliability

Non functional requirement is further classified as :-

- Product requirement.
- Organisational requirement
- & extra requirement

- Requirement analysis & negotiation :-

(In Ronak's book)

Data Modeling tools provides a software engineer the ability to represent data objects, their characteristics & their relationships., which is primarily used for large database application and other information systems.

1- Data objects.
2. Data attributes.
3. Relationships.

1. Data objects :- A data object is a representation information that must be understood by the softwar A data object can be any external entity, thing, occurance, goal, event, place & organisationa unit, structure that can be sensed and defined in terms of attributes.

2. Data attributes :- It defines the properti of data objects. They can be used to :-

(1) Name an instance of data object.
(2) Describe the instance.
(3) Make reference to another instance in another table.

Data object is connected to one another in different ways. These objects can be represented using simple lines. The connection established b/w 2 data objects shows their relationship b/w the objects.

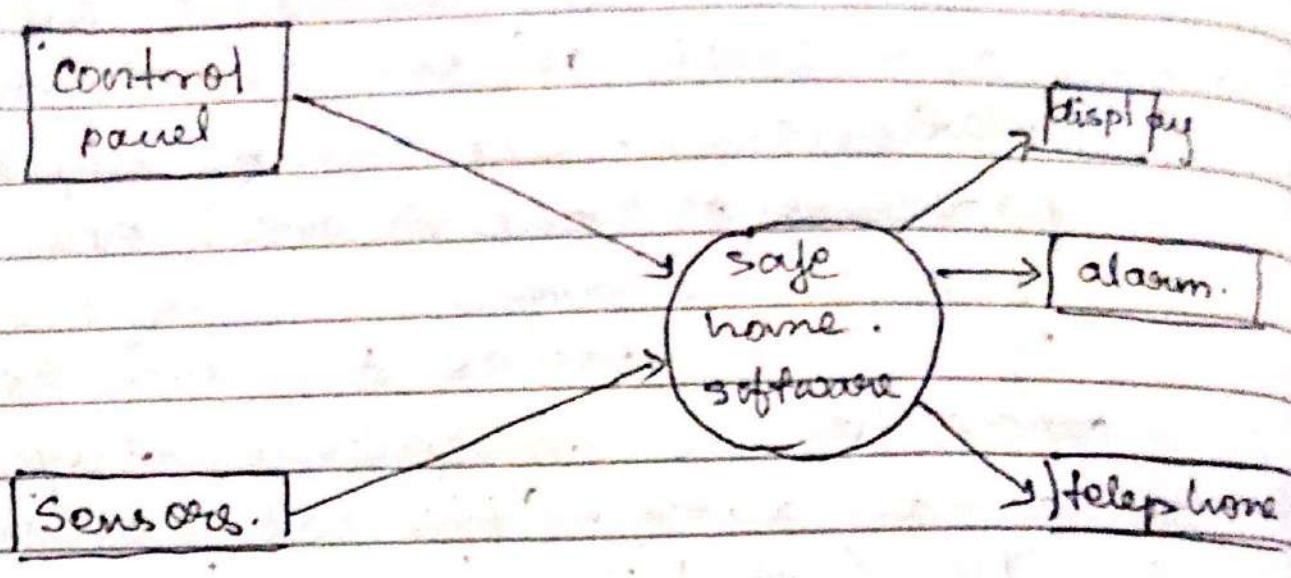
- ER diagrams :- It addresses the issues & represents all data objects that are entered, stored, transformed & produced within an application. It is also known as ER model.
It can be used in the foll. cases :-
1. Database design.
2. Database debugging.
3. Requirement gathering.
4. Database creation & patching.

- Cardinality or Modality :-
They are 2 data Modeling concepts which defines the relation b/w objects. The major difference b/w cardinality & Modality is defined as metrics used to specify the number of occurance of one object related to the number of occurance of another object. Whereas modality signifies whether a certain data object must participate in the relation or not.

