

Biyani's Think Tank

Concept based notes

System Design

(BCA Part-III)

Revised By: Neha Tewari
Dept. of Information Technology
Biyani Girls College, Jaipur



Published by :

Think Tanks

Biyani Group of Colleges

Concept & Copyright :

©Biyani Shikshan Samiti

Sector-3, Vidhyadhar Nagar,

Jaipur-302 023 (Rajasthan)

Ph : 0141-2338371, 2338591-95 • Fax : 0141-2338007

E-mail : acad@biyanicolleges.org

Website : www.gurukpo.com; www.biyanicolleges.org

ISBN: 978-93-82801-73-3

Edition: 2016

Price:

While every effort is taken to avoid errors or omissions in this Publication, any mistake or omission that may have crept in is not intentional. It may be taken note of that neither the publisher nor the author will be responsible for any damage or loss of any kind arising to anyone in any manner on account of such errors and omissions.

Leaser Type Setted by :

Biyani College Printing Department

Preface

I am glad to present this book, especially designed to serve the needs of the students. The book has been written keeping in mind the general weakness in understanding the fundamental concepts of the topics. The book is self-explanatory and adopts the “Teach Yourself” style. It is based on question-answer pattern. The language of book is quite easy and understandable based on scientific approach.

Any further improvement in the contents of the book by making corrections, omission and inclusion is keen to be achieved based on suggestions from the readers for which the author shall be obliged.

I acknowledge special thanks to Mr. Rajeev Biyani, *Chairman* & Dr. Sanjay Biyani, *Director (Acad.)* Biyani Group of Colleges, who are the backbones and main concept provider and also have been constant source of motivation throughout this Endeavour. They played an active role in coordinating the various stages of this Endeavour and spearheaded the publishing work.

I look forward to receiving valuable suggestions from professors of various educational institutions, other faculty members and students for improvement of the quality of the book. The reader may feel free to send in their comments and suggestions to the under mentioned address.

Author

Syllabus

B.C.A. Part-III

BCA302 : System Design Concepts

Max Marks: 100 (Main University Exam: 80 Internal Assessment: 20)

Question Paper pattern for Main University Examination

Max Marks: 80

Part – I (very short answer) consists 10 questions of one mark each with two questions from each unit. Maximum limit for each question is up to 20 words.

Part – II (short answer) consists 5 questions of four marks each with one question from each unit. Maximum limit for each question is up to 80 words.

Part – III (Long answer) consists 5 questions of ten marks each with one question from each unit with internal choice.

Unit – I

Introduction to Systems Design Environment:

Systems Development Approaches-Function Oriented, Data Oriented, Object Oriented, Development Process, Methodologies, Tools, Modeling Methods, Processing Types and Systems, Batch Processing, Realtime Processing.

System Development Life Cycle, Linear or Waterfall Cycle, Linear cycle phase problem definition, system specification, system design, system development, testing, maintenance Problems with Linear Life Cycle, Iterative Cycles, Spiral model Requirements analysis, Importance of Communication, Identifying Requirements, Data and Fact Gathering Techniques, Feasibility Studies, Introduction to Prototyping, Rapid Prototyping Tools, Benefits of prototyping.

Unit – II

System Design: Interface design tools, user interface evaluations, Introduction to Process Modeling, Introduction to Data Modeling.

System Design Techniques, Document Flow Diagrams, Documents, Physical Movement of documents, Usefulness of Document Flow diagram, Data Flow Diagrams, DFD notation, Context diagram DFD leveling, Process descriptions structured English, Decision Trees and Decision Tables, Entity Relationship Diagrams, Entities, Attributes, Relationship, Degree, Optionality, Resolving many to many relationship, Exclusive relationship, Structure Charts, Modules, Parameter passing. Execution sequence, Structured Design, Conversion from Data Flow Diagrams to Structure Charts.

Unit – III

Testing fundamentals: Objectives, principles, testability, Test cases: White box & Black box testing strategies: verification & validation, unit test, integration testing, validation, testing, system testing, System Implementation, Maintenance and documentation, Document Configuration Maintaining a Configuration.

Unit – IV

S/W Project planning Objectives, Decomposition techniques : S/W Sizing, Problem-based estimation, Process based estimation, Cost Estimation Models : COCOMO Model. S/W Design : Objectives, Principles, Concepts, Design methodologies Data design, Architectural design, procedural design, Object oriented concepts.

Unit – V An overview of Management Information System: Definition & Characteristics, Components of MIS, Frame Work for Understanding MIS : Information requirements & Levels of Management, Simon's Model of decision-Making, Structured Vs Un-structured decisions, Formal Vs. Informal systems

An overview of Management Information System: Definition & Characteristics, Components of MIS, Frame Work for Understanding MIS : Information requirements & Levels of



Content

S. No.	Name of Topic
1.	Introduction to System
2.	System Development
3.	Data Modeling
4.	Process Modeling
5.	Object Modeling
6.	Testing & System Maintenance
7.	Management Information System
8.	Unsolved Papers

Chapter-1

Introduction to System

Q.1 Define System and explain its characteristics.

Ans.: A System means an organised relationship among functioning units or components. It is an orderly grouping of interdependent components linked together according to a plan to achieve a specific objective. The elements of the system are as under :

- (1) **Outputs and Inputs :** A major objective of a system is to produce an output that has value to its user. Whatever the nature of the output, it must be in line with the expectations of the intended user. Inputs are the elements that enter the system for processing and output is the outcome of the processing.
- (2) **Processors :** The processor is the element of the system that involves the actual transformation of input into output. It is the operational component of a system. Processors modify the input totally or partially.
- (3) **Control :** The control element guides the system. It is the decision-making subsystem that controls the pattern of activities governing input, processing and output.
- (4) **Feedback :** Control in a dynamic system is achieved by feedback. Feedback measures output against a standard in some form that includes communication and control. Feedback may be positive or negative, routine or informational.
- (5) **Environment :** It is the source of external elements that impinge on the system. It determines how a system must function.
- (6) **Boundaries and Interfaces :** A system should be defined by its boundaries- the limits that identify its components, processes and interrelationships when it interfaces with another system.

The characteristics of a system are as under :

- (1) **Organisation** : It implies structure and order. It is the arrangement of components that helps to achieve objectives.
- (2) **Interaction** : It refers to the manner in which each component functions with other component of the system. In an organisation, for example, purchasing must interact with production, advertising with sales, etc.
- (3) **Interdependence** : It means that parts of the organisation or computer system depend on one another. They are coordinated and linked together according to a plan. One subsystem depends on the input of another subsystem for proper functioning.
- (4) **Integration** : It refers to the completeness of systems. It is concerned with how a system is tied together. It is more than sharing a physical part or location. It means that parts of a system work together within the system even though each part performs a unique function.
- (5) **Central Objective** : Objectives may be real or stated. Although a stated objective may be the real objective, it is not uncommon for an organisation to state one objective and operate to achieve another.

Q.2 Explain the different types of Processing Systems.

Ans.: **Batch processing** is execution of a series of programs ("jobs") on a computer without human interaction. Batch jobs are set up so they can be run to completion without human interaction, so all input data is preselected through scripts or command-line parameters. This is in contrast to "online" or interactive programs which prompt the user for such input.

Batch processing has these benefits :

- It allows sharing of computer resources among many users,
- It shifts the time of job processing to when the computing resources are less busy,
- It avoids idling the computing resources with minute-by-minute human interaction and supervision,
- By keeping high overall rate of utilization, it better amortizes the cost of a computer, especially an expensive one.

Distributed computing deals with hardware and software systems containing more than one processing element or storage element, concurrent processes, or multiple programs, running under a loosely or tightly controlled regime.

In distributed computing a program is split up into parts that run simultaneously on multiple computers communicating over a network. Distributed computing is a form of parallel computing, but parallel computing is most commonly used to describe program parts running simultaneously on multiple processors in the same computer. Both types of processing require dividing a program into parts that can run simultaneously, but distributed programs often must deal with heterogeneous environments, network links of varying latencies, and unpredictable failures in the network or the computers.

Distributed programming typically falls into one of several basic architectures or categories: Client-server, 3-tier architecture, N-tier architecture, Distributed objects, loose coupling, or tight coupling.

- **Client - Server** : Smart client code contacts the server for data, then formats and displays it to the user. Input at the client is committed back to the server when it represents a permanent change.
- **3-tier Architecture** : Three tier systems move the client intelligence to a middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are 3-Tier.
- **N-tier Architecture** : N-Tier refers typically to web applications which further forward their requests to other enterprise services. This type of application is the one most responsible for the success of application servers.
- **Tightly Coupled (Clustered)** : Refers typically to a cluster of machines that closely work together, running a shared process in parallel. The task is subdivided in parts that are made individually by each one and then put back together to make the final result.
- **Peer-to-Peer** : an architecture where there is no special machine or machines that provide a service or manage the network resources. Instead all responsibilities are uniformly divided among all machines, known as peers. Peers can serve both as clients and servers

The time between the presentation of a set of inputs and the appearance of all the associated outputs is called the response time. A **real-time system** is one that must satisfy explicit bounded response time constraints to avoid failure. Equivalently, a real-time system is one whose logical correctness is based both on the correctness of the outputs and their timeliness. Notice that response times of, for example, microseconds are not needed to characterize a real-time system - it simply must have response times that are constrained and thus predictable. In fact, the misconception that real-time systems must be "fast" is because in most instances, the deadlines are on the order of

microseconds. But the timeliness constraints or deadlines are generally a reflection of the underlying physical process being controlled. For example, in image processing involving screen update for viewing continuous motion, the deadlines are on the order of 30 microseconds.

An important concept in real-time systems is the notion of an event, that is, any occurrence that results in a change in the sequential flow of program execution. Events can be divided into two categories: synchronous and asynchronous. Synchronous events are those that occur at predictable times such as execution of a conditional branch instruction or hardware trap. Asynchronous events occur at unpredictable points in the flow-of-control and are usually caused by external sources such as a clock signal. Both types of events can be signaled to the CPU by hardware signals

Q3. What is the difference between object-oriented and function-oriented system design approach?

Ans 1 .Function Oriented Design: The basic abstractions, which are given to the user, are real world functions.
Object Oriented Design : The basic abstractions are not the real world functions but are the data abstraction where the real world entities are represented.

2 .Function Oriented Design: Functions are grouped together by which a higher level function is Page on obtained. an eg of this technique is SA/SD.

Object oriented Design: Functions are grouped together on the basis of the data they operate since the classes are associated with their methods.

3. Function Oriented Design: In this approach the state information is often represented in a centralized shared memory.

Object Oriented Design: In this approach the state information is not represented in a centralized memory but is implemented or distributed among the objects of the system.

4 Function Oriented Design: approach is mainly used for computation sensitive application,

Object oriented Design: whereas OOD approach is mainly used for evolving system which mimicks a business process or business case.

5. In Function oriented Design: - we decompose in function/procedure level
Object Oriented Design: - we decompose in class level

6. Function Oriented Design: Top down Approach
Object Oriented Design: Bottom up approach

7. Function oriented Design: It views system as Black Box that performs high level function and later decompose it detailed function so to be mapped to modules.

Object Oriented Design: Object-oriented design is the discipline of defining the objects and their interactions to solve a problem that was identified and documented during object-oriented analysis.

8. Function Oriented Design: Begins by considering the use case diagrams and Scenarios.

Object Oriented Design: Begins by identifying objects and classes

□ □ □

Chapter-2

System Development

Q.1 Describe System Development Life Cycle and explain its various phases.

Ans.: The Systems Development Life Cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project from an initial feasibility study through maintenance of the completed application. Various SDLC methodologies have been developed to guide the processes involved including the waterfall model (the original SDLC method), rapid application development (RAD), joint application development (JAD), the fountain model and the spiral model. Mostly, several models are combined into some sort of hybrid methodology. Documentation is crucial regardless of the type of model chosen or devised for any application, and is usually done in parallel with the development process. Some methods work better for specific types of projects, but in the final analysis, the most important factor for the success of a project may be how closely particular plan was followed.

Feasibility : The feasibility study is used to determine if the project should get the go-ahead. If the project is to proceed, the feasibility study will produce a project plan and budget estimates for the future stages of development.

Requirement Analysis and Design : Analysis gathers the requirements for the system. This stage includes a detailed study of the business needs of the organization. Options for changing the business process may be considered. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design (how the individual programs are going to work), interface design (what are the interfaces going to look like) and data design (what data will be required). During these phases, the software's overall structure is defined. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software development. Much care is taken during this phase. The logical system of the product is developed in this phase.

Implementation : In this phase the designs are translated into code. Computer programs are written using a conventional programming language

or an application generator. Programming tools like Compilers, Interpreters, Debuggers are used to generate the code. Different high level programming languages like C, C++, Pascal, Java are used for coding. With respect to the type of application, the right programming language is chosen.

Testing : In this phase the system is tested. Normally programs are written as a series of individual modules, these are subject to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

Maintenance : Inevitably the system will need maintenance. Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.

Q.2 What is the role of a Systems Analyst?

Ans.: System Analysts bridges the gap that always exists between those who need computer-based business solutions. They understand both business and computing. They study business problems and opportunities and then transform business and information requirements into specifications for information systems that will be implemented by various technical specialists including computer programmers. System Analysts initiate change within an organization. Every new system changes the business. System Analyst is basically a problem solver.

An analyst must possess various skills to effectively carry out the job. Specifically, they may be divided, into two categories: Interpersonal and technical skills. Both are required for system development. *Interpersonal* skills deal with, relationships and the interface .of the analyst with people in business. They are useful in establishing trust's resolving conflict, and communicating information. Technical skills, on the other hand, focus an procedures and techniques for operations analysis, systems analysis, and computer science

The **interpersonal skills** relevant to systems work include the following :

- Communication
- Understanding
- Foresightedness and Vision
- Adaptability and Flexibility Skills
- Teaching
- Selling
- Patience and Rationality
- Management Skills
- Leadership Quality
- Training and Documentation Capability

Technical skills include :

- Creativity-
- Problem solving-
- Project management-
- Dynamic interface-
- Questioning attitude and inquiring mind-
- Knowledge-

Q.3 Explain the Waterfall Model.

OR

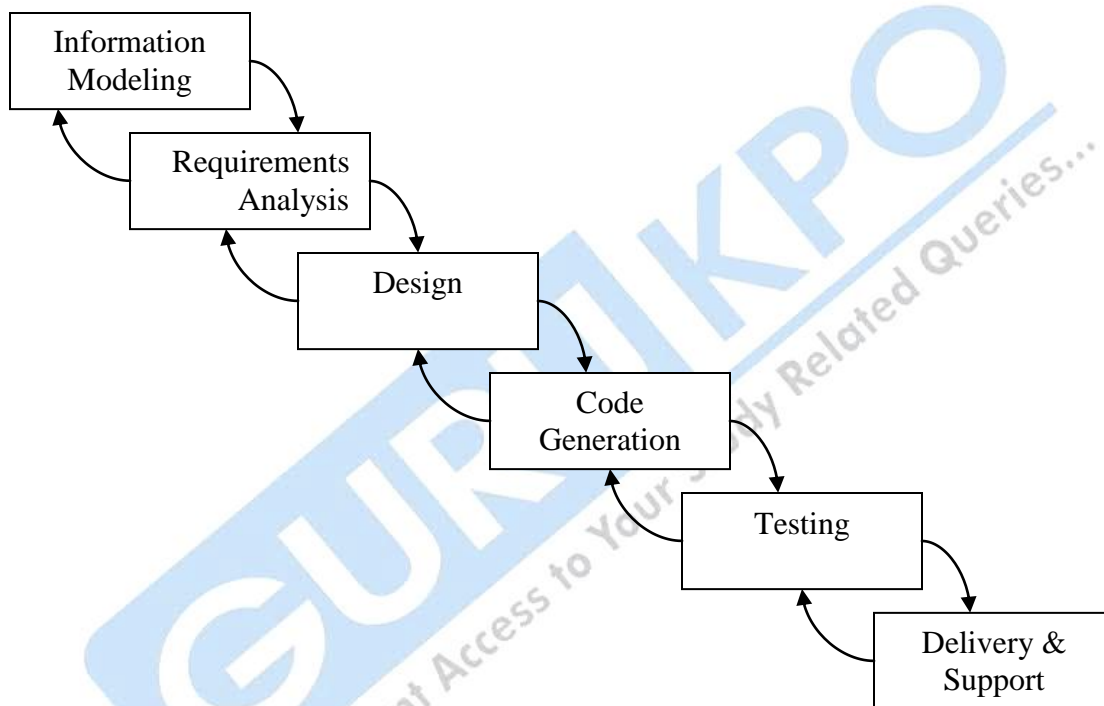
Describe the Classic Life Cycle Model.

