Women University Press
Jayoti Publication Desk
ISBN No. : 978-93-90892-07-5

System Analysis & Design

PART-I

JV'n Dr. Anamika Ahirwar

JAYOTI VIDYAPEETH WOMEN'S UNIVERSITY, JAIPUR
UGC Approved Under 2(f) & 12(b) | NAAC Accredited | Recognized by Statutory Councils

Printed by :
JAYOTI PUBLICATION DESK

Published by :
Women University Press
Jayoti Vidyapeeth Women's University, Jaipur

Faculty of Education & Methodology

Title: System analysis and design Part-1

Author Name Dr. Anamika Ahirwar

Published By: Women University Press

Publisher's Address: Jayoti Vidyapeeth Women's University, Jaipur
Vedaant Gyan Valley,
Village-Jharna, Mahala Jobner Link Road, NH-8
Jaipur Ajmer Express Way,
Jaipur-303122, Rajasthan (INDIA)

Printer's Detail: Jayoti Publication Desk

Edition Detail: I

ISBN: 978-93-90892-07-5

SYSTEM ANALYSIS

&

DESIGN - PART I

Contents

CHAPTER III: DATA INFORMATION AND RELATED ATTRIBUTES INFORMATION SYSTEMS

CHAPTER I

SYSTEM: DEFINITION AND CONCEPT

1.1. Introduction

Systems development is systematic method which incorporates phases like coming up with, analysis, design, deployment, and maintenance. Here, we are going to primarily concentrate on −

- Systems analysis
- Systems design

Systems Analysis

It is a method of assembling and deciphering facts, distinguishing the issues, and decomposition of a system into its elements.

System analysis is conducted for the aim of finding out a system or its elements so as to spot its objectives. it's a drag determination technique that improves the system and ensures that everyone the elements of the system work with efficiency to accomplish their purpose.

Analysis specifies what the system ought to do.

Systems Design

It is a method of designing a replacement business system or commutation of existing system by process its elements or modules to satisfy the precise necessities. Before coming up with, you wish to grasp the previous system totally and confirm however computers will best be utilized in order to work with efficiency.

System design focuses on a way to accomplish the target of the system.

System Analysis and design (SAD) principally focuses on −

- Systems
- Processes
- Technology

1.1.1. What is a System?

The word System comes from Greek word *Systema*, which implies an organized relationship between any set of parts to attain some common cause or objective.

A system is “an orderly grouping of dependent parts connected along in line with a concept to attain a particular goal.”

1.1.2. Constraints of a System

A system should have 3 basic constraints −

- A system should have some structure and behavior that is intended to attain a predefined objective.
- Interconnectivity and mutuality should exist among the system parts.
- The objectives of the organization have the next priority than the objectives of its subsystems.

For example, traffic management system, payroll system, automatic library system, human resources system.

1.1.3. Properties of a System

A system has the following the properties:

1.Organization

Organization implies structure and order. it's the arrangement of parts that helps to attain present objectives.

2.Interaction

It is outlined by the style during which the parts operate with one another.

For example, in a company, business department should move with production department and payroll with section.

3.Interdependence

Interdependence means that however the parts of a system rely upon each other. For correct functioning, the parts are coordinated and connected along in line with a nominal arrange. The output of one system is that the needed by alternative system as input.

4.Integration

Integration worries with however a system parts are connected along. It means the elements of the system work along at intervals the system although every half performs a novel performs.

5. Central Objective

The objective of system should be central. It should be real or expressed. It's not uncommon for a company to state an objective and operate to attain another.

The users should understand the most objective of a computer application early within the analysis for a successful design and conversion.

1.1.4. Elements of a System

The following Figure 1 shows the elements of a system –

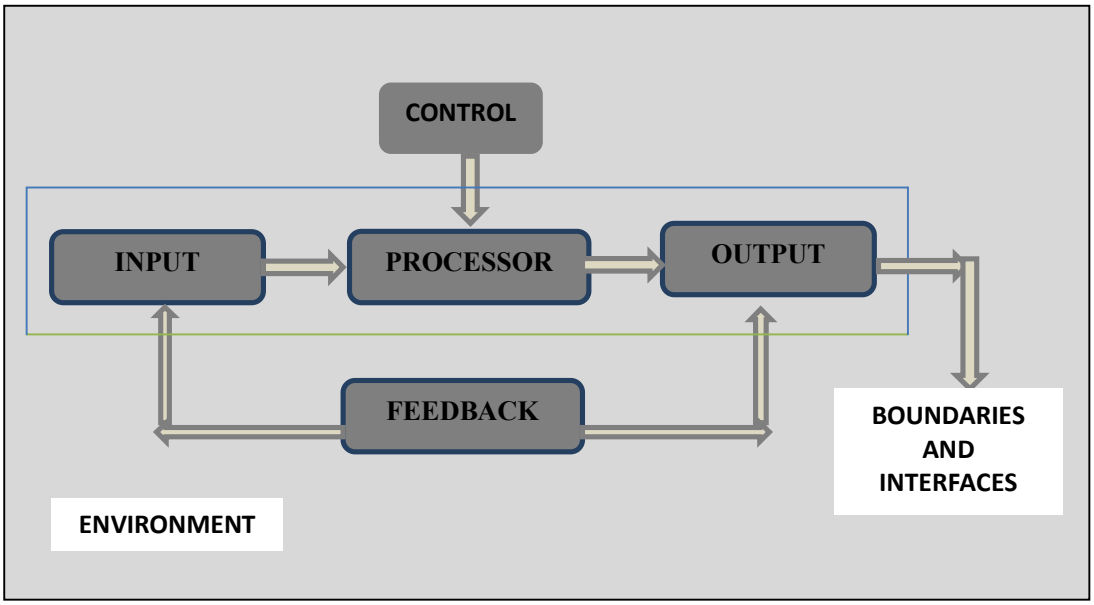

Figure 1: Components of System

Outputs and Inputs

- The main aim of a system is to provide an output that is beneficial for its user.
- Inputs area unit the data that enters into the system for process.
- Output is that the outcome of process.

Processor(s)

- The processor is that the component of a system that involves the particular transformation of input into output.
- It is that the operational element of a system. Processors might modify the input either wholly or part, betting on the output specification.
- As the output specifications modification, thus will the process. In some cases, input is additionally changed to change the processor for handling the transformation.

Control

- The management component guides the system.
- It is that the decision–making system that controls the pattern of activities governing input, processing, and output.

- The behavior of a computing system is controlled by the OS and software system. So as to stay system in balance, what and the way a lot of input is required is decided by Output Specifications.

Feedback

- Feedback provides the management in an exceedingly dynamic system.
- Positive feedback is routine in nature that encourages the performance of the system.
- Negative feedback is informational in nature that gives the controller with information for action.

Environment

- The surroundings is that the "supersystem" at intervals that a company operates.
- It is that the supply of external components that strike on the system.
- It determines however a system should operate. for instance, vendors and competitors of organization's surroundings, might offer constraints that have an effect on the particular performance of the business.

Boundaries and Interface

- A system ought to be outlined by its boundaries. Boundaries area unit the boundaries that establish its elements, processes, and interrelatedness once it interface with another system.
- Each system has boundaries that confirm its sphere of influence and management.
- The information of the boundaries of a given system is crucial in determining the character of its interface with different systems for roaring style.

1.2. Types of Systems

The systems may be divided into the following the following:

1.2.1. Physical or Abstract Systems

- Physical systems are tangible entities. We will bit and feel them.
- Physical System is also static or dynamic in nature. For instance, desks and chairs area unit the physical components of computer centrethat is static. A programmed computer may be a dynamic system during which programs, data, and applications will modification per the user's wants.
- Abstract systems area unit non-physical entities or abstract that will be formulas, illustration or model of a true system.

1.2.2. Open or Closed Systems

- An open system should move with its atmosphere. It receives inputs from and delivers outputs to the skin of the system. For instance, an information system should adapt to the ever-changing environmental conditions.

- A closed system doesn't move with its atmosphere. It's isolated from environmental influences. A totally closed system is rare essentially.

1.2.3. Adaptive and Non adaptational System

- Adaptive System responds to the modification within the atmosphere in a very way to improve their performance and to survive. For instance, human beings, animals.
- Non adaptational System is that the system that doesn't answer the environment. For instance, machines.

1.2.4. Permanent or Temporary System

- Permanent System persists for a while. For instance, business policies.
- Temporary System is formed for nominal time and afterward they're dismantled. for instance, A DJ system is ready up for a program and it's dissembled once the program.

1.2.4. Natural and manufactured System

- Natural systems area unit created by the character. For instance, scheme, seasonal system.
- Manufactured System is that the synthetic system. For instance, Rockets, dams, trains.

1.2.5. Deterministic or Probabilistic System

- Deterministic system operates in a very inevitable manner and therefore the interaction between system elements is thought with certainty. For instance, 2 molecules of H and one molecule of atomic number 8 makes water.
- Probabilistic System shows unsure behavior. The precise output isn't familiar. for instance, foretelling, mail delivery.

1.2.6. Social, Human-Machine, Machine System

- Social System is formed of individuals. For instance, social clubs, societies.
- In Human-Machine System, each human and machines area unit concerned to perform a selected task. For instance, computer programming.
- Machine System is wherever human interference is neglected. The entire tasks are unit performed by the machine. For example; a robot.

1.2.7. Man–Made information Systems

- It is an interconnected set of knowledge resources to manage data for explicit organization, underneath direct internal control.
- This system includes hardware, software, communication, data, and application for manufacturing data per the requirement of a corporation.

  Man-made data systems area unit divided into 3 types

- Formal data system data system relies on the flow of data within the type of memos, directions, etc., from prime level to lower levels of management.
- Informal data system data system is worker based mostly system that solves the day to day work connected issues.
- Computer based mostly System − this technique is directly enthusiastic about the pc for managing business applications. For instance, automatic library system, railway reservation system, industry, etc.

1.3. Systems Models

1.3.1. Schematic Models

- A schematic model may be a 2-D chart that shows system parts and their linkages.
- Different arrows are used to show data flow, material flow, and data feedback.

1.3.2. Flow System Models

- A flow system model shows the orderly flow of the fabric, energy, and data that hold the system along.
- Program analysis and Review Technique (PERT), for instance, is employed to abstract a true world system in model type.

1.3.3. Static System Models

- They represent one pair of relationships like activity–time or cost–quantity.
- The Gantt chart, for instance, offers a static image of an activity-time relationship.

1.3.4. Dynamic System Models

- Business organizations are dynamic systems. A dynamic model approximates the kind of organization or application that analysts contend with.
- It shows an ongoing, perpetually ever-changing standing of the system. It consists of:
  - Inputs that enter the system
  - The processor through that transformation takes place
  - The program(s) needed for process
  - The output(s) that result from process.

