Unit 1 - Overview of Systems Analysis and Design

1. Overview of Systems Analysis and Design **4 Hrs.**

1.1 Introduction to System Analysis and Design

1.2 Types of Information Systems and System Development

1.3 Developing Information Systems and Systems Development Life cycle

1.4 Systems Analysis and Design Tools

System and System Characteristics

A collection of components that work together to realize some objectives forms a ***system***. In a system the different components are connected with each other and they are interdependent. For example, human body represents a complete natural system. We are also bound by many national systems such as political system, economic system, educational system and so forth. The objective of the system demands that some output is produced as a result of processing the suitable inputs. A well-designed system also includes an additional element referred to as 'control' that provides a feedback to achieve desired objectives of the system.

Basically there are three major components in every system, namely input, processing and output.

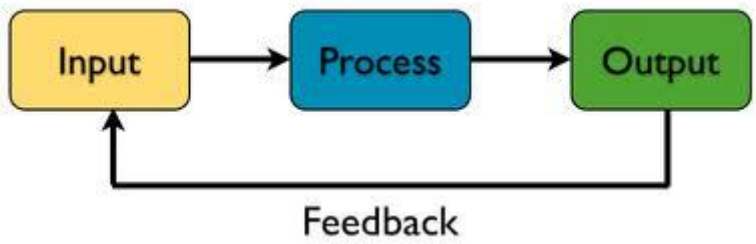


Figure: System and its components

Input: It involves capturing and assembling elements that enter the system to be processed. Inputs to the system are anything to be captured by the system from its environment. For example, raw materials.

Processing: It involves transformation processes that convert input to output. For example, a manufacturing process.

Output: It involves transferring elements that have been produced by a transformation process to their ultimate destinations. Outputs are the things produced by the system and sent into its environment. For example, finished products.

The system also includes other two additional activities. These activities include feedback and control.

Feedback: It is data about the performance of a system. It is the idea of monitoring the current system output and comparing it to the system goal. Any variation from the goal are then fed back in to the system and used to adjust it to ensure that it meets its goal. For example, data about sales performance is feedback to a sales manager.

Control: It involves monitoring and evaluating feedback to determine whether a system is moving toward the achievement of its goals. The control function then makes necessary adjustments to a system's input and processing components to ensure that it produces proper output. For example, a sales manager exercises control when reassigning salespersons to new sales territories after evaluating feedback about their sales performance.

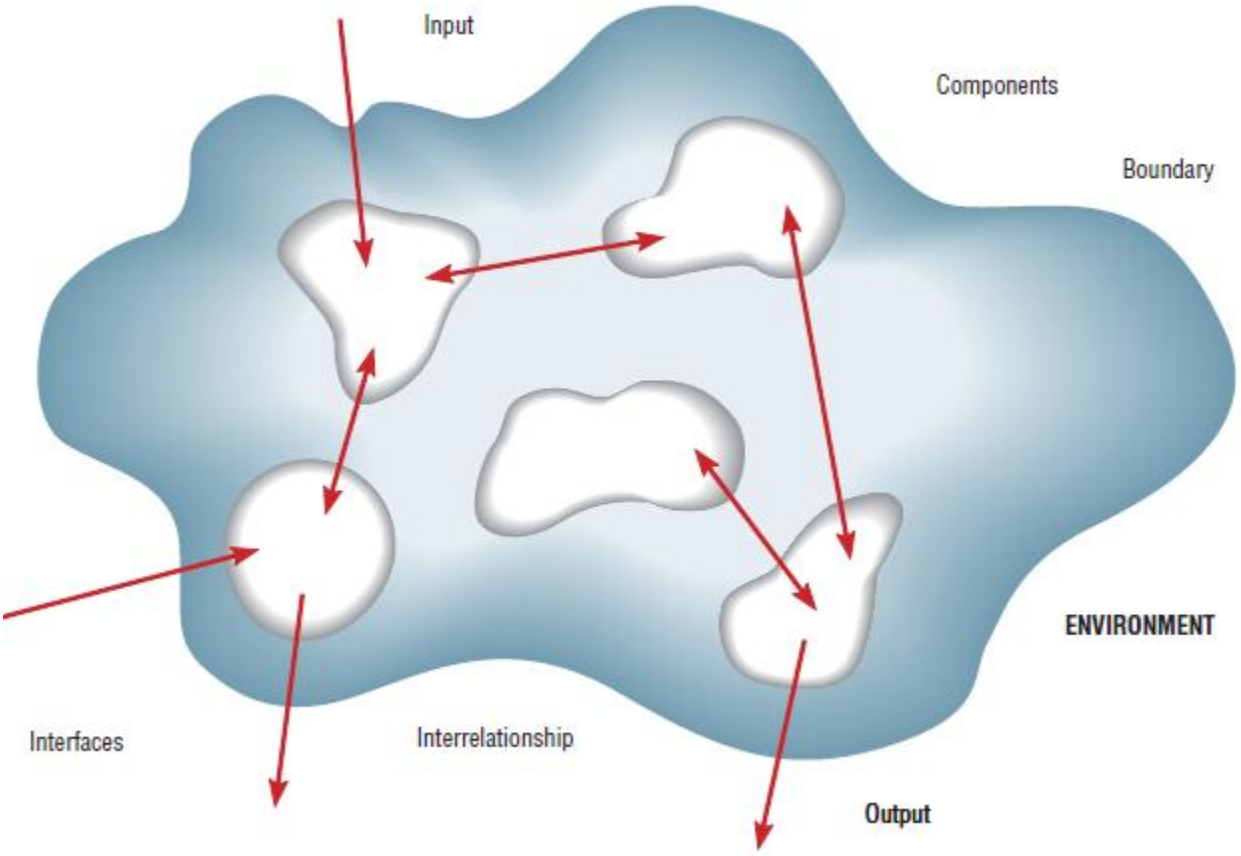


Figure: Characteristics of a System

The system has nine characteristics as shown in the above figure:
1. Components
2. Interrelated components
3. Boundary
4. Purpose
5. Environment
6. Interfaces
7. Constraints
8. Input
9. Output

b i i z a y @ g m a i l . c o m

A system is made up of ***components***. A component is either an irreducible part or an aggregate of parts, also called a subsystem. The components are ***interrelated***; that is, the function of one is somehow tied to the functions of the others. A system has a ***boundary***, within which all of its components are contained and which establishes the limits of a system, separating it from other systems. Components within the ***boundary*** can be changed, whereas systems outside the boundary cannot be changed. All of the components work together to achieve some overall ***purpose*** for the larger system: the system's reason for existing.

A system exists within an ***environment***—everything outside the system's boundary that influences the system. An information system interacts with its environment by receiving data (raw facts) and information (data processed in a useful format). The points at which the system meets its environment are called ***interfaces***; an interface also occurs between subsystems. In its functioning, a system must face ***constraints***—the limits (in terms of capacity, speed, or capabilities) to what it can do and how it can achieve its purpose within its environment. The system takes ***input*** from outside, processes it, and sends the resulting ***output*** back to its environment.

Introduction to System Analysis and Design

Computers are fast becoming our way of life and one cannot imagine life without computers in today's world. You go to an airport for airplane reservation, you want to web site a ticket for a cinema, you go to a library, or you go to a bank, you will find computers at all places. Since computers are used in every possible field today, it becomes an important issue to understand and build these computerized systems in an effective way. Building such systems is not an easy process but requires certain skills and capabilities to understand and follow a systematic procedure towards making of any information system

The ***System Analysis and Design*** (SAD) is the process of developing Information Systems (IS) that effectively use hardware, software, data, processes, and people to support the company's business objectives.

Classification of Systems:

A *system* is a combination of resources working together to transform inputs into usable outputs. An *information system (IS)* is an arrangement of people, data, processes, interfaces, networks, and technology that interact to support and improve both day-to-day operations (data processing, transaction processing), as well as support the problem-solving and decision-making needs of management (information services, management information systems, executive support). Some of the systems are classified as:

(1) Physical or Abstract System

Physical Systems are tangible entities that we can feel and touch. These may be static or dynamic in nature. For example, take a computer center. Desks and chairs are the static parts, which assist in the working of the center. Static parts don't change. The dynamic systems are constantly changing. Computer systems are dynamic system. Programs, data, and applications can change according to the user's needs.

Abstract Systems are conceptual. These are not physical entities. They may be formulas, representation or model of a real system.

(2) Open or Closed System

Systems interact with their environment to achieve their targets. Things that are not part of the system are environmental elements for the system. Depending upon the interaction with the environment, systems can be divided into two categories, open and closed.

Open Systems*:* They are the systems that interact with their environment. Practically most of the systems are open systems. An open system has many interfaces with its environment. It can also adapt to changing environmental conditions. It can receive inputs from, and delivers output to the outside of system. An information system is an example of this category.

Closed Systems*:* They are the systems that don't interact with their environment. Closed systems exist in concept only. Example: Chemicals in a sealed test tube

(3) Man-made Information System

The main purpose of information systems is to manage data for a particular organization. Maintaining files, producing information and reports are few functions. An information system produces customized information depending upon the needs of the organization. These are usually formal, informal, and computer based.

Formal Information Systems*:* It deals with the flow of information from top management to lower management. It is concerned with a pattern of authority, communication & work flow. Information is formally given in terms of instructions, memos or reports from top management to the intended user of the organization. This also includes policies, feedback & regulations. The output represents employee performance.

There are three categories of information system related to managerial level & decision managers make: Strategic information, Managerial information and Operational Information.

Informal Information Systems*:* It is an employee based system designed to meet personnel & vocational needs & to help solve work-related problems. The analyst should have knowledge of the chain of command, the power authority influence network.

Computer-Based Information Systems (CBIS)*:* This class of systems depends on the use of computer for managing business applications. The computer is now a required source of information. System analysis relies heavily on computers for problem solving. This suggests that the analyst must be familiar with computer technology and have experience in handling people in an organization context.

Types of Computer-based Information Systems

Information systems differ in their business needs. Also depending upon different levels in organization information systems differ.

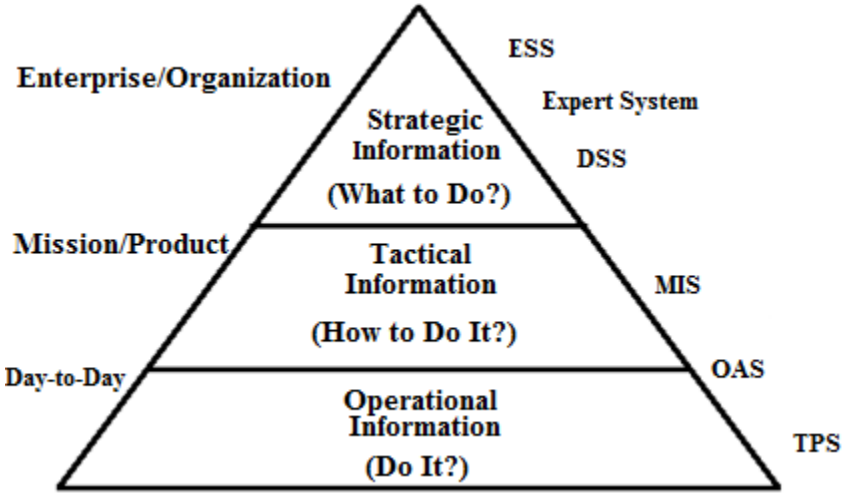


Figure: Various Information and Information Systems

Some major information systems are:

- Office automation Systems
- Transaction processing Systems
- Management information Systems
- Decision support Systems
- Expert Systems
- Executive Support Systems

Office Automation Systems (OAS)

Office automation system (OAS) is a collection of software and hardware products that increase productivity within the office setting. It can be described as a multi-function, integrated computer based system that allows many office activities to be performed in an electronic mode.

Office automation systems are among the newest and most rapidly expanding computer based information systems. They are being developed with the hopes and expectations that they will increase the efficiency and productivity of office workers, typists, secretaries, administrative assistants, staff professionals, managers and the like. Many organizations have taken the first step toward automating their offices.

The activities supported by these kinds of systems include:

✓ *Scheduling Resources*
Examples: Electronic Calendars with resource management (equipment, facilities, etc.)

✓ *Document Preparation*
Examples: Software (word processing and desktop publishing); hardware (printers)

✓ *Communicating*
Examples: e-mail, voice mail, Chat, videoconferencing and groupware

b i i z a y @ g m a i l . c o m

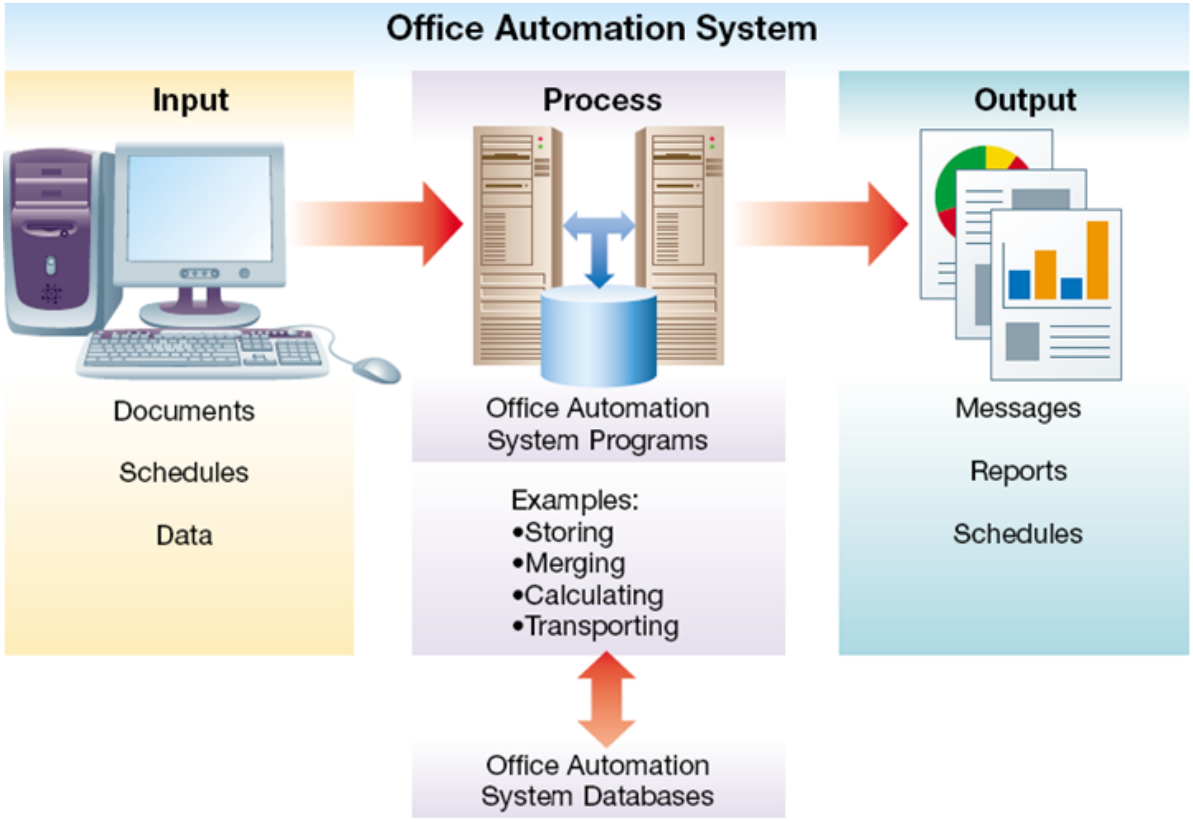


Figure: Office Automation Systems

Advantages:

- Office automation can get many tasks accomplished faster, eliminating the need for a larger staff.
- Less storage space is required for data, and copies can be easily transferred off-site for safekeeping in case of fire or other emergency.
- Multiple people can be updated simultaneously in the event of schedule changes.

Disadvantages:

- Older staff members may have a harder time adjusting to the new technology and be unable to use it efficiently.
- If something is "misfiled," it can be a lot harder to find.
- The amount of money required to implement and the cost of maintenance of certain equipment.

Transaction Processing System (TPS)

Transaction processing system (TPS) is a special class of information system designed to process business events and transactions. Two type of TPS exists: Batch transaction processing and Real-time processing.

Batch transaction processing collects the transaction data as a group, or batch, and processes it later. ***Real-time processing*** is the immediate processing of data. It provides instant confirmation of transaction but requires access to online database.

b i i z a y @ g m a i l . c o m

TPS are aimed at improving the routine business activities on which all organizations depend. They substitute computer based processing for manual procedure. TPS assists in carrying out the day-to-day, high volume activities/transactions of the organization. These transactions are processed using standard operating procedures. There are hardly any exceptions to these procedures. These routines are embedded in the computer programs that control the entry of data, processing of details, storage, and presentation of data and information.

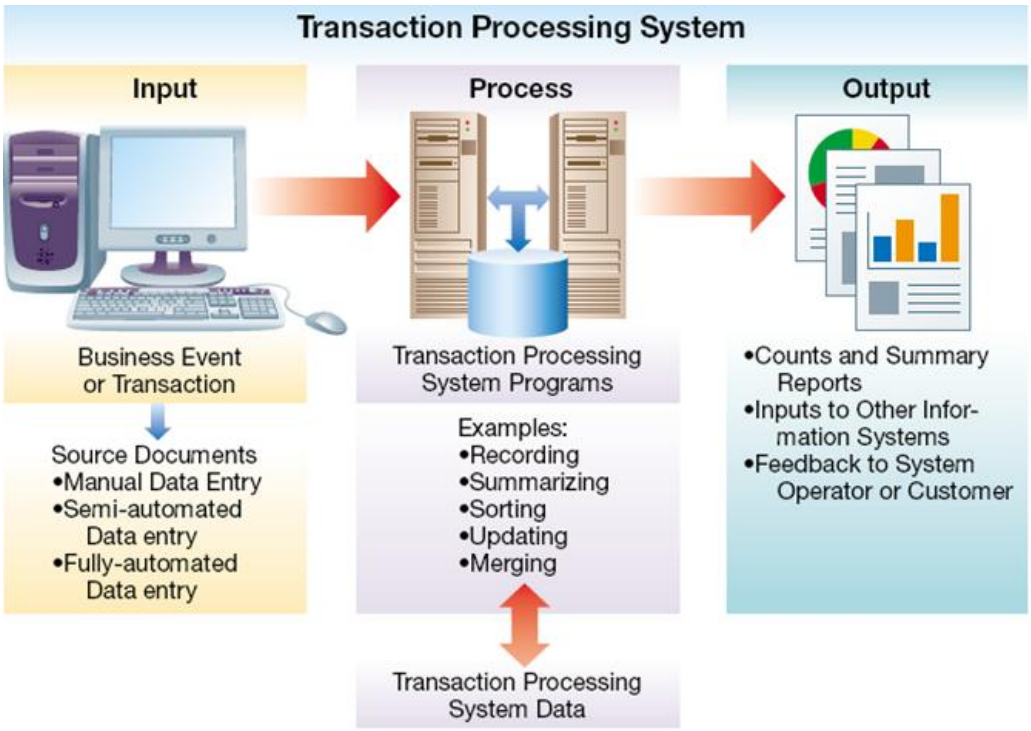


Figure: Transaction Processing Systems

This provides high speed and accurate processing of record keeping of basic operational processes. These include calculation, storage and retrieval. Transaction processing systems provide speed and accuracy, and can be programmed to follow routines functions of the organization.

Example: Pay cheques and other forms of paper output, Airline Reservation Systems, Banking Systems, Payroll Processing System, Purchase Order Entry System, or the Accounting System of almost any large company, etc.

Advantages:

- Handling of several thousand operations at once.
- In transaction processing there is no delay and the results of each transaction are immediately available.

Disadvantages:

- the need to handle hundreds, even thousands of simultaneous Users
- the need to allow many Users to work on the same set of data, with immediate updating

Management Information System (MIS)

b i i z a y @ g m a i l . c o m

Management information system (MIS) is used by managerial employees to support recurring decision making in managing a function or the entire business. These systems assist lower management in problem solving and making decisions. They use the results of transaction processing and some other information also. It is a set of information processing functions. It should handle queries as quickly as they arrive. An important element of MIS is database.

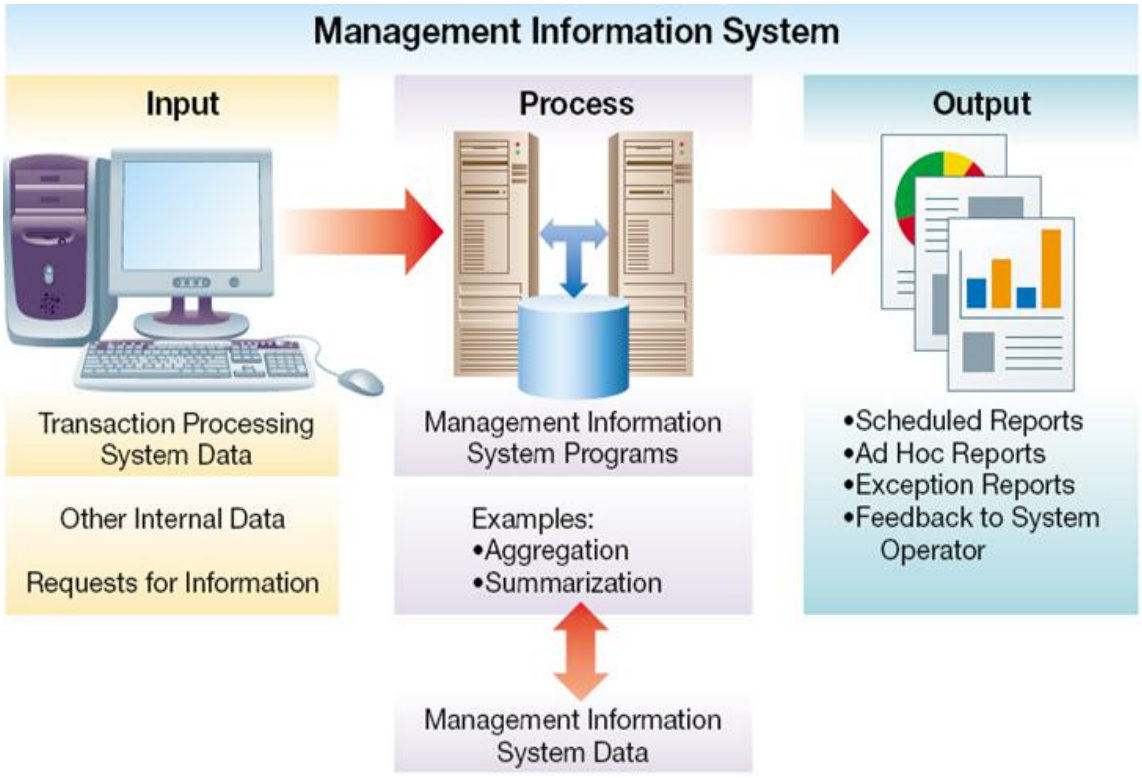


Figure: Management Information Systems

The information required at this level is used for making short term decisions and plans for the organization. Information like sales analysis for the past quarter/yearly production details etc. fall under this category. MIS caters to such information needs of the organization. Due to its capabilities to fulfill the managerial information needs of the organization, MIS have become a necessity for all big organizations. And due to its vastness, most of the big organizations have separate MIS departments to look into the related issues and proper functioning of the system.