OR

Explain the Linear Sequential Model.

Ans.: Sometimes called the *classic life cycle* or the *linear sequential model*, the *waterfall model* is a systematic, sequential approach to software development in which development is seen as flowing downwards (like a waterfall) that begins at the system level and progresses through analysis, design, coding, testing and support. To follow the waterfall model, one proceeds from one phase to the next in a sequential manner. For example, one first completes "requirements specification". When the requirements are fully completed, one proceeds to design. The software is designed (on paper) and this design should be a plan

for implementing the requirements given. When the design is fully completed, an implementation of that design, i.e. coding of the design is made by programmers. After the implementation phases are complete, the software product is tested and debugged; any faults introduced in earlier phases are removed here. Then the software product is installed, and later maintained to add any new functions that the user needs and remove bugs. Thus in a waterfall model, we can move to the next step only when the previous step is completed and removed of all errors. There is no jumping back and forth or overlap between the steps in a waterfall model.



The model consists of six distinct stages, namely :

- (1) In the *Information Modelling* phase
 - (a) Work begins by gathering information related to the existing system. This will consists of all items consisting of hardware, people, databases etc.
- (2) In the *requirements analysis* phase
 - (a) The problem is specified along with the desired objectives (goals).
 - (b) The constraints are identified.

- (c) All information about the functions, behaviour, and performance are documented and checked by the customers.
- (3) In the *design phase*, all inputs, computations and outputs of the system should be converted into a software model so that it can be coded by programmers. The hardware requirements are also determined at this stage along with a picture of the overall system architecture.
- (4) In the *code generation* phase, the design has to be translated into a machine-readable form using any of the programming languages available that is suitable for the project.
- (5) In the *testing* phase stage
 - (a) Once code is generated, testing begins.
 - (b) It focuses on all the statements of the software and removes all errors.
 - (c) It ensures that proper input will produce actual results.
 - (d) Detailed documentation from the design phase can significantly reduce the coding effort.
- (6) The *delivery and support* phase consists of delivering the final product to the customer and then taking care of the maintenance of the product. In this phase the software is updated to :
 - (a) Meet the changing customer needs
 - (b) Adapted to accommodate changes in the external environment
 - (c) Correct errors that were not previously known in the testing phases
 - (d) Enhancing the efficiency of the software

Q.4 Explain the Prototyping Process Model.

Ans.: The prototyping model begins with the requirements gathering. The developer and the customer meet and define the objectives for the software, identify the needs, etc. A 'quick design' is then created. This design focuses on those aspects of the software that will be visible to the customer. It then leads to the construction of a prototype. The prototype is then checked by the customer and any modifications or changes that are required are made to the prototype. Looping takes place in this process and better versions of the prototype are created. These are continuously shown to the user so that any new changes can be updated in the prototype. This process continues till the user is satisfied with the system. Once a user is satisfied, the prototype is

converted to the actual system with all considerations for quality and security.

The prototype is considered as the 'first system'. It is advantageous because both the customers and the developers get a feel of the actual system. But there are certain problems with the prototyping model too.

- (1) The prototype is usually created without taking into consideration overall software quality.
- (2) When the customer sees a working model in the form of a prototype, and then is told that the actual software is not created, the customer can get irritated.
- (3) Since the prototype is to be created quickly, the developer will use whatever choices he has at that particular time (eg, he may not know a good programming language, but later may learn. He then cannot change the whole system for the new programming language). Thus the prototype may be created with less-than-ideal choices.

Q.5 Describe the Rapid Application Development Model. State its disadvantages.

Ans.: Rapid Application Development (RAD) is an incremental software development process model that focuses on a very short development cycle. The RAD model is a 'high-speed' version of the linear sequential model. It enables a development team to create a fully functional system within a very short time period (e.g. 60 to 90 days).

Business Modeling : The information flow among business functions is modeled in a way that answers the following questions :

What information drives the business process?

What information is generated?

Who generates it?

Where does the information go?

Who processes it?

Data Modeling : It gives all the details about what data is to be used in the project. All the information found in the business modeling phase is refined into a set of data objects and the characteristics and the relationships between these objects are defined.

Process Modeling : Here all the processes are defined that are needed to use the data objects to create the system. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

Application Generation : RAD makes use of the fourth generation techniques and tools like VB, VC++, Delphi etc rather than creating software using conventional third generation programming languages. The RAD reuses existing program components (when possible) or creates reusable components (when necessary). In all cases, automated tools (CASE tools) are used to facilitate construction of the software.

Testing and Turnover : Since the RAD process emphasizes reuse, many of the program components have already been tested. This minimizes the testing and development time.

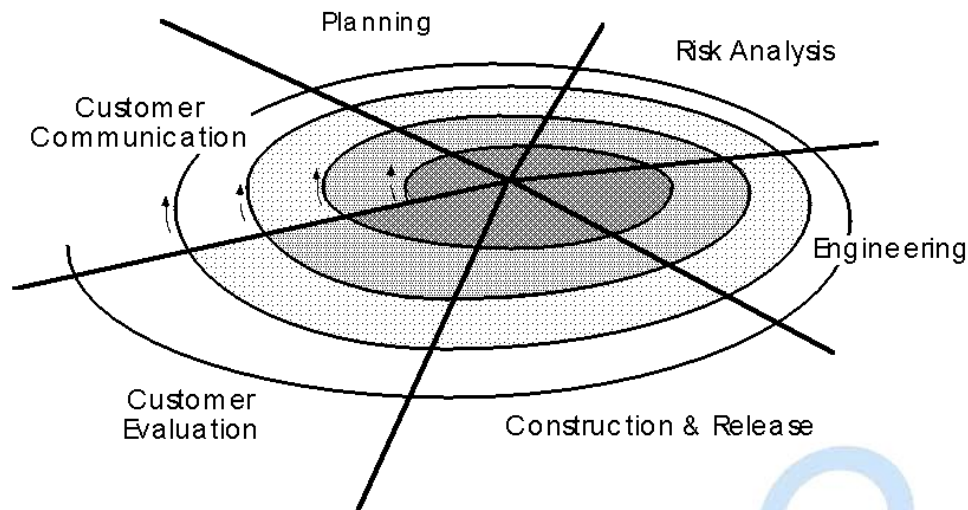
If a business application can be divided into modules, so that each major function can be completed within the development cycle, then it is a candidate for the RAD model. In this case, each team can be assigned a model, which is then integrated to form a whole.

Disadvantages :

- For Large projects, RAD requires sufficient resources to create the right number of RAD teams.
- If a system cannot be properly divided into modules, building components for RAD will be problematic
- RAD is not appropriate when technical risks are high, e.g. this occurs when a new application makes heavy use of new technology.

Q.6 Explain the Spiral Model. What are the advantages of this model?

Ans.: The spiral model, combines the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model, therein providing the potential for rapid development of incremental versions of the software. In this model the software is developed in a series of incremental releases with the early stages being either paper models or prototypes. Later iterations become increasingly more complete versions of the product.



As illustrated, the model is divided into a number of task regions.

These regions are :

- (1) The **customer communication** task - to establish effective communication between developer and customer.
- (2) The **planning** task - to define resources, time lines and other project related information..
- (3) The **risk analysis** task - to assess both technical and management risks.
- (4) The **engineering** task - to build one or more representations (prototypes) of the application.
- (5) The **construction and release** task - to construct, test, install and provide user support (e.g., documentation and training).
- (6) The **customer evaluation** task - to obtain customer feedback based on the evaluation of the software representation created during the engineering stage and implemented during the install stage.

The evolutionary process begins at the centre position and moves in a clockwise direction. Each traversal of the spiral typically results in a deliverable. For example, the first and second spiral traversals may result in the production of a product specification and a prototype, respectively. Subsequent traversals may then produce more sophisticated versions of the software.

An important distinction between the spiral model and other software models is the explicit consideration of risk. There are no fixed phases such as

specification or design phases in the model and it encompasses other process models. For example, prototyping may be used in one spiral to resolve requirement uncertainties and hence reduce risks. This may then be followed by a conventional waterfall development.

Advantages of the Spiral Model :

- The spiral model is a realistic approach to the development of large-scale software products because the software evolves as the process progresses. In addition, the developer and the client better understand and react to risks at each evolutionary level.
- The model uses prototyping as a risk reduction mechanism and allows for the development of prototypes at any stage of the evolutionary development.
- It maintains a systematic stepwise approach, like the classic life cycle model, but incorporates it into an iterative framework that more reflect the real world.
- If employed correctly, this model should reduce risks before they become problematic, as consideration of technical risks are considered at all stages.

Q.7 Explain Information Gathering Process for System Development.

OR

Explain Fact Finding Method of System Analysis.

Ans.: Fact finding means learning as much as possible about the present system. The tools used in information gathering or fact finding are

- (1) **Review of Written Documents :** In all organizations documents such as forms, records, reports, manuals, etc are available. These help in determining how the present system runs. The process of fact finding includes collection of all possible documents and evaluating them. Unfortunately, most manuals are not up to date and may not be readable. The analyst needs to find out how the forms are filled out, what changes need to be made and how easy they are to read.
- (2) **On-Site Observation :** The purpose of on-site observation is to get as close as possible to the real system being studied. It is the process of recognizing and noting people, objects and occurrences to obtain information. As an observer the analyst must follow a set of rules. He/she must listen than talk and not give advice or pass a moral

judgment, must not argue or show friendliness towards others. The following questions can serve as a guide for on-site observations:

- What kind of system is it? What does it do?
- Who runs the system? Who are the important people in it?
- What is the history of the system?

(3) **Interviews** : An interview is a face to face interpersonal situation in which a person called the interviewer asks a person being interviewed, questions designed to gather information about a problem. The analyst or interviewer can schedule interviews with key personnel of the organization. The analyst also needs to conduct detailed interviews with all the people who will actually use the system. This will provide all the details the analyst needs and also remove any fear from the users that the computers will replace the. Interviews help gather vital facts about the existing problems, such as lack of quality control or security, etc. Interviewing needs a friendly atmosphere so that the interviewer can ask questions properly, obtain reliable and correct answers and record the answers accurately and completely.

(4) **Questionnaires** : A questionnaire is a tool that has questions to which individuals respond. A questionnaire has the following advantages:

- It is economical and requires less skill than an interview.
- It can be used to gather data from large number of people simultaneously
- It is a uniform method in which all question asked are the same to all people
- The users are happy as they know that the answers they give are confidential
- User get time to think about the questions and so can give more accurate results than in an interview

Q.8 What is Feasibility? Describe the different types of Feasibility.

Ans.: Feasibility is the determination of whether or not a project is worth doing. The process followed in making this determination is called feasibility study. A feasibility study is carried out to select the best system that meets performance requirements. When conducting feasibility study, an analyst can consider 7 types of feasibility:

- **Technical Feasibility** : It is concerned with specifying the equipment and the computer system that will satisfy and support the proposed user requirements. Here we need to consider the configuration of the system which tells the analyst how many work stations are required, how the units are interconnected so that they can operate and communicate smoothly.
- **Operation Feasibility** : It is related to human organizational aspects. The points to be considered here are – what changes will be brought with the system?, what new skills will be required?, do the existing staff members have these skills and can they be trained?
- **Economic Feasibility** : It is the most frequently used technique for evaluating a proposed system. It is also called Cost/Benefit Analysis. It is used to determine the benefits and savings that are expected from the proposed system and compare them with the costs. If benefits are more than the cost, the proposed system is given an OK.
- **Social Feasibility** : It is a determination of whether the proposed system will be acceptable to the people or not. It finds out the probability of the project being accepted by the group of people who are directly affected by the changed system.
- **Management Feasibility** : It is a determination of whether the proposed system is acceptable to the management of the organization. The project may be rejected, if the management does not accept the proposed system.
- **Legal Feasibility** : It is a determination of whether the proposed project is under legal obligation of known Acts, Statutes, etc.
- **Time Feasibility** : It is a determination of whether the project will be completed within a specified time period. If the project takes too much time, it is likely to be rejected.

Q.9 What is Cost/Benefit Analysis? Explain its procedure.

Ans.: The costs associated with the system are expenses or losses arising from developing and using a system. But the benefits are the advantages received from installing and using this system. Cost/Benefit analysis is a procedure that gives a picture of the various costs, benefits and rules associated with a system. The determination of costs and benefits is done in the following steps :

- (1) Identify the costs and benefits of a project.
- (2) Categorize the costs and benefits for analysis: The different categories of costs and benefits are :
 - (a) Tangible or Intangible
 - (b) Direct or Indirect
 - (c) Fixed or Variable
- (3) Select a method of evaluation: When all data is identified and categorized, the analyst must select a method of evaluation. The methods are :
 - (a) Net Benefit analysis
 - (b) Present value analysis
 - (c) Net Present value
 - (d) Payback analysis
 - (e) Break even analysis
 - (f) Cash flow analysis
- (4) Get the result of analysis and Take action.

Q.10 What is Software Requirement Specification - [SRS]?

Ans A **software requirements specification** (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure

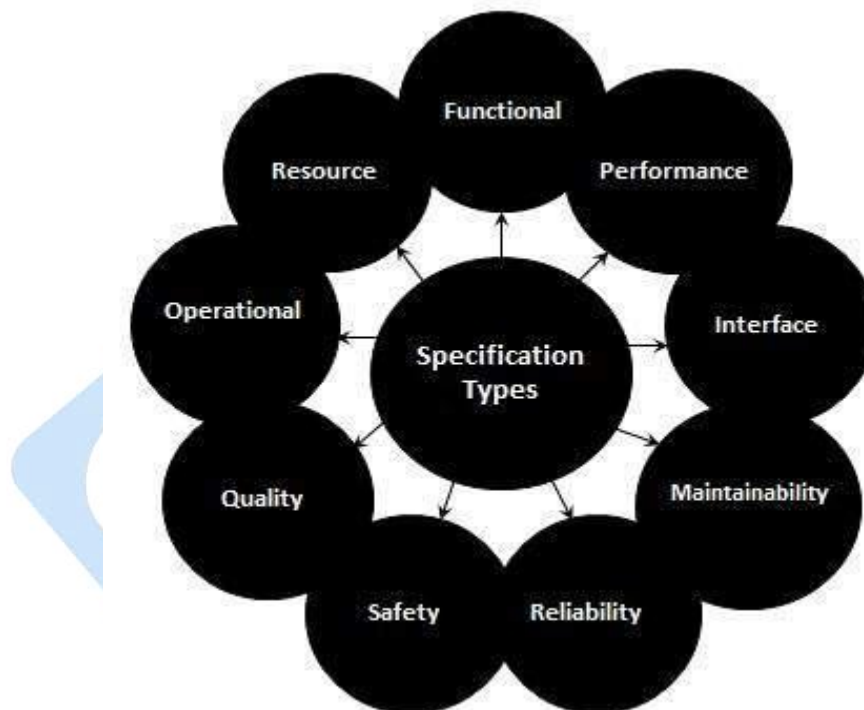
A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform. It is usually signed off at the end of requirements engineering phase.

Qualities of SRS:

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

Types of Requirements:

The below diagram depicts the various types of requirements that are captured during SRS.



Chapter-3

System Design & Modeling

Q.1 Explain Data Modeling and ER diagram with example.

Ans.: Data Modeling : It gives answers to questions regarding the data that is to be used in the application. We come to know the data objects, where they are stored, what is the relationship between objects, etc. Data modeling uses an Entity Relationship diagram to solve these questions. An Entity Relationship diagram will focus on all data that are entered, stored, transformed and produced within an application. The data model consists of three interrelated information – data objects, attributes that describe the data objects and relationships that connect data objects to one another.