1.4. Categories of Information

There are 3 classes of data associated with managerial levels and therefore the decision managers build shown in Figure 2.

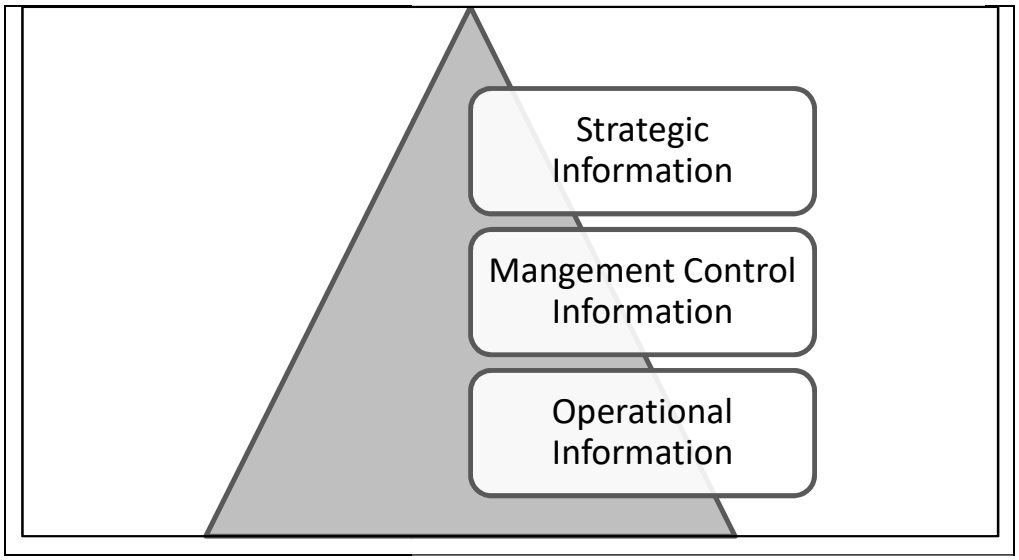

Figure 2: Categories of Information

1.4.1. Strategic information

- This data is needed by upmost management for long vary coming up with policies for next few years. For instance, trends in revenues, monetary investment, and human resources, and increment.
- This kind of data is achieved with the help of decision support system (DSS).

1.4.2. Managerial information

- This kind of data is needed by middle management for brief and intermediate vary coming up with that is in terms of months. for instance, sales analysis, income projection, and annual monetary statements.
- It is achieved with the help of Management data Systems (MIS).

1.4.3. Operational information

- This kind of data is needed by low management for daily and short term going to enforce regular operational activities. For instance, keeping worker attending records, delinquent purchase orders, and current stocks obtainable.
- It is achieved with the help of data processing Systems (DPS).

CHAPTER II

REAL TIME AND DISTRIBUTED SYSTEMS

2.1. Real-Time Systems

The design of period computing systems is that the most difficult and complicated task which will be undertaken by a programmer. By it's terribly nature, code for period systems makes demands on analysis, style and testing techniques that square measure unknown in alternative application areas.

Real-time code is extremely coupled to the external world. That is, period code should reply to the matter domain (the real world) in a very time-frame settled by the matter domain. As a result of period code should operate below rigorous performance constraints, code style is commonly driven by hardware yet as code design, package characteristics yet as application necessities, artificial language vagaries yet as style problems.

The computer is turning into ever additional gift within the daily lives of all people. Computers permit our watches to play games yet as tell time, optimize the fuel consumption rate of our latest generation cars, and sequence our appliances.... [In business, computers management machines coordinate processes, and more and more, replace manual skills and human recognition with machine-driven systems and "artificial intelligence."]

All these computing interactions –be they useful or intrusive –are samples of period computing. The computer is dominant one thing that interacts with reality on a timely basis. In fact, temporal arrangement is that the essence of the interaction... an unresponsive period system could also be worse than no system in any respect.

No quite a decade past, period code development was thought of a black magic, applied by anointed wizards who guarded their closed world with jealousy. Today, there simply don't seem to be enough wizards to travel around! Nonetheless, there's absolute confidence that the engineering of period code needs special skills.

2.1.1. System issues

Like any computer-based system, a real-time system should integrate hardware, software, human, and information base parts to properly come through a group of purposeful and performance needs. The matter for real-time systems is correct allocation. Real-time performance is commonly as necessary as operate, however allocation selections that relate to performance are typically tough to form with assurance.

• The planning of period of time system is resource forced. The first resource for a period of time system is time. It's essential to complete an outlined task inside a given range of central processor cycles. Additionally, alternative system resources, like memory size, are also listed against the clock to realize system objectives.

• Real-time systems are compact, however complicated. Though a classy real-time system might contain run over one million lines of code, the time vital portion of the package

represents an awfully little proportion of the overall. It's this little proportion of code that's generally the foremost complicated (from a recursive purpose of view).

• Period of time systems typically work while not the presence of an individual's user. Therefore, period of time package should discover issues that cause failure and mechanically live through these issues before harm to information and also the controlled surroundings happens.

2.1.2. Integration and Performance problems

Putting together a period of time system presents the system engineer with troublesome hardware and computer code selections. Once the computer code part has been allotted, careful computer code necessities are established and an elementary computer code style should be developed. Among several period of time style issues are: coordination between the period of time tasks; process of system interrupts; I/O handling to make sure that no information is lost; specifying the systems internal and external temporal order constraints, and guaranteeing the accuracy of its information base.

Each period of time style concern for computer code should be applied within the context of system performance. In most cases, the performance of a period of time system is measured united or additional time-related characteristics, however different measures like fault-tolerance may be used.

Some period of time systems are designed for applications during which solely the latency or the information transfer rate is crucial. Different period of time applications need improvement of each parameters below peak loading conditions. What is additional, period of time systems should handle their peak hundreds whereas acting variety of synchronous tasks.

Since the performance of a period of time system is decided primarily by the system latency and its information transfer rate, it's vital to know these two parameters. System latency is that the time at intervals that a system should observe an interior or external event associate degree respond with an action. Often, event detection and response generation are easy. It'sthe process of the data regarding the event to work out the acceptable response which will involve complicated, long algorithms.

Among the key parameters that have an effect on the latency are context shift and interrupt latency.Context shift involves the time and overhead to modify among tasks, and interrupt latency is that the wait before the switch is truly attainable. Different parameters that have an effect on latency are the speed of computation and of access to mass storage.

The data transfer rate indicates how fast serial or parallel, similarly as analog or digital information should be enraptured into or out of the system. Hardware vendors typically quote temporal order and capability values for performance characteristics. However, hardware specifications for performance are sometimes measured in isolation and are typically of very little worth in determinative overall period of time system performance. Therefore, I/O device performance, bus latency, buffer size, disk performance, and a bunch of different factors, though vital, are solely a part of the story of period of time system style.

Real-time systems are typically needed to method an eternal stream of incoming information. Style should assure that information isn't lost. Additionally, a period of time system should reply to events that are asynchronous. Therefore, the arrival sequence and information volume can't be simply expected ahead.

Although all computer code applications should be reliable, period of time systems create special demands on reliableness, restart, and fault recovery. as a result of the real-world is being monitored and controlled, loss of observation or management (or both) is intolerable in several circumstances (e.g., associate degree traffic management system). Consequently, period of time systems contain restart and fault-recovery mechanisms and regularly have integral redundancy to insure backup.

The need for reliableness, however, has spurred associate degree on-going dialogue regarding whether or not on-line systems, like airline reservation systems and automatic bank tellers, additionally qualify as period of time. On one hand, such on-line systems should reply to external interrupts at intervals prescribed response times on the order of one second. On the opposite hand, nothing ruinous happens if associate degree on-line system fails to satisfy response requirements; instead, solely system degradation results.

2.1.3. Interrupt Handling

One characteristic that serves to differentiate real-time systems from the other sort is interrupt handling. A real-time system should reply to external stimulae–interrupts–in a time-frame determined by the external world. As a result of multiple stimulae (interrupts) are typically gift, priorities and priority interrupts should be established. In alternative words, the foremost vital task should be repaired at intervals predefined time constraints in spite of alternative events.

Interrupt handling entails not solely storing data so the computer will properly restart the interrupted task, however conjointly avoiding deadlocks and endless loops. The approach to interrupt handling is illustrated in Figure 3. Traditional process flow is "interrupted" by an occurrence that's detected by processor hardware. Incidence is any occurrence that needs immediate service and should be generated by either hardware or software package. The state of the interrupted program is saved (i.e., all register contents, management blocks, etc. are saved) associated management is passed to an interrupt routine that branches to applicable software package for handling the interrupt. Upon completion of interrupt service, the state of the machine is repaired and traditional process flow continues.

In several things, interrupt service for one event might itself be interrupted by another, higher priority event. Interrupt priority levels (Figure 4) is also established. If a lower-priority method is accidentally allowed to interrupt a higher-priority one, {it might|it's going to|it should} be troublesome to restart the processes within the right order associated an endless loop may result.

To handle interrupts and still meet the system time constraints, several period of time in operation systems build dynamic calculations to see whether or not the system goals may be met. These dynamic calculations are supported the common frequency of incidence of events, the number of your time it takes to service them (if they'll be serviced), and therefore the routines that may interrupt them and if the dynamic calculations show that it's not possible to handle the events that may occur within the system and still meet the time constraints, the system should pick a theme of action. One potential theme involves buffering the info so it may be processed quickly once the system is prepared.