MIS helps in studying the decisions taken, the factors leading to a specific decision, and in developing reports that are useful in formulating future decisions (Demand reports, Exceptional Schedule reports). MIS deals with supporting well-structured decision situations. MIS aims at improving operational efficiency. It utilizes transaction data obtained as the result of transaction processing, however MIS may also use other information that is developed internally and from outside the organization. This information is often represented as the tactical information. In a MIS the typical information requirement can be anticipated.

Example: Annual Budgeting, Payroll System, Sales Order System and Personnel Management System

Advantages:

b i i z a y @ g m a i l . c o m

- It facilitates planning
- It minimizes information overload by changing the larger amount of data in to summarize form.
- MIS encourages decentralization by making necessary change in the organizational plans and procedures.
- It brings coordination as it connects all decision centers in the organization.
- It makes control easier as MIS serves as a link between managerial planning and control.
- MIS assembles, process, stores, retrieves, evaluates and disseminates the information.
- MIS allows managers to make different types of reports about a company activities.

Disadvantages:

- Highly sensitive, requires constant monitoring.
- Budgeting of MIS extremely difficult.
- Quality of outputs governed by quality of inputs.
- Lack of flexibility to update itself.
- Effectiveness decreases due to frequent changes in top management
- Takes into account only qualitative factors and ignores non-qualitative factors like morale of worker, attitude of worker etc.

Decision Support Systems (DSS)

Decision support system (DSS) is a special-purpose information system designed to support managerial-level employees in organizational decision making.

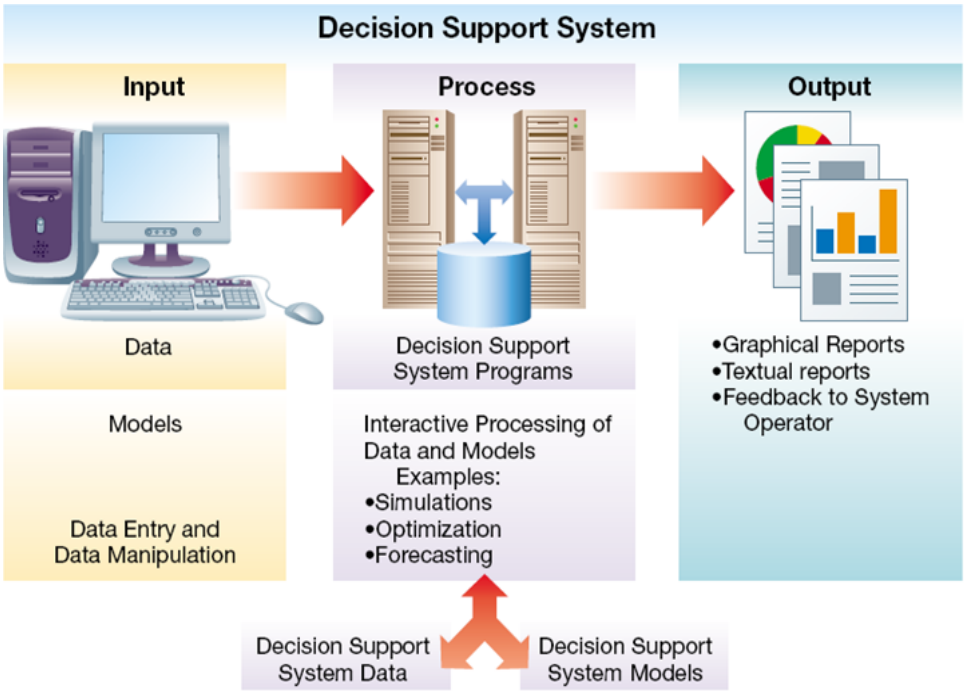


Figure: Decision Support Systems

b i i z a y @ g m a i l . c o m

DSS is an organized collection of people, procedures, databases, and devices used to support problem-specific decision making. These systems assist higher management to make long term decisions. These type of systems handle unstructured or semi structured decisions. A decision is considered unstructured if there are no clear procedures for making the decision and if not all the factors to be considered in the decision can be readily identified in advance.

DSS is tool that aids in the process of decision-making but it cannot take decisions. A manager in addition to the information gained by DSS relies on his experience and intuition. DSS are aimed at assisting managers who are faced with unique (non-recurring) and unstructured decision problems. DSS supports managers who are responsible for strategic planning and who take decisions based on long range considerations. DSS aids in decision-making in those situations where either information is not readily available or the information needs are difficult to predict.

Example: A DSS that facilitates use of simulation and what if mechanism for forecasting customer preferences about a product in the forthcoming decade.

Advantages:

- Reduced decision cycle time, increased employee productivity and more timely information for decision making.
- Improved decision making effectiveness and better decisions.
- Improved communication and collaboration among decision makers.
- Gaining a competitive advantage from computerized decision support.
- Cost saving from labor savings in making decisions and from lower infrastructure or technology costs.
- Increase decision maker satisfaction.
- Promote learning
- Increase organizational control.

Disadvantages:

- The DSS requires heavy investment in information system to collect data from many sources and analyze them to support the decision making.
- DSS may reinforce the rational perspective and overemphasize decision processes and decision making.
- Once DSS become common in organizations, that managers will use them inappropriately.
- Building DSS, especially knowledge-driven DSS, may be perceived as transferring decision authority to a software program.
- DSS is conceivable and it has been demonstrated that some DSS reduce the skill needed to perform a decision task.
- The computer does not make a "bad" decision, people do. Unfortunately some people may deflect personal responsibility to a DSS.
- Some managers argue using a DSS will diminish their status and force them to do clerical work. This perceptual problem can be a disadvantage of implementing a DSS.

Expert Systems (ES)

Expert system (ES) is a special-purpose system used by top level employees to make decisions usually made by more experienced employees or an expert in the field.

Expert system is a programmed decision–making information system that captures and reproduces the knowledge and expertise of an expert problem solver, managers, technicians etc. It replicates decision making process. It is actually a knowledge based system that describes the way an expert would approach the problem. It imitates the logic and reasoning of the experts within their respective fields. It is a branch of artificial intelligence (AI) and is also called knowledge-based systems.

ES is an information system that can function as a consultant to a problem solver, not only to by suggesting to a solution but also by explaining the line of reassuring that leads to the solution as what a human expert can do. These systems use inference engines that match facts and rules, sequence questions for the user, draw a conclusion, and present the user a recommendation.

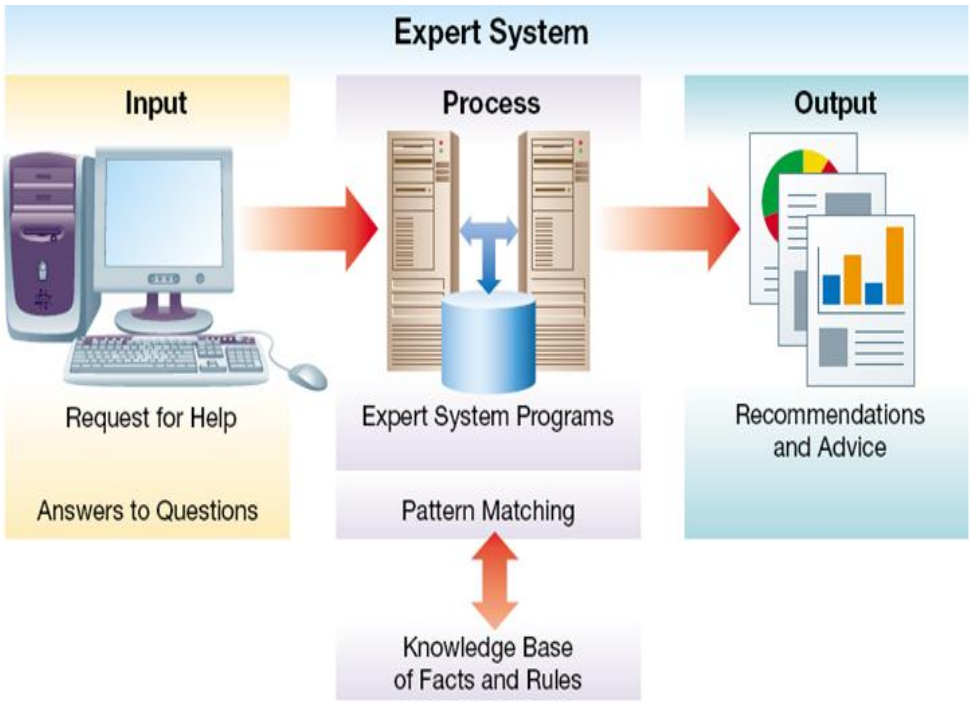


Figure: Expert Systems

Example: *MYCIN* is a medical diagnosis expert system, *Hearsay* was an early attempt at solving voice recognition through an expert systems approach, *SMH.PAL* is an expert system for the assessment of students with multiple disabilities.

These systems support many activities including:

- Medical Diagnosis
- Machine Configuration
- Financial Planning
- Software Application Assistance (help wizards)

Advantages:

- Provides consistent answers for repetitive decisions, processes and tasks
- Holds and maintains significant levels of information
- Encourages organizations to clarify the logic of their decision-making
- Never "forgets" to ask a question, as a human might
- Saving the human effort's time.
- Can work round the clock
- Can be used by the user more frequently
- A multi-user expert system can serve more users at a time
- It makes scarce expertise more widely available.
- It frees experts from constantly answering mundane or basic information-seeking questions.
- It provides guidance for novice users.
- It helps experienced searchers in a field that is not their specialty.

Disadvantages:

- Lacks common sense needed in some decision making
- Cannot make creative responses as human expert would in unusual circumstances
- Domain experts not always able to explain their logic and reasoning
- Errors may occur in the knowledge base, and lead to wrong decisions
- Cannot adapt to changing environments, unless knowledge base is changed
- The concept of the Expert Systems mainly involves a very narrow range of the codified domain.
- The Expert Systems are not generally adopted at managing the highly sophisticated sensory inputs.
- The Expert Systems mainly function in the domain of the extracted, cognitive, logical thinking process.
- The different types of the multi – dimensional problems that are faced by the various users while performing the various activities, cannot be efficiently tackled by the Expert Systems.
- Some of the typical Expert Systems at times are not able to make available common sense knowledge and the broad – ranging contextual information.
- Very narrow range of the knowledge is incorporated in the Expert Systems.
- The Expert Systems do – not respond well to the various situations out – side their range of the expertise.
- The Expert Systems remain what they are – the machine experts.
- The human self – awareness is lacking in the Expert Systems.
- The various Expert Systems lack the much needed self – analysis tools.
- The Expert Systems are non – self referral systems.
- In case of the Expert Systems, no introspection is possible.
- The Expert Systems have the ability of performing only with – in a specific, logical – oriented realm of the expertise.

Executive Support Systems (ESS)

Executive Support System (ESS) also called as Executive Information System (EIS) is a special purpose information system that support executive decision-making. It is used primarily by top management.

It is a user friendly, interactive system, and almost intuitive to use; it has excellent menus and graphic capabilities designed to meet the information needs of top management engaged in long range planning, crisis management, and other strategic decisions. Such systems assist in the making of decisions that require an in depth understanding of the firm and of the industry in which the firm operates.

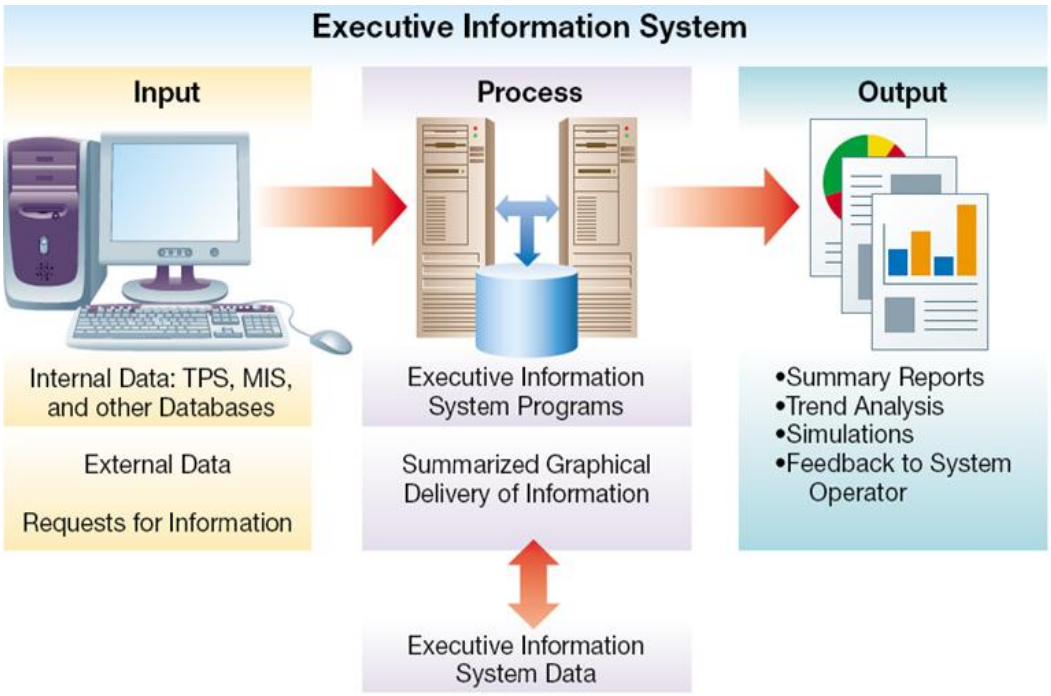


Figure: Executive Support Systems

Another special characteristic of an ESS is its drill down capability, which is the ability of the system to provide information at any level of detail desired by the decision maker.

For example, the CEO of a company may want the monthly sales of Product X for the entire company. Next, the CEO may want a breakdown of sales figures on a regional basis or on a store wide basis. The drill down facility can provide both.

The activities supported by these kinds of systems include:

- Executive Decision Making
- Long-range Strategic Planning
- Monitoring of Internal and External Events
- Crisis Management
- Staffing and Labor Relations

Advantages:

- Filters data for management
- Improves to tracking information
- Offers efficiency to decision makers
- Easy for upper-level executives to use, extensive computer experience is not required in operations
- Provides timely delivery of company summary information
- Information that is provided is better understood
- ESS provides timely delivery of information. Management can make decisions promptly.
- Improves tracking information
- Offers efficiency to decision makers
- Easy for upper level executive to use
- Ability to analyze trends
- Augmentation of managers' leadership capabilities
- Enhance personal thinking and decision making
- Contribution to strategic control flexibility
- Enhance organizational competitiveness in the market place
- Instruments of change
- Increased executive time horizons.
- Better reporting system
- Improved mental model of business executive
- Help improve consensus building and communication
- Improve office automation
- Reduce time for finding information
- Early identification of company performance
- Detail examination of critical success factor
- Better understanding
- Time management
- Increased communication capacity and quality

Disadvantages:

- Difficult to keep current data
- May lead to less reliable and insecure data
- Small companies may encounter excessive costs for implementation
- Too detailed-oriented
- System dependent
- Limited functionality, by design
- Information overload for some managers
- Benefits hard to quantify
- High implementation costs
- System may become slow, large, and hard to manage
- Need good internal processes for data management
- May lead to less reliable and less secure data

Systems Development

In systems analysis and design, we use various methodologies, techniques and tools that have been developed, tested, and widely used over the years to assist people during system analysis and design.

Methodologies are a sequence of step-by-step approaches that help develop your final product: the information system. Most methodologies incorporate several development techniques, such as direct observations and interviews with users of the current system.

Tools are computer programs, such as computer-aided software engineering (CASE) tools, that make it easy to use specific techniques.

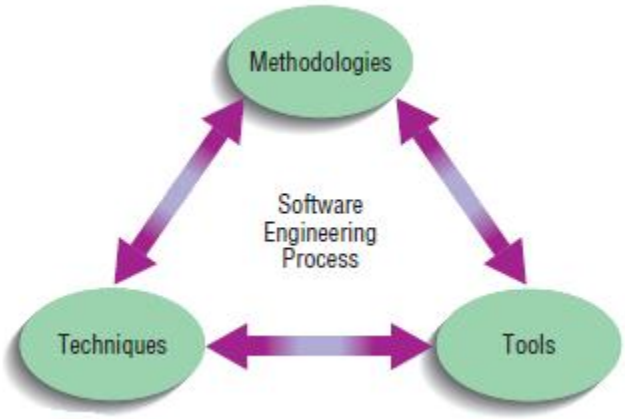


Figure: Methodologies, techniques, and tools.

Techniques are processes that you, as an analyst, will follow to help ensure that your work is well thought-out, complete, and comprehensible to others on your project team. Techniques provide support for a wide range of tasks, including conducting thorough interviews with current and future users of the information system to determine what your system should do, planning and managing the activities in a systems development project, diagramming how the system will function, and designing the reports, such as invoices, your system will generate for its users to perform their jobs. These three elements - methodologies, techniques, and tools - work together to form an organizational approach to systems analysis and design.

Developing Information Systems and the Systems Development Life cycle

System Development Life cycle (SDLC)

New information systems are created when existing systems do not adequately meet the needs of users of the information system, or when there is a need that could be met by an information system. The success of a new system depends upon how well the problem is understood, how the system is designed, how it is tested, evaluated and maintained over time.

b i i z a y @ g m a i l . c o m

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems. The concept generally refers to computer or information systems.

SDLC is the series of steps used to manage the phases of development for an information system. It is a common methodology for system development in all approaches. It is sometimes referred to as traditional SDLC.

It consists of six phases:

- Project Identification and Selection
- Project Initiation and Planning
- Analysis
- Design
- Implementation
- Maintenance

The first two phases are business analysis which results in conceptual solution. Every organization has its own life cycle model with more than these mentioned phases. The phases listed don't mean to come one to another serially.

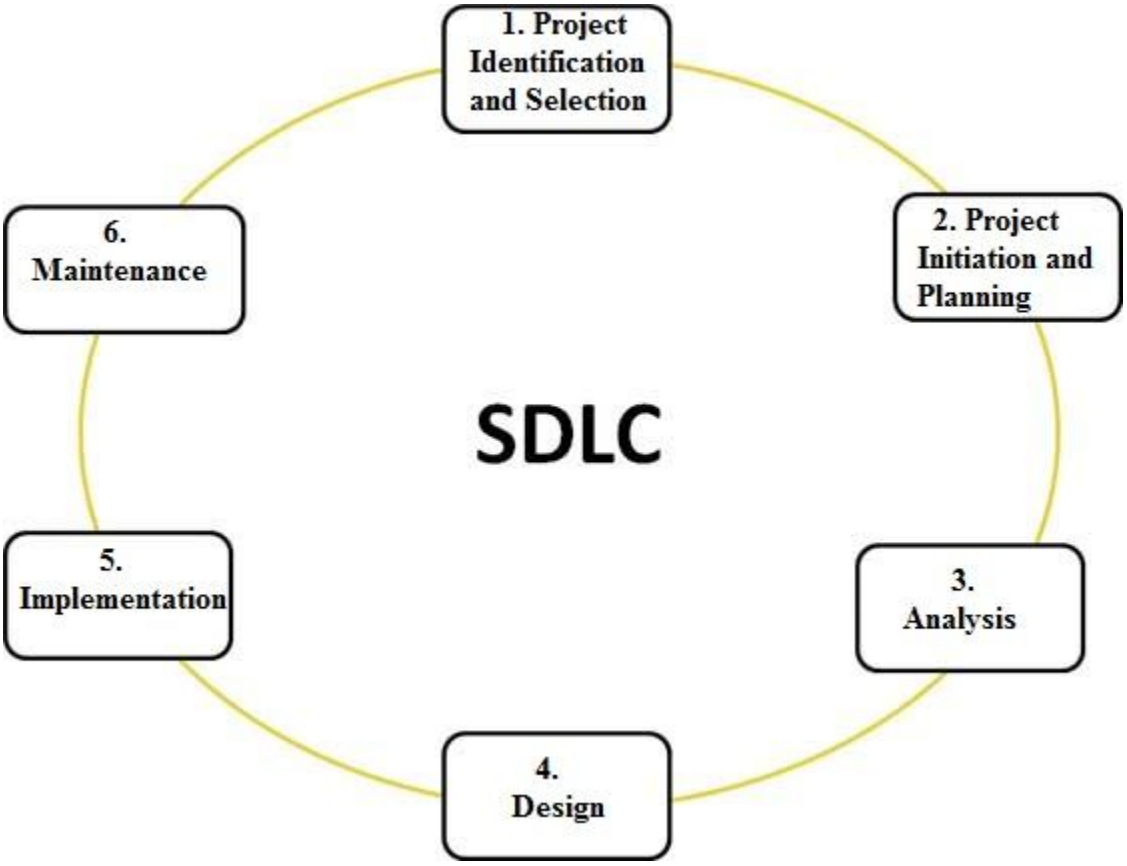


Figure: The System Analysis and Design (SDLC)

b i i z a y @ g m a i l . c o m

(1) Project Identification and Selection

It is the first phase of the SDLC in which an organization's total information system needs are identified, analyzed, prioritized, and arranged. This phase helps to translate the needs into a development schedule. It also helps organization to determine whether or not resources should be dedicated to a project.

(2) Project Initiation and Planning

It is 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. In this phase a detailed plan is also developed for conducting the remaining phases of the SDLC for the proposed system. It is actually a formal preliminary investigation of the problem at hand.

(3) Analysis

It is the third phase of the SDLC in which the current system is studied and alternative replacement systems are proposed. In this phase following things are analyzed: Determine requirements, Study current system, Structure requirements and eliminate redundancies, Generate alternative designs, Compare alternatives, and Recommend best alternative.

(4) Design

It is the fourth phase of the SDLC in which the description of the recommended solution is converted into logical and then physical system specifications.

Logical Design

In logical design, the requirement specifications are converted into independent computer platform design models. It includes external design, internal design and conceptual design.

Physical Design

In physical design, logical designs are converted into technical based design specifications.

(5) Implementation

It is the fifth phase of the SDLC in which the information system is coded, tested, installed, and supported in the organization.

(6) Maintenance

It is the final phase of the SDLC in which an information system is systematically repaired and improved.

Alternatives to Systems Analysis and Design / Software Development Models / System Development Methodologies

The development models are the various processes or methodologies that are being selected for the development of the project depending on the project's aims and goals. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out. There are various Software development models or methodologies. They are as follows:

Joint Application Design (JAD):

Joint Application Design (JAD) was developed by Chuck Morris of IBM Raleigh and Tony Crawford of IBM Toronto in the late 1970's.