Data Objects : A data object is something that has a number of different properties or attributes and that can be understood by software. For example a person or a car can be viewed as data objects. Data objects are related to one another. E.g. **person** can *own* **car**, where the relationship *own* denotes a connection between **person** and **car**. A data object reflects only data and not the operation that can be done on that data.

Attributes : Attributes define the properties of a data object. They can be used to name an instance of the data object, describe the instance or make reference to another instance in another table (e.g. attribute Owner). One or more attributes that uniquely identifies one and only one instance of an entity is defined as an identifier or primary key. E.g. employee no is a primary key for an employee.

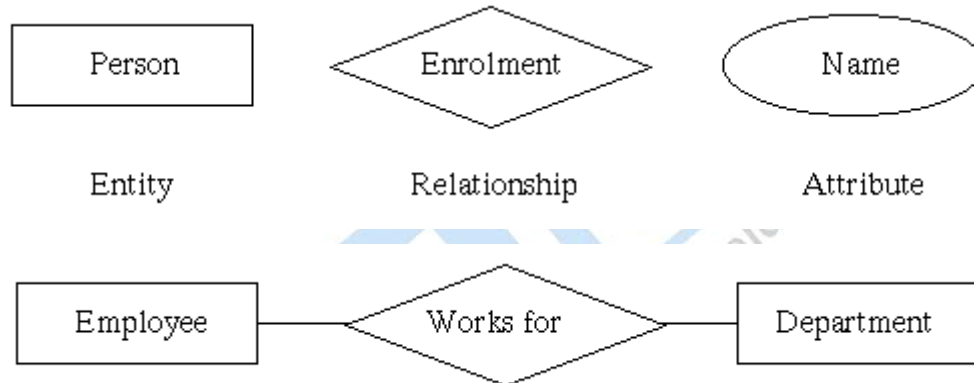
Relationships : Data objects are connected to one another in different ways. Consider two data objects – book and bookstore. A connection is established between book and bookstore because the two objects are related.

Entity – Relationship Diagrams : The object-relationship pair can be represented graphically using an ER diagram. An entity represents an object.

Examples: a computer, an employee, a song, a mathematical theorem. Entities are represented as rectangles.

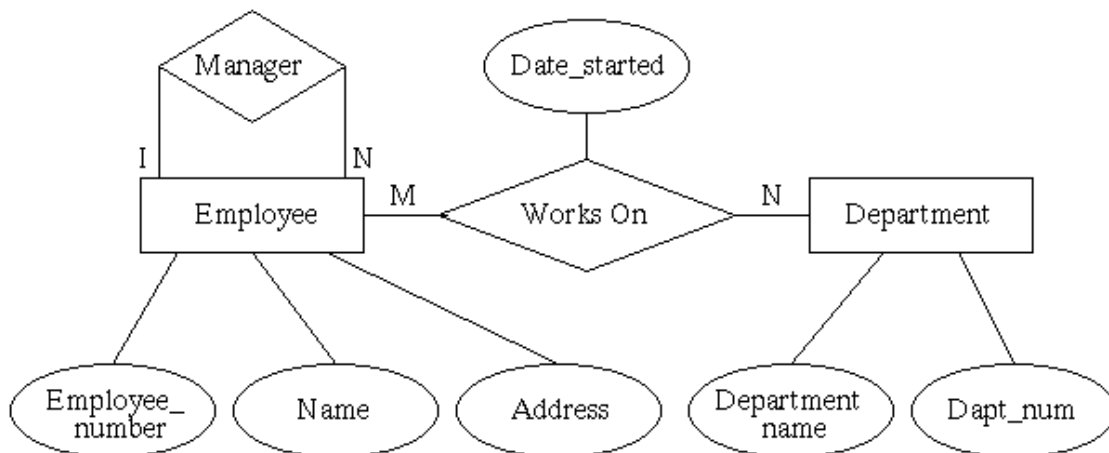
A relationship captures how two or more entities are related to one another. Examples: an *owns* relationship between a company and a computer, a *supervises* relationship between an employee and a department, a *performs* relationship between an artist and a song. Relationships are represented as diamonds, connected by lines to each of the entities in the relationship.

Entities and relationships can both have attributes. Examples: an employee entity might have an employee ID number attribute; the *proved* relationship may have a *date* attribute. Attributes are represented as ellipses connected to their entity by a line.



A Simple E-R Diagram

The following E-R diagram gives the attributes as well.



An E-R Diagram with Attributes

Q.2 In context with an ER diagram explain Cardinality and Modality. Give example.

Ans.: Cardinality : The elements of data modeling – data objects, attributes and relationships provide information only about which objects are related to one another. But this information is not sufficient for software engineering purpose. Cardinality specifies how many instances or occurrences of object X are related to how many occurrences of object Y. Cardinality is usually expressed as 'one' or 'many'. Thus two objects can be related as

- (1) **One-to-One (1:1) :** An occurrence of object A can relate to one and only one occurrence of object B and an occurrence of B can relate to only one occurrence of A.
- (2) **One-to-Many (1:N) :** One occurrence of object A can relate to one or many occurrences of object B but an occurrence of B can relate to only one occurrence of A. E.g. mother can have many children, but a child can have only one mother.
- (3) **Many-to-Many (M:N) :** An occurrence of object A can relate to one or many occurrences of object B and an occurrence of B can relate to only one or many occurrences of A. E.g. an uncle can have many nieces and a niece can have many uncles.

Cardinality defines the maximum number of objects that can participate in a relationship. It does not tell whether or not a data object must participate in the relationship.

Modality : If a particular relationship is optional or not needed then we say that the modality of that relationship is 0. The modality is 1 if an occurrence of the relationship is necessary.

Example : Consider 2 data objects Patient and Doctor. The relationship between the two data objects is *Treats*. A doctor needs a patient to treat, so the modality is 1 while it is not necessary for a patient to be treated by a doctor (he can be treated with home remedies too). So here the modality is 0.

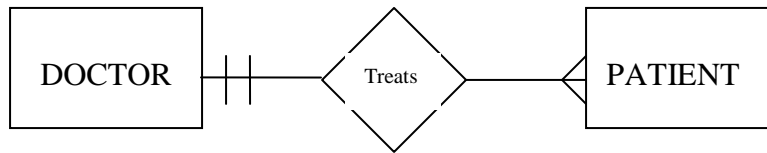
When we need to specify cardinality we use the symbols

- One = a line or dash |
- Many = crow's feet <

To specify modality we use the symbols

- One = a line or dash |
- Zero = a circle o

The following ER diagram specifies cardinality and modality.



The symbols on the relationship line that is closest to the data object will denote cardinality and the next will denote modality.

Q.3 What is a Data Dictionary? Give an example.

Ans.: A Data Dictionary (DD) is a structured repository of data about data. It is a set of accurate definitions of all DFD data elements and data structures. A data dictionary defines each term encountered during the analysis and design of a new system. Data dictionary is the place where we keep the details of the contents of data flows, data stores & processes.

Without a data dictionary the development of large systems becomes difficult. The data dictionary is an effective solution to the problem of complicated nature. The main purpose of a data dictionary is to provide a source of reference in which the analyst, the user, the designer can look up & find out its content and any other relevant information.

The main advantage of a DD is the documentation. It is a valuable reference to the organization which helps in communication between the analyst and the user. It is also important in building a database.

The Data Dictionary notations are

= is composed of

+ AND

() Optional value

[] Either/Or

{ } iteration

** comment

@ identifier (key field)

separates alternative choices in the [] construct

Examples of Data dictionary -

Name = Courtesy-Title + First-Name + (Middle-Name) + Last-Name

Courtesy-Title = [Mr. | Miss | Mrs. | Ms. | Dr. | Prof.]

First-Name = { Legal-Character }

Last-Name = { Legal-Character }

Legal-Character = [A-Z | a-z | 0-9 | ' | - | |]

Q.4 What is system design? Why is it important in the system development process .

Ans System designing in terms of software engineering has its own value and importance in the system development process as a whole. To mention it may though seem as simple as anything or simply the design of systems, but in a broader sense it implies a systematic and rigorous approach to design such a system which fulfills all the practical aspects including flexibility, efficiency and security. *Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.*

Before there is any further discussion of system design, it is important that some points be made clear. As it goes without saying that nothing is created that is not affected by the world in which it's made. So, the systems are not created in a vacuum. They are created in order to meet the needs of the users. They are not only intended to solve the existing problems, but they also come up with acceptable solutions to the problems that may arise in the future. The whole process of system development, from blueprint to the actual product, involves considering all the relevant factors and taking the required specifications and creating a useful system based on strong technical, analytical and development skills of the professionals.

Let's get back to our discussion about what the system design phase is and the importance of system design in the process of system development. Being another important step in the system development process, system designing phase commences after the system analysis phase is completed. It's appropriate to mention that the output or the specifications taken through the phase of system analysis become an input in the system design phase which in turn leads to workout based on the user defined estimations.

The importance of this phase may be understood by reason of the fact that it involves identifying data sources, the nature and type of data that is available. For example, in order to design a salary system, there is a need for using inputs, such as, attendance, leave details, additions or deductions etc. This facilitates understanding what kind of data is available and by whom it is supplied to the system so that the system may be designed considering all the relevant factors. In addition, system designing leads to ensure that the system is created in such a way that it fulfills the need of the users and keep them at ease being user-oriented. In terms of the flexibility, one of the main objectives of this phase is that it is intended to design such a system which can be dynamic in nature and responsive to the changes if required. Another important objective is that the phase of system designing is concerned with creating the system which can work efficiently providing the required output and being responsive to the time within a given time limit. The aspect of reliability and physical security of data cannot be ignored. With this respect, the system designing phase ensures security measures of the system effectively and efficiently.

□ □ □

Chapter-4

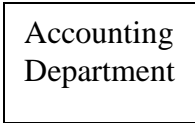
Process Modeling

Q.1 With examples explain what is a Data Flow Diagram.

Ans.: A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. It describes the system's data and how the processes transform the data in a graphical manner. Data flow diagrams can be used to provide a clear representation of any business function. It starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. It uses a top-down approach to show all the levels of the functions of the system. Initially a **context diagram** is drawn, which is a simple representation of the entire system under investigation. This is followed by a level 1 diagram; which provides an overview of the major functional areas of the business. The level 1 diagram identifies the major business processes at a high level and any of these processes can then be analyzed further - giving rise to a corresponding level 2 business process diagram. This process of more detailed analysis can then continue - through level 3, 4 and so on.

DFD Notation :


A rectangle



Accounting
Department


It denotes an external entity. It defines a source or destination of system data. It can represent a person, group of people, department, or some other system.

A circle



Compute
Sales Tax

It denotes a process or activity. It is also known as a bubble. It shows how the system transforms inputs into outputs. Each process is named.

A line with an arrowhead  Customer

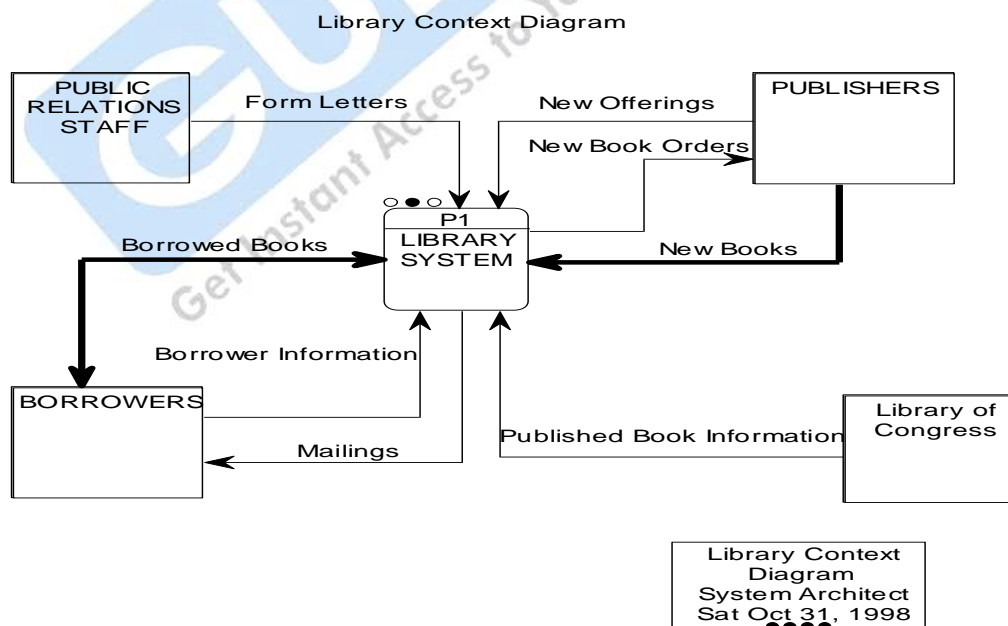
It denotes the direction of data flow. The input to, or output from, a given process, which is associated with each arrow in a DFD.

Open Rectangle

 CUSTOMER

It denotes a store that is used to model collection of data. It may refer to files or databases, or data stored on punched cards, optical disk, etc. It is shown by two parallel lines with the name of the data store between them

Context Diagrams :



The context diagram shown on this screen represents a book lending library. The library receives details of books, and orders books from one or more book suppliers. Books may be reserved and borrowed by members of the public, who are required to give a borrower number. The library will notify borrowers when a reserved book becomes available or when a borrowed book becomes overdue. In addition to supplying books, a book supplier will furnish details of specific books in response to library enquiries. After the context model is created the process is exploded to the next level to show the major processes in the system. Depending upon the complexity of the system each of these processes can also be exploded into their own process model. This continues until the goal of each process accomplishing a single function is reached. Because of this approach the context model is referred to as Level 0 (Zero) DFD, the next as Level 1 DFD, etc.

Q.2 Briefly describe a Decision Tree with example.

Ans.: Decision tree are graphical representation methods of representing a sequence of logical decisions. It is mainly used when decisions need to be taken or for defining policies. A decision tree has as many branches as there are logical alternatives. It is easy to construct, easy to read and easy to update. A decision tree is used to identify the strategy most likely to reach a goal. It is also used as a means for calculating probabilities or making financial or number based decisions. A decision making tree is essentially a diagram that represents, in a specially organized way, the decisions, the main external or other events that introduce uncertainty, as well as possible outcomes of all those decisions and events.

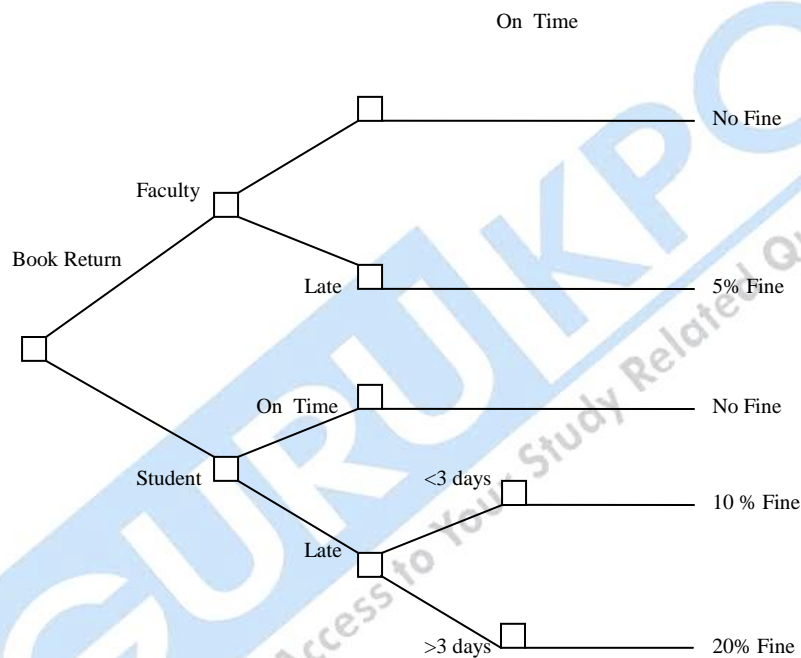
Q.3 How to draw a Decision Tree?