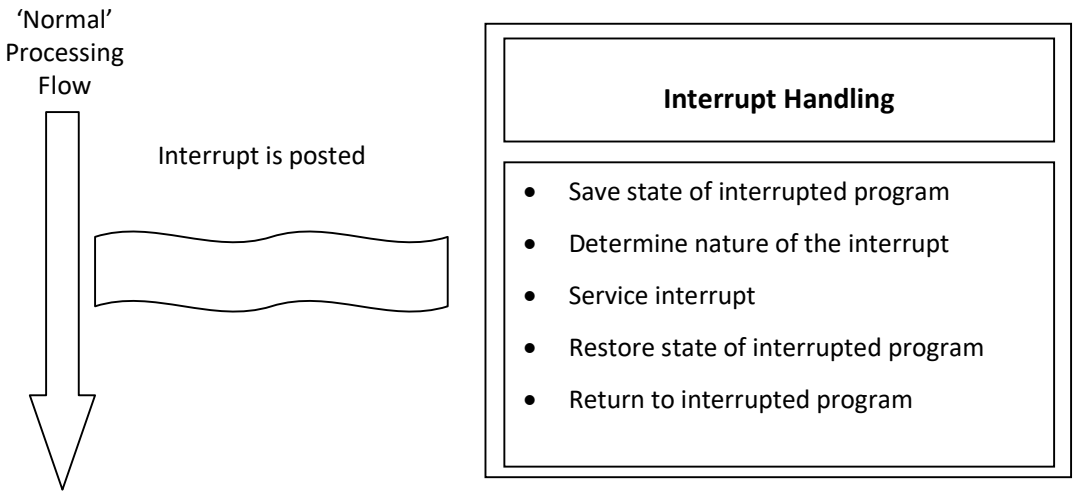

Figure 3: Interrupt handling

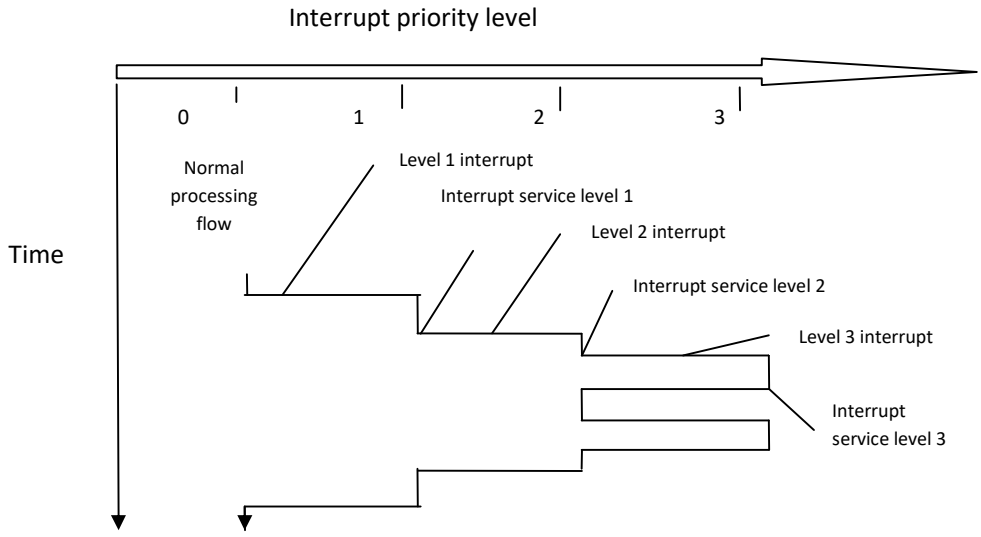

Figure 4:Interrupt priority levels

2.1.4. Real-Time Data Bases

Like several data-processing systems, real-time systems usually area unit as well as a knowledge base management operate. However, distributed knowledge bases would appear to be a most popular approach in period of time systems as a result of multi-tasking is commonplace and knowledge are usually processed in parallel. If the info base is distributed, individual tasks will access their knowledge quicker and a lot of dependably, and with fewer bottlenecks than with a centralized knowledge base. The use of a distributed knowledge base

for period of time applications divides input/output "traffic" and shortens queues of tasks looking forward to access to a knowledge base. Moreover, a failure of 1 knowledge base can seldom cause the failure of the complete system, if redundancy is made in.

The performance efficiencies achieved through the employment of a distributed knowledge base should be weighed against potential issues related to knowledge partitioning and replication. though knowledge redundancy improves time interval by provided multiple info sources, replication necessities for distributed files additionally produces logistic and overhead issues, since all the file copies should be updated. Additionally, the employment of distributed knowledge bases introduces the matter of concurrency management. Concurrency management involves synchronizing the info bases in order that all copies have the proper, identical info free for access.

The conventional approach to concurrency management relies on what area unit called protection and time stamps. At regular intervals, the subsequent tasks area unit initiated:

(1) the info base is "locked" in order that concurrency management is assured; no I/O is permitted;

(2) change happens as required;

(3) the info base is unlocked;

(4) files area unit valid to assure that everyone updates are properly made;

(5) the finished update is acknowledged. All protection tasks area unit monitored by a master clock (i.e., time stamps). The delays concerned in these procedures, moreover because the issues of avoiding inconsistent updates and dead end, mitigate against the widespread use of distributed knowledge bases.

Some techniques, however, are developed to hurry change and to unravel the concurrency downside. One among these, known as the exclusive-writer protocol maintains the consistency of replicated files by permitting solely one, exclusive writing task to update a file. It thus eliminates the high overhead of protection or time stamp procedures.

2.1.5. Real-Time Operating Systems

Choosing a real-time operating system (RTOS) for a selected application isn't any straightforward task. Some software package classifications are doable; however most don't match into neat classes with clear-cut blessings and downsides. Instead, there's extended overlap in capabilities, target systems, and alternative options.

Some real-time operating systems are applicable to a broad vary of system configurations, whereas others area unit intermeshed to a specific board or maybe microchip, despite the encompassing electronic surroundings. RTOS deliver the goods their capabilities through a mix of package options and (increasingly) a range of micro-coded capabilities enforced in hardware.

Today, 2 broad categories of operative systems are used for real-time work:

(1) dedicated RTOS designed completely for time period applications and

(2) all-purpose operative systems that are increased to supply time period capability.

The employment of a time period govt. makes time period performance possible for an all-purpose software package. Behaving like application package, the manager performs variety of software package functions–particularly those who have an effect on time period performance–faster and additional expeditiously than the overall purpose software package.

All operative systems should have a priority programming mechanism, however RTOS should offer a priority mechanism that permits high-priority interrupts to require precedence over reduced ones. Moreover, as a result of interrupts occur in response to asynchronous, nonrecurring events, they need to be repaired while not 1st taking time to swap in a very program from disk storage. Consequently, to ensure the specified latent period, a time period software package should have a mechanism for memory locking–that is, lockup a minimum of some programs in main memory in order that swapping overhead is avoided.

To determine which type of time period software package best matches associate degree application, measures of RTOS quality will be outlined and evaluated. Context change time and interrupt latency, (discussed earlier) verify interrupt-handling capability, the foremost necessary facet of a time period system. Context change time is that the time the software package takes to store the state of the pc and also the contents of the registers in order that it will come to a process task when sexual union the interrupt.

Interrupt latency, the most hold before the system gets around to change a task, happens as a result of in associate degree software package there are a unit usually non-re-entrant or important process methods that has got to be completed before associate degree interrupt will be processed.

The length of those methods (the variety of instructions) before the system will service an interrupt indicates the worst-case hold. The worst case happens if a high-priority interrupt is generated at once when the system enters a important path between associate degree interrupt and interrupt service. If the time is simply too long, the system might miss an irretrievable piece of information. It's necessary that the designer apprehend the hold in order that the system will make amends for it.

Many operative systems perform multitasking, or synchronous process, another major demand for time period systems. However to be viable for data processing, the system overhead should be low in terms of change time and memory area used.

2.1.6. Real-Time Languages

Because of the special needs for performance and dependableness demanded of real-time systems, the selection of a programming language is very important. Several general purpose programming languages (e.g., C, FORTRAN, Modula-2) are often used effectively for real-time applications. However, a category of questionable "real-time languages" (e.g., Ada, Jovial, HAL/S, Chill, and others) is usually utilized in specialised military and communications applications.

A combination of characteristics makes a period of time language completely different from a all-purpose language. These embody the multitasking capability, constructs to directly implement period of time functions, and fashionable programming options that facilitate guarantee program correctness.

A programming language that directly supports multitasking are very important as a result of a period of time system should answer asynchronous events occurring at the same time.

Though several RTOS offer multitasking capabilities, embedded period of time code typically exists while not associate OS. Instead, embedded applications are written during a language that has ample run-time support for real-time program execution. Run-time support needs less memory than associate OS, associated it are often tailored to an application, therefore increasing performance.

A real time system that has been designed to accommodate multiple tasks should conjointly accommodate intertask synchronization. A programming language that directly supports synchronization primitives like SCHEDULE, SIGNAL, and WAIT greatly simplifies the interpretation from style to code. The SCHEDULE command schedules a method supported time or associate event; SIGNAL and WAIT commands manipulate a special flag, known as a semaphore, that allows coinciding tasks to be synchronised.

Finally, options that facilitate reliable programming are necessary as a result of period of time programs are often giant and sophisticated. These options embody standard programming, powerfully enforced information typewriting, and a number of alternative management and information definition constructs

1.2.7. Task Synchronization and Communication

A multi-tasking system should furnish a mechanism for the tasks to pass data to every alternative yet on guarantee their synchronization. For these functions, in operation systems and languages with run-time support normally use queuing semaphores, mailboxes, or message systems. Semaphores offer synchronization and communication however contain no data. Messages are just like semaphores except that they carry the associated data. Mailboxes, on the other hand, don't signal data however instead contain it.

Queuing semaphores are software package primitives that facilitate manage traffic. they supply a technique of leading many queues–for example, queues of tasks watching for resources, data-base access, and devices, yet as queues of the resources and devices. The semaphores coordinate (synchronize) the waiting tasks with no matter they're watching for while not property tasks or resources interfere with one another.

In a period of time system, semaphores square measure normally accustomed implement and manage mailboxes. Mailboxes square measure temporary storage places (also referred to as a message pool or buffer) for messages sent from one method to a different. One method produces a chunk of knowledge, puts it within the mailbox, and then signals an intense method that there's a chunk of knowledge within the mailbox for it to use.

Some approaches to period of time in operation systems or run-time support systems read mailboxes because the best thanks to implement communications between processes. Some period of time in operation systems furnish an area to send and receive tips that could mailbox knowledge. This eliminates the necessity to transfer all of the data–thus saving time and overhead.

A third approach to communication and synchronization among processes may be a message system. With a message system, one method sends a message to a different. The latter is then mechanically activated by the run-time web or software to method the message. Such a system incurs overhead as a result of it transfers the particular data, however it provides larger flexibility and simple use.

2.1.8. Analysis and Simulation of Real-Time Systems

In the preceding section, we have a tendency to mention a collection of dynamic attributes that can't be unmarried from the practical necessities of a time period system:

• Interrupt handling and context switch

• Interval

• Knowledge transfer rate and outturn

• Resource allocation and priority handling

• Task synchronization and intertask communication

Each of those performance attributes will be such that, however it's very troublesome to verify whether or not system components can accomplish desired response, system resources are going to be ample to satisfy process necessities, or process algorithms can execute with ample speed.

The analysis of real time systems needs modeling and simulation that allows the system engineer to assess "timing and sizing" problems. Though variety of research techniques are planned within the literature, it's truthful to state that analytical approaches for the analysis and style of time period systems are still in their early stages of development.

2.1.9. Real-Time Design

The design of real-time software package should incorporate all of the basic ideas and principles related to prime quality software package. Additionally, real-time software package poses a collection of distinctive issues for the designer:

• Illustration of interrupts and context switching;

• Concurrency as manifested by multi-tasking and multi-processing;

• Intertask communication and synchronization;

• Wide variations in knowledge and communication rates;

• Illustration of temporal order constraints;

• Asynchronous processing;

• Necessary and unavoidable coupling with operational systems, hardware, and alternative external system parts.