Cardinality describes that a data model must be able to represent the number of occurance of an object in a given relationship. It can be expressed in patterns of one to one, one to many & many to many.
eg :- Pan cards to many accounts.
Modality is completely different from cardinality. It's value is computed as 'NULL' or 'zero'. when there is no requirement & the value is 'one' if there is compulsion for occurance.

- FLOW ORIENTED MODELING :-
This is an outdated technique by some software engineers. It was one of the most widely used requirement analysis model. Data flow diagrams are not a formal part of UML. But they can be used as a compliment for UML diagrams. (uniform modeling lang.)
Data flow diagrams takes the I/P process the I/P & gives the O/P of a system. D.F. the data objects are represented by circles. and labelled arrows.
Data flow diagram is also called as context diagram.

- Data diagram for a safe ~~mode~~ software home.



- Data Dictionary :

Data Dictionary is a collection of discriptions of ~~the~~ a Data object in a data model. For the benefit of programmers and others who refers to them.

There are 2 different types :

1. Active data dictionary.
2. Passive ——"——

MODULE - 4.

MODULARITY :- It is the process of dividing a software into multiple independent modules where each modules work independ-ently. There are many advantages of modularity.

1. It is easy to understand the system.
2. System maintainance is easy.
3. Reduces development time.
4. Program need not be re-written several times.
5. Testing & debugging can be more efficient.
6. Maintainance can be conducted without serious side effects.

COUPLING :- It is a measurement of the degree of independence b/w modules. A good software must have low coupling b/w modules.

There are different types :-

1. Data coupling.
2. Stamp coupling.
3. Control coupling.
4. External coupling.
5. common coupling.
6. Content coupling.

- COHESION : It is the measure of the degree to which the elements of module are functionally related - It is a degree to which all elements are directed towards performing a single task contained in a component. Basically cohesion is an internal connection that keeps the module together. A good software will have high cohesion.

Types :- 1. Functional cohesion.
2. Sequential —"—
3. Communication "
4. Procedural "
5. Temporary "
6. Logical "
7. Coincidal "

- Functional Independence: It is achieved by developing modules with single minded functions & avoid excessive interactions with other modules. A developer should design software so that each module addresses a specific subset of requirement and has a simple interface with other parts of program structures & modules.
Independence is assessed using 2 qualitative criteria.

1. Coupling.
2. Cohesion.

- DESIGN DOCUMENTATION :-

papergrid
Date: / /

Object Diagram

Object Diagram describes the static structure of a system at particular time.
They can be used to test case diagrams for accuracy.

Component Diagram

It describes the organization of physical software components including source codes and executables.

Deployment Diagram

It explains the physical resource in a system such as nodes, interface, components and connections

Activity Diagram

It explains the nature of the system by modelling the flow of control from activity to activity.
An activity represents an operation on some class in the system that results in a change in the state of the system.

Use Case Diagram

It is used for visualizing the functional requirements of a system and gives a

relationship b/w an actor and a use case.

Sequence Diagram

It describes the interaction among the classes in terms of an exchange of message overtime.

Data Modeling

If software requirements include the need to create an interface with database or if complex data structures must be used to construct and manipulate, the software team must create a data model as a part of requirement model.

The software engineer or analyst defines all the data object that are used within the system and clearly represents the relationships b/w the data objects and other informations.

Data Modeling Tools

It provides a software engineer the ability to represent data objects their characteristics and their relationships. Which is primarily used for large database applications and other information systems.

Structural Diagram

i) Class Diagram
ii) Package Diagram
iii) Object Diagram
iv) Component Diagram
v) Deployment Diagram

Behavioural Diagram

i) Activity Diagram
ii) UseCase "
iii) Sequence "
iv) State "
v) Communication "
vi) Timing "

Class Diagram

A class diagram is a static structure of a system. It shows the relationship b/w the classes objects, attributes and operation.

Package Diagram

Package diagrams are subset of class diagrams but developers sometime treat them as a separate technique. Package diagrams organize elements of a system into related groups to minimize dependency b/w packages.