Ans.: You start a decision tree with a decision that needs to be made. This decision is represented by a small square towards the left of a large piece of paper. From this box draw out lines towards the right for each possible solution, and write that solution along the line. At the end of each solution line, consider the results. If the result of taking that decision is uncertain, draw a small circle. If the result is another decision that needs to be made, draw another square. **Squares represent decisions; circles represent uncertainty or random factors.** Write the decision or factor to be considered above the square or circle. If you have completed the solution at the end of the line, just

leave it blank. Starting from the new decision squares on your diagram, draw out lines representing the options that could be taken. From the circles, draw out lines representing possible outcomes. Again mark a brief note on the line saying what it means. Keep on doing this until you have drawn down as many of the possible outcomes and decisions as you can see leading on from your original decision.

Example : Book return policy in library

If a Faculty returns a book late, a fine of 5% of the book rate is charged. If a Student returns a book late by 3 days, fine is 10%, else 20% of book rate.



Q.4 What are Decision Tables? Explain with example.

Ans.: **Decision tables** are a precise yet compact way to model complicated logic. Decision tables, like if-then-else and switch-case statements, associate conditions with actions to perform. But, unlike the control structures found in traditional programming languages, decision tables can associate many independent conditions with several actions in an elegant way. Decision tables are typically divided into four quadrants, as shown below.

The four quadrants	
Conditions	Condition alternatives
Actions	Action entries

Each decision corresponds to a variable, relation or predicate whose possible values are listed among the condition alternatives. Each action is a procedure or operation to perform, and the entries specify whether (or in what order) the action is to be performed for the set of condition alternatives the entry corresponds to. Many decision tables include in their condition alternatives the **don't care** symbol, a hyphen. Using don't cares can simplify decision tables, especially when a given condition has little influence on the actions to be performed. In some cases, entire conditions thought to be important initially are found to be irrelevant when none of the conditions influence which actions are performed. The limited-entry decision table is the simplest to describe. The condition alternatives are simple boolean values, and the action entries are check-marks, representing which of the actions in a given column are to be performed.

A technical support company writes a decision table to diagnose printer problems based upon symptoms described to them over the phone from their clients.

Printer troubleshooter									
		Rules							
Conditions	Printer does not print	Y	Y	Y	Y	N	N	N	N
	A red light is flashing	Y	Y	N	N	Y	Y	N	N
	Printer is unrecognized	Y	N	Y	N	Y	N	Y	N
Actions	Check the power cable			X					
	Check the printer-computer cable	X		X					

	Ensure printer software is installed	X		X		X		X	
	Check/replace ink	X	X			X	X		
	Check for paper jam		X		X				

Decision tables make it easy to observe that all possible conditions are accounted for. In the example above, every possible combination of the three conditions is given. In decision tables, when conditions are omitted, it is obvious even at a glance that logic is missing. Compare this to traditional control structures, where it is not easy to notice gaps in program logic with a mere glance --- sometimes it is difficult to follow which conditions correspond to which actions!

Just as decision tables make it easy to audit control logic, decision tables demand that a programmer think of all possible conditions. With traditional control structures, it is easy to forget about corner cases, especially when the else statement is optional. Since logic is so important to programming, decision tables are an excellent tool for designing control logic.

□ □ □

Chapter-5

Object Modeling

Q.1 What is Structured English?

Ans.: Structured English or pseudo code or program design language (PDL) uses the vocabulary of English and the syntax of a structured programming. Structured English looks like a modern programming language. The difference between structured English and a real programming language is in the use of narrative text which is placed within the structured English statements. Structured English cannot be compiled. It should have the following characteristics:

- A fixed syntax of keywords used for structured constructs, data declaration
- A free syntax of natural language that describes processing
- Data declaration facilities that include simple(array) and complex(linked list or tree) data structures
- Facility to declare subprograms and call them

Decisions in Structured English are made through IF, THEN, ELSE, SO, etc.

Q.2 What are Structure Charts? Describe.

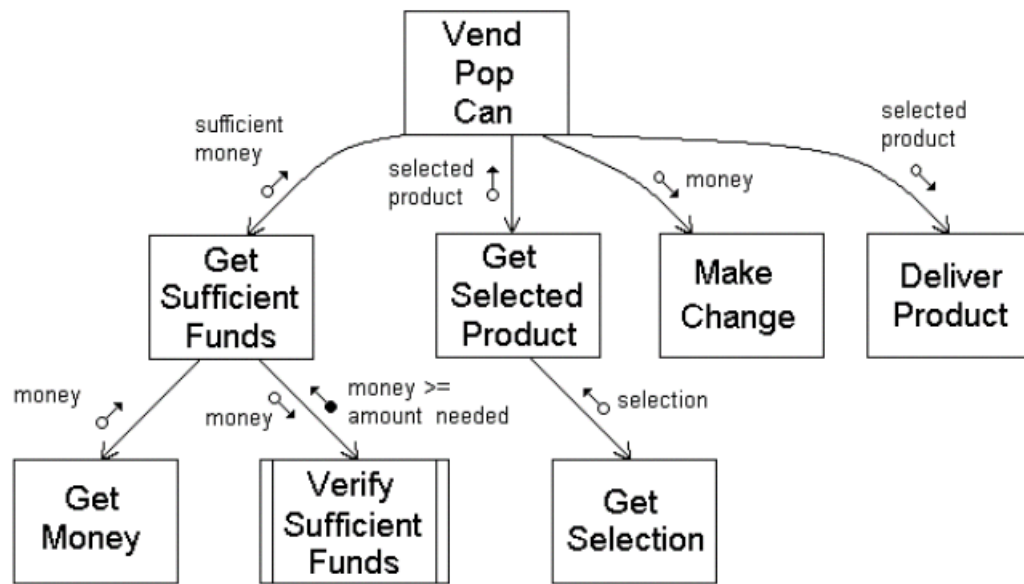
Ans.: **Structure Chart :** A hierarchical diagram showing the relationships between the modules of a computer program. A module is the basic component of a structure chart and is used to identify a function. Modules are relatively simple and independent components. Higher-level modules are “control” modules that control the flow of execution. Lower level modules are “worker bee” modules and contain the program logic to actually perform the functions.

The vertical lines connecting the modules indicate the calling structure from the high-level modules to the lower-level modules. The little arrows next to

the lines show the data that is passed between modules and represent the inputs and outputs of each module. At the structure chart level, we are not concerned with what is happening inside the module yet. We only want to know that somehow it does the function indicated by its name using the input data and producing the output data. A program call is when one module invokes a lower-level module to perform a needed service or calculation. Program call: The transfer of control from a module to a subordinate module to perform a requested service. The arrows with the open circle, called data couples, represent data being passed into and out of the module. A data couple can be an individual data item (e.g., a flag or a customer account number) or a higher-level data structure (e.g., an array, record, or other data structure). The arrow with the darkened circle is a "flag." A flag is purely internal information that is used between modules to indicate some result. Data couples: The individual data items that are passed between modules in a program call.

A basic idea of structured programming is that each module only has to do a very specific function. The module at the very top of the tree is the "boss" module. Its functions will be to call the modules on the next tier, pass information to them, and receive information back. The function of each middle-level module is to control the processing of the modules below it. Each has control logic and any error-handling logic that is not handled by the lower-level module. The modules at the extremities, or the leaves, contain the actual algorithms to carry out the functions of the program.

Structure charts are developed to design a hierarchy of modules for a program. A structure chart is in the form of a tree with a root module and branches. A subtree is simply a branch that has been separated from the overall tree. When the subtree is placed back in the larger tree, the root of the subtree becomes just another branch in the overall tree.



Q.3 What is a HIPO Chart? Explain.

Ans.: HIPO charts show relationships between modules. It describes the data input and output from the processes and defines the data flow. It provides a structure by which the functions of a system can be understood. It also provides a visual description of input to be used and output to be produced for each level of the diagram. It makes the transformation from input to output data visible.

There are two parts to a HIPO chart, a hierarchy chart and an IPO chart.

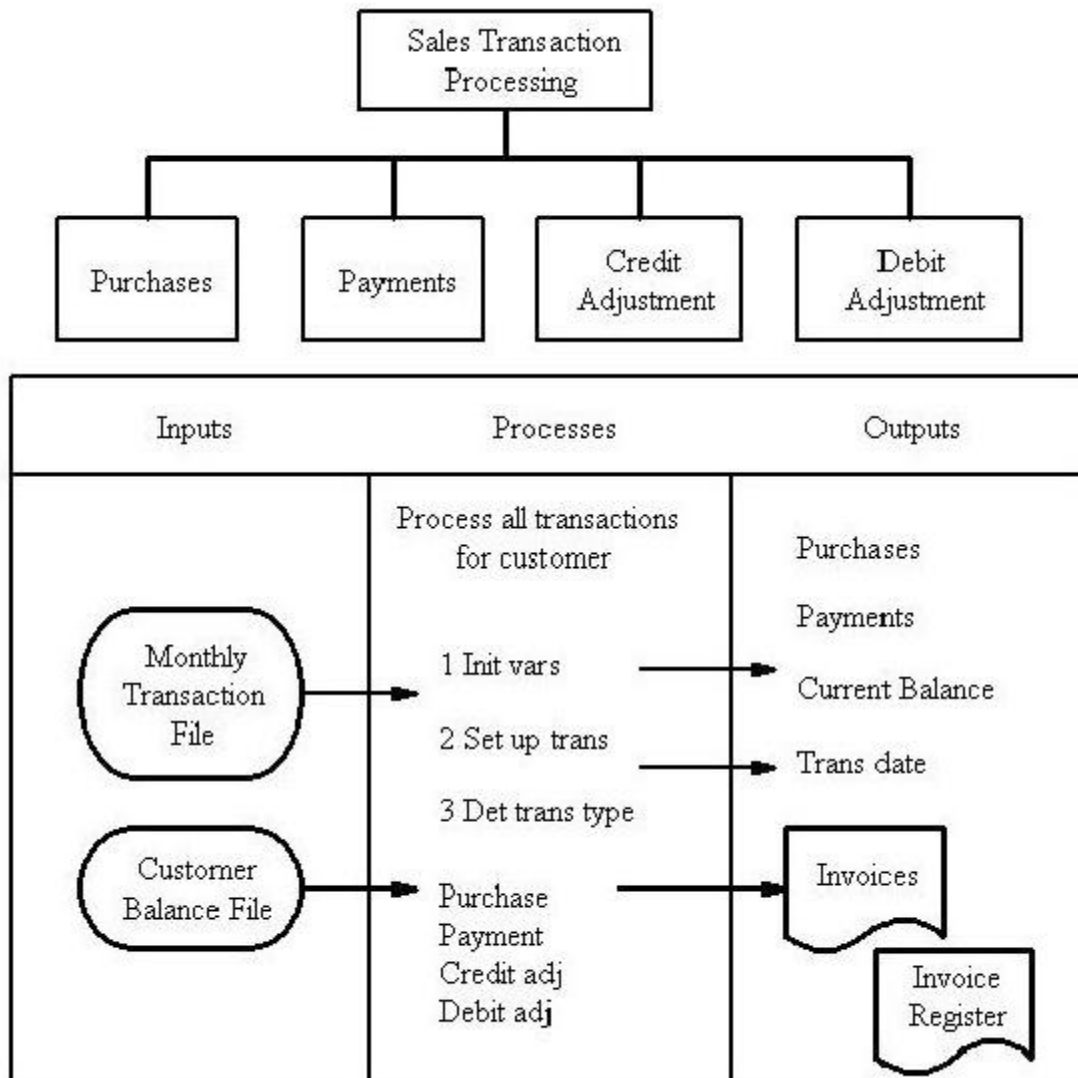
The **hierarchy chart** is useful for showing hierarchy of procedures within a program. Hierarchy charts are also called structure charts, top-down charts, or VTOC (Visual Table of Contents) charts. All these names refer to planning diagrams that are similar to a company's organization chart. Hierarchy charts depict the organization of a program but omit the specific processing logic. They describe what each part, or module, of the program does and how the modules relate to each other.

The **IPO** chart describes the system in terms of its inputs, outputs and the processes that are performed on the inputs to transform them into outputs. It provides the following :

- The Input section that contains the data items used by the process steps.
- The Output section that contains the data items created by the process steps.

- (c) Process section that contains numbered steps that describe the functions to be performed. Arrows connect them to the output steps and the input/output data items.

The IPO chart is in the form of a table with three columns, one for each of Input, Output and Process. The flow between screens is indicated by the use of arrows.



Chapter-6

Testing & System Maintenance

Q.1 Explain Input Design.

Ans.: Inaccurate input data are the most common cause of errors in data processing. Errors entered by data entry operators can be controlled by input design. Input design is the process of converting user-originated inputs to a computer based format. In the system design phase, the expanded data flow diagram identifies logical data flows, data stores, sources and destinations. The goal of designing input data is to make data entry as easy, logical and free from errors as possible. In entering data, operators need to know the following:

- (1) The allocated space for each field.
- (2) Field sequence, which must match that in the source document.
- (3) The format in which data fields are entered.

Source data are input into the system in a variety of ways, the media and devices used are Punch cards, Key-to-diskette, MICR, OCR, Optical bar code readers, CRT screens, etc. We also input data online. The three major approaches for entering data into the computer are menus, formatted forms and prompts. Menu is a selection list that simplifies computer data access or entry. Instead of remembering what to enter, the user chooses from a list of options and types the option letter associated with it. A formatted form is a preprinted form or a template that requests the user to enter data in appropriate locations. It is a fill-in-the-blank type form. In prompt the system displays one inquiry at a time, asking the user for a response.

Q.2 Explain Output Design.

Ans.: Computer output is the most important and direct source of information to the user. Efficient, intelligible output design should improve the systems relationships with the user and help in decision making. A major form of output is a hard copy from the printer. Printouts should be designed around the output requirements of the user. The output devices to consider depend of factors such as compatibility of the device with the system, response time requirements, expected print quality and number of copies needed. The media devices used are MICR, Line, matrix and daisy wheel printers, Computer output microfilm, CRT screen, graph plotters and audio response. The output design considerations are as under :

- (1) Give each output a specific name or title
- (2) Provide a sample of the output layout, including areas where printing may appear and the location of each field
- (3) State whether each output field is to include significant zeros, spaces, etc.
- (4) Specify the procedure for proving the accuracy of output data.

In online applications, information is displayed on the screen. The layout sheet for displayed output is similar to the layout chart used for designing input.

Q.3 Describe File Structure and Organisation.

Ans.: Given that a file consists, generally speaking, of a collection of records, a key element in file management is the way in which the records themselves are organized inside the file, since this heavily affects system performances as far as record finding and access. Note carefully that by "organization" we refer here to the *logical* arrangement of the records in the file (their ordering or, more generally, the presence of "closeness" relations between them based on their content), and not instead to the physical layout of the file as stored on a storage media. To prevent confusion, the latter is referred to by the expression "record blocking", and will be treated later on.

Choosing a file organization is a design decision, hence it must be done having in mind the achievement of good performance with respect to the most likely usage of the file. The criteria usually considered important are :

- (1) Fast access to single record or collection of related records.
- (2) Easy record adding/update/removal, without disrupting.
- (3) Storage efficiency.
- (4) Redundance as a warranty against data corruption.

Needless to say, these requirements are in contrast with each other for all but the most trivial situations, and it's the designer job to find a good compromise among them, yielding an adequate solution to the problem at hand. For example, easiness of adding is not an issue when defining the data organization of a CD-ROM product, whereas fast access is, given the huge amount of data that this media can store. However, as it will become apparent shortly, fast access techniques are based on the use of additional information about the records, which in turn competes with the high volumes of data to be stored.