Before considering a number of these issues, it's worthy to handle a collection of specialised style principles that are significantly relevant throughout the look of period of time systems. Kurki-Suonodiscusses the look model for real-time ("reactive") software:

All reasoning, whether or not formal or intuitive, is performed with some abstraction. Therefore, it's necessary to know that sorts of properties are describable within the abstraction in question. In reference to reactive systems, this can be emphasised by the lot of tight would like for formal strategies, and by the very fact that no general accord has been

reached concerning the models that ought to be used. Rigorous formalisms for reactive systems vary from method algebras and temporal logics to concrete state-based models and Petri nets, and totally different colleges keep contention concerning their relative deserves.

He then defines variety of "modeling principles" that ought to be thought-about within the style of period of time software package:

Explicit atomicity. It's necessary to outline "atomic actions" expressly as a part of the period of time style model. An atomic action or event may be a well strained and restricted operate that may be dead by one task or dead at the same time by many tasks. An atomic action is invoked solely by those tasks ("participants") that need it and also the results of its execution have an effect on solely those participants; no alternative elements of the system are affected.

Interleaving. Though process may be synchronic, the history of some computation ought to be characterised in an exceedingly manner that may be obtained by a linear sequence of actions. Beginning with an initial state, a primary action is enabled and executed. As a results of this action, the state is changed and a second action happens. As a result of many actions will occur in any given state, totally different results (histories) may be spawned from constant initial state. "This nondeterminism is important in interleaved modelling of concurrency".

Nonterminating histories and fairness. The process history of a reactive system is assumed to be infinite;by this we tend to mean that process continues indefinitely or "stutters" till some event causes it to continue process. Fairness necessities forestall a system from stopping at some discretional purpose.

Closed system principle. A style model of a period of time system ought to embrace the software package and also the surroundings within which the software package resides. "Actions will thus be partitioned off into those that the system itself is accountable, and to those who are assumed to be dead by the surroundings."

Structuring of state. A real time system may be sculpturesque as a collection of objects every of that contains a state of its own.

The coder ought to think about every of the ideas noted higher than because the style of a period of time system evolves.

Over the past 20 years, variety of period of time software package style strategies are planned to grapple with some or all of the issues noted higher than. Some approaches to period of time style extend the look strategies (e.g., knowledge flow, organization, or object-oriented methods). Others introduce a completely separate approach, exploitation finite state machine models or message passing systems, Petri nets, or a specialised language as a basis for style.

2.2. Distributed system

It's one amongst those things that is onerous to outline while not 1st shaping several different things. Here could be a "cascading" definition of a distributed system:

A program:is the code you write.

A process: is what you get once you run it.

A message: is used to communicate between processes.

A packet: is a fragment of a message that may travel on a wire.

A protocol: is a formal description of message formats and also the rules that 2 processes should follow so as to exchange those messages.

A network: is the infrastructure that links computers, workstations, terminals, servers, etc. It consists of routers that are connected by communication links.

A element: can be a method or any piece of hardware needed to run a method, support communications between processes, store data, etc.

A distributed system: is an application that executes a group of protocols to coordinate the actions of multiple processes on a network, specified together to perform one or tiny set of connected tasks.

2.2.1. Why build a distributed system?

There are variant benefits together with the flexibility to attach remote users with remote resources in an open and scalable means. After we say open, we have a tendency to mean every element is regularly receptive interaction with different parts. After we say scalable, we have a tendency to mean the system will simply be altered to accommodate changes within the variety of users, resources and computing entities.

Thus, a distributed system may be abundant larger and additional powerful given the combined capabilities of the distributed parts, than mixtures of complete systems. However it isn't simple - for a distributed system to be helpful, it should be reliable. This can be a troublesome goal to attain attributable to the quality of the interactions between at the same time running parts.

To be really reliable, a distributed system should have the subsequent characteristics:

- **Fault-Tolerant:** It will live through element failures while not performing arts incorrect actions.

- **Highly Available**: It will restore operations, allowing it to resume providing services even once some parts have failing.

- **Recoverable:** failing parts will restart themselves and re-join the system, when the reason behind failure has been repaired.

- **Consistent:** The system will coordinate actions by multiple parts typically within the presence of concurrency and failure. This underlies the flexibility of a distributed system to act sort of a non-distributed system.

- **Scalable**: It will operate properly while some facet of the system is scaled to a bigger size. For instance, we'd increase the dimensions of the network on that the system is running. This will increase the frequency of network outages and will degrade a "non-scalable" system. Similarly, we'd increase the quantity of users or servers, or overall load on the system. During a scalable system, this could not have a major impact.

- **Predictable Performance:** the flexibility to produce desired responsiveness during a timely manner.

- **Secure:** The system authenticates access to information and services

Handling failures is a vital theme in distributed systems style. Failures make up 2 obvious categories: hardware and software system. Hardware failures were a dominant concern till the late 80's, however since then internal hardware reliableness has improved tremendously. belittled heat production and power consumption of smaller circuits, reduction of off-chip connections and wiring, and high-quality producing techniques have all contend a positive role in rising hardware reliableness. Today, issues are most frequently related to connections and mechanical devices, i.e., network failures and drive failures.

Software failures are a major issue in distributed systems. Even with rigorous testing, software system bugs account for a considerable fraction of unplanned time period (estimated at 25-35%). Residual bugs in mature systems may be classified into 2 main classes.

- Heisenbug: A bug that appears to disappear or alter its characteristics once it's discovered or researched. a standard example could be a bug that happens in an exceedingly release-mode compile of a program, however not once researched beneath debug-mode. The name "heisenbug" could be a pun on the "Heisenberg indeterminacy principle," a physical science term that is often (yet inaccurately) accustomed see the approach during which observers have an effect on the measurements of the items that they're perceptive, by the act of perceptive alone (this is really the observer impact, and is often confused with the Heisenberg uncertainty principle).

- Bohrbug: A bug (named once the Bohr atom model) that, in distinction to a heisenbug, doesn't disappear or alter its characteristics once it's researched. A Bohrbug usually manifests itself faithfully beneath a well-defined set of conditions. [6]

Heisenbugs tend to be additional prevailing in distributed systems than in native systems. One reason for this can be the problem programmers have in getting a coherent and comprehensive read of the interactions of synchronic processes.

2.2.2. Distributed Design Principles

Given what we've lined to date, we will outline some basic style principles which each and every distributed system designer and applied scientist ought to recognize. a number of these could appear obvious, however it'll be useful as we tend to proceed to possess a decent beginning list.

- As Ken Arnold says: "You have to be compelled to design distributed systems with the expectation of failure." Avoid creating assumptions that any part within the system is in an exceedingly specific state. A classic error situation is for a method to send knowledge to a method running on a second machine. the method on the primary machine receives some knowledge back and processes it, so sends the results back to the second machine presumptuous it's able to receive. Any variety of things might have failing within the interim and also the causing method should anticipate these potential failures.

- Explicitly outline failure eventualities and establish however doubtless all may occur. check that your code is completely lined for the foremost doubtless ones.

- Both purchasers and servers should be able to cope with unresponsive senders/receivers.

- Think rigorously regarding what proportion knowledge you send over the network. Minimize traffic the maximum amount as potential.

- Latency is that the time between initiating letter of invitation for knowledge and also the starting of the particular knowledge transfer. Minimizing latency typically comes all the way down to an issue of whether or not you must build several very little calls/data transfers or one huge call/data transfer. The way to build this call is to experiment. Do little tests to spot the most effective compromise.

- Don't assume that knowledge sent across a network (or even sent from disk to disk in an exceedingly rack) is that the same knowledge once it arrives. If you want to make sure, do checksums or validity checks on knowledge to verify that the information has not modified.

- Caches and replication methods square measure ways for addressing state across parts. we tend to try and minimize stateful parts in distributed systems, however it's difficult. State are a few things command in one place on behalf of a method that's in another place, one thing that can't be reconstructed by the other part. If it may be reconstructed it is a cache. Caches may be useful in mitigating the risks of maintaining state across parts. however cached knowledge will become stale, therefore there might have to be a policy for validating a cached knowledge item before victimisation it.

- If a method stores info that cannot be reconstructed, then issues arise. One potential question is, "Are you currently one purpose of failure?" I even have to speak to you currently - i can not sit down with anyone else. therefore what happens if you go down? To cope with this issue, you'll be replicated. Replication methods also are helpful in mitigating the risks of maintaining state. however there square measure challenges here too: What if I sit down with one replicant and modify some knowledge, then I sit down with another? Is that modification certain to have already found out the other? What happens if the network gets divided and also the replicants cannot sit down with every other? will anybody proceed?

- There square measure a collection of tradeoffs when deciding however and wherever to keep up state, and once to use caches and replication. It's harder to run little tests in these eventualities as a result of the overhead in putting in place the various mechanisms.

- Be sensitive to hurry and performance. Take time to see that elements of your system will have a big impact on performance: wherever square measure the bottlenecks and why? Devise little tests you'll do to evaluate alternatives. Profile and measure to find out additional. sit down with your colleagues regarding these alternatives and your results, and choose on the most effective answer.

- Acks square measure pricy and have a tendency to be avoided in distributed systems where potential.

- Retransmission is costly. it is important to experiment therefore you'll tune the delay that prompts a retransmission to be best.

CHAPTER III

DATA INFORMATION AND RELATED ATTRIBUTES INFORMATION SYSTEMS

3.1. Introduction

Information systems are just one of associate nearly infinite kind of systems which can operate at intervals a company. a crucial requirement to understanding the particular role associated operation of knowledge system in a company is thus an appreciation of the importance of the term system and of the main parts of any system, regardless of whether or not the system relates to info or another resource within the organization. (Information system in business, By- Bob Ritchie, David Marshall and Alan Eardlye).

A system collects, processes, store, analyses and disseminates info for a selected purpose. Like several alternative system, associate system includes inputs. It processes the inputs and produces outputs that square measure sent to the user or the opposite system. A feedback mechanism that controls the operation could also be enclosed. Like several alternative system, associate system operates at intervals associate setting.

3.1.1. Role of Information System in an Organisation

Information system and organizations influence each other. Data systems are designed by managers to serve the interest of the business. At identical time the corporate should remember of and open to the influences of data systems to learn from new technologies. The interaction between data technology and organization is advanced and is influenced by several mediating factors, as well as the organizations structure, business method, politics, culture, encompassing surroundings, and management selections.