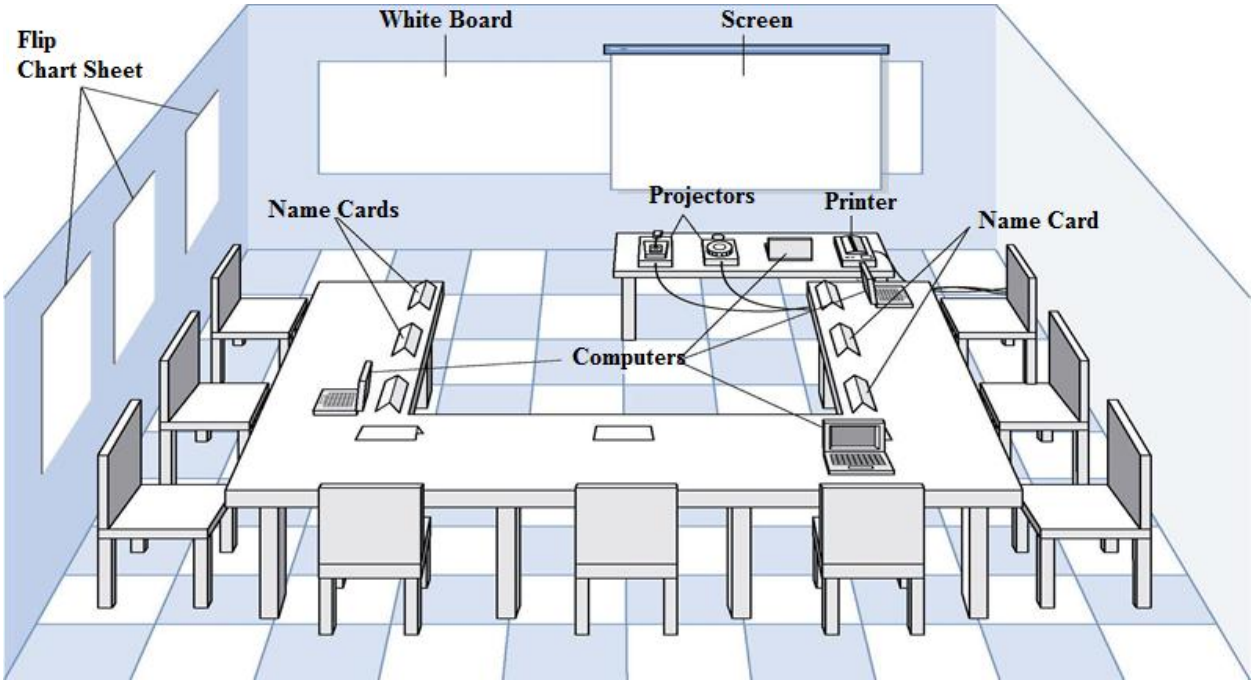


Figure: A JAD Room

JAD is 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. It is a management process - a people process - which allows IS to work more effectively with users in a shorter time frame. Because of bringing the people directly affected by the system in one place and time, time and organizational resources are better managed. Also, group members develop a shared understanding of what the system is supposed to do.

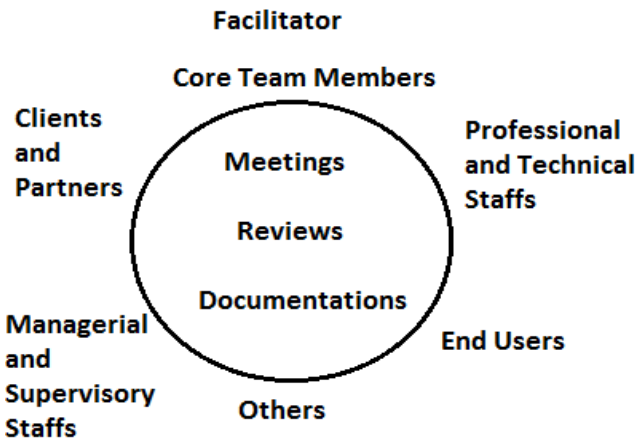


Figure: Joint Application Design (JAD) Model

b i i z a y @ g m a i l . c o m

The JAD process also includes approaches for enhancing user participation, expediting development, and improving the quality of specifications. It consists of a workshop where "knowledge workers and IT specialists meet, sometimes for several days, to define and review the business requirements for the system." The attendees include high level management officials who will ensure the product provides the needed reports and information at the end. This acts as "a management process which allows Corporate Information Services (IS) departments to work more effectively with users in a shorter time frame."

Advantages/Benefits:

- This technique allows for the simultaneous gathering and consolidating of large amounts of information.
- This technique produces relatively large amounts of high-quality information in a short period of time.
- JAD decreases time and costs associated with requirements elicitation process.
- Discrepancies are resolved immediately with the aid of the facilitator.
- This technique provides a forum to explore multiple points of view regarding a topic.
- JAD sessions help bring experts together giving them a chance to share their views, understand views of others, and develop the sense of project ownership.
- Time is saved, compared with traditional interviewing.
- Rapid development of systems.
- Improved user ownership of the system.
- Creative idea production is improved.

Disadvantages/Problems/Limitations:

- Requires significant planning and scheduling effort.
- Requires significant stakeholder commitment of time and effort.
- Requires trained and experienced personnel for facilitation and recording.
- JAD requires a large block of time to be available for all session participants.
- If preparation is incomplete, the session may not go very well.
- If the follow-up report is incomplete, the session may not be successful.
- The organizational skills and culture may not be conducive to a JAD session.

Rapid Application Design (RAD):

Rapid application development (RAD) is a software development methodology that uses minimal planning in favor of rapid prototyping. The "planning" of software developed using RAD is interleaved with writing the software itself. The lack of extensive pre-planning generally allows software to be written much faster, and makes it easier to change requirements. The fundamental principle of any RAD methodology is to delay producing detailed system design documents until after user requirements are clear.

According to Whitten (2004), it is a merger of various structured techniques, especially data-driven Information Engineering, with prototyping techniques to accelerate software systems development. It is a merger of various other technologies from structure design i.e. it involves JAD, Prototyping, CASE tools and code generators.

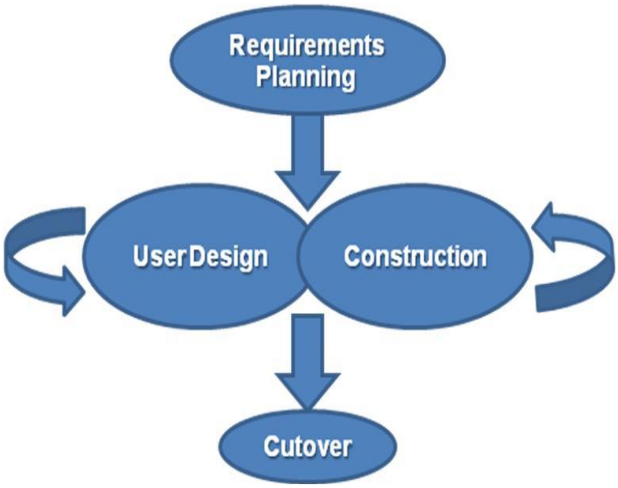


Figure: Rapid Application Development (RAD) Model

The development process starts with the development of preliminary data models and business process models using structured techniques. In the next stage, requirements are verified using prototyping, eventually to refine the data and process models. These stages are repeated iteratively; further development results in "a combined business requirements and technical design statement to be used for constructing new systems".

Advantages/Benefits:

- Flexible and adaptable to changes
- Can handle large projects without a doubt
- RAD realizes an overall reduction in project risk
- Generally Rad incorporates short development cycles
- Reduced development time.
- Increases reusability of components
- Quick initial reviews occur
- Encourages customer feedback
- Integration from very beginning solves a lot of integration issues.
- Results in reduction of manual coding due to code generators and code reuse.

Disadvantages/Problems/Limitations:

- For large projects, RAD requires sufficient human resources to develop the increments in parallel
- Projects will fail if people don't commit to rapid activities
- If a system can't be modularized, you can't build components
- Might not work if high performance is an issue
- Might not be appropriate when technical risks are high
- This method cannot be a success if the team is not sufficiently motivated and nor is unable to work cohesively together.
- Requires more resources and money to implement RAD
- Not suitable for small projects.

- Depends on strong team and individual performances for identifying business requirements.
- Only system that can be modularized can be built using RAD
- Requires highly skilled developers/designers.
- High dependency on modeling skills
- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.
- Not appropriate when technical risks are high. This occurs when the new application utilizes new technology or when new software requires a high degree of interoperability with existing system.

Prototyping:

Prototyping is an iterative process of systems development in which requirements are converted to a working system that is continually revised through close work between an analyst and users. Designing and building a scaled-down but functional version of a desired system is the process known as prototyping. Prototyping is part of the analysis phase of the systems development life cycle. It is the process of building a model of a system.

In prototyping, the analyst works with users to determine the initial or basic requirements for the system. The analyst then quickly builds a prototype. When the prototype is completed, the users work with it and tell the analyst what they like and do not like about it. The analyst uses this feedback to improve the prototype and takes the new version back to the users. This iterative process continues until the users are relatively satisfied with what they have seen.

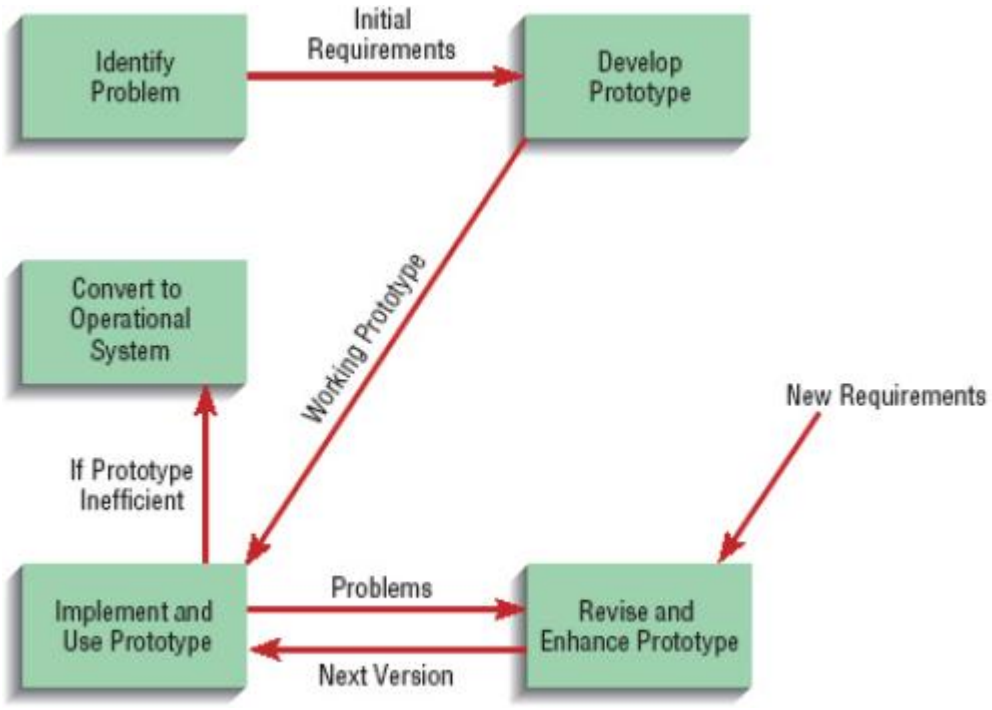


Figure: Prototyping Model

b i i z a y @ g m a i l . c o m

Prototyping is a form of rapid application development (RAD). Prototyping is a rapid, iterative, and incremental process of systems development in which requirements are converted to a working system that is continually revised through close work between the development team and the users.

Ideally, the prototype serves as a mechanism for identifying information system requirements. In this case, we throw away the prototype (also called throwaway prototype) after identifying requirements. The actual information system is developed with an eye toward quality and maintainability based on the requirements.

Advantages/Benefits:

- Useful for projects in which user requirements are uncertain or imprecise.
- A powerful, bottom up alternative or supplement to logical modeling.
- It encourages active user and management participation.
- Projects have higher visibility and support because of the extensive user involvement.
- Users and management see working, software based solutions more rapidly.
- Errors and omissions tend to be detected earlier in prototypes.
- Testing and training are natural by-products.
- It is more natural process.
- It is most popular for small to medium-size projects.
- Users are actively involved in the development
- Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
- Errors can be detected much earlier.
- Quicker user feedback is available leading to better solutions.
- Missing functionality can be identified easily
- Confusing or difficult functions can be identified

Disadvantages/Problems/Limitations:

- It increases lifetime cost to operate, support and maintain the system.
- It can solve the wrong problems since problem analysis is abbreviated or ignored.
- The product may have less quality because of speed in development.
- Leads to implementing and then repairing way of building systems.
- Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
- Incomplete application may cause application not to be used as the full system was designed
- Incomplete or inadequate problem analysis.

Object-Oriented Analysis and Design:

In object-oriented design, the processes, data and flows are combined into single entity called object. It helps in easy conversion from analysis to design models and supports multimedia. It is often called the third approach to systems development, after the process-oriented and data-oriented approaches.

Object-oriented analysis and design (OOAD) is a software engineering approach that models a system as a group of interacting objects. Each object represents some entity of interest in the system being modeled, and is characterized by its class, its state (data elements), and its behavior. Various models can be created to show the static structure, dynamic behavior, and run-time deployment of these collaborating objects. There are a number of different notations for representing these models, such as the Unified Modeling Language (UML).

Object-oriented analysis (OOA) applies object-modeling techniques to analyze the functional requirements for a system. Object-oriented design (OOD) elaborates the analysis models to produce implementation specifications. OOA focuses on what the system does, OOD on how the system does it.

Advantages/Benefits:

- Real-World Modeling
- Improved Reliability and Flexibility
- High Code Reusability. Reusability is provided through inheritance no need to test same piece of code again n again since it is already being tested.
- Easy to understand. Objects may be understood as stand-alone entities.
- Easy to operate compared to structured programs.
- Modular approach to problem solving
- Easier and reduced maintenance.
- Produce safer, more reliable and maintainable code

Disadvantages/Problems/Limitations:

- Some developers and managers today believe that OOAD simply means defining classes, objects, and methods. They may confuse style with substance.
- Managers and developers may not recognize all the implications of OOAD. They often assume that using OOAD will eliminate development bottlenecks.

Waterfall Model:

Waterfall Model is the most basic Life-Cycle model. It was developed by Winston Royce in the early 1970. The waterfall model is a sequential software development model in which development is seen as flowing steadily downwards (like a waterfall) through several phases. Each phase has a set of well-defined goals and activities. The important contribution of the waterfall model is for management. It enables management to track development progress.

Thus the waterfall model maintains that one should move to a phase only when it's preceding phase is completed and perfected. However, there are various modified waterfall models (including Royce's final model) that may include slight or major variations upon this process.

In Royce's original waterfall model, the following phases are followed in order:

- Requirements specification
- Design
- Construction (i.e. implementation or coding)

- Integration
- Testing and debugging (i.e. Validation)
- Installation
- Maintenance

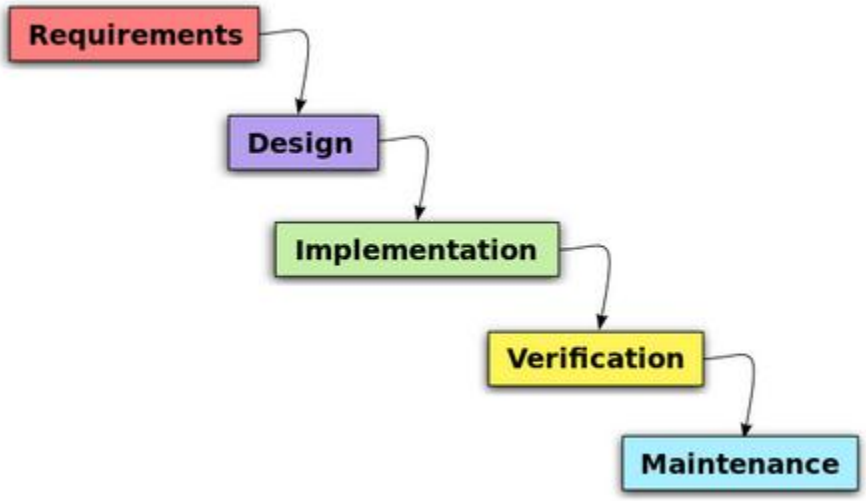


Figure: Waterfall Model

Advantages/Benefits:

- Simple and easy to understand and use.
- Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Being a linear model, it is very simple to implement.
- The amount of resources required to implement this model are minimal.
- Documentation is produced at every stage of the software's development. This makes understanding the product designing procedure, simpler.
- After every major stage of software coding, testing is done to check the correct running of the code.

Disadvantages/Problems/Limitations:

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.
- Real projects rarely follow a sequential flow
- The customer has to state all requirements explicitly
- Working version of the program appears late in the project

Spiral Model:

The spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. Also known as the spiral lifecycle model (or spiral development), it is a systems development method used in information technology. This model of development combines the features of the prototyping model and the waterfall model. The spiral model is intended for large, expensive and complicated projects.

The spiral model was defined by Barry Boehm in his 1986 article "A Spiral Model of Software Development and Enhancement". This model was not the first model to discuss iterative development. The spiral model is based on continuous refinement of key products for requirements definition and analysis, system and software design, and implementation (the code). At each iteration around the cycle, the products are extensions of an earlier product. This model uses many of the same phases as the waterfall model, in essentially the same order, separated by planning, risk assessment, and the building of prototypes and simulations.

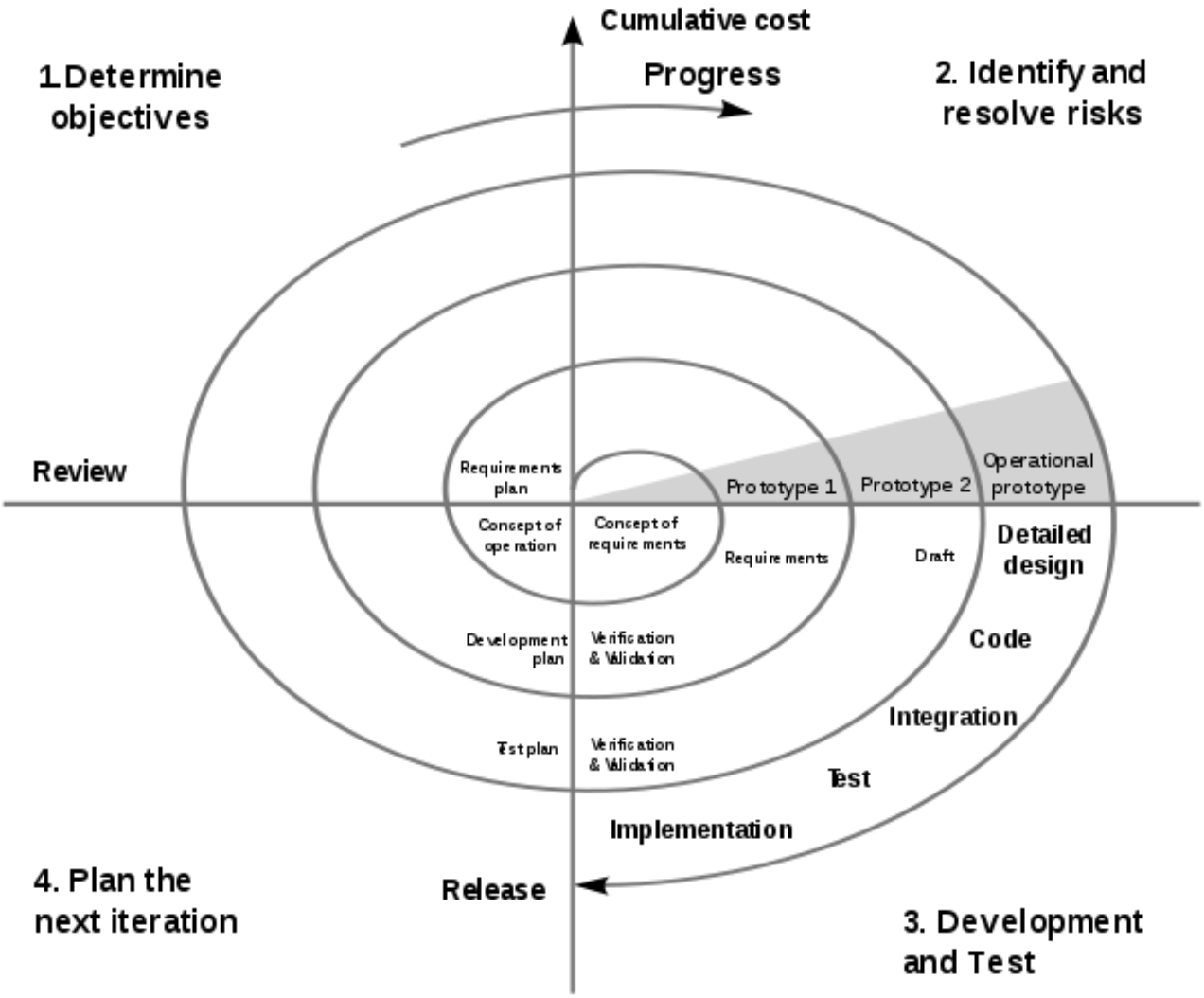


Figure: Spiral Model

b i i z a y @ g m a i l . c o m

Starting at the center, each turn around the spiral goes through several task regions:

- Determine the objectives, alternatives, and constraints on the new iteration.
- Evaluate alternatives and identify and resolve risk issues.
- Develop and verify the product for this iteration.
- Plan the next iteration.

1. *First quadrant (Objective Setting)*
   - During the first quadrant, it is needed to identify the objectives of the phase.
   - Examine the risks associated with these objectives.

2. *Second Quadrant (Risk Assessment and Reduction)*
   - A detailed analysis is carried out for each identified project risk.
   - Steps are taken to reduce the risks. For example, if there is a risk that the requirement are inappropriate, a prototype system may be developed.

3. *Third Quadrant (Development and Validation)*
   - Develop and validate the next level of the product after resolving the identified risks.

4. *Fourth Quadrant (Review and Planning)*
   - Review the results achieved so far with the customer and plan the next iteration around the spiral.
   - Progressively more complete version of the software gets built with each iteration around the spiral.

Advantages/Benefits:

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the software life cycle.
- Repeated or continuous development helps in risk management. The developers or programmers describe the characteristics with high priority first and then develop a prototype based on these. This prototype is tested and desired changes are made in the new system. This continual and steady approach minimizes the risks or failure associated with the change in the system.
- Adaptability in the design in software engineering accommodates any number of changes that may happen, during any phase of the project.
- Since the prototype building is done in small fragments or bits, cost estimation becomes easy and the customer can gain control on administration of the new system.
- As the model continues towards final phase, the customer's expertise on new system grows, enabling smooth development of the product meeting client's needs.

Disadvantages/Problems/Limitations:

- May be hard to convince customers that it’s controllable
- Demands considerable risk assessment and expertise

- Problems if a major risk is not uncovered and managed
- The models work best for large projects only, where the costs involved are much higher and system pre requisites involves higher level of complexity.
- It needs extensive skill in evaluating uncertainties or risks associated with the project and their abatement.
- It works on a protocol, which needs to be followed strictly for its smooth operation. Sometimes it becomes difficult to follow this protocol.
- Evaluating the risks involved in the project can shoot up the cost and it may be higher than the cost for building the system.
- There is a requirement for further explanation of the steps involved in the project such as breakthrough, blueprint, checkpoints and standard procedure.

Incremental/Iterative Model:

In incremental model the whole requirement is divided into various builds. This model is an intuitive approach to the waterfall model. Multiple development cycles take place here, making the life cycle a "multi-waterfall" cycle. Cycles are divided up into smaller, more easily managed iterations. Each iteration passes through the requirements, design, implementation and testing phases.

A working version of software is produced during the first iteration, so you have working software early on during the software life cycle. Subsequent iterations build on the initial software produced during the first iteration.