Sequential : This is the most common structure for large files that are typically processed in their entirety, and it's at the heart of the more complex schemes. In this scheme, all the records have the same size and the same field format, with the fields having fixed size as well. The records are sorted in the file according to the content of a field of a scalar type, called "key". The key must identify uniquely a record, hence different records have different keys. This organization is well suited for batch processing of the entire file, without adding or deleting items: this kind of operation can take advantage of the fixed size of records and file; moreover, this organization is easily stored both on disk and tape. The key ordering, along with the fixed record size, makes this organization amenable to dicotomic search. However, adding and deleting records to this kind of file is a tricky process: the logical sequence of records typically matches their physical layout on the media storage, so to ease file navigation, hence adding a record and maintaining the key order requires a reorganization of the whole file. The usual solution is to make use of a "log file" (also called "transaction file"), structured as a pile, to perform this kind of modification, and periodically perform a batch update on the master file.

Indexed Sequential : An index file can be used to effectively overcome the above mentioned problem, and to speed up the key search as well. The simplest indexing structure is the single-level one: a file whose records are pairs key-pointer, where the pointer is the position in the data file of the record with the given key. Only a subset of data records, evenly spaced along the data file, are indexed, so to mark intervals of data records.

A key search then proceeds as follows: the search key is compared with the index ones to find the highest index key preceding the search one, and a linear search is performed from the record the index key points onward, until the search key is matched or until the record pointed by the next index entry is reached. In spite of the double file access (index + data) needed by this kind of search, the decrease in access time with respect to a sequential file is significant. Consider, for example, the case of simple linear search on a file with 1,000 records. With the sequential organization, an average of 500 key comparisons are necessary (assuming uniformly distributed search key among the data ones). However, using an evenly spaced index with 100 entries, the number of comparisons is reduced to 50 in the index file plus 50 in the data file: a 5:1 reduction in the number of operations. This scheme can obviously be hierarchically extended: an index is a sequential file in itself, amenable to be indexed in turn by a second-level index, and so on, thus exploiting more and more the hierarchical decomposition of the searches to decrease the access time. Obviously, if the layering of indexes is pushed too far, a point is reached when the advantages of indexing are hampered by the increased storage costs, and by the index access times as well.

Q.5 Explain Database Design.

Ans.: **Database design** is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity. The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an Object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the Database Management System or DBMS. The process of doing database design generally consists of a number of steps which will be carried

out by the database designer. Not all of these steps will be necessary in all cases. Usually, the designer must :

- Determine the data to be stored in the database.
- Determine the relationships between the different data elements.
- Superimpose a logical structure upon the data on the basis of these relationships.

Within the relational model the final step can generally be broken down into two further steps, that of determining the grouping of information within the system, generally determining what are the basic objects about which information is being stored, and then determining the relationships between these groups of information, or objects. This step is not necessary with an Object database. The tree structure of data may enforce a hierarchical model organization, with a parent-child relationship table. An Object database will simply use a one-to-many relationship between instances of an object class. It also introduces the concept of a hierarchical relationship between object classes, termed inheritance.

Q.7 Explain System Testing.

Ans.: Once source code has been generated, software must be tested to remove and correct as many errors as possible before delivery to the customer. The goal of system testing is to design a series of test cases that have a high likelihood of finding errors. Testing is the process of examining a product to determine what defects it contains. An information system is an integrated collection of software components. Components can be tested individually or in groups, or the entire system can be tested as a whole. Testing is necessary for the success of the system. A small system error can explode into a much larger problem.

The proper choice of test data is as important as the test itself. If the test data that is inputted is not valid or according to the requirements, the reliability of the output will be low. Test data may be artificial or live. Artificial data is created only for testing purposes. Live data on the other hand, is taken from the users actual files. So there can be bias toward correct values. The design of tests for software products is also a very important topic. The designs may be White Box testing or Black Box testing.

Q.8 What is Unit Testing?

Ans.: A strategy for software testing may be viewed as a spiral. Unit testing begins at the center of the spiral. Testing progresses by moving outward to integration testing, then towards validation testing and finally system testing.

Unit testing is the process of testing individual code modules before they are integrated with other modules. The unit being testing can be a function, subroutine, procedure or method. Units can also be very small groups of interrelated modules that are always executed as a group. The goal of unit testing is to identify and fix as many errors as possible before modules are combined into large units. Errors become more difficult and expensive to locate and fix when many modules are combined. Here the module interface is tested to see that information flows in and out of the program unit properly. It makes use of white box testing. Because a component is not a stand-alone program, a driver and/or stub software must be developed for each unit test. A driver is like a main program that accepts test case data, passes the data to the component and prints the results. A stub replaces modules that are subordinate the component to be tested. It uses the subordinate modules interface, does data manipulation, prints the result of entry and then returns control to the module undergoing the test.

Q.9 Explain the different conversion methods during the System Implementation Phase.

Ans.: Implementation involves all those activities that take place, to convert from the old system to the new one. The news system may be completely new, replacing an existing manual or automated system or it may be a major modification to an existing system.

Conversion is the process of changing from the old system to the new one. It must be properly planned and executed. Four common methods are used for this purpose. They are :

- (1) **Parallel Systems :** The most secure method of converting from an old to new system is to run both systems in parallel. Under this approach, users continue to operate the old system in the usual manner but they also start using the new system. This method is the safest because it ensures that in case of any problems in the new system, the organization can still fall back to the old system without loss of time or

money. The disadvantages are that it doubles the operating cost and that the new system may not get a fair trial.

- (2) **Direct Conversion :** This method converts from the old to the new system abruptly, sometimes over a week end or even overnight. The old system is used until a planned conversion day, when it is replaced by the new system. There are no parallel activities. The main disadvantages of this approach are: no other system to fall back on, if problems arise, secondly careful planning is required.
- (3) **Pilot System :** Pilot approach is often preferred in the case when the new system involves new techniques or some drastic changes in the organization performance. In this method, a working version of the system is implemented in one part of the organization, such as a single department. Based on the feedback, changes are made and the system is then installed in the remaining departments of the organization, either all at once or gradually.
- (4) **Phase-In Method :** This method is used when it is not possible to install a new system throughout an organization all at once. The conversion of files, training of personnel, etc may force the process of implementation over a period of time, ranging from weeks to months.

Q.10 Briefly describe what is Software Quality Assurance.

Ans.: Quality is a characteristic and attribute of something, which is measurable. There can be two types of quality: *quality of design* – it is the characteristics that the designers specify which will include the materials used, performance specifications, etc. and *quality of conformance* – which is the degree to which the design specifications are followed during the manufacturing process. Software Quality Assurance (SQA) consists of a means of monitoring the software engineering processes to ensure quality. It provides management with the data necessary to be informed about product quality. Software today is being developed in rapid speeds and this affects its quality. Software that is developed needs to meet certain standards for it to be certified and used by users. Software quality assurance is thus useful to keep the software development process in check and see that quality products are created for the market. Just as a team of members that are used for the development process, a SQA group is a group that assists the software team in achieving a high quality end product.

The software life cycle includes various stages of development, and each stage has a goal of quality assurance. Several factors determine the quality of a system. Among them are correctness, reliability, efficiency, usability, accuracy, etc. There are three levels of quality assurance: testing, validation and certification.

In system testing, the goal is to remove the errors in the software. This is extremely difficult and time-consuming. The system needs to be put through a “fail-test” so that we know what will make the system fail. A successful test is one that can uncover the errors so that the system can then be corrected to reach a good level of quality.

System validation checks the quality of the software in both simulated and live environments. First the software is passed through the simulated environment (not live) where the errors and failures are checked based on artificial data and user requirements. This is also known as *alpha testing*. The software is tested and verified and all changes are then made to the software. This modified software is then sent through the second phase that is the live environment. This is called *beta testing* where the software is sent to the user's site. Here the system will go through actual user data and requirements. After a scheduled time, failures and errors are documented and final correction and enhancements are made before the software is released for use.

The third level is to certify that the program or software package is correct and conforms to all standards. Nowadays, there is trend towards buying of ready-to-use software. So certification is of utmost importance. A package that is certified goes through a team of specialists who test, review, and determine how well it meets the vendor's claims. Certification is actually issued after the package passes the test.

Q.11 Explain Software Maintenance. Describe its classification.

Ans.: The last part of the system development life cycle is system maintenance which is actually the implementation of the post-implementation plan. When systems are installed, they are generally used for long periods. This period of use brings with it the need to continually maintain the system. Maintenance accounts for 50-80% of the total system development. Maintenance is not as rewarding and exciting as developing systems.

Maintenance can be classified as :

- (1) **Corrective** : It means repairing, processing or performance failures or making changes because of previously uncorrected problems.
- (2) **Adaptive** : It means changing the program functions.
- (3) **Perfective** : It means enhancing the performance or modifying the programs to respond to the users additional or changing needs

The greatest amount of time is spent on perfective. Maintenance covers a wide range of activities including correcting coding and design errors, updating documentation and test data.

Q.12 Describe the different types of Documentation.

Ans.: There are five types of documentation :

- (1) **Program** : Before a program is developed, the systems analyst should provide the programmer with the required documentation. The logic in some programs is best described by a flowchart. Sometimes decision tables are also useful. The main responsibility in documentation is to provide enough information to enable future programmers to understand and make necessary changes. Since programmers do not retain their jobs for a very long time, it becomes necessary that there be some kind of documentation that will be useful for the new programmers who are assigned the same system.
- (2) **Operations** : For smooth running of the system, the data entry operator must have complete knowledge about the job. The instructions must be in a form that is easily accessible to the console operator and written in simple and understandable style.
- (3) **User** : System users should have a manual that describes everything the users must know to do their job correctly. Users require two general type of information: complete details to handle everything the system processes, and an overall picture of the system.
- (4) **Management** : The documentation required by management differs a lot from that required by users. The manual should enable management to perform three functions:
 - (a) Evaluate progress on the development of system.
 - (b) Monitor the existing systems.

- (c) Understand the objectives and methods of the new and existing system.
- (5) **Systems :** This manual document the complete life cycle of the system. It documents the results of the feasibility study, the team assigned, etc. It also documents the file specification, transaction specification and output specification.

Q.13 What are CASE Tools?

Ans.: Use of automated tools to improve the speed and quality of system development work is very essential and important. One type of automated tool is a CASE tool. CASE tools are specifically designed to help system analysts complete system development tasks. A CASE tool contains a database of information about the project, called a repository. The repository stores information about the system, including models, descriptions and references that link the various models together. The CASE tool can check the models to make sure they are complete and follow the correct diagramming rules. If system information is stored in a repository, the development team can use the information in a variety of ways. Every time a team member adds information about the system, it is immediately available for everyone else.

CASE tools are often categorized as Upper CASE or Lower CASE tools. Upper CASE tools provide support for analysts during the analysis and design phases. Lower CASE tools provide support for implementation, generating programs based on specifications in the repository. CASE tools that combine support for the full life cycle are called Integrated CASE or ICASE tools.

Around the CASE repository is a collection of tools or facilities for creating system models and documentation. To use the repository, the CASE tools provide some combination of the following facilities :

- (1) Diagramming Tools
- (2) Design Generator and Code Generator
- (3) Testing Tools
- (4) Quality Management Tools
- (5) Reverse-Engineering Tools

Q.14 What is Data Warehousing?

Ans.: A data warehouse is a repository of all the data of an organization. It contains data that is necessary and useful for the management's decision support system. A data analyst can perform complex queries and analysis, such as data mining, on the data in the warehouse, without slowing down the operational system. The data warehouse is :

Subject-Oriented : The data in the database is organized so that all the data elements relating to the same real-world event or object are linked together;

Time-Variant : The changes to the data in the database are tracked and recorded so that reports can be produced showing changes over time;

Non-Volatile : Data in the database is never over-written or deleted - once committed, the data is static, read-only, but retained for future reporting; and

Integrated : The database contains data from most or all of an organization's operational applications, and that this data is made consistent.

The data warehouse architecture consists of various interconnected elements which are: 1) Operational and external database layer: the source data for the data warehouse. 2) Informational access layer: the tools, the end user access to extract and analyze the data. 3) Data Access Layer: the interface between the operational and informational access layer. 4) Metadata Layer: The data directory or repository of metadata information. The goal of a data warehouse is to bring data together from a variety of existing databases to support management and reporting needs. The generally accepted principle is that data should be stored at its most elemental level because this provides for the most useful and flexible basis for use in reporting and information analysis.

There are many advantages to using a data warehouse, some of them are :

- Data warehouses enhance end-user access to a wide variety of data.
- Decision support system users can obtain specified trend reports, e.g. the item with the most sales in a particular area within the last two years.
- Data warehouses can significantly enable commercial business applications, particularly customer relationship management (CRM) systems.

Q.15 Explain Data Mining.

Ans.: Data mining, the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. Data mining tools can answer business questions that traditionally were too time-consuming to resolve. Data mining is ready for application in the business community because it is supported by three technologies that are now sufficiently mature:

- Massive Data Collection
- Powerful Multiprocessor Computers
- Data Mining Algorithms

Computers are loaded up with lots of information about a variety of situations where an answer is known and then the data mining software on the computer must run through that data and distill the characteristics of the data that should go into the model. Once the model is built it can then be used in similar situations where you don't know the answer.

Q.16 Difference between validation & Verification?

Ans

Verification and Validation example is also given just below to this table.

Verification	Validation
1. Verification is a static practice of verifying documents, design, code and program.	1. Validation is a dynamic mechanism of validating and testing the actual product.
2. It does not involve executing the code.	2. It always involves executing the code.
3. It is human based checking of documents and files.	3. It is computer based execution of program.
4. Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking etc.	4. Validation uses methods like black box (functional) testing, gray box testing, and white box (structural) testing etc.
5. Verification is to check whether the software conforms to specifications.	5. Validation is to check whether software meets the customer expectations and requirements.

6. It can catch errors that validation cannot catch. It is low level exercise.	6. It can catch errors that verification cannot catch. It is High Level Exercise.
7. Target is requirements specification, application and software architecture, high level, complete design, and database design etc.	7. Target is actual product-a unit, a module, a bent of integrated modules, and effective final product.
8. Verification is done by QA team to ensure that the software is as per the specifications in the SRS document.	8. Validation is carried out with the involvement of testing team.
9. It generally comes first-done before validation.	9. It generally follows after verification .

Q.14 Differences Between Black Box Testing and White Box Testing?

Ans The Differences Between Black Box Testing and White Box Testing are listed below.

Criteria	Black Box Testing	White Box Testing
<i>Definition</i>	Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester	White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
<i>Levels Applicable To</i>	Mainly applicable to higher levels of testing: Acceptance Testing System Testing	Mainly applicable to lower levels of testing: Unit Testing Integration Testing
<i>Responsibility</i>	Generally, independent Software Testers	Generally, Software Developers
<i>Programming Knowledge</i>	Not Required	Required
<i>Implementation Knowledge</i>	Not Required	Required
<i>Basis for Test Cases</i>	Requirement Specifications	Detail Design

Chapter 7

Management Information System

Q.1 What is MIS? Discuss in detail?

OR

Describe the three words of MIS: Management, Information, System.

OR

Discuss the objectives and characteristics of MIS.

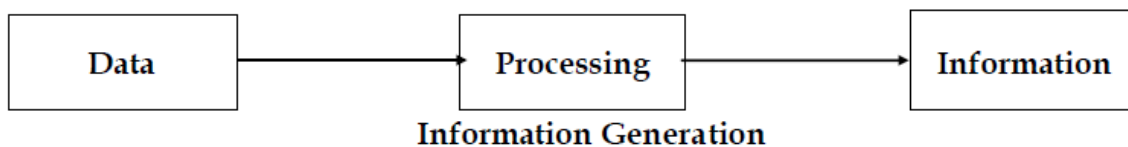
Ans Management Information Systems (MIS), referred to as Information Management and Systems, is the discipline covering the application of people, technologies, and procedures collectively called information systems, to solving business problems.

"MIS' is a planned system of collecting, storing and disseminating data in the form of information needed to carry out the functions of management."

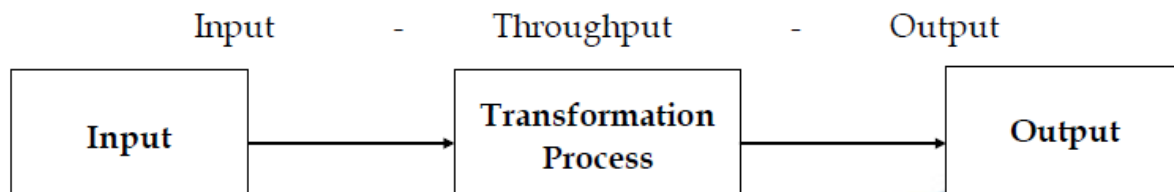
Academically, the term is commonly used to refer to the group of information management methods tied to the automation or support for human decision making, e.g. Decision Support Systems, Expert Systems, and Executive Information Systems.