- Organization
- Information
- Technology
- Mediating issue
- Environment
- Culture
- Structure
- Business process
- Politics
- Management decision

Information system became integral, online, interactive tools deeply concerned within the minute-to-minute operations and deciding of huge organizations. Over the last decade, data

system have basically altered the social science of organizations and greatly enhanced the probabilities for organizing work.

3.1.2. Level in an Organisation

Senior management makes long-range strategic choices regarding product and services further as ensures monetary performance of the firm.

Middle management carries out the programs and plans of senior management and operational management is answerable for observance the daily activities of the business.

Knowledge employees like engineers, scientists or architects, style product or services and make new information for the firm. Whereas knowledge employees like secretaries or clerks, assist with work in the least levels of the firm.

Production and repair employees really turn out the merchandise and deliver the service.

Middle Management

Scientists and information

Worker

Senior management

Operational Management

Production and service workers,

Data employees

3.1.3. System Types and Characteristics

Organizational data systems may be handily being placed into the subsequent categories:-

1. Transaction Processing System (TPS) or Data Processing System
2. Management Information System (MIS)
3. Executive Information System (EIS)
4. Decision Support System (DSS)

Transaction Processing System (TPS)

In the starting, computers were all rather restricted, extremely specialised machines that needed an obsessive atmosphere and specialist personnel to create them operate properly. Such machines were known as 'mainframes' and also the organization required a computer department to supply the mandatory processing service to supply reports. The system that these computer installations supported was known as {data process|processing} (DP) or dealing processing system (TPS). Some example of TPS systems in typical organizations are:-

1. A payroll system
2. A stock system
3. An order entry system

Management Information System (MIS)

If the primary part of business computing was to modify the manual and clerical processes of business with the aim of accelerating potency, the second part was to stress the role of data.

Early management information systems sought-after to use the output from existing process/processing system (and the info processing department) in some type that created it additional appropriate for middle management to know.

Executive Information System (EIS)

In spite of the success of MISs at the military science level of management, it became apparent that IT had not created any important edges for commanding executives. The strategic decision-making processes (e.g. new development, changes in market position, growth by acquisition) were typically perceptibly sick aware. Several chief executives within the late Eighties had not used any kind of automatic data processing system the least bit. Owing to this, it absolutely was apparent that new developments like windows and up to date developments in human computer interaction (HCI) like buttons and bit sensitive screen would kind the basis of EISs.

Decision Support System (DSS)

Decision support systems were developing to beat the rigidity of MIS-type coverage structures and also the limitations of spreadsheets. Additionally there have been advances in company info technology and EIS-type interfaces on that to draw. DSS support no routine deciding for middle management. They specialize in issues that square measure distinctive and rapidly ever-changing, that the procedure for inbound at an answer might not be absolutely predefined in advanced.

3.1.4. Bright Aspects of information System

MIS facilitates planning: – MIS improves the standard of plants by providing relevant info for sound decision-making. Thanks to increase within the size and complexness of organization, managers have lost personal contact with the sense of operation.

It minimizes info overload: – MIS amendment the larger quantity of knowledge into summarized type and there by avoids the confusion which can arise once managersare flooded with elaborate facts.

It encourages decentralization: – Decentralization of authority is presumably once there's a system for watching operation at lower levels. MIS is with success used for mensuration performance and creating necessary amendment within the structure plans and procedures.

It brings co-ordination:- MIS facilities integration of specialised activities by keeping every department alert to the matter and needs of alternative department. It connects all call centers within the organization.

3.1.5. Dark Aspects of information System

Hacker activities have broadened on the far side mere system intrusion to incorporate stealing of products and data, yet as system harm and cyber mischief, the intentional disruption, defacement, or perhaps destruction of a web-site or company system.

No one is aware of the magnitude of the computer crime problem- what percentage systems are invaded, what percentage folks have interaction within the follow, or the full economic harm. in keeping with the 2007 CSI, computer Crime and Security Survey of nearly five hundred corporations, participant's average annual loss from computer crime and security attacks was $350,420 (Richardson, 2007)..

System malfunction if element breaks down, isn't designed properly, or is broken by improper use or criminal acts. Error in programming, improper installation, or unauthorized changes cause laptop package to fail. Equipment failure, floods, fires, or alternative natural disaster may disrupt automatic data processing system.

Vulnerability has additionally exaggerated from wide unfold use of e-mail, instant messaging (IM), and peer-to-peer file sharing programs. E-mail might contain attachments that function springboards for malicious package or unauthorized access to internal company systems.

3.2. System Analysis and Analyst Decision Making Process

3.2.1. Software Analysis & Design Tools

Software analysis and design includes all activities that facilitate the transformation of requirement specification into implementation. Demand specifications specify all useful and non-functional expectations from the code. These demand specifications are available the form of human clear and graspable documents, to that a computer has nothing to try and do.

Software analysis and style is that the intermediate stage, that helps human-readable necessities to be reworked into actual code.

Let us see few analysis and style tools employed by code designers:

3.2.1.1. Data flow Diagram (DFD)

Data flowchart is graphical illustration of flow of information in the system. It's capable of portraying incoming information flow, outgoing information flow and hold on information. The DFD doesn't mention something concerning however information flows through the system.

There is a distinguished distinction between DFD and flow sheet. The flow sheet depicts flow of management in program modules. DFDs depict flow of information within the system at varied levels. DFD doesn't contain any management or branch components.

3.2.1.1.1. Types of DFD

1. **Logical DFD -** this kind of DFD concentrates on the system method, and flow of information within the system.For example in an exceedingly banking code, however information is emotional between completely different entities.
2. **Physical DFD -** this kind of DFD shows however the information flow is really enforced within the system. It's additional specific and shut to the implementation.

3.2.1.1.2. DFD elements

DFD will represent supply, destination, storage and flow of information victimization the subsequent set of elements shown in Figure 6

Figure 6: DFD elements

- **Entities** - Entities are source and destination of information data. Entities are delineated by a rectangle with their respective names.

- **Process** - Activities and action taken on the information are delineated by Circle or Round-edged rectangles.

- **Data Storage** - There are 2 variants of information storage - it will either be delineated as a parallelogram with absence of each smaller side or as an open-sided parallelogram with only 1 side missing.

- **Data Flow** - Movement of information is shown by pointed arrows. Information movement is shown from the bottom of arrow as its supply towards head of the arrow as destination.

3.2.1.1.3. Levels of DFD

- **Level 0 -** Highest abstraction level DFD is thought as Level 0 DFD, that depicts the complete system mutually diagram concealing all the underlying details. Level 0 DFDs are called context level DFDs shown in Figure 7.

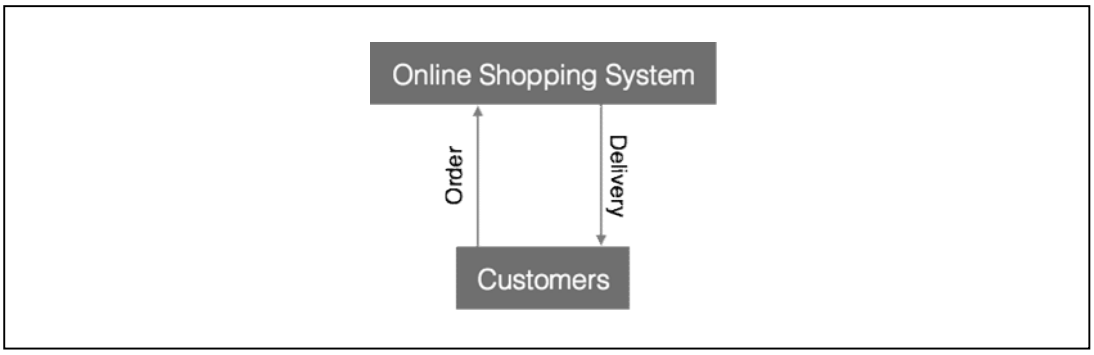

Figure 7: Level 0 DFD

- **Level 1** - the extent zero DFD is broken down into additional specific, Level one DFD. Level one DFD depicts basic modules within the system and flow of information among varied modules. Level one DFD conjointly mentions basic processes and sources of data shown in Figure 8.

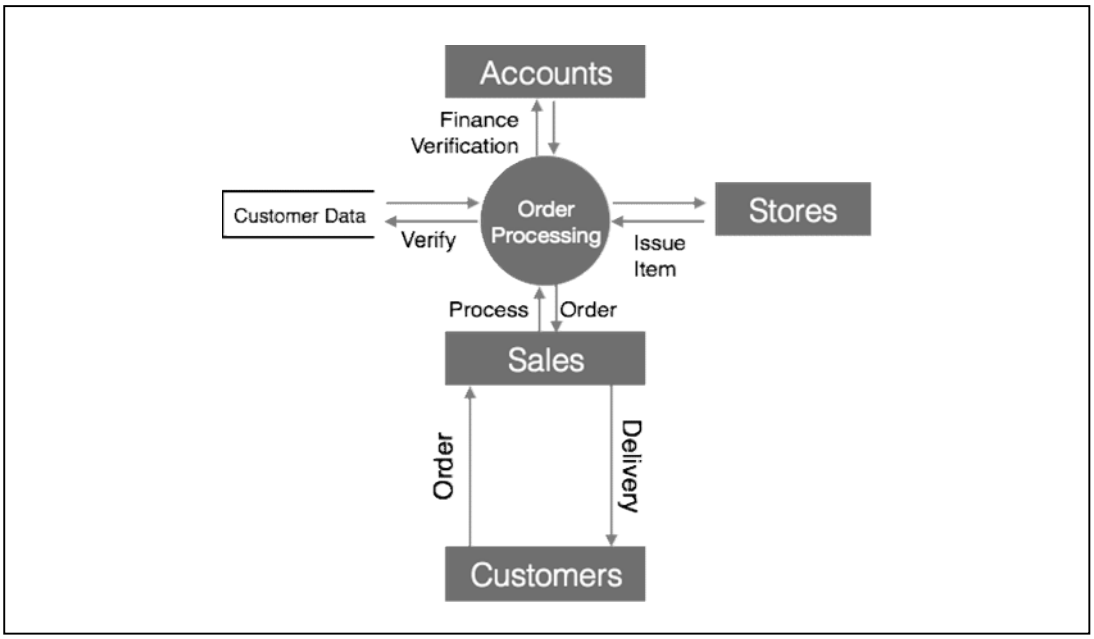

Figure 8: Level 1 DFD

- **Level 2** - At this level, DFD shows however information flows within the modules mentioned in Level 1.

Higher level DFDs is reworked into additional specific lower level DFDs with deeper level of understanding unless the required level of specification is achieved.

3.2.1.2. Structure Charts

Structure chart is a chart derived from data flow diagram. It represents the system in additional detail than DFD. It breaks down the complete system into lowest functional modules, describes functions and sub-functions of every module of the system to a bigger detail than DFD.