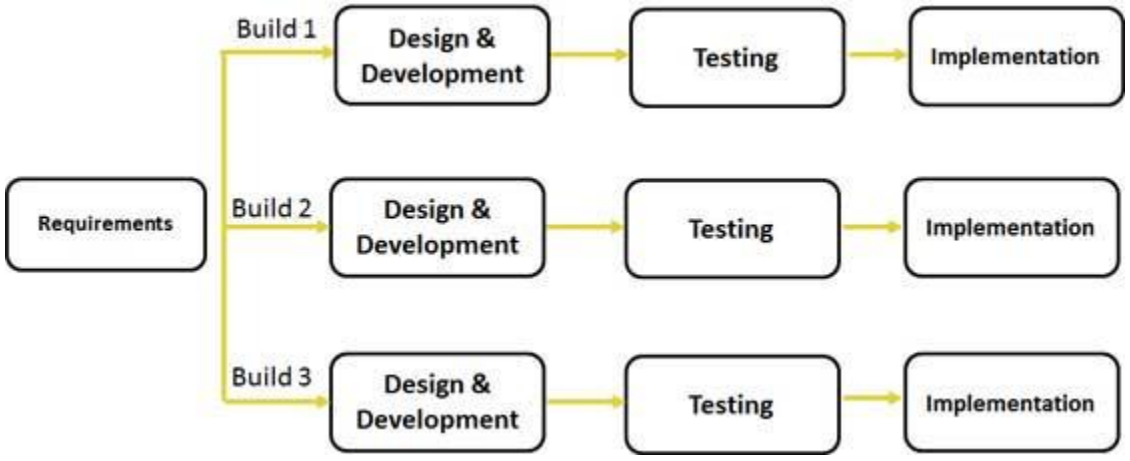


Figure: Iterative/Incremental Model

Advantages/Benefits:

- Generates working software quickly and early during the software life cycle.
- More flexible – less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Each iteration is an easily managed milestone
- In this model customer can respond to each built.
- Lowers initial delivery cost.

- Easier to manage risk because risky pieces are identified and handled during it'd iteration.
- In iterative model we can only create a high-level design of the application before we actually begin to build the product and define the design solution for the entire product. Later on we can design and built a skeleton version of that, and then evolved the design based on what had been built.
- In iterative model we are building and improving the product step by step. Hence we can track the defects at early stages. This avoids the downward flow of the defects.
- In iterative model we can get the reliable user feedback. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.
- In iterative model less time is spent on documenting and more time is given for designing.

Disadvantages/Problems/Limitations:

- Each phase of an iteration is rigid and do not overlap each other.
- Needs good planning and design.
- Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
- Total cost is higher than waterfall.
- Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle

Systems Analysis and Design Tools

Some other tools used in system analysis and design are covered in the upcoming chapters. These tools are:

(1) CASE Tools
(2) Data Flow Diagram (DFD)
(3) Entity-Relationship Diagram (ERD)
(4) Data Dictionary (DD)
(5) Structure Chart
(6) State Diagram
(7) Decision Table
(8) Decision Tree
(9) Structured English
(10) System Flow Chart

References:

- ✓ Hoffer, J.A., George, J.F. and Valacich J.S., "Modern Systems Analysis and Design", 3rd Edition, Pearson Education, 2003.
- ✓ Jeffrey L. Whitten, Loonnie D. Bentley, 5rd Edition, "Systems Analysis and Design Methods".
- ✓ Englewood Cliffs, New Jersey, "Systems Analysis and Design".
- ✓ Joseph S. Valacich, Joey F. George, Jeffrey A. Hoffer, "Essentials of Systems Analysis and Design" - 5th Edition.
- ✓ K.E. Kendall and J.E. Kendall, "Systems Analysis and Design", 5th Edition, Pearson Education, 2003.

- ✓ V. Rajaraman, “Analysis and Design of Information Systems”, 2nd Edition, Prentice Hall of India, New Delhi, 2002
- ✓ Shelly Cashman, “System Analysis and Design” - 9th Edition.

Assignments:

(1) What are the advantages of computer-based information systems over manual information systems?
(2) Compare and contrast between batch processing and real-time transaction processing.
(3) What are the differences between Prototype Model and Waterfall Model?
(4) What are the differences between Waterfall Model and Spiral Model?
(5) Discuss the importance of system analysis and design in the development of a system?
(6) What type of information is required by the top level of management, and why?
(7) Differentiate between transaction processing systems (TPS) and management information system (MIS). (T.U. 2067)
(8) What are the system analysis and design tools? (T.U. 2067)
(9) What do you mean by system analysis? Explain the system development life cycle with example. (T.U. 2067)
(10) Explain development methodology used in developing information system in detail with example. (T.U. 2068)
(11) What do you mean by JAD? Explain. (T.U. 2068, 2069)
(12) Explain the types of information with example and compare each of them. (T.U. 2069)
(13) Differentiate between DSS and MIS. (T.U. 2069)
(14) Mention the key steps of system development life cycle and explain each steps with example. (T.U. 2069)
(15) What are the types of information systems? (T.U. 2069)

A Gentle Advice:

Please go through your text books and reference books for detail study!!! Thank you all.

Notes Compiled By:

Bijay Mishra
biizay.blogspot.com
9813911076 or 9841695609

Unit 2 - Modeling Tools for Systems Analyst

2. Modeling Tools for System Analysts **5 Hrs.**

2.1 System Analyst (Introduction, Roles, and Skills)
2.2 Modeling with Context Diagram and Data Flow Diagrams (Level 0, 1 and 2)
2.3 Modeling with Entity- Relationship Diagrams
2.4 CASE Tools

System Analyst

A systems analyst is an IT professional who specializes in analyzing, designing and implementing information systems. System analysts assess the suitability of information systems in terms of their intended outcomes and liaise with end users, software vendors and programmers in order to achieve these outcomes.

System Analyst Roles

Systems analysts play a key organizational role in systems development. They act as liaisons between business users on one hand and technical personnel on the other. Although many people in organizations are involved in systems analysis and design, the systems analyst has the primary responsibility. A career as a systems analyst will allow you to have a significant impact on how your organization operates. This fast-growing and rewarding position is found in both large and small companies.

The primary role of a systems analyst is to study the problems and needs of an organization in order to determine how people, methods, and information technology can best be combined to bring about improvements in the organization. A systems analyst helps system users and other business managers define their requirements for new or enhanced information services.

Among several roles, some important roles are:

1. ***Change Agent:*** An analyst can be viewed as an agent of change. Actually a candidate system is established to introduce change and re-orientation in how the user organization handles information or makes decision. So it's important that user accepts changes. For example an analyst prefers participation of users while designing and implementation. Analyst plans, monitors and implements change into the user domain. So as agent analyst may use different approaches to introduce to user organization.
2. ***Investigator and monitor:*** A system analyst may work as investigator to know the reasons why existing system became fail. The role of system analyst is to extract the problems from existing systems and create information structures that uncover previously unknown trends that may have direct impact on organization.in this role the analyst must monitor programs in relation to time, cost and quality.
3. ***Architect:*** The analyst's another role also can be of an architect. Here he/she plays role of liaison between logical design and detailed physical design, as an architect the analyst also creates a detailed physical design on the basis of end users requirements.
4. ***Psychologist:*** In system development, the systems are built around people. The analyst plays role of psychologist in the way s/he reaches people, interprets their thoughts, assesses their behavior and draws conclusions from these interactions and finds facts.

b i i z a y @ g m a i l . c o m

5. ***Motivator:*** As the system acceptance is achieved then analyst starts to give training to user with motivation towards new system. It happens just during few weeks after implementation and during times when turnover results in new people being trained to work.
6. ***Intermediary:*** The analyst tries to appease all parties involved. Diplomacy in dealing with people can improve acceptance of the system. The goal of analyst is to have support of all users. The analyst represents their way of thinking and tries achieve their goal with computerization.

System Analyst Skills

Systems analysts are key to the systems development process. To succeed as a systems analyst, you will need to develop four types of skills: analytical, technical, managerial, and interpersonal.

Analytical Skills:

The system analyst possesses four sets of analytical skills:

- Systems Thinking
- Organizational Knowledge
- Problem Identification
- Problem Analyzing and Solving

Analytical skills enable system analyst to understand the organization and its functions, to identify opportunities and problems, and to analyze and solve problems. One of the most important analytical skills he can develop is systems thinking, or the ability to see organizations and information systems as systems. Systems thinking provides a framework from which to see the important relationships among information systems, the organizations they exist in, and the environment in which the organizations themselves exist. Problem identification is process of defining differences, i.e. difference between an existing situation and a desired situation. After analyzing the problem he should be able to solve it.

Technical Skills:

Technical skills help system analyst understand the potential and the limitations of information technology. As an analyst, he must be able to envision an information system that will help users solve problems and that will guide the system's design and development. He must also be able to work with programming languages such as C++ and Java, various operating systems such as Windows and Linux, and computer hardware platforms such as IBM and Mac.

Management Skills:

Management skills help system analyst in

- Resource Management
- Project Management
- Risk Management
- Change Management

Systems analyst needs to know how to get the most out of the resources of an organization, including team members. He must assists management in keeping track of project's progress. He must minimize risk and/or minimize damage that might result. He must have an ability to assist people in making transition to new system.

Interpersonal Skills:

Interpersonal skills help system analyst to work with end users as well as with other analysts and programmers. Mastery of interpersonal skills is paramount to success as a Systems Analyst. He should have four types of skills:

- Communication skills
- Working alone and with a team
- Facilitating groups
- Managing expectations

As a systems analyst, he will play a major role as a liaison among users, programmers, and other systems professionals. Effective written and oral communication, including competence in leading meetings, interviewing end users, and listening, are key skills that analysts must master. Team work involves establishing standards of cooperation and coordination. Effective listening leads to understanding of problem and generates additional questions. He should involve guiding a group without being a part of the group. Managing expectations is directly related to successful system implementation.

Process Modeling:

Process modeling graphically represents the processes that capture, manipulate, store, and distribute data between a system and its environment and among system components. It utilizes the information gathered during requirements determination. Main focus is to model the processes and data structures. It is a formal way of representing how a business system operates. It also illustrates the activities that are performed and how data moves among them.

A *process model* is a formal way of representing how a business operates. It is a core diagram in structured analysis and design. It shows the flow of information through a system. Each process transforms inputs into outputs. Process models are based on behavior and actions. Example: Data flow diagram

Data Flow Diagram (DFD)

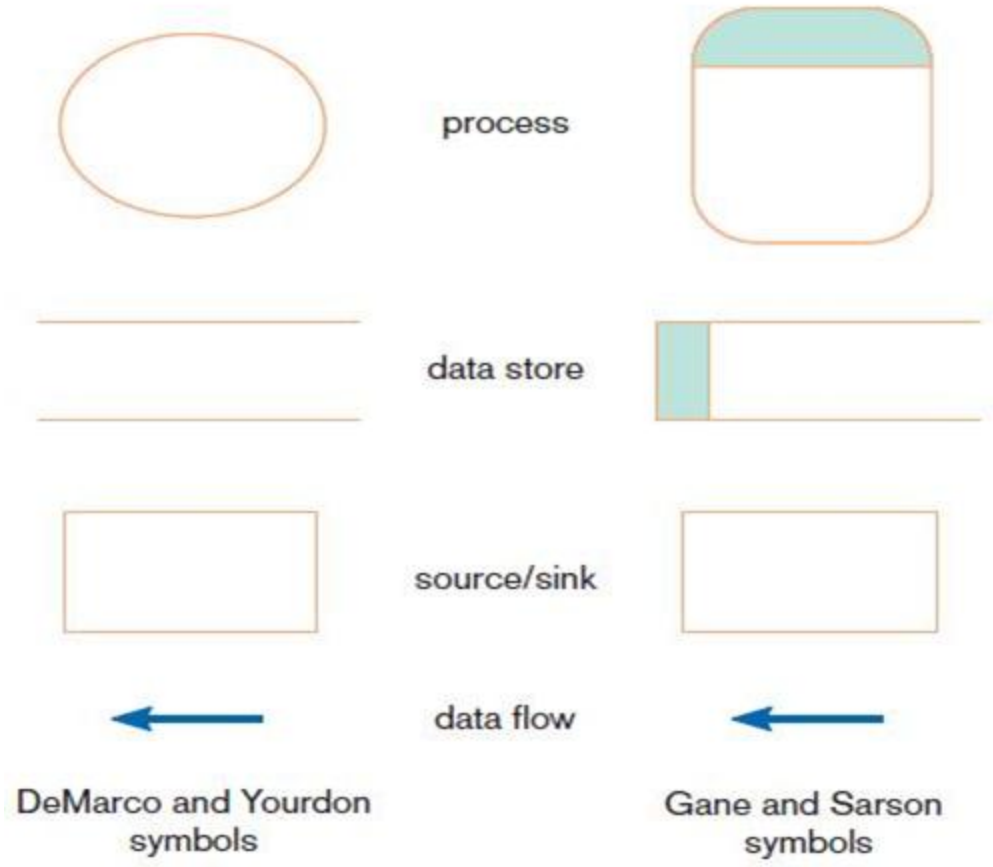
A DFD is a pictorial representation of the movement of data between external entities and the processes and data stores within a system. It is a common technique for creating process models. A DFD shows how data moves through an information system but does not show program logic or processing steps. DFDs can also be used for the visualization of data processing (Structured Design).

Data Flow Diagramming Mechanics

Four symbols are used to represent DFDs (See Figure below). Two different standard sets can be used as proposed by:

1. DeMarco and Yourdan
2. Gane and Sarson

b i i z a y @ g m a i l . c o m



Data Flow

A data flow is data that are in motion and moving as a unit from one place in a system to another. A data flow is depicted as an arrow. The arrow is labeled with a meaningful name for the data in motion; for example, customer order, sales receipt, or paycheck.

Process

A process is the work or actions performed on data so that they are transformed, stored, or distributed. The symbol for a process is a rectangle with rounded corners. Inside the rectangle are written both the number of the process and a name, which indicates what the process does. For example, the process may generate paychecks, calculate overtime pay, or compute grade-point average.

Data Store

A data store is data at rest. A data store may represent one of many different physical locations for data, including a file folder, one or more computer-based file(s), or a notebook. The symbol for a data store is a rectangle with the right vertical line missing. Its label includes the number of the data store (e.g., D1 or D2) and a meaningful label, such as student file, transcripts, or roster of classes.

Source/Sink (External Entities)

Source/Sink is the origin and/or destination of the data (outside of the system). Source/sinks are sometimes referred to as external entities because they are outside the system and define

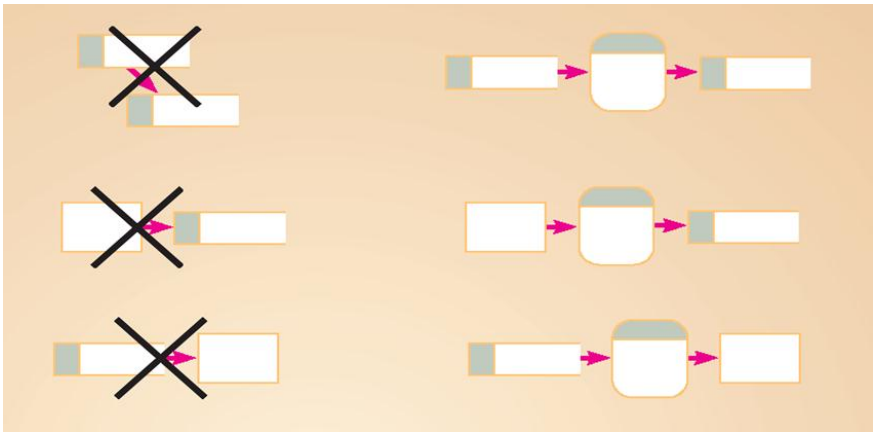
the system's boundaries. Data must originate outside a system from one or more sources, and the system must produce information to one or more sinks. It is drawn as a square symbol. The name states what the external agent is. Because they are external, many characteristics are not of interest to us. A person, organization, or system that is external to the system but interacts with it.

DFD Diagramming Rules:

Rules for Process:

1. No process can have only outputs. It is making data from nothing (a miracle). If an object has only outputs, then it must be a source.
2. No process can have only inputs (a black hole). If an object has only inputs, then it must be a sink.
3. A process has a verb-phrase label.

Rules for Data Store:



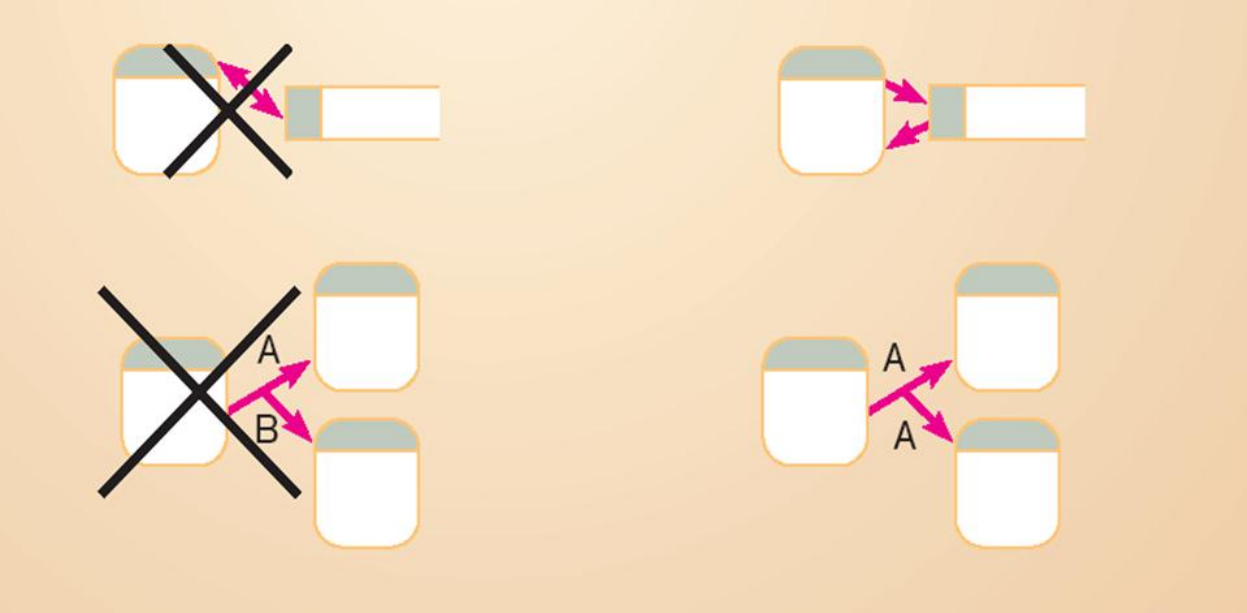
4. Data cannot move directly from one data store to another data store. Data must be moved by a process.
5. Data cannot move directly from an outside source to a data store. Data must be moved by a process that receives data from the source and places the data into the data store.
6. Data cannot move directly to an outside sink from a data store. Data must be moved by a process.
7. A data store has a noun-phrase label.

b i i z a y @ g m a i l . c o m

Rules for Source/Sink (External Entities):

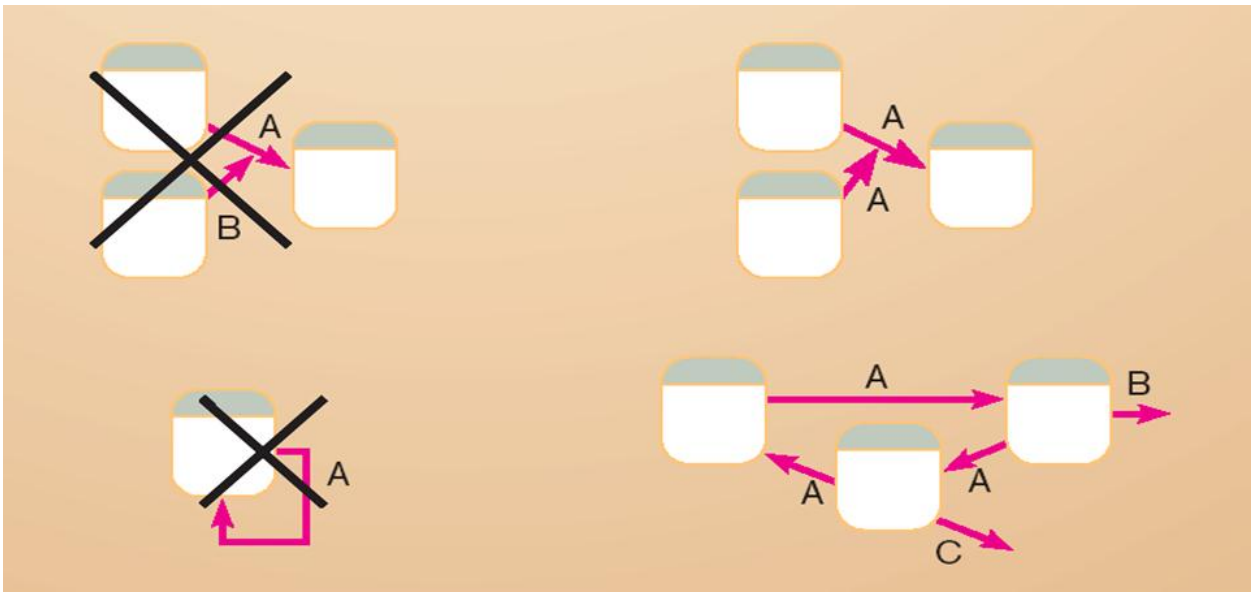
8. Data cannot move directly from a source to a sink. They must be moved by a process if the data are of any concern to our system. Otherwise, the data flow is not shown on the DFD.
9. A source/sink has a noun-phrase label.

Rules for Data Flow:



10. A data flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The latter is usually indicated, however, by two separate arrows because the read and update usually happen at different times.
11. A fork in a data flow means that exactly the same data go from a common location to two or more different processes, data stores, or sources/sinks (it usually indicates different copies of the same data going to different locations).
12. A join in a data flow means that exactly the same data come from any of two or more different processes, data stores, or sources/sinks to a common location.
13. A data flow cannot go directly back to the same process it leaves. At least one other process must handle the data flow, produce some other data flow, and return the original data flow to the beginning process.
14. A data flow to a data store means update (delete or change).

15. A data flow from a data store means retrieve or use.
16. A data flow has a noun-phrase label. More than one dataflow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.



Summary of the Rules:

From/To	Process	Data Store	External Entity
Process	√	√	√
Data Store	√	×	×
External Entity	√	×	×

Plus, additionally, each data store must have at least one input flow and one output flow (read & write).

Functional Decomposition:
It is an iterative process of breaking a system description 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 decomposition is an act of going from one single system to many component processes. It is a repetitive procedure. Decomposition goes on until you have reached the point where no sub process can logically be broken down any further. The lowest level of DFDs is called a ***primitive DFD***.

DFD Levels:

- ✓ Context Diagram
  Overview of the organizational system

b i i z a y @ g m a i l . c o m

- ✓ Level-0 DFD
  Representation of system's major processes at high level of abstraction
- ✓ Level-1 DFD
  Results from decomposition of Level 0 diagram
- ✓ Level-n DFD
  Results from decomposition of Level n-1 diagram

Context Diagram:
Context diagram shows the system boundaries, external entities that interact with the system, and major information flows between entities and the system. It is the first DFD in every business process. It shows the context into which the business process fits. It also shows the overall business process as just one process (process 0). It shows all the external entities that receive information from or contribute information to the system.