Management : Management is art of getting things done through and with the people in formally organized groups. The basic functions performed by a manager in an organization are: Planning, controlling, staffing, organizing, and directing.

Information : Information is considered as valuable component of an organization. Information is data that is processed and is presented in a form which assists decision maker.



System : A system is defined as a set of elements which are joined together to achieve a common objective. The elements are interrelated and interdependent. Thus every system is said to be composed of subsystems. A system has one or multiple inputs, these inputs are processed through a transformation process to convert these input(s) to output. These subsystems are interrelated through a process of



Objectives of MIS :

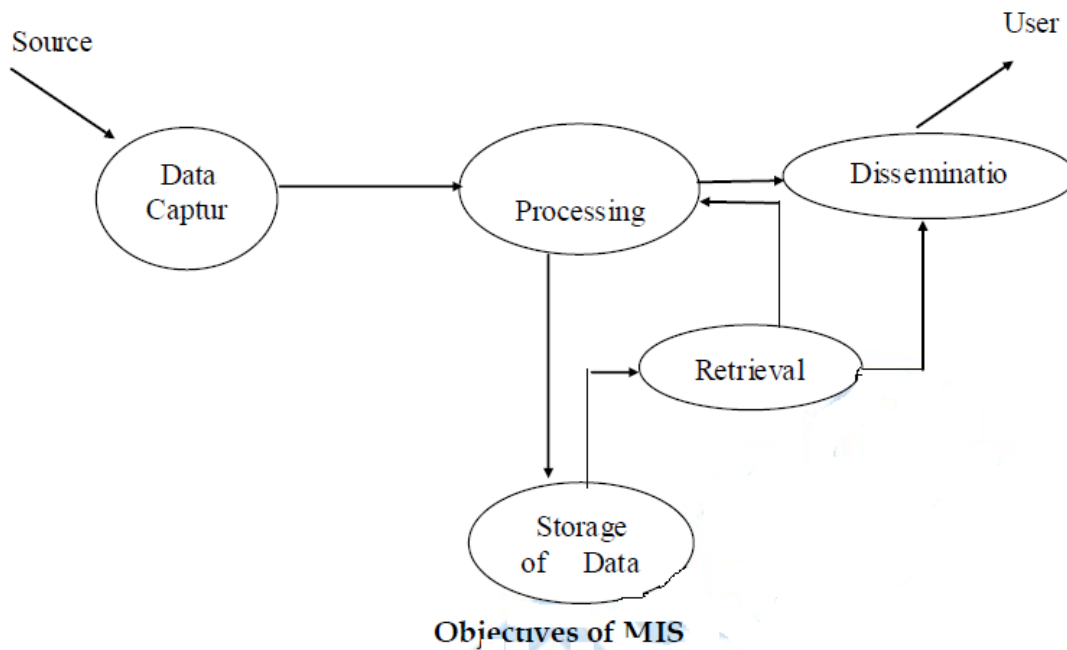
Data Capturing : MIS capture data from various internal and external sources of organization. Data capturing may be manual or through computer terminals.

Processing of Data : The captured data is processed to convert into required information. Processing of data is done by such activities as calculating, sorting, classifying, and summarizing.

Storage of Information : MIS stores the processed or unprocessed data for future use. If any information is not immediately required, it is saved as an organization record, for later use.

Retrieval of Information : MIS retrieves information from its stores as and when required by various users.

Dissemination of Information : Information, which is a finished product of MIS, is disseminated to the users in the organization. It is periodic or online through computer terminal.



Characteristics of MIS :

Systems Approach : The information system follows a systems approach. Systems approach means taking a comprehensive view or a complete look at the interlocking sub-systems that operate within an organization.

Management Oriented : Management oriented characteristic of MIS implies that the management actively directs the system development efforts. For planning of MIS, top-down approach should be followed. Top down approach suggests that the system development starts from the determination of management's needs and overall business objective. To ensure that the implementation of system's policies meet the specification of the system, continued review and participation of the manager is necessary.

Need Based : MIS design should be as per the information needs of managers at different levels.

Exception Based : MIS should be developed on the exception based also, which means that in an abnormal situation, there should be immediate reporting about the exceptional situation to the decision -makers at the required level.

Future Oriented : MIS should not merely provide past of historical information; rather it should provide information, on the basis of future projections on the actions to be initiated.

Integrated : Integration is significant because of its ability to produce more meaningful information. Integration means taking a comprehensive view or looking at the complete picture of the interlocking subsystems that operate within the company.

Common Data Flow : Common data flow includes avoiding duplication, combining similar functions and simplifying operations wherever possible. The development of common data flow is an economically sound and logical concept, but it must be viewed from a practical angle.

Long Term Planning : MIS is developed over relatively long periods. A heavy element of planning should be involved.

Sub System Concept : The MIS should be viewed as a single entity, but it must be broken down into digestible sub-systems which are more meaningful.

Central database : In the MIS there should be common data base for whole system.

Q.2 Highlight the Salient Features of Computer which makes it an essential component of MIS

OR

With the Penetration of Computer in Business Society, Information System has got a new meaning, explain.

Ans Characteristics of Computerized MIS :

- (i) Ability to process data into information with accuracy and high speed. It involves complex computation, analysis, comparisons and summarization.
- (ii) Organizing and updating of huge amount of raw data of related and unrelated nature, derived from internal and external sources at different periods of time.
- (iii) The information processing and computer technology have been so advanced that managers are able to obtain real time information about ongoing activities and events without any waiting period.
- (iv) The input data in computer can be converted into different output formats for a variety of purpose. The system is so organized that managers at different levels and in different activity units are in a position to obtain information in whatever form they want, provided that relevant — programmes or instructions have been designed for the purpose.

(v) Super-human memory, tremendous volume of data and information and the set of instructions can be stored in the computer and can be retrieved as and when needed. Management can get bit of stored information from the computer in seconds.

Advantages of Computer : The usage of computer gives following advantages in comparison to manual MIS :

- a) **Speed :** The speed of carrying out the given instructions logically and numerically is incomparable between computers and human beings. A computer can perform and give instructions in less than a millionth of second
- b) **Accuracy :** Computer can calculate very accurately without any errors.
- c) **Reliability :** The information stored in the computer is in digital format. The information can be stored for a long time and have long life. A user may feel comfortable and be rely on, while using information stored in computer.
- d) **Storage :** Computer can store huge data for a long time in comparison to human brain.
- e) **Automaticity :** Computers perform work automatically through user friendly and menu driven program.
- f) **Repetitiveness :** Computer can be used repetitively to process information without any mental fatigue as in case of human brain.
- g) **Diligence :** A computer is an electronic device. It does not suffer from the human traits of lack of concentration.
- h) **No Feeling :** Computers are devoid of any emotions. They have no feelings and no instincts because they are machines

Limitations of Computer :

- a) **Lack of Common Sense :** Computer is only an electronic device. It can not think. If we provide an incorrect data, it does not have a commonsense to question the correctness of the data.
- b) **Memory Without Brain :** Computer can store data in its memory; however, if a wrong instruction is given to computer it does not have a brain to correct the wrong instruction

Q.3 Discuss an Organizational Need for MIS in a Company?

Ans To facilitate the management decision making at all levels of company, the MIS must be integrated. MIS units are company wide. MIS is available for the Top management. The top management of company should play an active

role in designing, modifying and maintenance of the total organization wide management information system.

Information system and Information technology have become a vital component of any successful business and are regarded as major functional areas just like any other functional area of a business organization like marketing, finance, production and HR. Thus it is important to understand the area of information system just like any other functional area in the business. MIS is important because all businesses have a need for information about the tasks which are to be performed. Information and technology is used as a tool for solving problems and providing opportunities for increasing productivity and quality.

Information has always been important but it has never been so available, so current and so overwhelming. Efforts have been made for collection and retrieval of information, However, challenges still remain in the selection analysis and interpretation of the information that will further improve decision making and productivity.

MIS for Business Organization :

Support the Business Process : Treats inputs as a request from the customer and outputs as services to customer. Supports current operations and use the system to influence further way of working.

Support Operation of a Business Organization : MIS supports operations of a business organization by giving timely information, maintenance and enhancement which provides flexibility in the operation of an organizations.

To Support Decision Making : MIS supports the decision making by employee in their daily operations. MIS also supports managers in decision making to meet the goals and objectives of the organization. Different mathematical models and IT tools are used for the purpose evolving strategies to meet competitive needs.

Strategies for an Organization : Today each business is running in a competitive market. MIS supports the organization to evolve appropriate strategies for the business to assented in a competitive environment.

Q4. What do you understand by Decision Making? Discuss the nature and characteristics of Decision?

Ans The word —**decision** —is derived from the Latin word —decido||. Which means —A decision, therefore is
A Settlement

A fixed intuition to bringing to a conclusive result

A judgment

A resolution

Decision : A decision is the choice out of several options made by the decision maker to achieve some objective in a given situation.

Business Decision : Business decisions are those which are made in the process of conducting business to achieve its objective in a given situation.

Characteristic of Business Decision Making :

- a) Sequential in nature.
- b) Exceedingly complex due to risk and trade off.
- c) Influenced by personal values
- d) Made in institutional setting and business environment.

Rational Decision Making : A rational decision is the one which, effectively and efficiently, ensure the achievement of the goal for which the decision is made .In reality there is no right or wrong decision but a rational decision or irrational decision which depends on situation.

Type of Rationality :

Objectively : Maximum the value of the objectives.

Subjective : If it is minimize the attainment of value in relation to the knowledge and awareness of subject.

Consciously : Extent the process of the decision making is a conscious one

Organizationally : degree of the orientation towards the organization.

Personal: Rational to the extent is achieve's an individual's personal reason (goals).

Type of Decision Making System : There are two types of decision making system on the basis of knowledge about the environment.

(i) **Closed :** If the manager operates in a known environment then it is called closed decision making system.

Conditions :

- a) Manager knows the set of decision alternative and know their outcome in term of values.
- b) Manager has a model, by which decision alternatives can be generated, tested and ranked.

- c) The manager can choose one of them, based on some goal or objective.
- (ii) **Open** : If the manager operates in unknown environment then it is called open decision making.

Conditions :

- a) Manager does not know all alternatives.
- b) Outcome is not known.
- c) No methods or models are used.
- d) Decide objective or goal; select one where his aspirates or desire are met best.

Types of Decision : Types of decision are based on the degree of knowledge about the out come of the events which are yet to take place.

Certainty : If the manager has full knowledge of event or outcome then it is a situation of certainty.

Risk : If the manager has partial knowledge or probabilistic knowledge then it is decision under risk.

Uncertainty : If the manager does not have any knowledge, it is decision making under uncertainty

MIS converts the uncertainty to risk and risk to certainty. The decision at the low level management is certain, at middle level of the management the decision is under risk and at the top level management the decision is in under uncertain.

Nature of decision : Decision making is a complex task. To resolve the complexity the nature of decision are of two types :

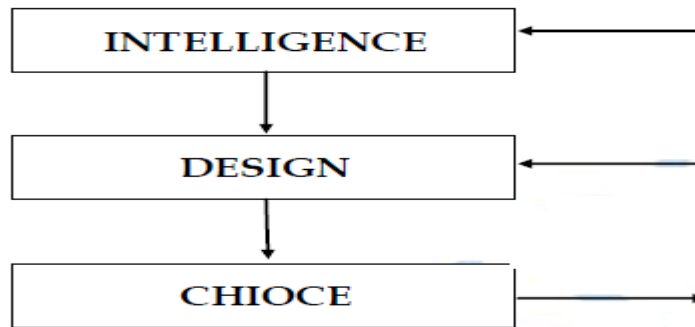
Programmed and Non-Programmed Decision :

- a) If a decision can be based on a rule, methods or even guidelines, it is called the programmed decision.
- b) A decision which can not be made by using a rule or model is the non-programmed decision.

- Q5. Discuss in brief the Hebert A. Simon 'Decision Support System Model'. Define the term Intelligence, Design and Choice as Model.**
OR

Discuss the essential steps in process of decision making.

Ans.: There are three phases in Hebert Simon model :



Intelligence : In this phase MIS collects the raw data. Further the data is sorted and merged with other data and computation are made, examined and presented. In this phase, the attention of the manager is drawn to the entire problem situation, calling for a decision.

Design : Manager develops a model of problem situation on which he can generate and test, summarizing the different decision alternatives and test the feasibility of implementation. Assess the value of the decision outcome.

Choice : In this phase the manager evolves a selection criterion and selects one alternative as decision based on selection criteria.

In these three phases if the manager fails to reach a decision, he starts the process all over again from intelligence phase where additional data and information is collected, the decision making process is refined, the selection criteria is changed and a decision is arrived at.

KEY TERMS

Abstract Class	A class that has no direct instances, but whose descendants may have direct instances.
Abstract operation	Defines the form or protocol of the operation, but not its implementation.
Acceptance testing	The process whereby actual users test a completed information system, the end result of which is the users acceptance of the system.
Access method	An operating system algorithm for storing and locating data in secondary memory.
Action stubs	That part of a decision table that lists the actions that result for a given set of conditions.
Activation	The time period during which an object performs an operation.
Actor	An external entity that interacts with the system (similar to an external entity in data flow diagramming).
Adaptive maintenance	Changes made to a system to evolve its functionality to changing business needs or technologies.
Afferent module	A module of a structure chart related to input to the system.
Affinity clustering	The process of arranging planning matrix information so that clusters of information with some predetermined level or type of affinity are placed next to each other on a matrix report.
Aggregation	A part-of relationship between a component object and an aggregate object.
Alias	An alternative name given to an attribute.
Alpha testing	User testing of a completed information system using simulated data.
Analysis	The third phase of the SDLC in which the current system is studied and alternative replacement systems are proposed.
Analysis tools	CASE tools that enable automatic checking for incomplete, inconsistent, or incorrect specifications in diagrams, forms, and reports.
Anomalies	Errors or inconsistencies that may result when a user attempts to update a table that contains redundant data. There are three

	types of anomalies: insertion, deletion, and modification anomalies.
Application independence	The separation of data and the definition of data from the applications that use these data.
Application program interface (API)	Software which allows a specific front-end program development platform to communicate with a particular back-end database engine, even when the front-end and back-end were not built to be compatible.
Application server	A computing server where data analysis functions primarily reside.
Application software	Computer software designed to support organizational functions or processes.
Association	A relationship between object classes
Association class	An association that has attributes or operations of its own, or that participates in relationships with other classes.
Association role	The end of an association which connects it to a class.
Associative entity	An entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances. Also called a gerund.
Asynchronous message	A message in which the sender does not have to wait for the recipient to handle the message.
Attribute	A named property or characteristic of an entity that is of interest to the organization.
Audit trail	A list of changes to a data file which allows business transactions to be traced. Both the updating and use of data should be recorded in the audit trail, since the consequences of bad data should be discovered and corrected.
Authorization rules	Controls incorporated to restrict access to systems and data and also to restrict the actions that people may take once in the system.
Backward recovery (rollback)	An approach to rebuilding a file in which before images of changed records are restored to the file in reverse order until some earlier state is achieved.
Balancing	The conservation of inputs and outputs to a data flow diagram

	process when that process is decomposed to a lower level.
Baseline modules	Software modules that have been tested, documented, and approved to be included in the most recently created version of a system.
Baseline Project Plan	A major outcome and deliverable from the project initiation and planning phase which contains the best estimate of a project's scope, benefits, costs, risks, and resource requirements.
Batch processing	Information that is collected or generated at some predetermined time interval and can be accessed via hard copy or on-line devices.
Behavior	Represents how an object acts and reacts.
Beta testing	User testing of a completed information system using real data in the real user environment.
Binary relationship	A relationship between instances of two entity types. This is the most common type of relationship encountered in data modeling.
Biometric device	An instrument that detects personal characteristics such as fingerprints, voice prints, retina prints, or signature dynamics.
Blocking factor	The number of physical records per page.
Bottom-up planning	A generic information systems planning methodology that identifies and defines IS development projects based upon solving operational business problems or taking advantage of some business opportunities.
Boundary	The line that marks the inside and outside of a system and which sets off the system from its environment.
Build routines	Guidelines that list the instructions to construct an executable system from the baseline source code.
Business case	The justification for an information system, presented in terms of the tangible and intangible economic benefits and costs, and the technical and organizational feasibility of the proposed system.
Business Process Reengineering (BPR)	The search for, and implementation of, radical change in business processes to achieve breakthrough improvements in products and services.