Structure chart represents data structure of modules. At every layer a particular task is performed.

Here are the symbols employed in construction of structure charts -

- Module - It represents method or subprogram or task. a control module branches to quite one sub-module. Library Modules are re-usable and invokable from any module shown in Figure 9.

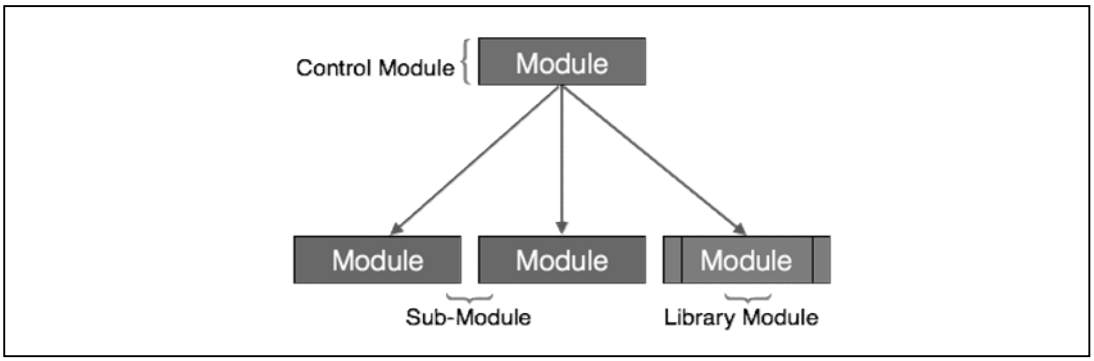

Figure 9: Module

• Condition - it's described by tiny diamond at the bottom of module. It depicts that management module will choose any of sub-routine supported some condition shown in Figure 10.

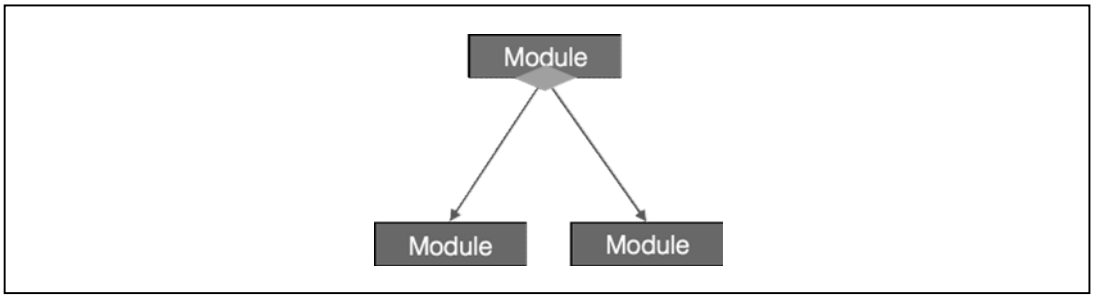

Figure 10: Condition

• Jump - an arrow is shown inform within the module to depict that the management can jump within the middle of the sub-module shown in Figure 11.

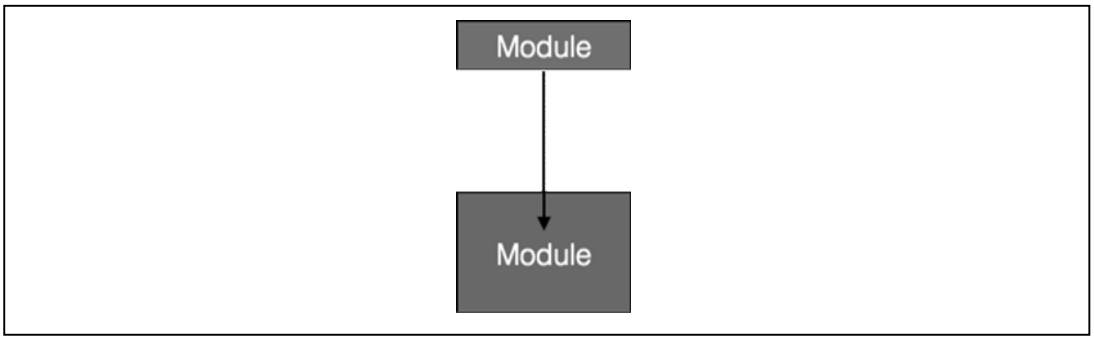

Figure 11: Jump

• Loop - A curved arrow represents loop within the module. All sub-modules lined by loop repeat execution of module shown in Figure 12.

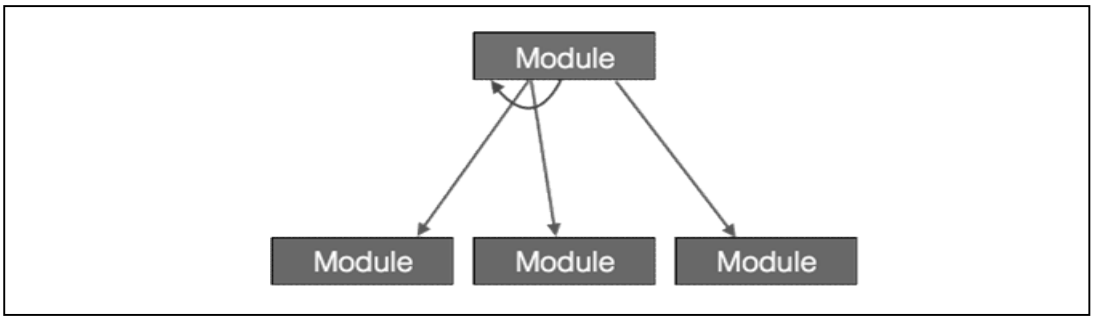

Figure 12: Loop.

• Data flow - A directed arrow with empty circle at the end represents data flow shown in Figure 13.

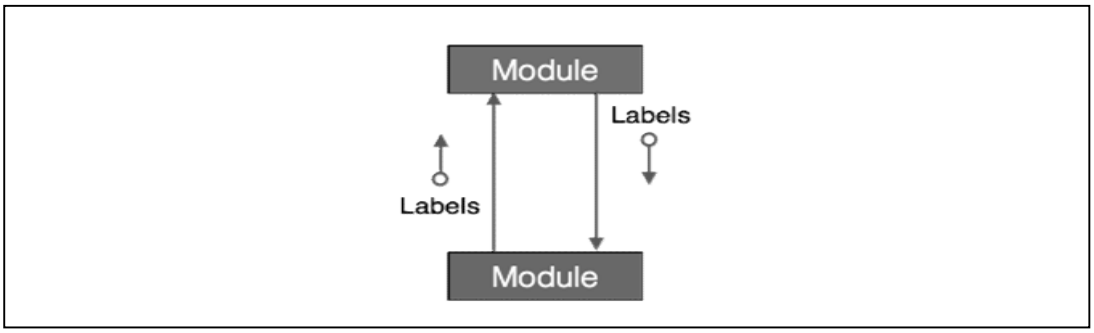

Figure 13: Data Flow

• Control flow - A directed arrow with crammed circle at the tip represents control flow shown in Figure 14.

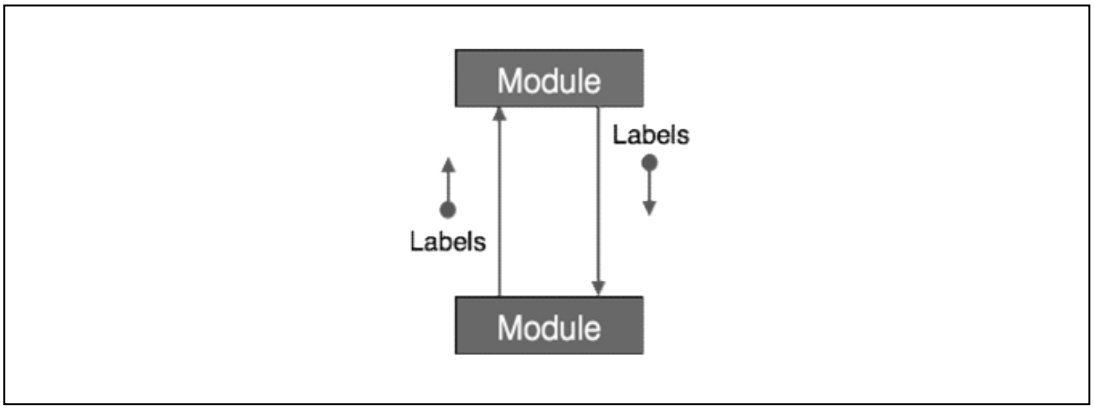

Figure 14: Control Flow.

3.2.1.3. HIPO Diagram

HIPO (Hierarchical Input process Output) diagram could be a combination of 2 organized technique to research the system and supply the means that of documentation. HIPO model was developed by IBM in year 1970.

HIPO diagram represents the hierarchy of modules in the software system. Analyst uses HIPO diagram so as to get high-level read of system functions. It decomposes functions into sub-functions in a very graded manner. It depicts the functions performed by system.

HIPO diagrams square measure sensible for documentation purpose. Their graphical illustration makes it easier for designers and managers to urge the pictorial plan of the system structure shown in Figure 21.

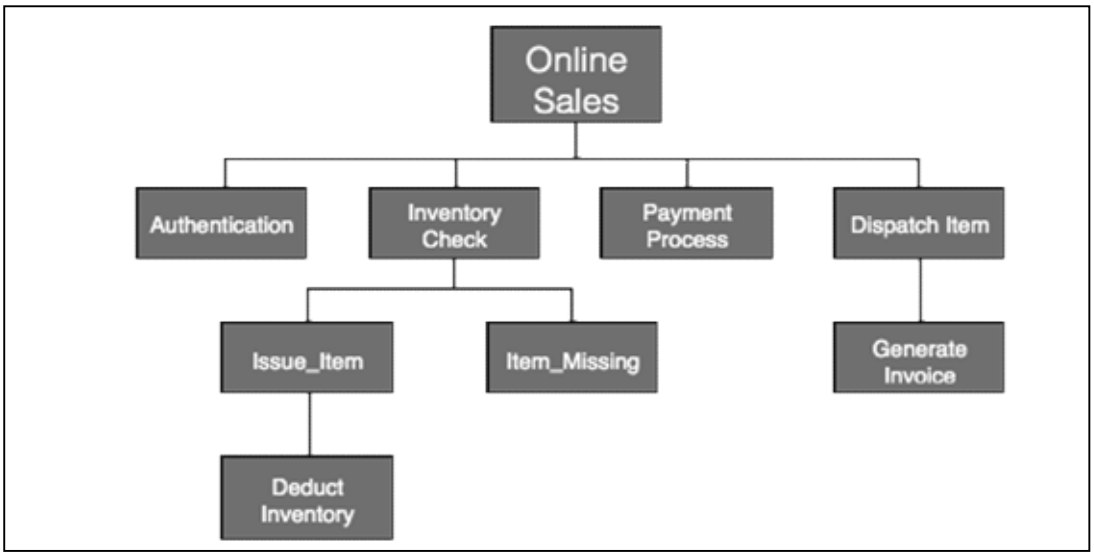

Figure 15: HIPO

In distinction to IPO (Input process Output) diagram shown in Figure 16, that depicts the flow of control and information in a very module, HIPO doesn't offer any data concerning information flow or management flow.