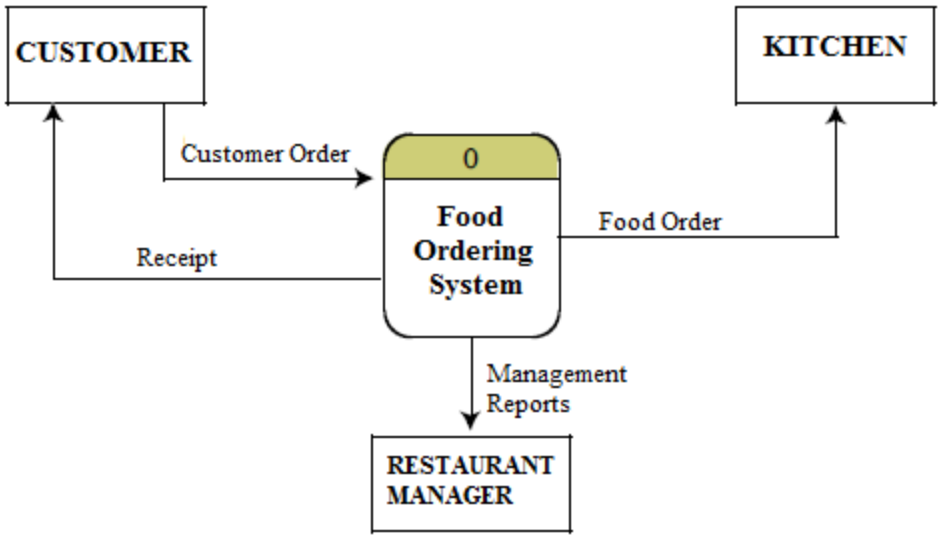


Figure: Context Diagram of Food Ordering System

Note: Only one process symbol and no data stores are shown in context diagram.

Level-0 DFD:
Level-0 DFD shows the system's major processes, data flows, and data stores at a high level of abstraction.

It shows all the major processes that comprise the overall system – the internal components of process 0. It also shows how the major processes are interrelated by data flows. It shows external entities and the major processes with which they interact. It adds data stores.

Note: Processes are labeled 1.0, 2.0, etc. These will be decomposed into more primitive (lower-level) DFDs.

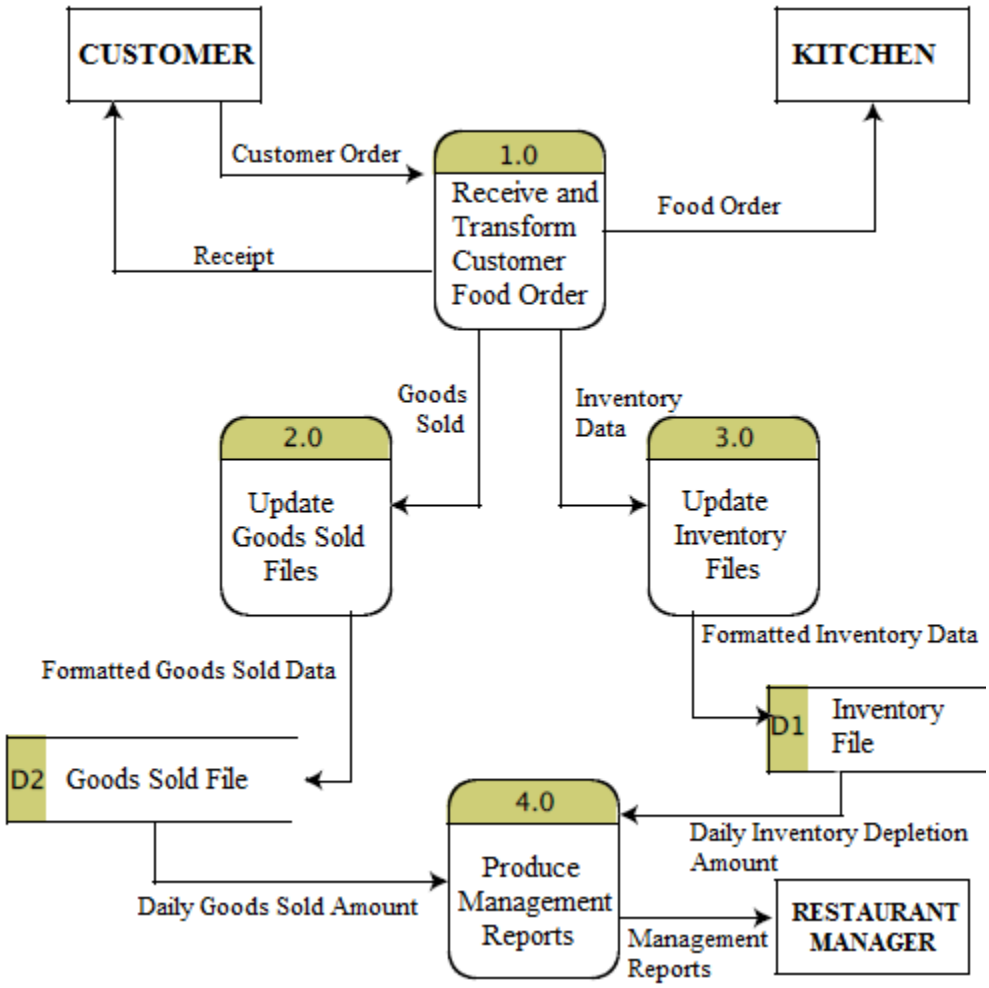


Figure: Level-0 DFD of Food Ordering System

Level-1 DFD:
Level-1 DFD shows the sub-processes of one of the processes in the Level-0 DFD.

Generally, one level 1 diagram is created for every major process on the level 0 diagram. It shows all the internal processes that comprise a single process on the level 0 diagram. It also shows how information moves from and to each of these processes.

If a parent process is decomposed into, for example, three child processes, these three child processes wholly and completely make up the parent process.

Note: Processes are labeled 1.1, 1.2, 4.1, 4.2, etc. These can be further decomposed in more primitive (lower-level) DFDs if necessary. Sources and sinks are optional on level-1 diagrams.

b i i z a y @ g m a i l . c o m

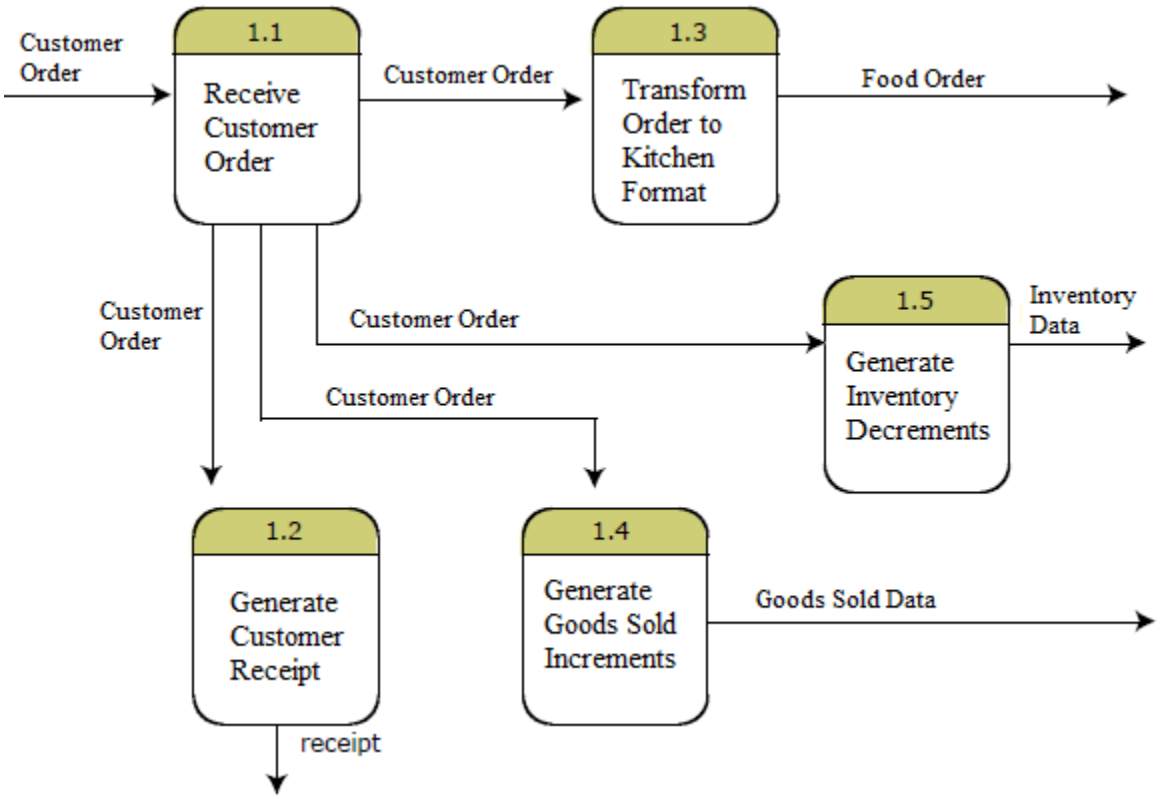


Figure: Level-1 Diagram Showing Decomposition of Process 1.0 from the Level-0 Diagram

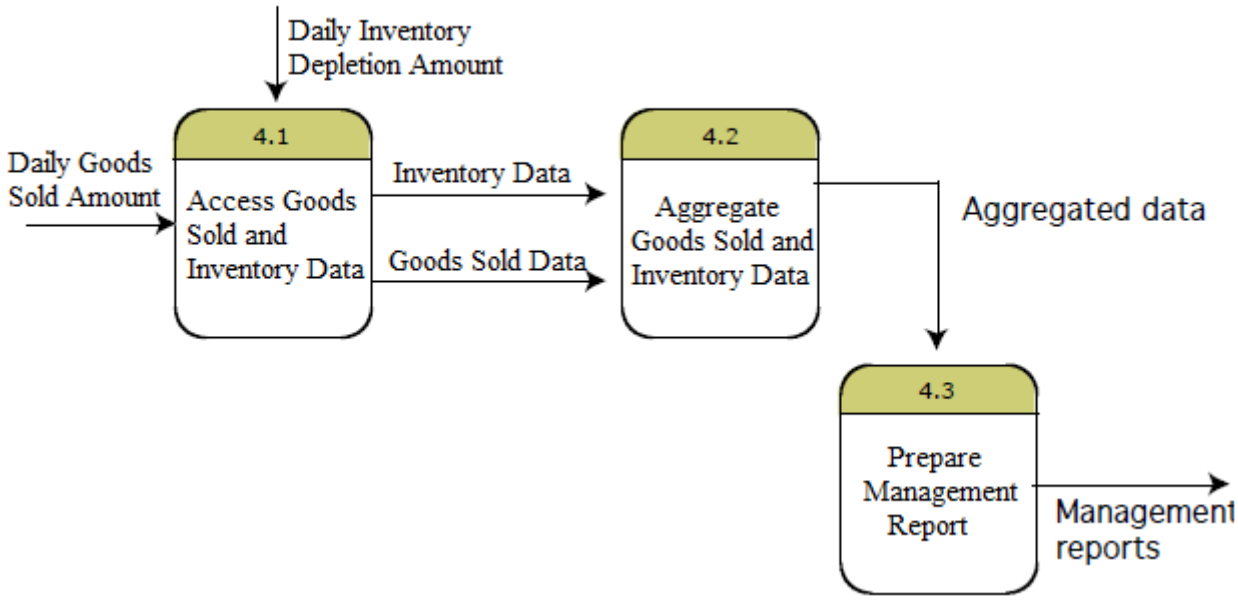


Figure: Level-1 Diagram Showing the Decomposition of Process 4.0 from the Level-0 Diagram

b i i z a y @ g m a i l . c o m

Level-n DFD:
Level-n DFD shows the sub-processes of one of the processes in the Level n-1 DFD. It shows all processes that comprise a single process on the level 1 diagram. It also shows how information moves from and to each of these processes. Level 2 diagrams may not be needed for all level 1 processes. Correctly numbering each process helps the user understand where the process fits into the overall system.

Note: Processes are labeled 4.3.1, 4.3.2, etc. If this is the lowest level of the hierarchy, it is called a primitive DFD. Sources and sinks are optional on level-n diagrams.

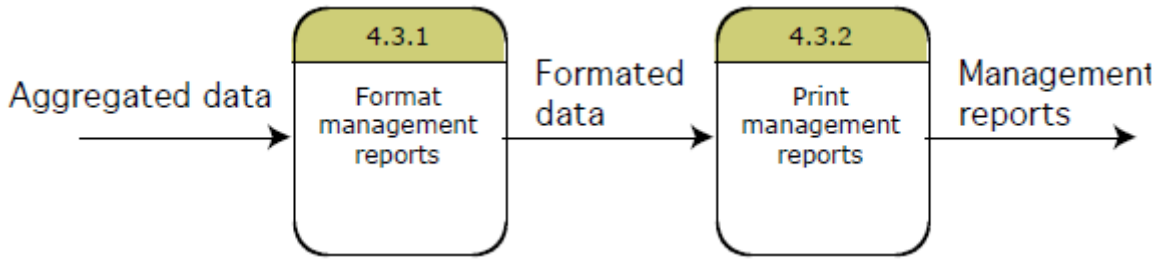
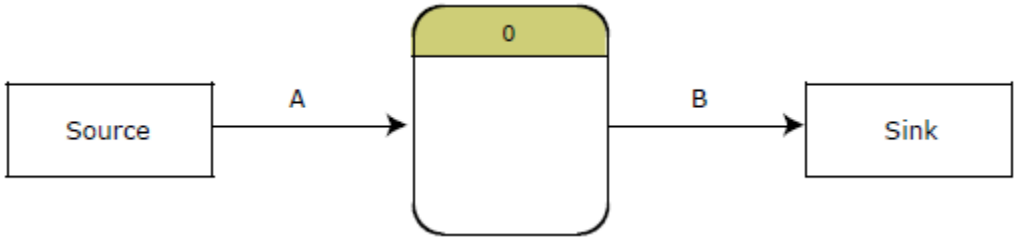


Figure: Level-2 Diagram Showing the Decomposition of Process 4.3 from the Level-1 Diagram for Process 4.0

DFD Balancing:
When decomposing a DFD, you must conserve inputs to and outputs from a process at the next level of decomposition. This conservation of inputs and outputs is called ***balancing***.

Balanced means:

- ✓ Number of inputs to lower level DFD equals number of inputs to associated process of higher-level DFD
- ✓ Number of outputs to lower level DFD equals number of outputs to associated process of higher-level DFD



b i i z a y @ g m a i l . c o m

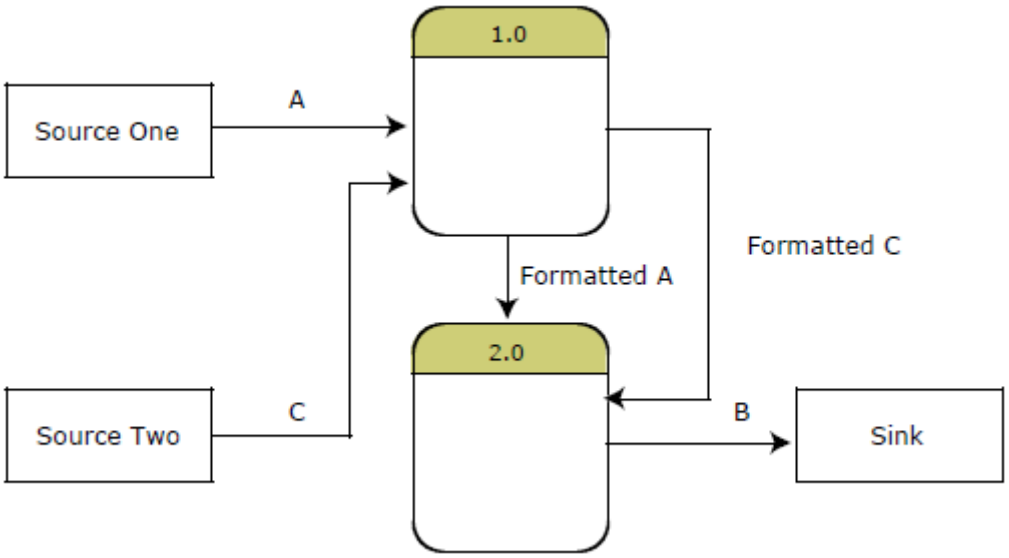
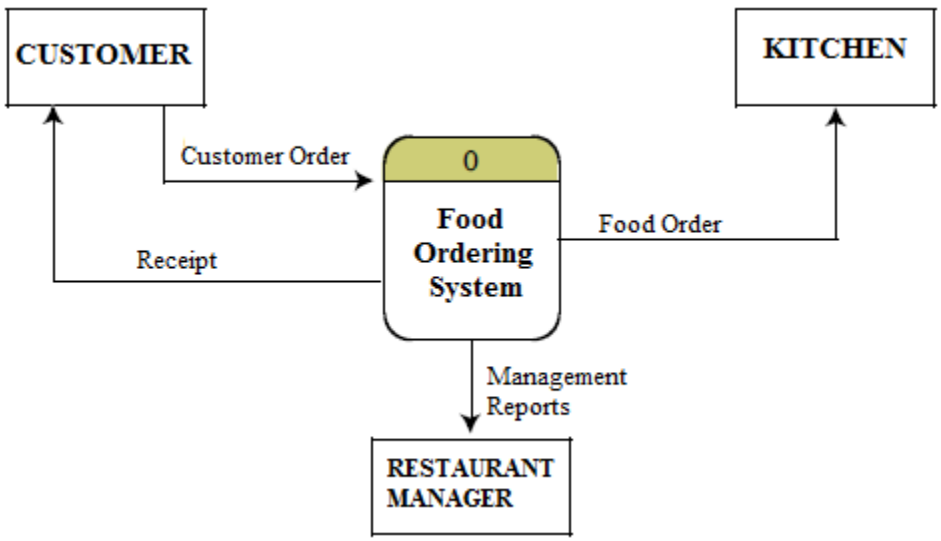
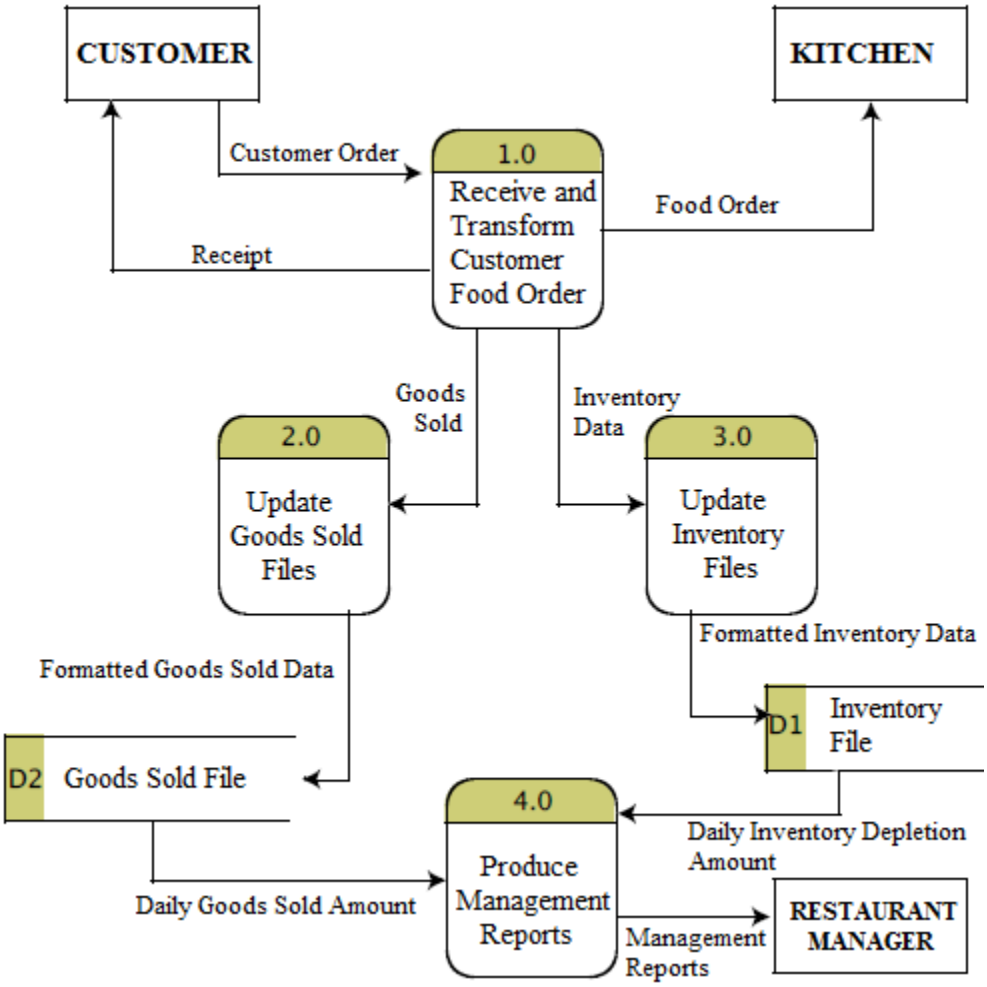


Figure: An unbalanced set of data flow diagrams – Figure A: Context Diagram and Figure B: Level-1 DFD

The figure above is unbalanced because the process of the context diagram has only one input but the Level-0 diagram has two inputs.



b i i z a y @ g m a i l . c o m



These are balanced because the numbers of inputs and outputs of context diagram process equal the number of inputs and outputs of Level-0 diagram.

When to stop decomposing DFDs?

Ideally, a DFD has at least three processes and no more than seven to nine. Some other points may be:

- ✓ When each process has been reduced to a single decision, calculation or database operation
- ✓ When each data store represents data about a single entity
- ✓ When the system user does not care to see any more detail
- ✓ When every data flow does not need to be split further to show that data are handled in various ways
- ✓ When you believe that you have shown each business form or transaction, online display and report as a single data flow
- ✓ When you believe that there is a separate process for each choice on all lowest-level menu options

How broad should the DFD be? (How many processes should be on a level?)

7 ± 2 is a reasonable heuristic.

How deep should be the DFD be? (How many levels should a DFD have?)
If the process has only one input or one output, probably cannot partition further.

Logical DFD vs. Physical DFD

Data flow diagrams are categorized as either logical or physical. A logical data flow diagram focuses on the business and how the business operates. It is not concerned with how the system will be constructed.