Date: / /

Building the Requirement Model

The intent of the requirement model is to provide description of the required functional, behavioural and informational domains for the computer based system

Elements of Requirement Model

The requirement model for a computer based system can be represented in different models. There are certain approaches that has higher probability of uncovering omissions, inconsistencies and ambiguity.

Types of Requirement Model

1. Scenario based Model
2. Class based Model
3. Behavioural Model

What is UML Diagram?

UML is a way of visualizing a software program using a collection of diagrams. There are different types of UML Diagrams which is organized into two distinct groups:

i) Structural Diagrams
ii) Behavioural / Interaction Diagrams

v) A definition mechanism should be used Quality Function Deployment

It is a quality management technique that translates the requirement of the customer into technical requirements for software.

i) Normal Requirements
The objectives and goals that are stated for a product or system during meetings with a customer. If these requirements are present the customer is satisfied.

ii) Expected Requirements
These requirements are implicit to the product or system and may not be so fundamental for the customer. Their absense might not dissatisfy the customer.

iii) Exciting Requirements
These features go beyond the customer's expectations and prove to be very satisfying

Usage Scenario

As requirements are gathered and overall vision of system function and features begins to develop. However it is difficult to move into more detailed technical software engineering activities until ge

Date: / /

understand how these functions and features will be used by different end users.
To overcome this developers and users can create a set of ~~usage~~ scenarios that identifies the usage of system for different end users.

Elicitation of Work Products

The work products produced as a consequence of requirement elicitation will vary depending upon the size of the system or product to be built. For most systems, the work product include the following :-

i) Statement of need and feasability
ii) A bounded statement of scope for the system or product.
iii) A list of customers users and other stakeholders who participated in requirement gathering
iv) A description of a system's technical environment.
v) A list of requirements that categorize
vi) A list of usage scenario.
vii) A prototype to be developed

Date: / /

Requirement Elicitation

It is also called as requirement gathering which combines elements like problem solving, elaboration, negotiation and specification.

In order to encourage a collaborative team oriented approach to requirement gathering the stakeholders work together for the above mentioned elements

Collaborative Requirement Gathering

Many different approaches for collaborative requirement gathering have been proposed with the following basic guidelines:

i) Meetings are conducted and attended by both software engineers and other stakeholders

ii) Rules for preparation and participation are established.

iii) An agenda is ~~adjusted~~ suggested that is formal enough to cover all the important points.

iv) A facilitator can be customer, developer or an outsider who controls the meeting

Date: / /

Functional and Non-functional Requirements

Software System Requirements are classified as functional requirements and non-functional requirement

Functional Requirements

These are statements of services the system should provide or how the system should reacts to a particular input and how the system should behave in particular situation.

~~Non-functional Requirements~~

Example:-
Business Rules, Transaction correction, authentication

Non-functional Requirements

These are constraints on the service or function offered by the system.

Example:
Performance, scalability, capability, availability

Non-functional Requirement is further classified as
Product Requirements
Organizational "
External "

Date: / /

→ What problems will the solution address?
→ Am I asking questions to the right person?
→ Am I asking too many questions?
→ Are my questions relevant to the problem that you have?
→ Are you the right person to ask these questions?

Elicitation Requirements or Requirement Gathering

1) Collaborative Requirements Gathering.
2) Quality function requirement.
3) Usage Scenarios
4) Elicitation work products.

Date: / /

Inception
It describes on how does a software project gets started.
The inception task doesn't have a definite answers.
Here a casual conversation is all that is needed to have a perspective on the product.
This perspective on the product can be figured from the potential stakeholders.

Elicitation
There are number of problems that are encountered as elicitation occurs:-
i) Problem of scope.
ii) " " understanding
iii) " " volatility.

Elaboration

Establishing the ground work for understanding software requirements:
1) Stakeholders.
2) Recognizing multiple view points
3) Working towards collaboration
4) Asking the first question
→ Who is behind the request for this work?
→ Who will use the solution?
What will be the economic benefit of the successful solution?
→ Is there any other source for the solution that you need?