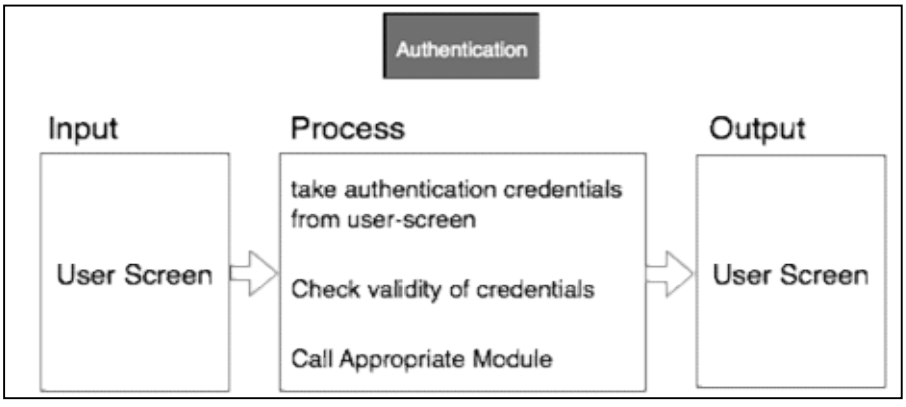

Figure 16: IPO

Example

Both components of HIPO diagram, graded presentation and IPO Chart are used for structure design of software package program yet as documentation of the same.

3.2.1.4. Structured English

Most programmers are unaware of the big image of software package so that they solely have faith in what their managers tell them to try and do. It's the responsibility of upper software package management to produce correct forms of to the programmers to develop correct nonetheless quick code.

Other forms of ways that use graphs or diagrams, May are sometimes taken otherwise by completely different individuals.

Hence, analysts and designers of the software package return up with tools like Structured English. It's nothing however the outline of what's needed to code and the way to code it. Structured English helps the technologist to put in writing error-free code.

Other styles of ways, that use graphs or diagrams, might square measure typically taken otherwise by completely different individuals. Here, each Structured English and Pseudo-Code tries to mitigate that understanding gap.

Structured English is uses plain English words in structured programming paradigm. It's not the final word code however a form of description what's needed to code and the way to code it. The subsequent square measure some tokens of structured programming.

3.2.1.5. Decision Tables

A Decision table represents conditions and therefore the several actions to be taken to deal with them, in an exceedingly structured tabular format.

It is a robust tool to rectify and stop errors. It helps cluster similar information into one table so by combining tables it delivers straightforward and convenient decision-making.

Creating decision table

To create the decision table, the developer should follow basic four steps:

- Identify all attainable conditions to be self-addressed
- Determine actions for all known conditions
- Create most attainable rules
- Define action for every rule

Decision Tables ought to be verified by end-users and may latterly be simplified by eliminating duplicate rules and actions.

Example

Let us take an easy example of every day problem with our internet connectivity. We start by distinguishing all issues that may arise whereas beginning the net and their several attainable solutions.

We list all attainable issues below column conditions and therefore the prospective actions below column Actions.

	Conditions/Actions	Rules							
Conditions	Shows Connected	N	N	N	N	Y	Y	Y	Y
	Ping is Working	N	N	Y	Y	N	N	Y	Y
	Opens Website	Y	N	Y	N	Y	N	Y	N
Actions	Check network cable	X							
	Check internet router	X				X	X	X	
	Restart Web Browser							X	
	Contact Service provider		X	X	X	X	X	X	
	Do no action								

Table: Decision Table – In-house Internet Troubleshooting

3.2.1.6. Entity-Relationship Model

Entity-Relationship model could be a style of information model supported the notion of globe entities and relationship among them. We are able to map globe state of affairs onto ER information model. ER Model creates a group of entities with their attributes, a group of constraints and relation among them.

ER Model is best used for the abstract style of information. ER Model may be drawn as follows shown in Figure17:

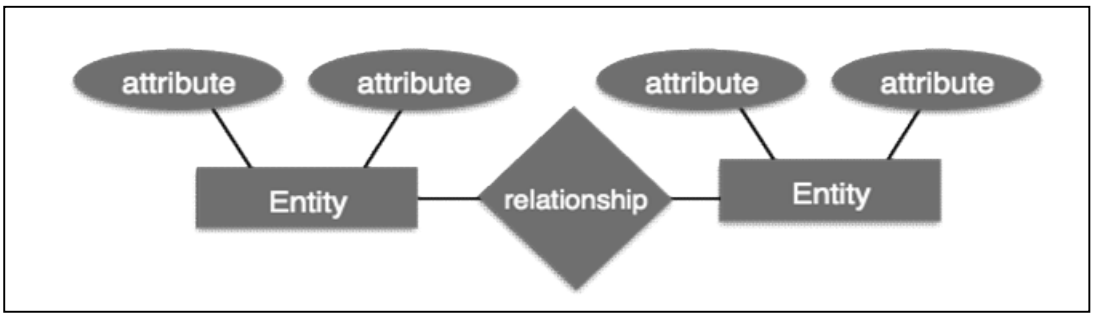

Figure 17:Entity-Relationship model

- **Entity** - an entity in ER Model may be a planet being, that has some properties referred to as attributes. Each attribute is outlined by its corresponding set of values, referred to as domain.

For example, think about faculty information. Here, a student is an entity. Student has numerous attributes like name, id, age and class etc.

- **Relationship** - The logical association among entities is termed relationship. Relationships are mapped with entities in numerous ways in which. Mapping cardinalities outline the amount of associations between 2 entities.

Mapping cardinalities:

- one to one
- one to many
- many to one
- many to many

Data Dictionary

Data dictionary is that the centralized assortment information regarding data. It stores that means and origin of knowledge, its relationship with alternative knowledge, data formatting for usage etc. data dictionary has rigorous definitions of all names so as to facilitate user and software package designers.

Data dictionary is commonly documented as meta-data (data regarding data) repository. It's created together with DFD (Data Flow Diagram) model of software program and is expected to be updated whenever DFD is modified or updated.

Requirement of data dictionary

The data is referenced via data dictionary whereas coming up with and implementing software. Data dictionary removes any probabilities of ambiguity. It helps keeping work of programmers and designers synchronised whereas mistreatment same object reference everyplace within the program.

Data dictionary provides how of documentation for the whole information system in one place. Validation of DFD is disbursed mistreatment data dictionary.

Contents

Data dictionary ought to contain info regarding the subsequent

- Data Flow
- Data Structure
- Data components

- Data Stores
- Data process

Data Flow is represented by suggests that of DFDs as studied earlier and delineate in algebraic type as represented.

= Composed of

{} Repetition

() Optional

+ And

[/] Or

Example

Address = House No + (Street / Area) + town + State

Course ID = Course number + Course Name + Course Level + Course Grades

Data components

Data components comprises Name and descriptions of knowledge and management things, Internal or External knowledge stores etc. with the subsequent details:

- Primary Name
- Secondary Name (Alias)
- Use-case (How and wherever to use)
- Content Description (Notation etc.)
- Supplementary information (present values, constraints etc.)

Data Store

It stores the information from wherever the data enters into the system and exists out of the system. The info Store might include -

- **Files**
  - Internal to software.
  - External to software however on the same machine.
  - External to software and system, situated on completely different machine.
- **Tables**

- Naming convention
- Indexing property

Data process

There are two types of dataprocessing:

- Logical: As user sees it
- Physical: As software sees it

3.3. System Development Life Cycle (SDLC)

An effective System Development Life Cycle (SDLC) ought to lead to a prime quality system that meets client expectations, reaches completion inside time and value evaluations, and works effectively and expeditiously within the current and planned data Technology infrastructure.

System Development Life Cycle (SDLC) may be a abstract model which incorporates policies and procedures for developing or altering systems throughout their life cycles.

SDLC is employed by analysts to develop a data system. SDLC includes the subsequent the following

- Requirements
- Design
- Implementation
- Testing
- Deployment
- Operations
- Maintenance

3.3.1. Phases of SDLC

System Development Life Cycle may be a systematic approach that expressly breaks down the work into phases that square measure needed to implement either new or changed data system shown in Figure 18.

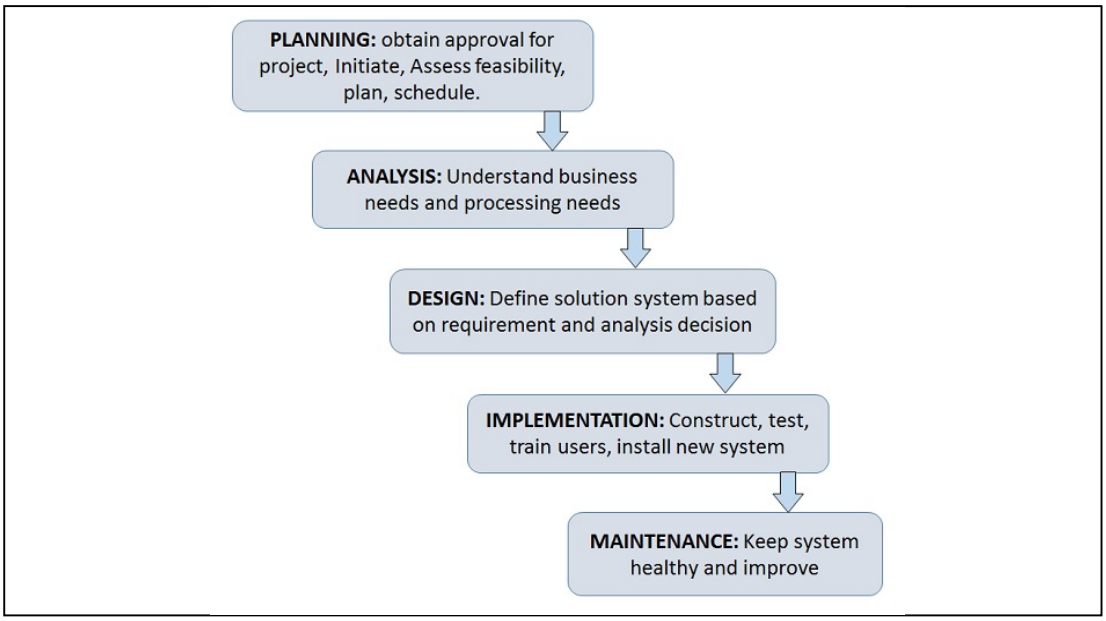

Figure 18:Phases of SDLC

3.3.2. Feasibility Study or designing

- Define the matter and scope of existing system.
- Overview the new system and confirm its objectives.
- Confirm project practicableness and manufacture the project Schedule.
- During this section, threats, constraints, integration and security of system are thought of.
- A feasibility report for the whole project is formed at the tip of this section.