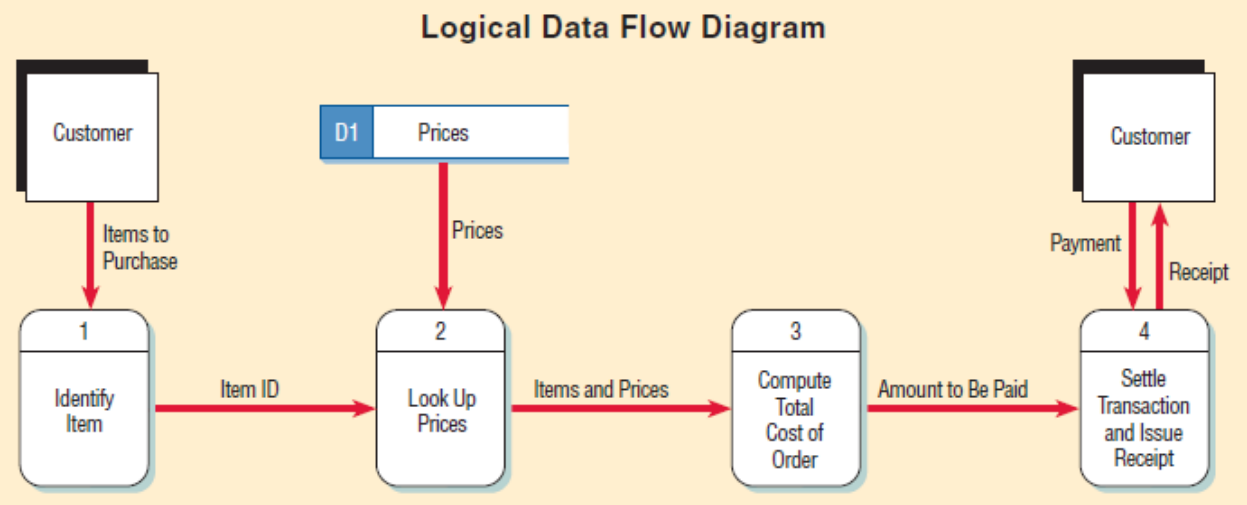
A physical data flow diagram shows how the system will be implemented, including the hardware, software, files, and people in the system. It is developed such that the processes described in the logical data flow diagrams are implemented correctly to achieve the goal of the business.

A logical DFD makes it easier to communicate for the employees of an organization, leads to more stable systems, allows for better understanding of the system by analysts, is flexible and easy to maintain, and allows the user to remove redundancies easily. On the other hand, a physical DFD is clear on division between manual and automated processes, gives detailed description of processes, identifies temporary data stores, and adds more controls to make the system more efficient and simple.

A logical DFD captures the data flows that are necessary for a system to operate. It describes the processes that are undertaken, the data required and produced by each process, and the stores needed to hold the data. On the other hand, a physical DFD shows how the system is actually implemented, either at the moment (Current Physical DFD), or how the designer intends it to be in the future (Required Physical DFD). Thus, a Physical DFD may be used to describe the set of data items that appear on each piece of paper that move around an office, and the fact that a particular set of pieces of paper are stored together in a filing cabinet.

Design Feature	Logical	Physical
What the model depicts	How the business operates.	How the system will be implemented (or how the current system operates).
What the processes represent	Business activities.	Programs, program modules, and manual procedures.
What the data stores represent	Collections of data regardless of how the data are stored.	Physical files and databases, manual files.
Type of data stores	Show data stores representing permanent data collections.	Master files, transition files. Any processes that operate at two different times must be connected by a data store.
System controls	Show business controls.	Show controls for validating input data, for obtaining a record (record found status), for ensuring successful completion of a process, and for system security (example: journal records).

Table: Features common to both logical and physical data flow diagrams.



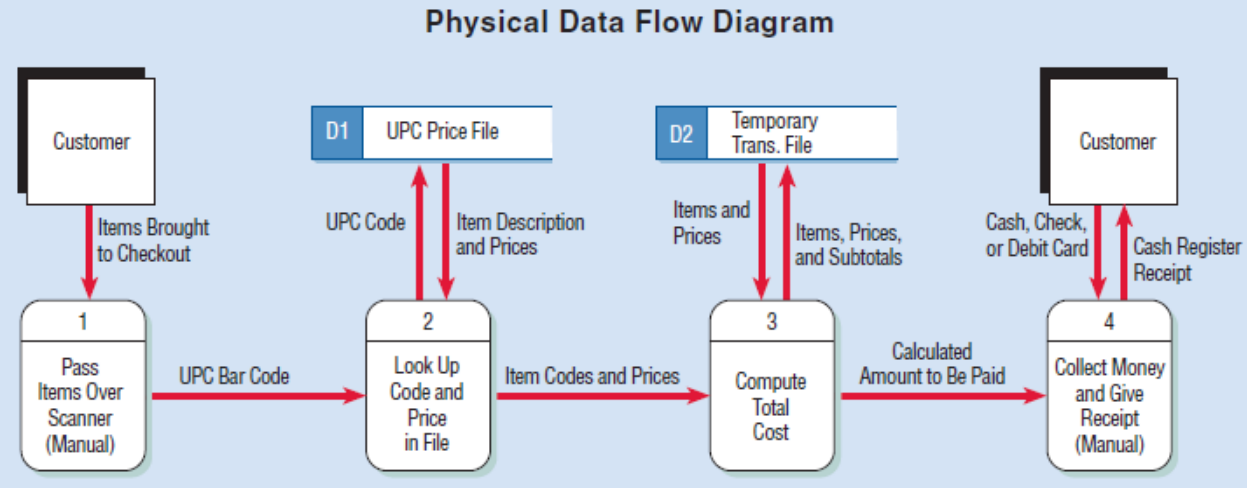


Figure: The physical data flow diagram (below) shows certain details not found on the logical data flow diagram (above).

Advantages of Logical Data Flow Diagram

1. *Better communication:* A logical diagram is easier to use when communicating with the users of a system, because it is centered on business activities.
2. *More stable system:* Systems formed using a logical data flow diagram are often more stable than those that are not because they are based on business events and not on a particular technology or method of implementation.
3. *Better understanding of business:* A logical data flow diagram has a business emphasis and helps the analyst to understand the business being studied, to grasp why procedures are performed, and to determine the expected result of performing a task.
4. *Flexibility and maintenance:* The new system will be more flexible and easier to maintain if its design is based on a logical model and the business functions are not subject to frequent change. Physical aspects of the system change more frequently than do business functions.
5. *Simple to develop:* The logical model is easy to create and simpler to use because it does not often contain data stores other than files or a database.
6. *Easy conversion to physical model:* Once the logical data flow diagrams are created, the creation of physical data flow diagrams becomes easy.

Advantages of Physical Data Flow Diagram

1. *Clarifying which processes are manual and which are automated:* Manual processes require detailed documentation and automated process require computer programs to be developed.
2. *Describing processes in more detail than do logical DFDs:* Describes all steps for processing of data.
3. *Sequencing processes that have to be done in a particular order:* Sequence of activities that lead to a meaningful result are described. For example, update must be performed before a producing a summary report.
4. *Identifying temporary data storage:* Temporary storage such as a sales transaction file for a customer receipt (report) in a grocery store, are described.
5. *Specifying actual names of files and printouts:* Logical data flow diagrams describes actual filenames and reports, so that the programmers can relate those with the data dictionary during the developmental phase of the system.
6. *Adding controls to ensure the processes are done properly:* These are conditions or validations of data that are to be met during input, update, delete, and other processing of data.

Advantages of DFDs

1. Since they are easily understood by users (i.e. presented in a non-technical format), they are easier to validate for correctness. It is therefore easy to determine whether requirements are correct. The probability of a better system is increased.
2. It is argued that a picture can convey meaning more quickly than more traditional methods such as textual narrative.
3. DFDs allow the analyst to abstract to whatever level of detail is required so that it is possible to examine a system in overview and at a more detailed level whilst still keeping sight of the links and interfaces among the different levels.
4. The DFD specifies the system at a logical level rather than a physical level. This means that it shows what the system will do rather than how it will be done. The benefit of this separation is that the users can specify their requirements without any restriction being imposed of a design nature, for example, the exact hardware requirements. Therefore, an independence exists between logical and physical implementation which means that changes or upgrades can be made to the hardware without affecting the functions of the system.
5. Further understanding of the inter-relatedness of systems and subsystems are developed.
6. Provides a means of analysis of a proposed system to determine if the necessary data and processes have been defined.
7. It gives further understanding of the interestedness of the system and sub-systems
8. It is useful from communicating current system knowledge to the user
9. Used as part of the system documentation files
10. Data flow diagram helps to substantiate the logic underlining the dataflow of the organization
11. It gives the summary of the system
12. DFD is very easy to follow errors and it is also useful for quick reference to the development team for locating and controlling errors

Disadvantages of DFDs

1. For large systems it can be a time consuming and complex task to produce all necessary levels of DFDs.
2. It can be difficult to read and understand/appreciate what is going on at a first glance.
3. The symbols used are not common to all DFDs. Different models use different symbols for the structure of a DFD.
4. DFD is likely to take many alteration before agreement with the user
5. Physical consideration are usually left out
6. It is difficult to understand because it ambiguous to the user who have little or no knowledge

Conceptual Data Modeling

Conceptual data model is a detailed model that captures the overall structure of data in an organization. It is independent of any database management system (DBMS) or other implementation considerations. Conceptual data modeling is the representation of organizational data. Its purpose is to show rules about the meaning and interrelationships among data. Entity-Relationship (E-R) diagrams are commonly used to show how data are organized. The main goal of conceptual data modeling is to create accurate E-R diagrams. Methods such as interviewing, questionnaires, and JAD are used to collect information. Consistency must be maintained among process flow, decision logic, and data modeling descriptions.

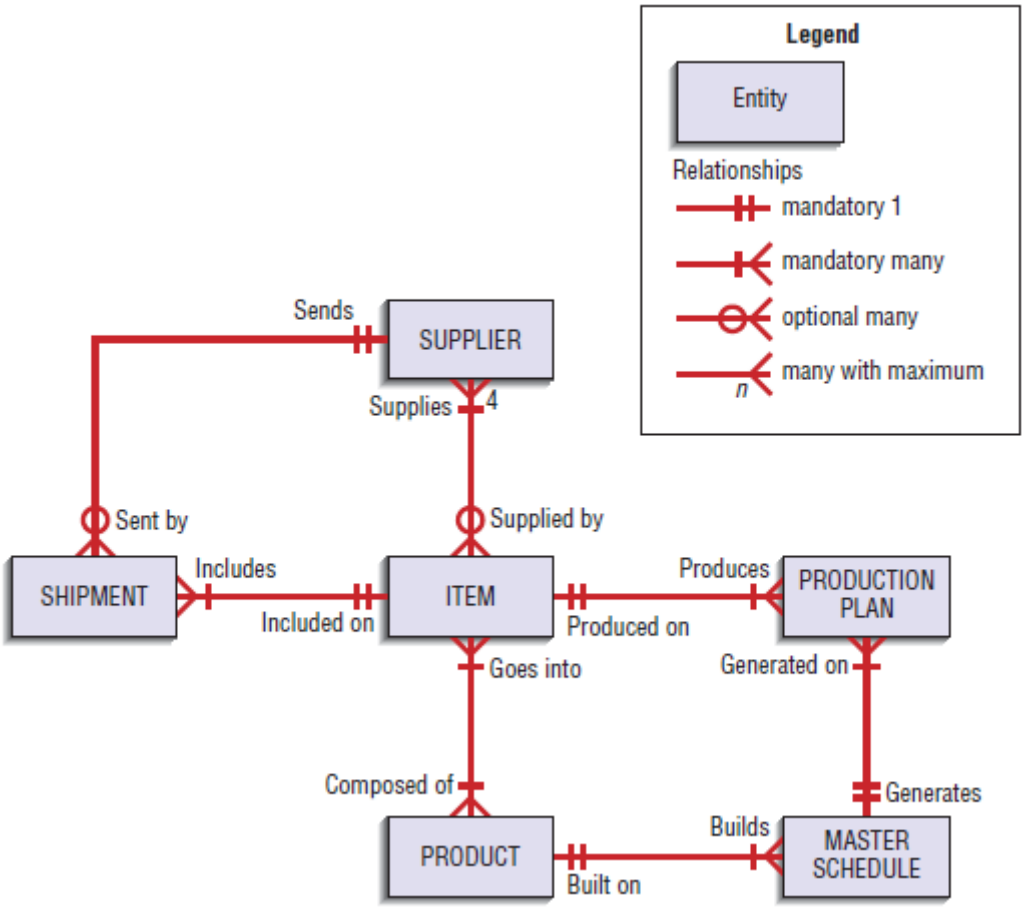


Figure: Sample conceptual data model diagrams: Standard E-R notation.

b i i z a y @ g m a i l . c o m

The Process of Conceptual Data Modeling

First step is to develop a data model for the system being replaced. Next, a new conceptual data model is built that includes all the requirements of the new system. In the design stage, the conceptual data model is translated into a physical design. Project repository links all design and data modeling steps performed during SDLC. Primary deliverable is the entity-relationship diagram.

Introduction to Entity-Relationship Modeling

The basis entity-relationship modeling notation uses three main constructs: Data Entities, Relationships, and Attributes.

Entity-Relationship Data Model

E-R data model is a detailed, logical, and graphical representation of the entities, associations and data elements for an organization or business area.

Entity-Relationship (E-R) Diagram

An entity-relationship diagram (or, E-R diagram) is a detailed, logical, and graphical representation of the data for an organization or business area. It is a graphical representation of an E-R model. ERD illustrate the logical structure of databases. Peter Chen developed ERDs in 1976. ERD is a picture showing the information created, stored, and used by a business system.

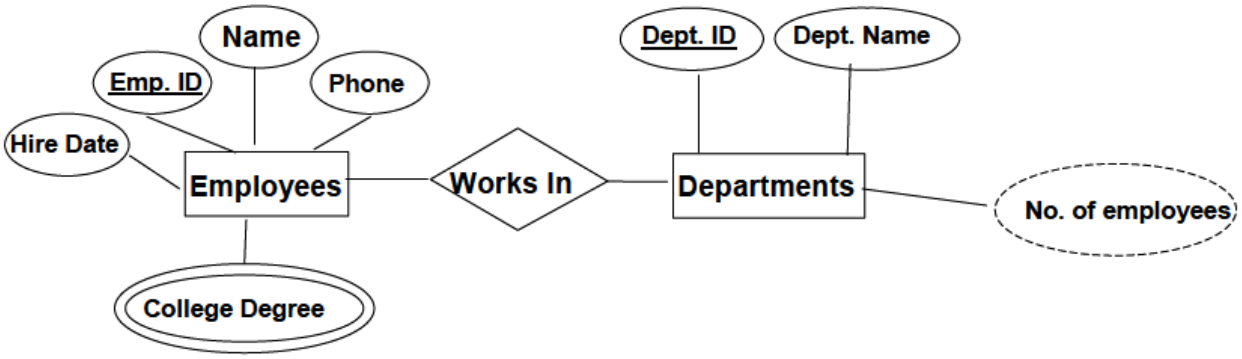


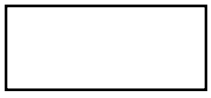
Figure: Example of an ERD

ERD complements DFD. While DFD focuses on processes and data flow between them, ERD focuses on data and the relationships between them. It helps to organize data used by a system in a disciplined way. It helps to ensure completeness, adaptability and stability of data.

It is an effective tool to communicate with senior management (what is the data needed to run the business), data administrators (how to manage and control data), database designers (how to organize data efficiently and remove redundancies).

Entities

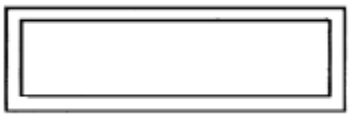
An entity is a person, place, object, event or concept in the user environment about which the organization wishes to maintain data. It is represented by a rectangle in E-R diagrams. An entity has its own identity, which distinguishes it from every other entity.



Example:
Person: EMPLOYEE, STUDENT, PATIENT
Place: STATE, REGION, COUNTRY, BRANCH
Object: MACHINE, BUILDING, AUTOMOBILE, PRODUCT
Event: SALE, REGISTRATION, RENEWAL
Concept: ACCOUNT, COURSE, WORK CENTER

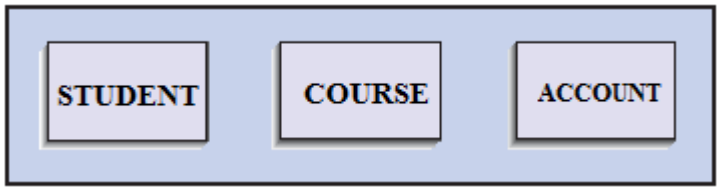
Weak Entity

A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone.



Entity Type

Entity type is a collection of entities that share common properties or characteristics.



Entity Instance

Entity instance is a single occurrence of an entity type. For example, most organizations have one STUDENT (entity type), but hundreds (or even thousands) of instances of this entity type may be stored in the database.

Attributes of STUDENT Entity Type →

Entity Instances →

Student ID	Last Name	First Name
2144	Mishra	Bijay
3122	Mishra	Jijibisha
3843	Mishra	Babita
9844	Mishra	Ajay
2837	Mishra	Sanjay
5123	Mishra	Manju
2293	Mishra	Govinda

b i i z a y @ g m a i l . c o m

Attributes

An attribute is a property or characteristic of an entity that is of interest to the organization. Ovals (or ellipses) are used to represent attributes. Lines links attributes to entity sets and entity sets to relationship sets. We use nouns with an initial capital letter followed by lowercase letters in naming an attribute.

Following are some typical entity types and associated attributes:
STUDENT: Student_ID, Student_Name, Address, Phone_Number, Major
AUTOMOBILE: Vehicle_ID, Color, Weight, Horsepower
EMPLOYEE: Employee_ID, Employee_Name, Address, Skill

Key Attribute (Identifier)

A key attribute is the unique, distinguishing characteristic of the entity. An identifier is a candidate key that has been selected as the unique identifying characteristic for an entity type. One or more attributes can serve as the entity identifier, uniquely identifying each entity instance. For example, an employee's social security number might be the employee's key attribute.

Candidate Keys

Each entity type must have an attribute or set of attributes that distinguishes one instance from other instances of the same type. A candidate key is an attribute (or combination of attributes) that uniquely identifies each instance of an entity type. A candidate key for a STUDENT entity type might be Student_ID.

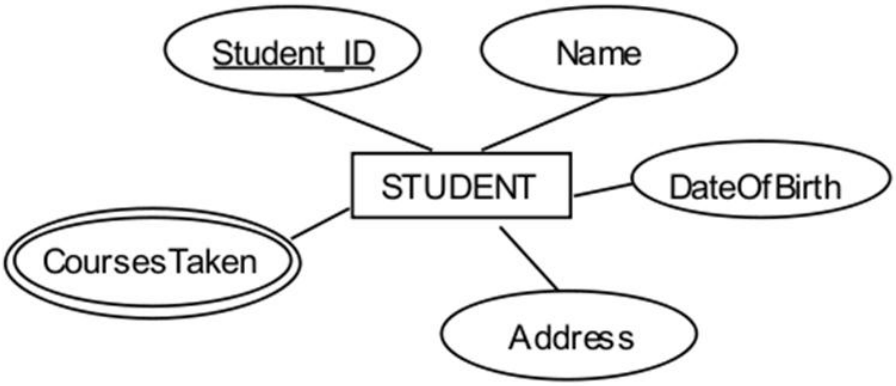


Figure**:** Entity, Attributes, and Multivalued Attributes

Multivalued Attribute

A multivalued attribute is an attribute that may take on more than one value for each entity instance. For example, an employee entity can have multiple ***skill*** values. It is represented on E-R diagram by double-lined ellipse

b i i z a y @ g m a i l . c o m

Composite attribute

Composite attributes are made up of simple attributes. Example: Address(Street, City, State, Zip) and Name (FirstName, MiddleName, LastName).

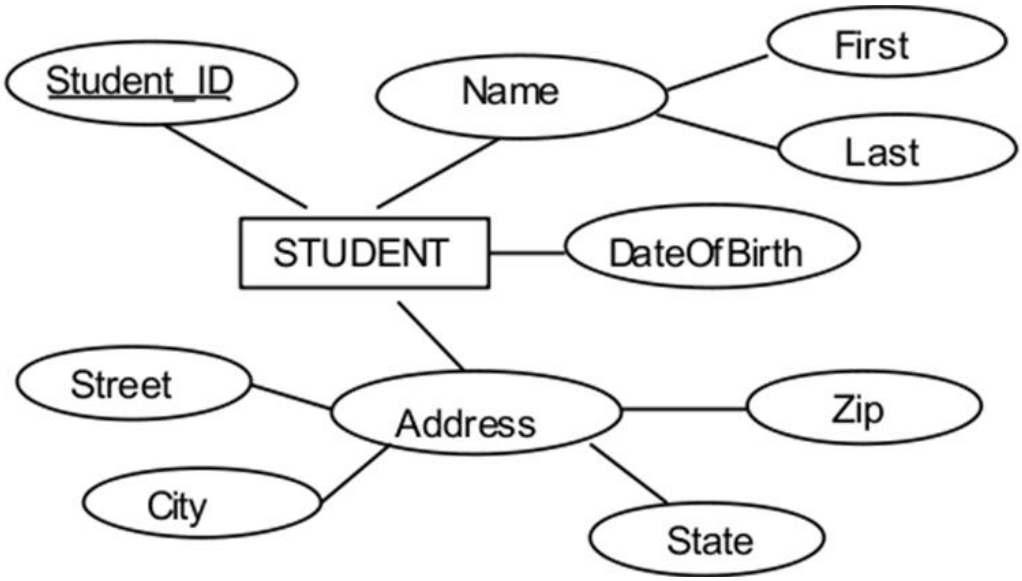
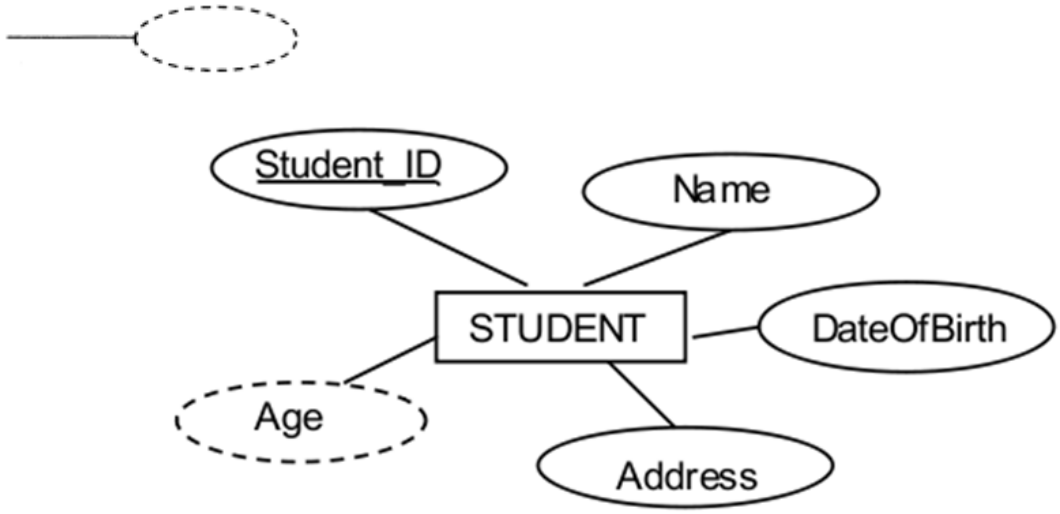


Figure: Example of composite attribute

Derived attribute

A derived attribute is based on another attribute. For example, an employee's “monthly salary” is based on the employee's “annual salary”. Similarly, “Age” can be derived from “Date_of_birth” attribute of an employee. It is denoted as:



b i i z a y @ g m a i l . c o m

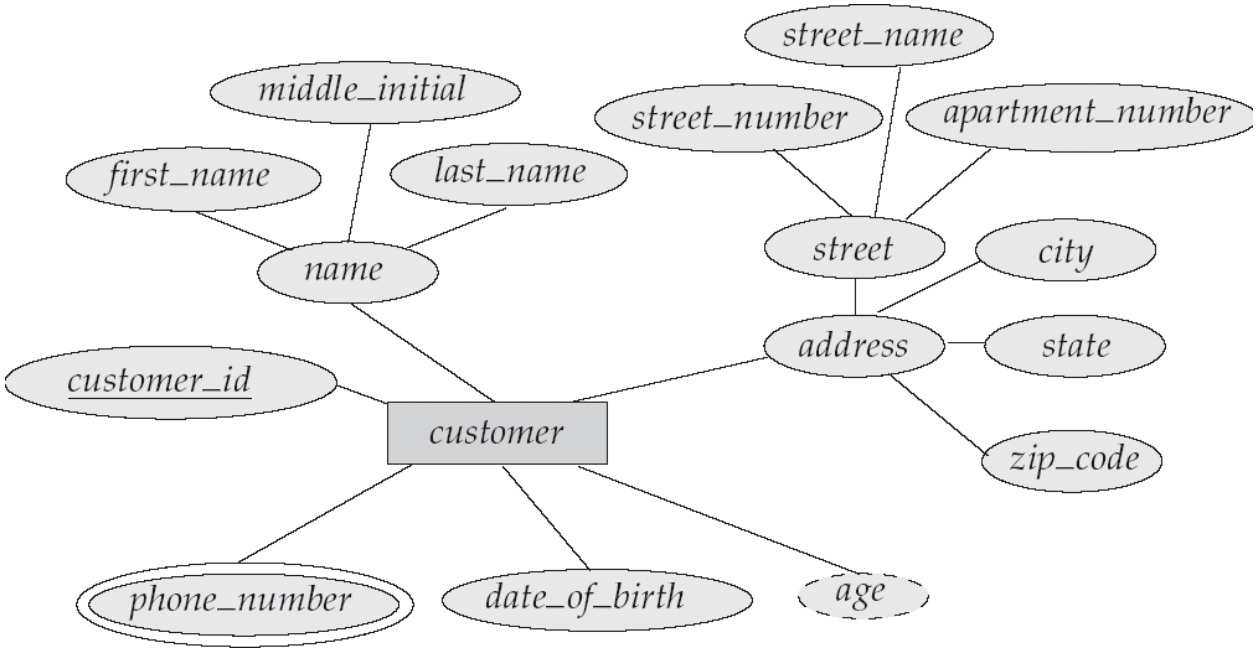
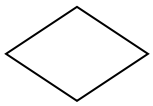


Figure: Composite Attributes, Derived Attributes and Multi-valued Attributes

Relationship

Relationships are the glue that hold together the various components of an E-R model. It is an association between the instances of one or more entity types that is of interest to the organization. Association indicates that an event has occurred or that there is a natural link between entity types. The first entity in the relationship is the parent entity; the second entity in the relationship is the child entity. Relationships go in both directions. Relationships are always labeled with verb phrases. Diamonds are normally used to represent relationships.