Business rules	Specifications that preserve the integrity of a conceptual or logical data model.
Calculated field	A field which can be derived from other database fields. Also called computed or derived field.
Candidate key	An attribute (or combination of attributes) that uniquely identifies each instance of an entity type.
Cardinality	The number of instances of entity B that can (or must) be associated with each instance of entity A.
Central transform	The area of a transform-centered information system where the most important derivation of new information takes place.
Class diagram	Shows the static structure of an object-oriented model: the object classes, their internal structure, and the relationships in which they participate.
Class-scope attribute	An attribute of a class that specifies a value common to an entire class, rather than a specific value for an instance.
Client	The (front-end) portion of the client/server database system that provides the user interface and data manipulation functions.
Client/server architecture	A LAN-based computing environment in which a central database server or engine performs all database commands sent to it from client workstations, and application programs on each client concentrate on user interface functions.
Closed-ended questions	Questions in interviews and on questionnaires that ask those responding to choose from among a set of prespecified responses.
Closed system	A system that is cut off from its environment and does not interact with it.
Code generators	CASE tools that enable the automatic generation of program and database definition code directly from the design documents, diagrams, forms, and reports stored in the repository
Cohesion	The extent to which a system or a subsystem performs a single function.
Command language interaction	A human-computer interaction method where users enter explicit statements into a system to invoke operations.

Competitive strategy	The method by which an organization attempts to achieve its mission and objectives.
Component	An irreducible part or aggregation of parts that make up a system, also called a subsystem.
Component diagram	Shows the software components or modules and their dependencies.
Composition	A part object that belongs to only one whole object and lives and dies with the whole.
Computer-aided software engineering (CASE)	Software tools that provide automated support for some portion of the systems development process.
Computing infrastructure	All the resources and practices required to help people adequately use computer systems to do their primary work.
Conceptual data model	A detailed model that captures the overall structure of organizational data while being independent of any database management system or other implementation considerations.
Concrete class	A class that can have direct instances.
Concurrency control	A method for preventing loss of data integrity due to interference between users in a multiuser environment.
Condition stubs	That part of a decision table that lists the conditions relevant to the decision.
Configuration management	The process of assuring that only authorized changes are made to a system.
Constraint	A limit to what a system can accomplish.
Constructor operation	An operation that creates a new instance of a class.
Context diagram	An overview of an organizational system that shows the system boundary, external entities that interact with the system, and the major information flows between the entities and the system.
Corporate strategic planning	An ongoing process that defines the mission, objectives, and strategies of an organization.
Corrective maintenance	Changes made to a system to repair flaws in its design, coding, or implementation.
Coupling	The extent to which subsystems depend on each other.

Critical path scheduling	A scheduling technique where the order and duration of a sequence of activities directly affect the completion date of a project.
Cross life cycle CASE	CASE tools designed to support activities that occur across multiple phases of the systems development life cycle.
Cross referencing	A feature performed by a data dictionary that enables one description of a data item to be stored and accessed by all individuals so that a single definition for a data item is established and used.
Data	Raw facts about people, objects, and events in an organization.
Data compression technique	Pattern matching and other methods which replace repeating strings of characters with codes of shorter length.
Data couple	A diagrammatic representation of the data exchanged between two modules in a structure chart.
Data dictionary	The repository of all data definitions for all organizational applications.
Data flow	Data in motion, moving from one place in a system to another.
Data flow diagram	A picture of the movement of data between external entities and the processes and data stores within a system.
Data-oriented approach	An overall strategy of information systems development that focuses on the ideal organization of data rather than where and how data are used.
Data store	Data at rest, which may take the form of many different physical representations.
Data type	A detailed coding scheme recognized by system software for representing organizational data.
Database	A shared collection of logically related data designed to meet the information needs of multiple users in an organization.
Database engine	The (back-end) portion of the client/server database system running on the server and providing database processing and shared access functions.
Database management system (DBMS)	Software that is used to create, maintain, and provide controlled access to user databases.

Decision support systems (DSS)	Computer-based systems designed to help organization members make decisions; usually composed of a database, model base, and dialogue system.
Decision table	A matrix representation of the logic of a decision, which specifies the possible conditions for the decision and the resulting actions.
Decision tree	A graphical representation of a decision situation in which decision points (nodes) are connected together by arcs (one for each alternative on a decision) and terminate in ovals (the action which is the result of all of the decisions made on the path that leads to that oval).
Default value	A value a field will assume unless an explicit value is entered for that field.
Degree	The number of entity types that participate in a relationship.
Design strategy	A high-level statement about the approach to developing an information system. It includes statements on the system's functionality, hardware and system software platform, and method for acquisition.
Desk checking	A testing technique in which the program code is sequentially executed manually by the reviewer.
DFD completeness	The extent to which all necessary components of a data flow diagram have been included and fully described.
DFD consistency	The extent to which information contained on one level of a set of nested data flow diagrams is also included on other levels.
Diagramming tools	CASE tools that support the creation of graphical representations of various system elements such as process flow, data relationships, and program structures.
Dialogue	The sequence of interaction between a user and a system.
Dialogue diagramming	A formal method for designing and representing human-computer dialogues using box and line diagrams.
Direct installation	Changing over from the old information system to a new one by turning off the old system when the new one is turned on.
Discount rate	The rate of return used to compute the present value of future cash flows.

Disruptive technologies	Technologies that enable the breaking of long-held business rules that inhibit organizations from making radical business changes.
Distributed database	A single logical database that is spread across computers in multiple locations which are connected by a data communications link.
Documentation	<i>See</i> External documentation, Internal documentation, System documentation, User documentation.
Documentation generators	CASE tools that enable the easy production of both technical and user documentation in standard formats.
Domain	The set of all data types and values that an attribute can assume.
Drop-down menu	A menu positioning method that places the access point of the menu near the top line of the display; when accessed, menus open by dropping down onto the display.
DSS generators	General purpose computer-based tools used to develop specific decision support systems.
Economic feasibility	A process of identifying the financial benefits and costs associated with a development project.
Efferent module	A module of a structure chart related to output from the system.
Electronic performance support system (EPSS)	Component of a software package or application in which training and educational information is embedded. An EPSS can take several forms, including a tutorial, an expert system shell, and hypertext jumps to reference material.
Encapsulation	The technique of hiding the internal implementation details of an object from its external view.
Encryption	The coding (or scrambling) of data so that they cannot be read by humans.
End users	Non-information-system professionals in an organization who specify the business requirements for and use software applications. End users often request new or modified applications, test and approve applications, and may serve on project teams as business experts.
End-user development	An approach to systems development in which users who are not computer experts satisfy their own computing needs

	through the use of high-level software and languages such as electronic spreadsheets and relational database management systems.
Entity instance (instance)	A single occurrence of an entity type.
Entity-relationship data model (E-R model)	A detailed, logical representation of the entities, associations, and data elements for an organization or business area.
Entity-relationship diagram (E-R diagram)	A graphical representation of an E-R model.
Entity type	A collection of entities that share common properties or characteristics.
Environment	Everything external to a system which interacts with the system.
Event	Something that takes place at a certain point in time; a noteworthy occurrence that triggers a state transition.
Exclusive relationships	A set of relationships for which an entity instance can participate in only one of the relationships at a time.
Executive support systems	Computer-based systems developed to support the information-intensive but limited-time decision making of executives (also referred to as executive information systems).
Expert systems	Computer-based systems designed to mimic the performance of human experts.
External documentation	System documentation that includes the outcome of structured diagramming techniques such as data flow and entity-relationship diagrams.
External information	Information that is collected from or created for individuals and groups external to an organization.
Feasibility	<i>See</i> Economic feasibility, Legal and contractual feasibility, Operational feasibility, Political feasibility, Schedule feasibility, Technical feasibility.
Field	The smallest unit of named application data recognized by system software.
File organization	A technique for physically arranging the records of a file on secondary storage devices.

File server	A device that manages file operations and is shared by each client PC attached to a LAN.
First normal form (1NF)	A relation that contains no repeating data.
Flag	A diagrammatic representation of a message passed between two modules.
Foreign key	An attribute that appears as a nonkey attribute in one relation and as a primary key attribute (or part of a primary key) in another relation.
Form	A business document that contains some pre-defined data and may include some areas where additional data are to be filled in. An instance of a form is typically based on one database record.
Form and report generators	CASE tools that support the creation of system forms and reports in order to prototype how systems will "look and feel" to users.
Form interaction	A highly intuitive human-computer interaction method whereby data fields are formatted in a manner similar to paper-based forms.
Formal system	The official way a system works as described in organizational documentation.
Forward recovery (rollforward)	An approach to rebuilding a file in which one starts with an earlier version of the file and either reruns prior transactions or replaces a record with its image after each transaction.
Functional decomposition	An iterative process of breaking the description of a system down into finer and finer detail which creates a set of charts in which one process on a given chart is explained in greater detail on another chart.
Functional dependency	A particular relationship between two attributes. For any relation R, attribute B is functionally dependent on attribute A if, for every valid instance of A, that value of A uniquely determines the value of B. The functional dependence of B on A is represented as $A > B$.
Gantt chart	A graphical representation of a project that shows each task activity as a horizontal bar whose length is proportional to its time for completion.
Hashed file organization	The address for each record is determined using a hashing

	algorithm.
Hashing algorithm	A routine that converts a primary key value into a relative record number (or relative file address).
Help desk	A single point of contact for all user inquiries and problems about a particular information system or for all users in a particular department.
Homonym	A single name that is used for two or more different attributes (for example, the term invoice to refer to both a customer invoice and a supplier invoice).
Horizontal partitioning	Distributing the rows of a table into several separate tables.
I-CASE	An automated systems development environment that provides numerous tools to create diagrams, forms, and reports; provides analysis, reporting, and code generation facilities; and seamlessly shares and integrates data across and between tools.
Icon	Graphical pictures that represent specific functions within a system.
Identifier	A candidate key that has been selected as the unique, identifying characteristic for an entity type.
Implementation	The sixth phase of the SDLC in which the information system is coded, tested, installed, and supported in the organization.
Incremental commitment	A strategy in systems analysis and design in which the project is reviewed after each phase and continuation of the project is rejustified in each of these reviews.
Index	A table or other data structure used to determine the location of rows in a file that satisfy some condition.
Indexed file organization	The records are either stored sequentially or non sequentially and an index is created that allows software to locate individual records.
Indifferent condition	In a decision table, a condition whose value does not affect which actions are taken for two or more rules.
Informal system	The way a system actually works.
Information	Data that have been processed and presented in a form suitable for human interpretation, often with the purpose of revealing trends or patterns.

Information center	An organizational unit whose mission is to support users in exploiting information technology.
Information repository	Automated tools to manage and control access to organizational business information and application portfolios as components within a comprehensive repository.
Information systems analysis and design	The complex organizational process whereby computer-based information systems are developed and maintained.
Information systems planning (ISP)	An orderly means of assessing the information needs of an organization and defining the systems, databases, and technologies that will best satisfy those needs.
Inheritance	The property that occurs when entity types or object classes are arranged in a hierarchy and each entity type or object class assumes the attributes and methods of its ancestors; that is, those higher up in the hierarchy. Inheritance allows new but related classes to be derived from existing classes.
Input	Whatever a system takes from its environment in order to fulfill its purpose.
Inspections	A testing technique in which participants examine program code for predictable language-specific errors.
Installation	The organizational process of changing over from the current information system to a new one.
Intangible benefit	A benefit derived from the creation of an information system that cannot be easily measured in dollars or with certainty. (6) <i>See also</i> Tangible benefit.
Intangible cost	A cost associated with an information system that cannot be easily measured in terms of dollars or with certainty.
Integration testing	The process of bringing together all of the modules that a program comprises for testing purposes. Modules are typically integrated in a top-down, incremental fashion.
Interface	In systems theory, the point of contact where a system meets its environment or where subsystems meet each other.
Internal documentation	System documentation that is part of the program source code or is generated at compile time.
Internal information	Information that is collected, generated, or consumed within an organization.

Interrelated components	Dependence of one subsystem on one or more subsystems.
JAD session leader	The trained individual who plans and leads Joint Application Design sessions.
Joint Application Design (JAD)	A structured process in which users, managers, and analysts work together for several days in a series of intensive meetings to specify or review system requirements.
Key business processes	The structured, measured set of activities designed to produce a specific output for a particular customer or market.
Knowledge engineers	Computer professionals whose job it is to elicit knowledge from domain experts in order to develop expert systems. (Website)
Legal and contractual feasibility	The process of assessing potential legal and contractual ramifications due to the construction of a system.
Level-0 diagram	A data flow diagram that represents a systems major processes, data flows, and data stores at a high level of detail.
Level-n diagram	A DFD that is the result of n nested decompositions of a series of subprocesses from a process on a level-0 diagram.
Local area network (LAN)	The cabling, hardware, and software used to connect workstations, computers, and file servers located in a confined geographical area (typically within one building or campus).
Location transparency	A design goal for a distributed database which says that a user (or user program) requesting data need not know at which site those data are located.
Logical database model	A description of data using a notation that corresponds to an organization of data used by database management systems.
Logical design	The fourth phase of the SDLC in which all functional features of the system chosen for development in analysis are described independently of any computer platform.
Logical system description	Description of a system that focuses on the systems function and purpose without regard to how the system will be physically implemented.
Lower CASE	CASE tools designed to support the implementation and maintenance phases of the systems development life cycle.

Maintainability	The ease with which software can be understood, corrected, adapted, and enhanced.
Maintenance	The final phase of the SDLC in which an information system is systematically repaired and improved; or changes made to a system to fix or enhance its functionality.
Management information systems (MIS)	Computer-based systems designed to provide standard reports for managers about transaction data.
Mean time between failures (MTBF)	A measurement of error occurrences that can be tracked over time to indicate the quality of a system.
Menu interaction	A human-computer interaction method where a list of system options is provided and a specific command is invoked by user selection of a menu option.
Method	The implementation of an operation.
Middleware	A combination of hardware, software, and communication technologies that bring together data management, presentation, and analysis into a three-tiered client/server environment.
Mission statement	A statement that makes it clear what business a company is in.
Modularity	Dividing a system up into chunks or modules of a relatively uniform size.
Module	A self-contained component of a system, defined by function.
Multiple classification	Shows that an object is an instance of more than one class.
Multiplicity	Indicates how many objects participate in a given relationship.
Multivalued attribute	An attribute that may take on more than one value for each entity instance.
Natural language interaction	A human-computer interaction method where inputs to and outputs from a computer-based application are in a conventional speaking language such as English.
Normal form	A state of a relation that can be determined by applying simple rules regarding dependencies to that relation.
Normalization	The process of converting complex data structures into simple, stable data structures.