Date: / /

MODULE - 2

System Engineering

Requirement Engineering

It builds a bridge to design and construction.

The task or technique that leads to understanding of requirements is called as Requirement Engineering

For a software process perspective requirement engineering is a major software engineering action that begins during the communication activity and continues into modelling activity

Requirement engineering provides a mechanism for understanding what customer wants, analysing needs, assessing feasibility, negotiating a reasonable solution, specifying a solution properly, validating the specification and managing the requirements

Requirement Engineering encompasses 7 distinct tasks:

i) Inception
ii) Elicitation
iii) Elaboration
iv) Negotiation
v) Specification
vi) Validation
vii) Management

27.Write short note on requirements specification

A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.

Qualities of SRS:

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable
- Modifiable
- Traceable

28.What is requirement engineering? Explain seven distinct task of requirement engineering

Introduction to requirement engineering

- The process of collecting the software requirement from the client then understand, evaluate and document it is called as requirement engineering.
- Requirement engineering constructs a bridge for design and construction.

Requirement engineering consists of seven different tasks as follow:

1. Inception

- Inception is a task where the requirement engineering asks a set of questions to establish a software process.
- In this task, it understands the problem and evaluates with the proper solution.
- It collaborates with the relationship between the customer and the developer.
- The developer and customer decide the overall scope and the nature of the question.

2. Elicitation

Elicitation means to find the requirements from anybody.
The requirements are difficult because the **following problems occur in elicitation**.

Problem of scope: The customer give the unnecessary technical detail rather

than clarity of the overall system objective.

Problem of understanding: Poor understanding between the customer and the developer regarding various aspect of the project like capability, limitation of the computing environment.

Problem of volatility: In this problem, the requirements change from time to time and it is difficult while developing the project.

3. Elaboration

- In this task, the information taken from user during inception and elaboration and are expanded and refined in elaboration.
- Its main task is developing pure model of software using functions, feature and constraints of a software.

4. Negotiation

- In negotiation task, a software engineer decides the how will the project be achieved with limited business resources.
- To create rough guesses of development and access the impact of the requirement on the project cost and delivery time.

5. Specification

- In this task, the requirement engineer constructs a final work product.
- The work product is in the form of software requirement specification.
- In this task, formalize the requirement of the proposed software such as informative, functional and behavioral.
- The requirement are formalize in both graphical and textual formats.

6. Validation

- The work product is built as an output of the requirement engineering and that is accessed for the quality through a validation step.
- The formal technical reviews from the software engineer, customer and other stakeholders helps for the primary requirements validation mechanism.

7. Requirement management

- It is a set of activities that help the project team to identify, control and track the requirements and changes can be made to the requirements at any time of the ongoing project.

- These tasks start with the identification and assign a unique identifier to each of the requirement.
- After finalizing the requirement traceability table is developed.
- The examples of traceability table are the features, sources, dependencies, subsystems and interface of the requirement.

29.Explain in detail the any four people involved in software engineering process.

PROJECT SPONSOR

Project Sponsors play a critical role in all projects. Project sponsors have the bandwidth to take on the Project Sponsor role, their day job and no other project role, therefore Project Sponsors are not Project Managers, Scrum Masters or Product Owners.

SUBJECT MATTER EXPERTS (SME)

A Subject Matter Expert (SME) or Domain Expert is a person who is an authority in a particular area or topic. A Subject Matter Expert has superior (expert) knowledge of a discipline, technology, product, business process or entire business area.

PRODUCT OWNER

Product Owner is a software development role for **a person who represents the business or end-users and is responsible for working with the user group to determine what features will be in the product release**.

PROJECT MANAGER (PM)

The Project Manager (PM) is **responsible for knowing the "who, what, where, when and why" of the software project**. This means knowing the stakeholders of the project and being able to effectively communicate with each of them.