Conceptual Data Modeling and the E-R Model

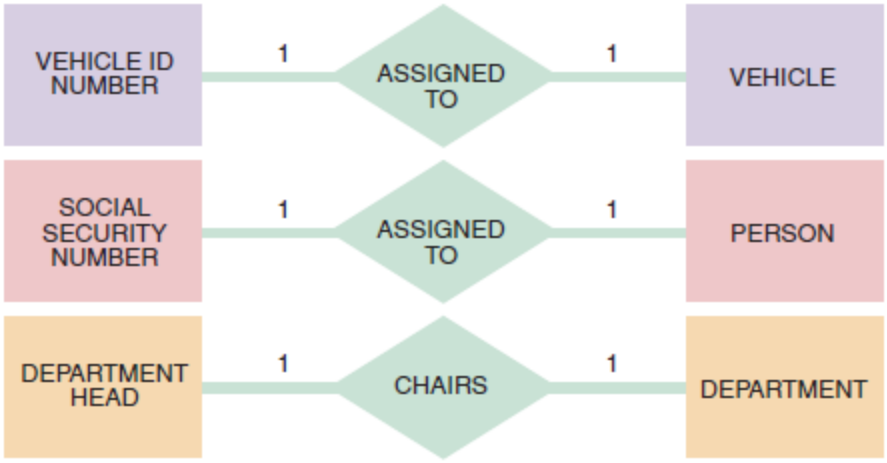
The goal of conceptual data modeling is to capture as much of the meaning of data as possible. The more details about data that we can model, the better the system we can design and build.

Types of Relationship

Three types of relationships can exist between entities: one-to-one, one-to-many, and many-to-many.

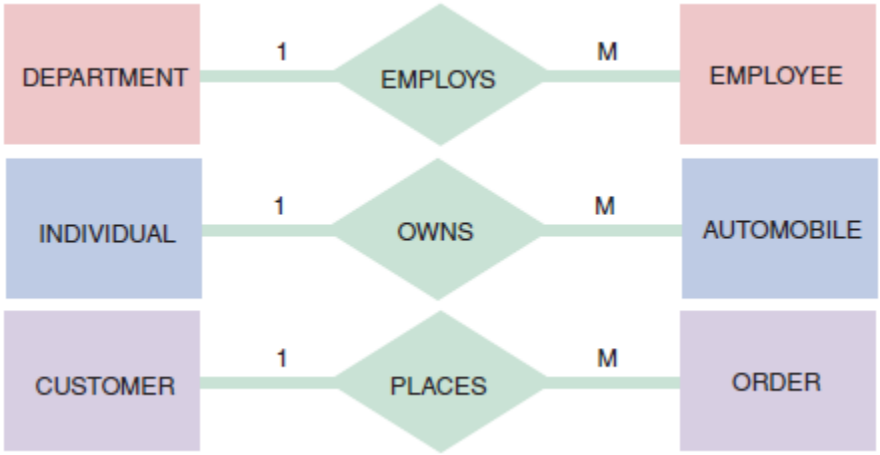
One-to-One Relationship

A one-to-one relationship, abbreviated 1:1, exists when exactly one of the second entity occurs for each instance of the first entity. Figure below shows examples of several 1:1 relationships. A number 1 is placed alongside each of the two connecting lines to indicate the 1:1 relationship.



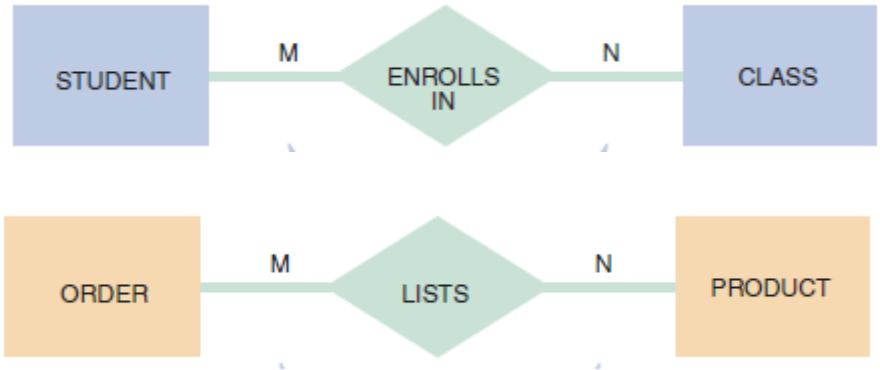
One-to-Many Relationship

A one-to-many relationship, abbreviated 1:M, exists when one occurrence of the first entity can relate to many instances of the second entity, but each instance of the second entity can associate with only one instance of the first entity. The line connecting the many entity is labeled with the letter M, and the number 1 labels the other connecting line.



Many-to-Many Relationship

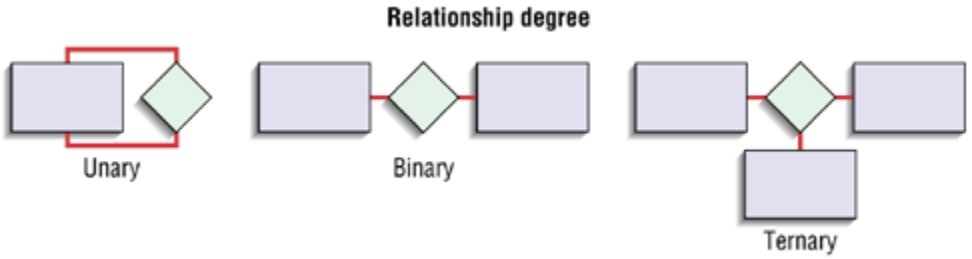
A many-to-many relationship, abbreviated M:N, exists when one instance of the first entity can relate to many instances of the second entity, and one instance of the second entity can relate to many instances of the first entity. One of the connecting lines is labeled with the letter M, and the letter N labels the other connection.



b i i z a y @ g m a i l . c o m

Degree of Relationship

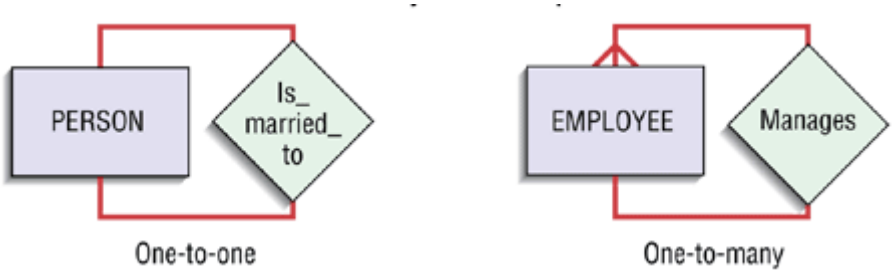
The degree of a relationship is the number of entity types that participate in a relationship. The three most common relationships in E-R diagrams are unary (degree one), binary (degree two), and ternary (degree three).



Unary Relationship

- ✓ A relationship between the instances of one entity type
- ✓ Also called as recursive relationship

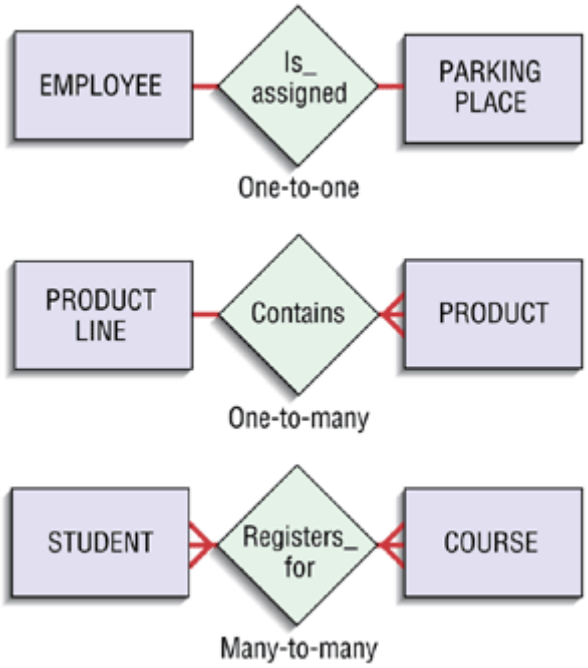
Example:



Binary Relationship

- ✓ A relationship between the instances of two entity types
- ✓ Most common type of relationship among all

Example:

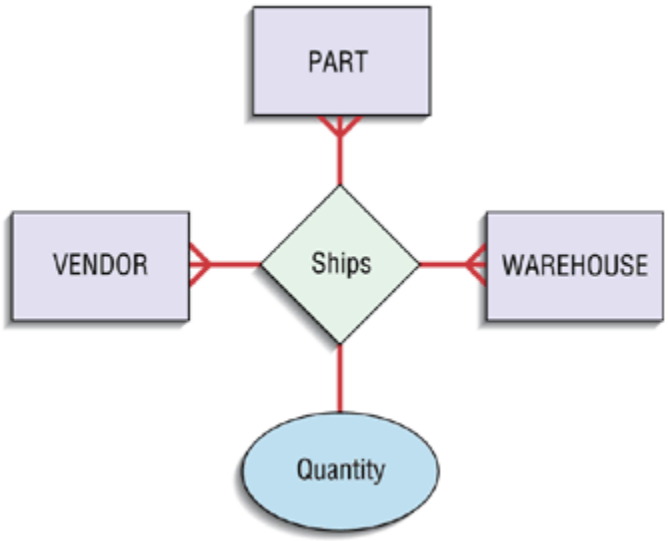


b i i z a y @ g m a i l . c o m

Ternary Relationship

- ✓ A simultaneous relationship among the instances of three entity types
- ✓ Not the same as three binary relationships

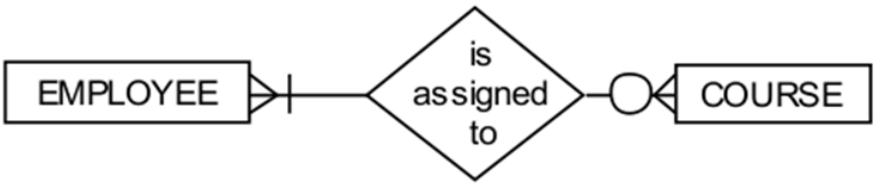
Example:



Cardinalities in Relationships

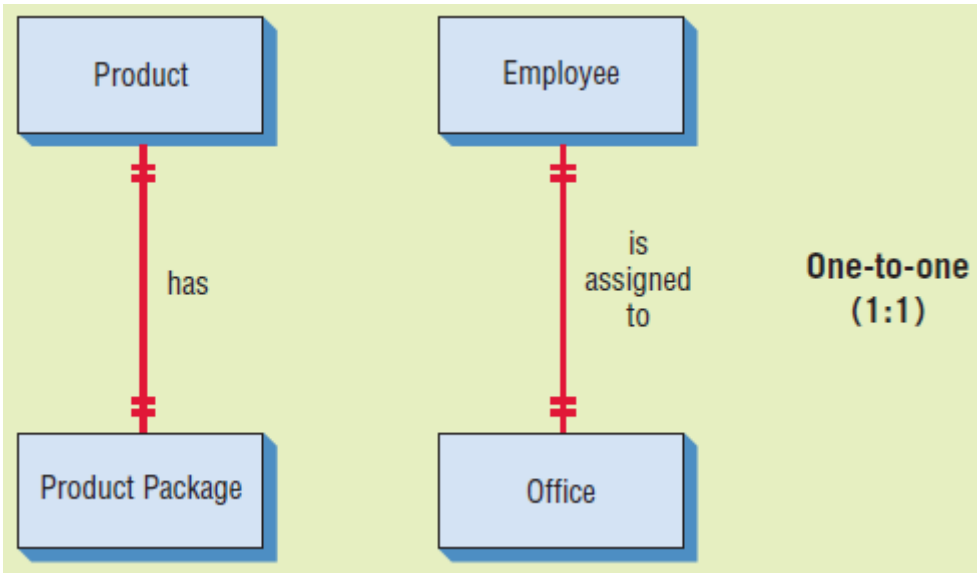
The cardinality of a relationship is the number of instances of entity B that can be associated with each instance of entity A. It refers to the number of times instances in one entity can be related to instances in another entity.

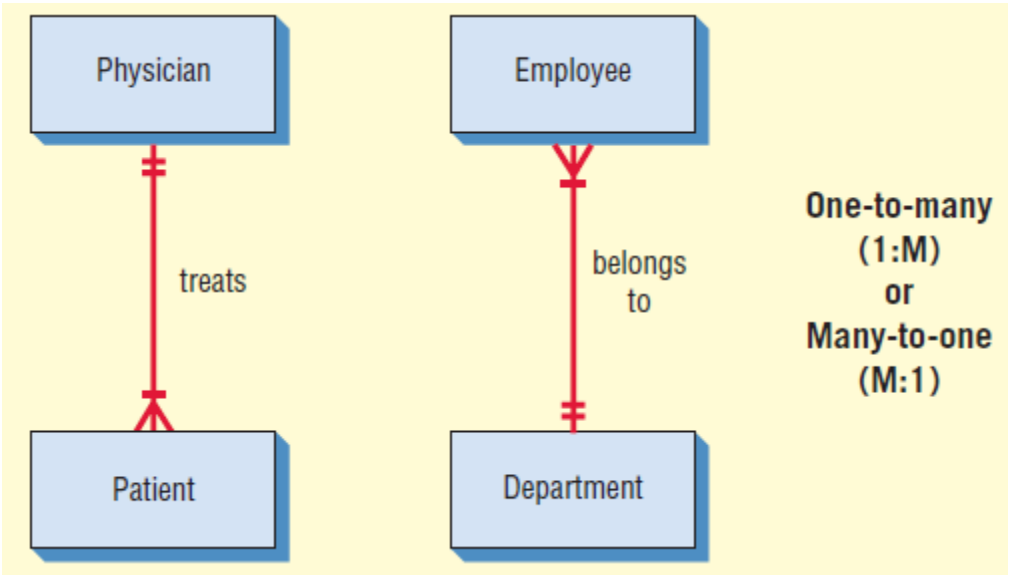
Mandatory cardinalities

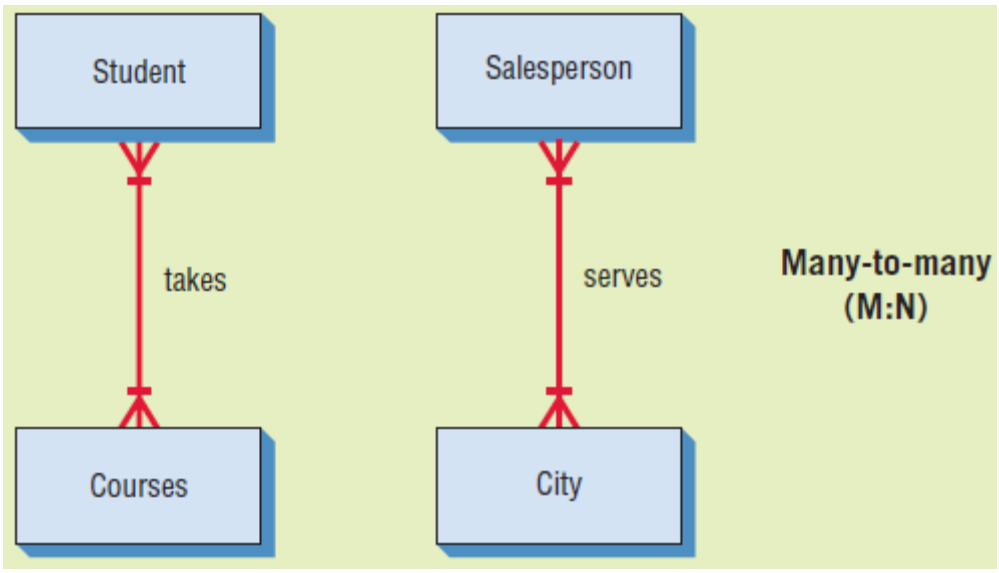


One optional, One mandatory cardinality

b i i z a y @ g m a i l . c o m







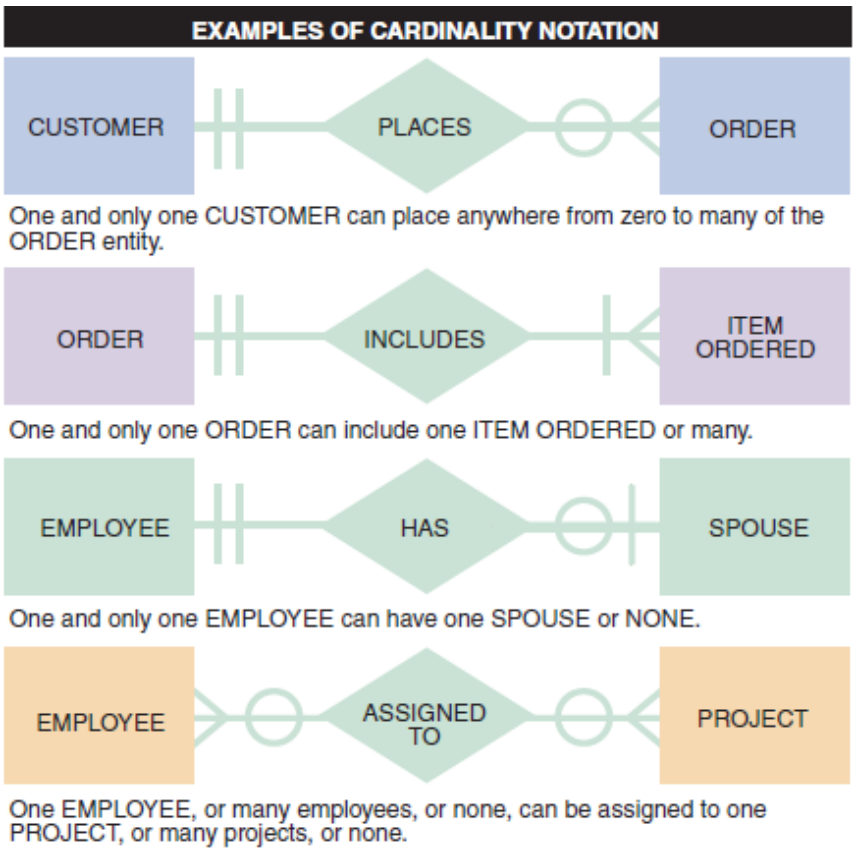
b i i z a y @ g m a i l . c o m

	To 1 relationship	Exactly one
	To many relationship	One or more
	To 0 or 1 relationship	Only zero or one
	To 0 or more relationship	Can be zero, one, or more
	To more than 1 relationship	Greater than one

Modality

Modality refers to the minimum number of times that an instance in one entity can be related to an instance in another entity.

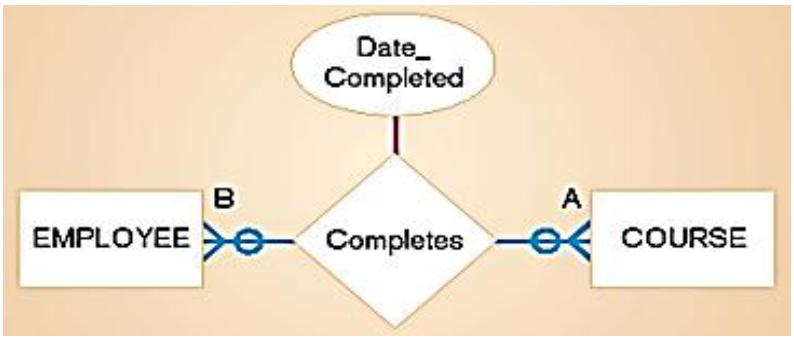
- ✓ One means that an instance in the related entity must exist for an instance in another entity to be valid
- ✓ Zero means that no instance in the related entity is necessary for an instance in another entity to be valid



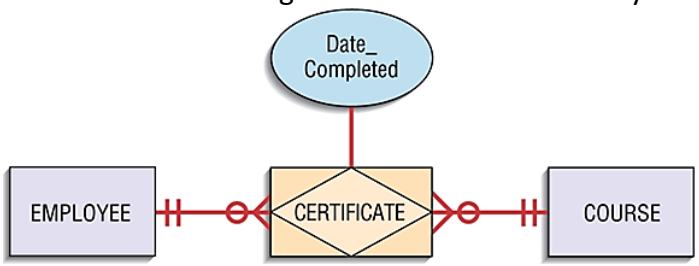
b i i z a y @ g m a i l . c o m

Associative Entities

An associative entity is 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. It is also called as ***gerund***. It is used to join two entities.