3.3.3. Analysis and Specification

- Gather, analyze, and validate the data.
- Define the necessities and prototypes for brand spanking new system.
- Evaluate the alternatives and order the necessities.
- Examine the data desires of end-user and enhances the system goal.
- A computer code demand Specification (SRS) document, that specifies the computer code, hardware, functional, and network necessities of the system is ready at the tip of this section.

3.3.4. System design

- Includes the planning of application, network, databases, user interfaces, and system interfaces.

- Transform the SRS document into logical structure, that contains elaborate and complete set of specifications which will be enforced during a artificial language.
- Create a contingency, training, maintenance, and operation set up.
- Review the projected style. Make sure that the ultimate style should meet the necessities expressed in SRS document.
- Finally, prepare a style document which can be used throughout next phases.

Implementation

- Implement the planning into ASCII text file through writing.
- Combine all the modules along into coaching surroundings that detects errors and defects.
- A take a look at report that contains errors is ready through take a look at set up that has taken a look at connected tasks like legal action generation, testing criteria, and resource allocation for testing.
- Integrate the data system into its surroundings and install the new system.

3.3.5. Maintenance/Support

- Include all the activities like phone support or physical on-site support for users that's needed once the system is putting in.
- Implement the changes that computer code would possibly endure over a amount of your time, or implement any new necessities once the computer code is deployed at the client location.
- It conjointly includes handling the residual errors and resolves any problems which will exist within the system even once the testing section.
- Maintenance and support is also required for a extended time for giant systems and for a brief time for smaller systems.

3.3.6. Life Cycle of System Analysis and Design

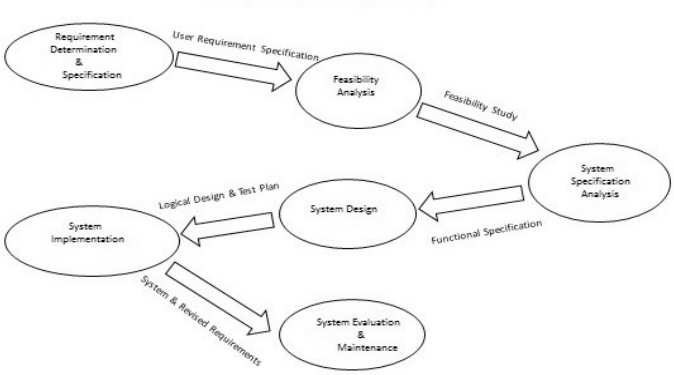

Figure 19 shows the entire life cycle of the system throughout analysis and design section.

3.3.7. Role of System Analyst

The system analyst may be a one who is completely conscious of the system and guides the system development project by giving correct directions. He's an expert having technical and social skills to hold out development tasks needed at every part.

He pursues to match the objectives of data system with the organization goal.

Main Roles

- Defining and understanding the need of user through varied reality finding techniques.
- Prioritizing the wants by getting user consensus.
- Gathering the facts or data and acquires the opinions of users.
- Maintains analysis and analysis to attain acceptable system that is additional user friendly.
- Suggests several versatile different solutions, choose the most effective answer, and quantify value and advantages.
- Draw bound specifications that are simply understood by users and computer programmer in precise and elaborated type.
- Implemented the logical design of system that should be standard.
- Plan the regularity for analysis when it's been used for a few times, and modify the system as needed.

REFERENCES

Recommended text Books:

1. System Analysis & Design – Elias M. Awad, Galgotia Publication
2. System Analysis and Design, Capron, Tom, Addison Wesley.

Website references:

1. www.mhhe.com
2. www.tutorialspoint.com
3. vinhphuc121583.wordpress.com
4. www.scribd.com
5. www.systemanalysisanddesigns.com
6. www.auessays.com
7. www.ynb.no
8. www.w3schools.in
9. www.coursehero.com
10. www.kscst.iisc.ernet.in
11. pt.scribd.com

MCQ FOR PRACTICE

1. _____ system consists of programs, data files and documentation
   A. Conceptual
   B. Logical
   C. Physical
   D. None of the above
2. System study involves
   A. Study of an existing system
   B. Documenting the existing system
   C. Identifying current deficiencies and establishing new goals
   D. All the above
3. The main ingredient of the report documenting the____ is the cost benefit analysis.
   A. System Analysis
   B. Feasibility Study
   C. System Analyst
   D. System Design
4. The organized process or set of steps that needs to be followed to develop an information system is known as
   A. analytical cycle
   B. design cycle
   C. program specification
   D. system development life cycle
5. A data flow can
   A. Only emanate from an external entity
   B. Only terminate in an external entity
   C. May emanate and terminate in an external entity
   D. May either emanate or terminate in an external entity but not both

6. What is the real time system?
   A. Used for monitoring events as they occur
   B. Primarily used on mainframe computers
   C. Used for real time interactive users
   D. Used for program development
7. In real time operating system is _____
   A. Kernel is not required
   B. Process scheduling can be done only once task
   C. Must be serviced but its deadline period
   D. All processes have the same priority
8. Statement of scope and objectives, opportunities and performance criteria …….
   A. Problem definition

B. System analysis
C. System Design
D. Documentation

9. The output of problem definition stage is ……..
   A. Master Development Plan
   B. Terms of reference
   C. Feasibility report
   D. Final product

10. A context diagram is used
   A. as the first step in developing a detailed DFD of a system
   B. in systems analysis of very complex systems
   C. as an aid to system design
   D. as an aid to the programmer

11. Advantages of system flowcharts …….
   A. Effective communication
   B. Effective analysis
   C. Queasier group or relationships
   D. All A, B, C

12. The key considerations involved in the feasibility analysis is include
   i) Economical ii) Technical iii) Behavioral iv) Personal
   A. i, ii, iv
   B. i, ii, iii
   C. ii, iii, iv
   D. All of the above

13. …….. Phase is a time-consuming phase and yet a very crucial phase
   A. Feasibility Study
   B. Requirement Phase
   C. Analysis Phase
   D. Testing Phase

14. Which of the following is not a fact-finding technique?
   A. Third party enquiry
   B. Interview
   C. Questionnaire
   D. Record reviews

15. System study involves

   A. Study of an existing system
   B. Documenting the existing system
   C. Identifying current deficiencies and establishing new goals

D. All the above

16. In ER diagrams, the rectangles are used to denote

A. entity types
B. attribute types
C. key types
D. structure types

17. In constructing ER diagrams, the double ovals are used to denote

A. multi-value table
B. multi-value entity
C. multi-value attributes
D. multi-value key

18. All the resources are shared and integrated within one OS, in the computing paradigm named

A. Distributed computing
B. Parallel computing
C. Cloud computing
D. Centralized computing

19. In a distributed system, information is exchanged through

A. Memory sharing
B. Memory sharing
C. Message passing
D. Exceptions

20. External Entities may be a

A. Source of input data only
B. Source of input data or destination of results
C. Destination of results only
D. Repository of data

21. By an external entity, we mean a

A. The unit outside the system being designed which can be controlled by an analyst.
B. The unit outside the system whose behavior is independent of the system is designed
C. A unit external to the system is designed
D. A unit which is not part of a DFD

22. A context diagram is used

A. as the first step in developing a detailed DFD of a system
B. in systems analysis of very complex systems

C. as an aid to system design
D. as an aid to the programmer

23. Changing the relationship with and services provided to customers in such a way that they will not think of changing suppliers is called …….
A. Lock in customers
B. Lockout customers
C. Lock in competitors
D. Lockout competitors

24. The primary tool used in structured design is a:

A. structure chart

B. data-flow diagram

C. program flowchart

D. module

25. In a _____ one module of the new information system is activates at a time.

A. System Development Life Cycle

B. CASE tool

C. Phased Conversion

D. Success factors

26. The step-by-step instructions that solve a problem are called _____.

A. An algorithm

B. A list

C. A plan

D. A sequential structure

27. The approach used in top-down analysis and design is

A. to identify the top level functions by combining many smaller components into a single entity

B. to prepare flow charts after programming has been completed

C. to identify a top level function an d then create a hierarchy of lower-level modules and components.

D. All of the above

28. System Study involves

A. study of an existing system

B. documenting the existing system.

C. identifying current deficiencies and establishing new goals

D. All of the above

29. Documentation is prepared

A. at every stage

B. at system design

C. at system analysis

D. at system development

30. System Implementation Phase entails

A. System check outs

B. Pilot run

C. Parallel runs

D. All of the above

QUESTIONS FOR PRACTICE

1. What are the different Types of Systems?
2. What is the role of system analysis?
3. What are the types of software design?
4. What is the role of system analysis?
5. What are information and its importance?
6. What are the principles of literature review?
7. How real time database works?
8. Explain the process of literature review.
9. What is the role of system analysis and design?
10. Benefits of Object-Oriented Systems Analysis and Design
11. Describe the stages of software life cycle.
12. Explain how system and program test preparation?
13. Describe operational feasibility.
14. What do you mean by QUESTIONNAIRES**?**
15. What is the purpose of an entity relationship diagram?
16. What are the types of distributed systems?
17. What is the relationship between classes and objects?
18. What is system? Explain characteristics and types of system.
19. What is the drawback of DFD? Draw a DFD for the university admission system.
20. Explain various types of feasibility studies that the analyst should consider.
21. What do you understand by system testing? Explain different types of testing.
22. What are the different methodologies of system design? What do you understand by module coupling and cohesion?
23. Draw a labelled DFD of a Library Automation System.
24. Illustrate the concepts of logical and physical design with a suitable example.
25. Explain various types of feasibility studies that the analyst should consider.
26. . Which basic qualities must system analysts have to perform its basic roles?
27. Explain needs, roles and qualities of system analyst.
28. Which basic qualities must system analysts have to perform its basic roles?
29. What are the requirements to preparing for the JAD Session?
30. Benefits of Using JAD Model

Contact Us:

University Campus Address:

Jayoti Vidyapeeth Women's University

Vadaant Gyan Valley, Village-Jharna, Mahala Jobner Link Road,

Jaipur Ajmer Express Way, NH-8, Jaipur- 303122, Rajasthan (INDIA)

(Only Speed Post is Received at University Campus Address, No. any Courier Facility is available at Campus Address)

Pages	: 47
Book Price	: ₹ 150/-

JPD Women University Press

Jayoti Publication Desk