Null value	A special field value, distinct from 0, blank, or any other value, that indicates that the value for the field is missing or otherwise unknown.
Object	An entity that has a well-defined role in the application domain and has state, behavior, and identity.
Object-based interaction	A human-computer interaction method where symbols are used to represent commands or functions.
Object class (class)	A set of objects that share a common structure and a common behavior.
Object diagram	A graph of instances that are compatible with a given class diagram.
Object-oriented analysis and design (OOAD)	Systems development methodologies and techniques based on objects rather than data or processes.
Objective statements	A series of statements that express organizations qualitative and quantitative goals for reaching a desired future position.
On-line processing	The collection and delivery of the most recent available information, typically through an on-line workstation. (14)
One-time cost	A cost associated with project start-up and development, or system start-up. (6)
Open-ended questions	Questions in interviews and on questionnaires that have no prespecified answers.
Open system	A system that interacts freely with its environment, taking input and returning output.
Operation	A function or a service that is provided by all the instances of a class.
Operational feasibility	The process of assessing the degree to which a proposed system solves business problems or takes advantage of business opportunities.
Output	Whatever a system returns to its environment in order to fulfill its purpose.
Outsourcing	The practice of turning over responsibility of some to all of an organization's information systems applications and operations to an outside firm.
Overriding	The process of replacing a method inherited from a super class by a more specific implementation of that method in a

	subclass.
Package	A set of cohesive, tightly coupled classes representing a subsystem.
Page	The amount of data read or written in one secondary memory (disk) input or output operation. For I/O with a magnetic tape, the equivalent term is record block.
Parallel installation	Running the old information system and the new one at the same time until management decides the old system can be turned off.
Partial functional dependency	A dependency in which one or more nonkey attributes are functionally dependent on part, but not all, of the primary key.
Participatory Design (PD)	A systems development approach that originated in Northern Europe in which users and the improvement in their work lives are the central focus.
Perfective maintenance	Changes made to a system to add new features or to improve performance.
PERT chart	A diagram that depicts project activities and their inter-relationships. PERT stands for Program Evaluation Review Technique.
Phased installation	Changing from the old information system to the new one incrementally, starting with one or a few functional components and then gradually extending the installation to cover the whole new system.
Physical design	The fifth phase of the SDLC in which the logical specifications of the system from logical design are transformed into technology-specific details from which all programming and system construction can be accomplished.
Physical file	A named set of contiguous records.
Physical record	A group of fields stored in adjacent memory locations and retrieved together as a unit.
Physical system description	Description of a system that focuses on how the system will be materially constructed.
Picture (or template)	A pattern of codes that restricts the width and possible values for each position of a field.

Pointer	A field of data that can be used to locate a related field or record of data.
Political feasibility	The process of evaluating how key stakeholders within the organization view the proposed system.
Polymorphism	The same operation may apply to two or more classes in different ways.
Pop-up menu	A menu positioning method that places a menu near the current cursor position.
Present value	The current value of a future cash flow.
Preventive maintenance	Changes made to a system to avoid possible future problems.
Primitive DFD	The lowest level of decomposition for a data flow diagram.
Process	The work or actions performed on data so that they are transformed, stored, or distributed.
Process-oriented approach	An overall strategy to information systems development that focuses on how and when data are moved through and changed by an information system.
Processing logic	The steps by which data are transformed or moved and a description of the events that trigger these steps.
Project	A planned undertaking of related activities to reach an objective that has a beginning and an end.
Project close-down	The final phase of the project management process that focuses on bringing a project to an end.
Project execution	The third phase of the project management process in which the plans created in the prior phases (project initiation and planning) are put into action.
Project identification and selection	The first phase of the SDLC in which an organizations total information system needs are identified, analyzed, prioritized, and arranged.
Project initiation	The first phase of the project management process in which activities are performed to assess the size, scope, and complexity of the project and to establish procedures to support later project activities.
Project initiation and planning	The second phase of the SDLC in which a potential information systems project is explained and an argument for

	continuing or not continuing with the project is presented; a detailed plan is also developed for conducting the remaining phases of the SDLC for the proposed system.
Project management	A controlled process of initiating, planning, executing, and closing down a project.
Project manager	An individual with a diverse set of skills--management, leadership, technical, conflict management, and customer relationship--who is responsible for initiating, planning, executing, and closing down a project.
Project planning	The second phase of the project management process which focuses on defining clear, discrete activities and the work needed to complete each activity within a single project.
Project workbook	An on-line or hard copy repository for all project correspondence, inputs, outputs, deliverables, procedures, and standards that is used for performing project audits, orientation of new team members, communication with management and customers, scoping future projects, and performing post-project reviews.
Prototyping	An iterative process of systems development in which requirements are converted to a working system which is continually revised through close work between an analyst and users.
Pseudocode	A method for representing the instructions in a module with language very similar to computer programming code.
Purpose	The overall goal or function of a system.
Query operation	An operation that accesses the state of an object but does not alter the state.
Rapid Application Development (RAD)	Systems development methodology created to radically decrease the time needed to design and implement information systems. RAD relies on extensive user involvement, Joint Application Design sessions, prototyping, integrated CASE tools, and code generators.
Record partitioning	The process of splitting logical records into separate physical segments based on affinity of use.
Recurring cost	A cost resulting from the ongoing evolution and use of a system.
Recursive foreign key	A foreign key in a relation that references the primary key

	values of that same relation.
Reengineering	Automated tools that read program source code as input, perform an analysis of the programs data and logic, and then automatically, or interactively with a systems analyst, alter an existing system in an effort to improve its quality or performance.
Referential integrity	An integrity constraint specifying that the value (or existence) of an attribute in one relation depends on the value (or existence) of an attribute in the same or another relation.
Relation	A named, two-dimensional table of data. Each relation consists of a set of named columns and an arbitrary number of unnamed rows.
Relational database model	A data model that represents data in the form of tables or relations.
Relationship	An association between the instances of one or more entity types that is of interest to the organization.
Repeating group	A set of two or more multi valued attributes that are logically related.
Report	A business document that contains only pre-defined data; that is, it is a passive document used solely for reading or viewing. A report typically contains data from many unrelated records or transactions.
Repository	A centralized database that contains all diagrams, form and report definitions, data structure, data definitions, process flows and logic, and definitions of other organizational and system components; it provides a set of mechanisms and structures to achieve seamless data-to-tool and data-to-data integration.
Resource	Any person, group of people, piece of equipment, or material used in accomplishing an activity.
Reusability	The ability to design software modules in a manner so that they can be used again and again in different systems without significant modification.
Reverse engineering	Automated tools that read program source code as input and create graphical and textual representations of program design-level information such as program control structures, data structures, logical flow, and data flow.

Rules	That part of a decision table that specifies which actions are to be followed for a given set of conditions.
Schedule feasibility	The process of assessing the degree to which the potential timeframe and completion dates for all major activities within a project meet organizational deadlines and constraints for affecting change.
Scribe	The person who makes detailed notes of the happenings at a Joint Application Design session.
Second normal form (2NF)	A relation is in second normal form if it is in first normal form and every non key attribute is fully functionally dependent on the primary key. Thus no non key attribute is functionally dependent on part (but not all) of the primary key.
Secondary key	One or a combination of fields for which more than one record may have the same combination of values.
Sequence diagram	Depicts the interactions among objects during a certain period of time.
Sequential file organization	The records in the file are stored in sequence according to a primary key value.
Single location installation	Trying out a new information system at one site and using the experience to decide if and how the new system should be deployed throughout the organization.
Slack time	The amount of time that an activity can be delayed without delaying the project.
Smart card	A thin plastic card the size of a credit card with an embedded microprocessor and memory.
Source/sink	The origin and/or destination of data, sometimes referred to as external entities.
Stakeholder	A person who has an interest in an existing or new information system. A stakeholder is someone who is involved in the development of a system, in the use of a system, or someone who has authority over the parts of the organization affected by the system.
State	Encompasses an objects properties (attributes and relationships) and the values those properties have.

State diagram	A model of the states of an object and the events that cause the object to change from one state to another.
State transition	Changes in the attributes of an object or in the links an object has with other objects.
Statement of Work (SOW)	Document prepared for the customer during project initiation and planning that describes what the project will deliver and outlines generally at a high level all work required to complete the project.
Structure chart	Hierarchical diagram that shows how an information system is organized.
Structured English	Modified form of the English language used to specify the logic of information system processes. Although there is no single standard, Structured English typically relies on action verbs and noun phrases and contains no adjectives or adverbs.
Stub testing	A technique used in testing modules, especially where modules are written and tested in a top-down fashion, where a few lines of code are used to substitute for subordinate modules.
Support	Providing ongoing educational and problem solving assistance to information system users. For in-house developed systems, support materials and jobs will have to be prepared or designed as part of the implementation process.
Synchronous message	A type of message in which the caller has to wait for the receiving object to finish executing the called operation before it can resume execution itself.
Synonyms	Two different names that are used to refer to the same data item (for example, car and automobile).
System	An inter-related set of components, with an identifiable boundary, working together for some purpose.
System documentation	Detailed information about a systems design specifications, its internal workings, and its functionality.
System librarian	A person responsible for controlling the checking-out and checking-in of baseline modules for a system when a system is being developed or maintained.
System testing	The bringing together of all the programs that a system comprises for testing purposes. Programs are typically integrated in a top-down, incremental fashion.

Systems analyst	The organizational role most responsible for the analysis and design of information systems.
Systems development life cycle (SDLC)	The traditional methodology used to develop, maintain, and replace information systems.
Systems development methodology	A standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.
Tangible benefit	A benefit derived from the creation of an information system that can be measured in dollars and with certainty.
Tangible cost	A cost associated with an information system that can be measured in terms of dollars and with certainty.
Technical feasibility	A process of assessing the development organizations ability to construct a proposed system.
Ternary relationship	A simultaneous relationship among instances of three entity types.
Third normal form (3NF)	A relation is in third normal form if it is in second normal form and no transitive dependencies exist.
Three-tiered client/server	Advanced client/server architectures in which there are three logical and distinct applications--data management, presentation, and analysis--which are combined to create a single information system.
Top-down planning	A generic information systems planning methodology that attempts to gain a broad understanding of the information system needs of the entire organization.
Transaction analysis	The process of turning data flow diagrams of a transaction-centered system into structure charts.
Transaction-centered system	An information system that has as its focus the dispatch of data to their appropriate locations for processing.
Transaction processing systems (TPS)	Computer-based versions of manual organization systems dedicated to handling the organizations transactions; e.g., payroll.
Transactions	Individual, simple events in the life of an organization that contain data about organizational activity.
Transform analysis	The process of turning data flow diagrams of a transform-centered system into structure charts.

Transform-centered system	An information system that has as its focus the derivation of new information from existing data.
Transitive dependency	A functional dependency between two (or more) non key attributes in a relation.
Triggering operation (trigger)	An assertion or rule that governs the validity of data manipulation operations such as insert, update, and delete.
Turnaround document	Information that is delivered to an external customer as an output that can be returned to provide new information as an input to an information system.
Unary relationship (recursive relationship)	A relationship between the instances of one entity type.
Unit testing	Method in which each module is tested alone in an attempt to discover any errors in its code.
Update operation	An operation that alters the state of an object.
Upper CASE	CASE tools designed to support information planning and the project identification and selection, project initiation and planning, analysis, and design phases of the systems development life cycle.
Usability	An overall evaluation of how a system performs in supporting a particular user for a particular task.
Use case	A complete sequence of related actions initiated by an actor, it represents a specific way of using the system.
Use-case diagram	A diagram that depicts the use cases and actors for a system.
User documentation	Written or other visual information about an application system, how it works, and how to use it.
Value chain analysis	The process of analyzing an organizations activities to determine where value is added to products and/or services and the cost are incurred for doing so; usually also includes a comparison with the activities, added value, and costs of other organizations for the purpose of making improvements in the organizations operations and performance.
Vertical partitioning	Distributing the columns of a table into several separate tables.
View	A subset of the database that is presented to one or more users.

Walkthrough	A peer group review of any product created during the systems development process. Also called structured walkthrough.
Well-structured relation	A relation that contains a minimum amount of redundancy and allows users to insert, modify, and delete the rows in a table without errors or inconsistencies.
Work breakdown structure	The process of dividing the project into manageable tasks and logically ordering them to ensure a smooth evolution between tasks.





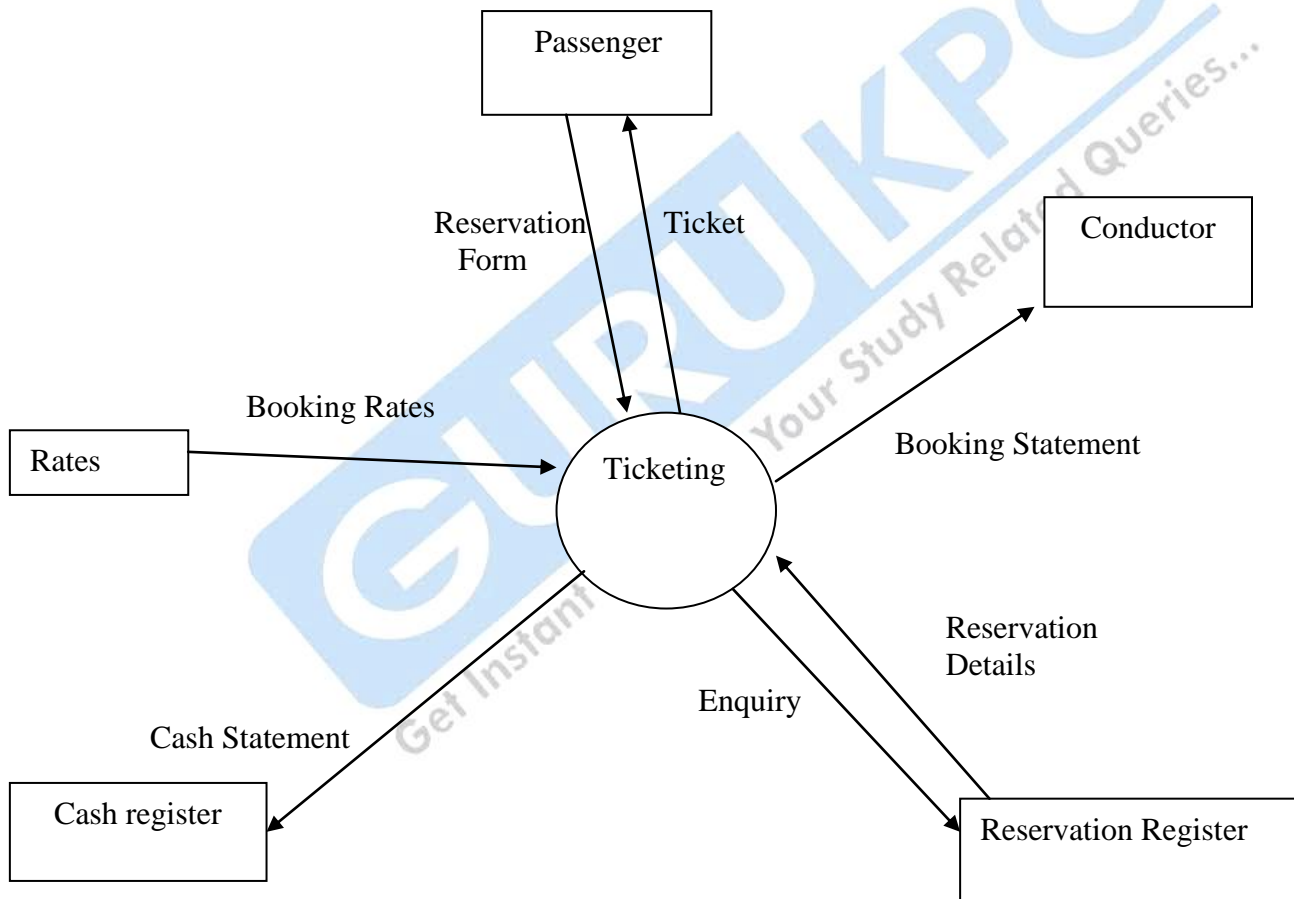
CASE STUDY

CASE 1: A Railway reservation system functions as follows:

The passenger fills in a reservation form giving his/her particulars and source and destination details. The counter clerk ensures whether seats are available or not from the reservation register. If seat is not available, the form is returned back to the passenger. Otherwise the clerk will prepare the tickets, compute the charges for the tickets and a booking statement is composed. One copy of the booking statement is retained as office copy, one is given to the train conductor and one copy is pasted on the compartment. A cash statement is prepared at the end of each shift.

PREPARE A DATAFLOW DIAGRAM FOR THE ABOVE SYSTEM

SOLUTION:



Context Diagram for Railway Reservation System

First Level Data Flow Diagram