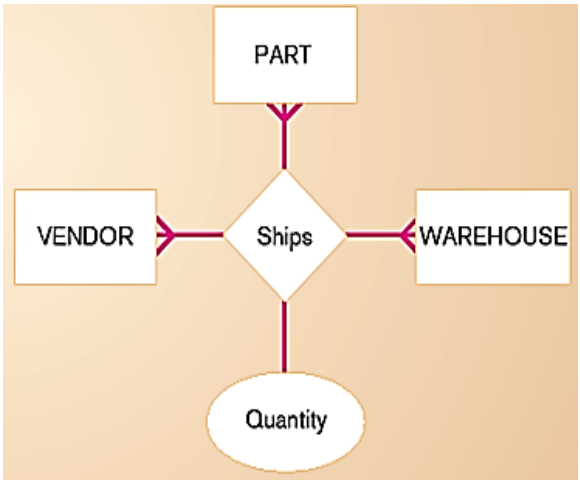


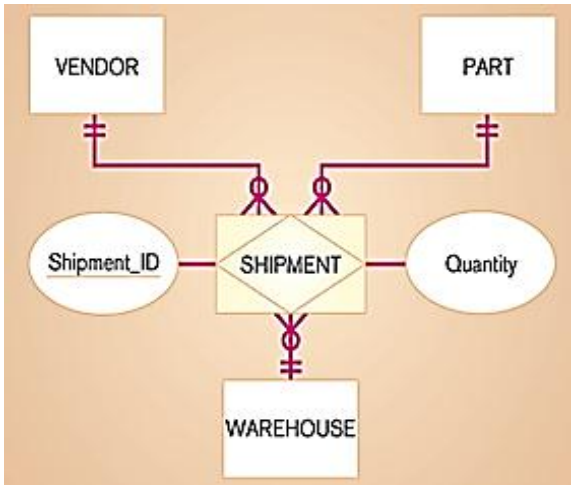
A relationship with an attribute in above figure as an associative entity is shown below:



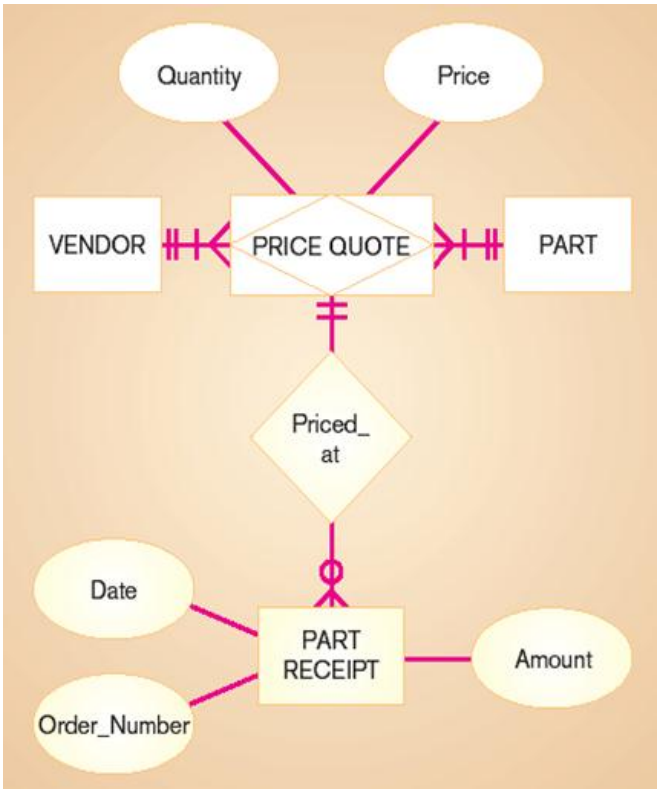
A relationship that itself is related to other entities via another relationship must be represented as an associative entity.

Some other examples of associative entities:





b i i z a y @ g m a i l . c o m



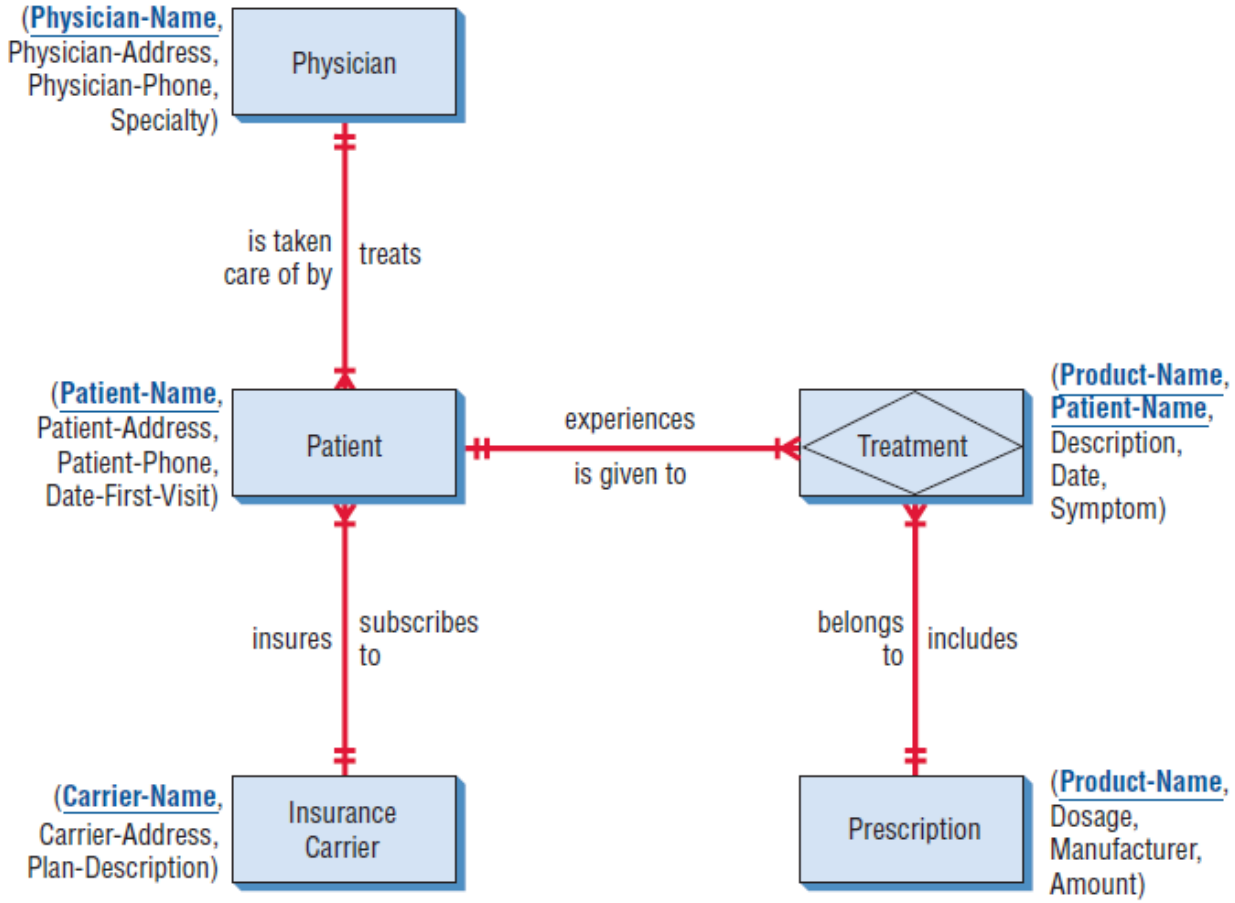


Figure: The entity-relationship diagram for patient treatment. Attributes can be listed alongside the entities. In each case, the key is underlined.

b i i z a y @ g m a i l . c o m

Steps for Designing the ERD

1. Identify Entities	Identify the roles, events, locations, tangible things or concepts about which the end-users want to store data.
2. Find Relationships	Find the natural associations between pairs of entities using a relationship matrix.
3. Draw Rough ERD	Put entities in rectangles and relationships on line segments connecting the entities.
4. Fill in Cardinality	Determine the number of occurrences of one entity for a single occurrence of the related entity.
5. Define Primary Keys	Identify the data attribute(s) that uniquely identify one and only one occurrence of each entity.
6. Draw Key-Based ERD	Eliminate Many-to-Many relationships and include primary and foreign keys in each entity.
7. Identify Attributes	Name the information details (fields) which are essential to the system under development.
8. Map Attributes	For each attribute, match it with exactly one entity that it describes.
9. Draw fully attributed ERD	Adjust the ERD from step 6 to account for entities or relationships discovered in step 8.
10. Check Results	Does the final Entity Relationship Diagram accurately depict the system data

Advantages of ERD

1. Straightforward relation representation: Having designed an E-R diagram for a database application, the relational representation of the database model becomes relatively straightforward.
2. Easy conversion for E-R to other data model: Conversion from E-R diagram to a network or hierarchical data model can· easily be accomplished.
3. Graphical representation for better understanding: An E-R model gives graphical and diagrammatical representation of various entities, its attributes and relationships between entities. This is turn helps in the clear understanding of the data structure and in minimizing redundancy and other problems.
4. Conceptual simplicity
5. Effective communication
6. Integration with the relational database model
7. Gives a higher level description of the system
8. Intuitive and helps in Physical Database creation

Disadvantages of ERD

1. No industry standard for notation: There is no industry standard notation for developing an E-R diagram.
2. Popular for high-level design: The E-R data model is especially popular for high level.
3. Limited constraint representation
4. Limited relationship representation
5. No representation of data manipulation

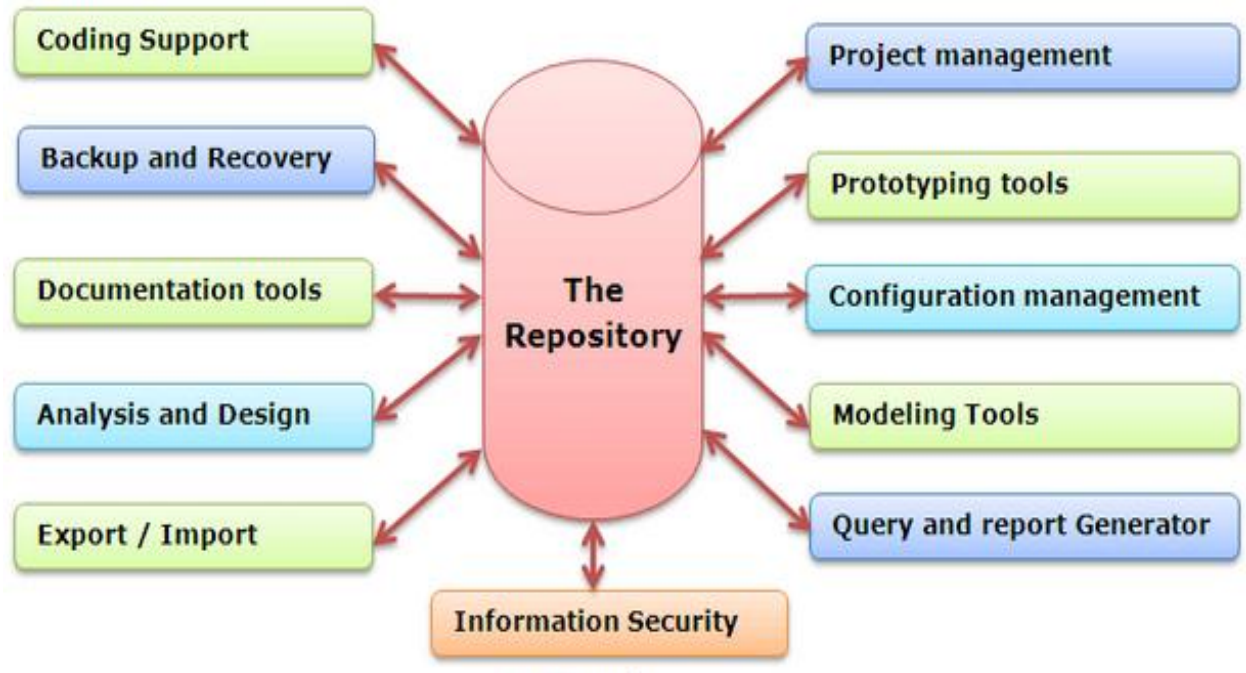
6. Loss of information
7. Physical design derived from E-R Model may have some amount of ambiguities or inconsistency
8. Sometime diagrams may lead to misinterpretations

CASE Tool:

Computer-aided software engineering (CASE) is the scientific application of a set of tools and methods to a software system which is meant to result in high-quality, defect-free, and maintainable software products. It also refers to methods for the development of information systems together with automated tools that can be used in the software development process.

C - Computer
A - Aided, Assisted, Automated
S - System, Software
E – Engineering

Computer Aided Software Engineering (CASE) is a way that enables you to spend maximum time designing software and minimum time trying to work out why it will not work the way you want it to. CASE support each phase of the system development life cycle with a set of labor saving tools. CASE is a category of software tools which aid a developer to create and maintain software. CASE is intended to accelerate the process of developing systems and to improve the quality of the resulting systems. CASE is not a methodology or an alternative to methodologies. CASE is an enabling technology that supports a methodology's preferred strategies, techniques, and deliverables.



CASE Tools

b i i z a y @ g m a i l . c o m

CASE can be used to mean any computer-based tool for software planning, development, and evolution. Various people regularly call the following 'CASE': Structured Analysis (SA), Structured Design (SD), Editors, Compilers, Debuggers, Edit-Compile-Debug environments, Code Generators, Documentation Generators, Configuration Management, Release Management, Project Management, Scheduling, Tracking, Requirements Tracing, Change Management (CM), Defect Tracking, Structured Discourse, Documentation editing, Collaboration tools, Access Control, Integrated Project Support Environments (IPSEs), Reverse Engineering, Metric Analyzers.

CASE software supports the software process activities such as requirement engineering, design, program development and testing. Almost all the phases of the software development life cycle are supported by them such as analysis; design, etc., including umbrella activities such as project management, configuration management etc. In general, standard software development methods such as structured system analysis and design method are also supported by CASE tools.

CASE also refers to the methods dedicated to an engineering discipline for the development of information system using automated tools. CASE is mainly used for the development of quality software which will perform effectively. The objective of introducing Computer Aided Software Engineering (CASE) tools is to reduce the time of development, reduce the cost of software, and the enhancement of the quality of the software. CASE technologies are tools that provide automated assistance for software development to the developers.

Whenever a new system is installed, the implementation integrates a number of related and different tasks. The process has to be efficiently organized and it is for this very reason that CASE tools are developed. With the help of CASE, the installation process can be automated and coordinated within the developed and adopted system life cycle.

CASE has three major components:

1. Methodologies
2. Techniques
3. Tools

Methodologies:

Methodologies provide the organizers framework for systems development. It defines:

- The states into which the development will be broken.
- The tasks to be performed.
- The deliverables from these tasks
- The standards to be used.
- Quality checks to be applied.

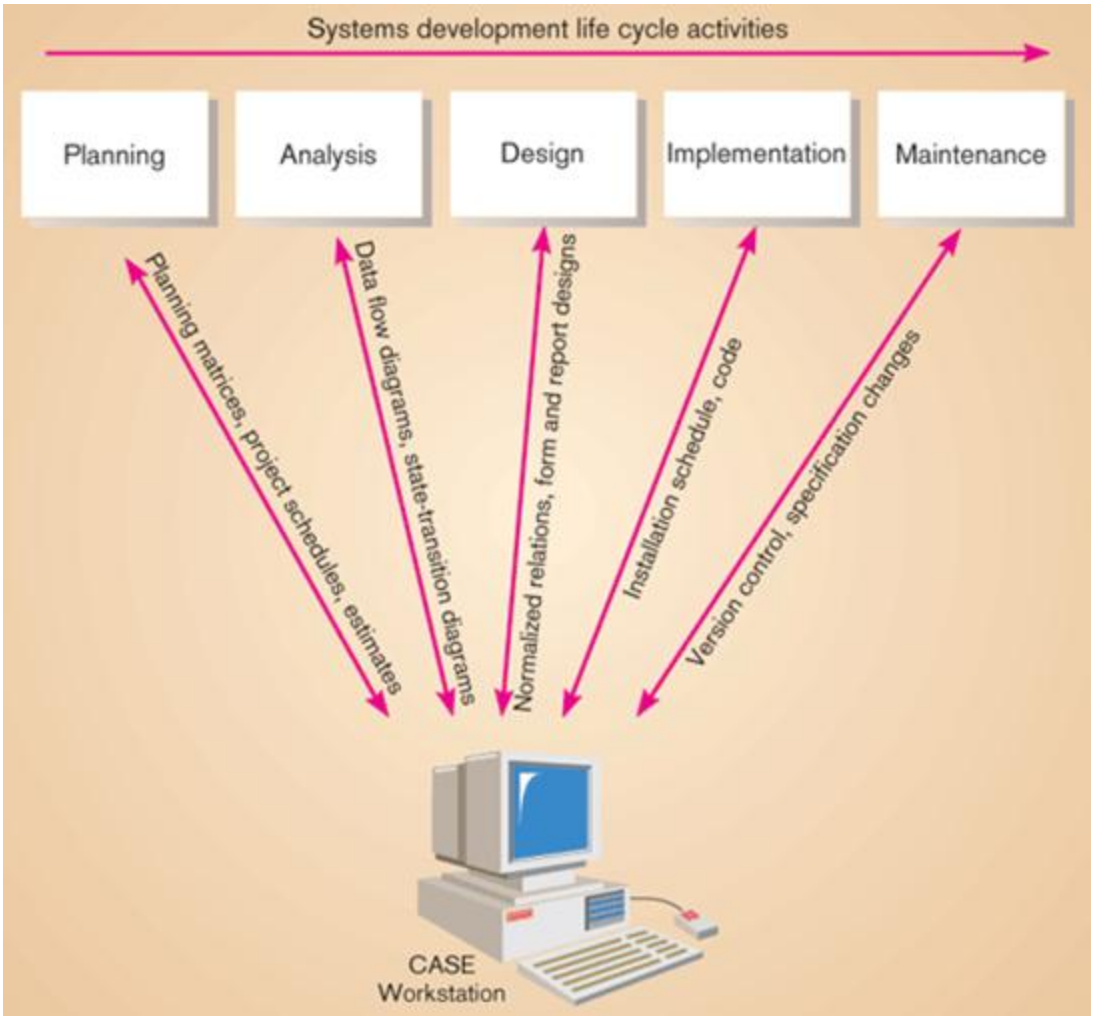


Figure: CASE can provide efficient supports for SDLC activities

Techniques:
A set of techniques is used to implement the structured method. These techniques are used to provide the descriptions of the business system requirements from various viewpoints. Examples: Data Flow Diagram, Entity-Relationship Diagram, Class Diagram, etc.

Tools:
General aim of the software tools is to:

- *Decrease the human effort required to develop software.*
- *Increase the quality of software*

Software tools provide implementations of the techniques to enable the descriptions of the system requirements to be *capture, modified and elaborated.* Software tools allow elements of the system to be drawn, described, stored and where appropriate to be generated automatically.

b i i z a y @ g m a i l . c o m

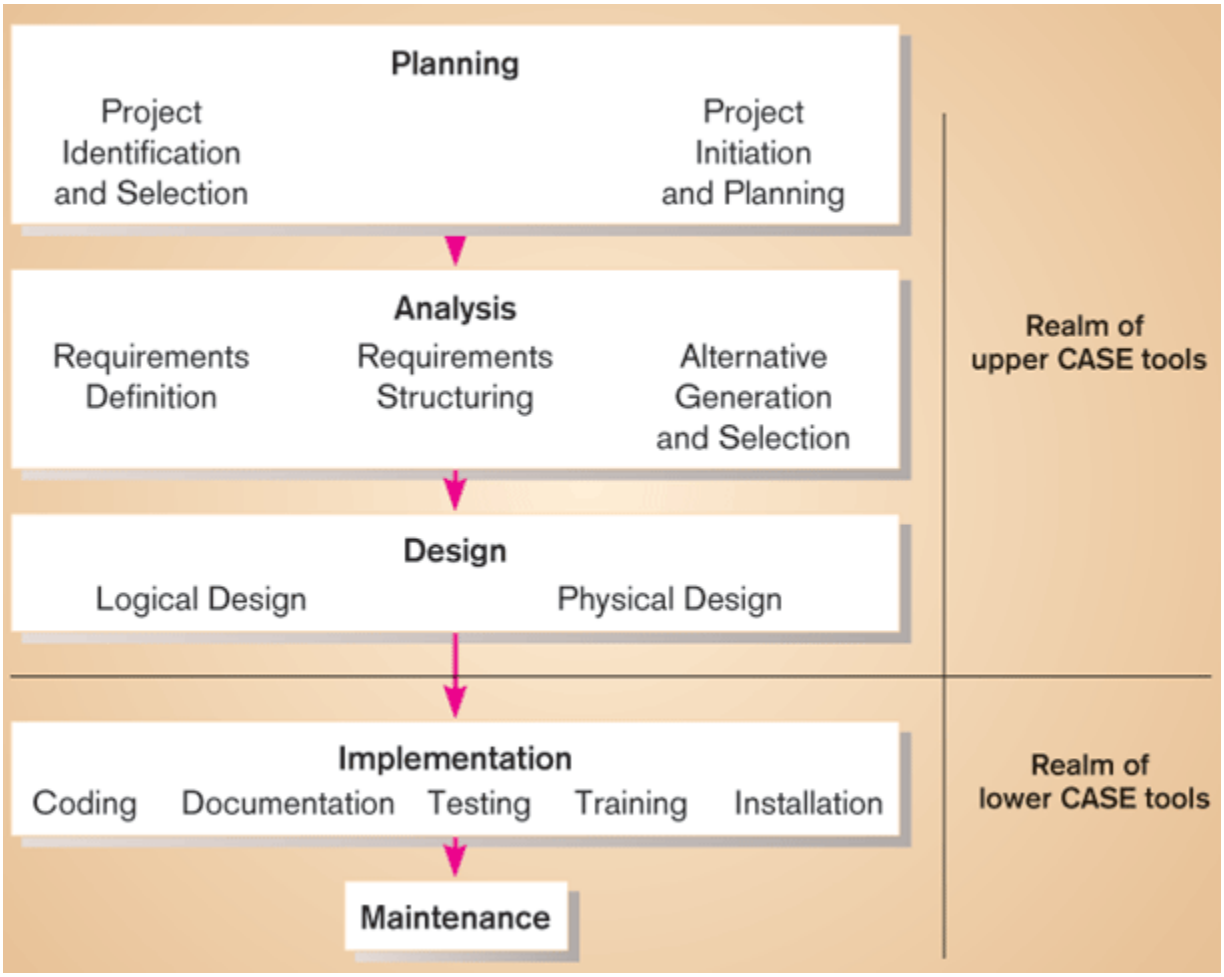


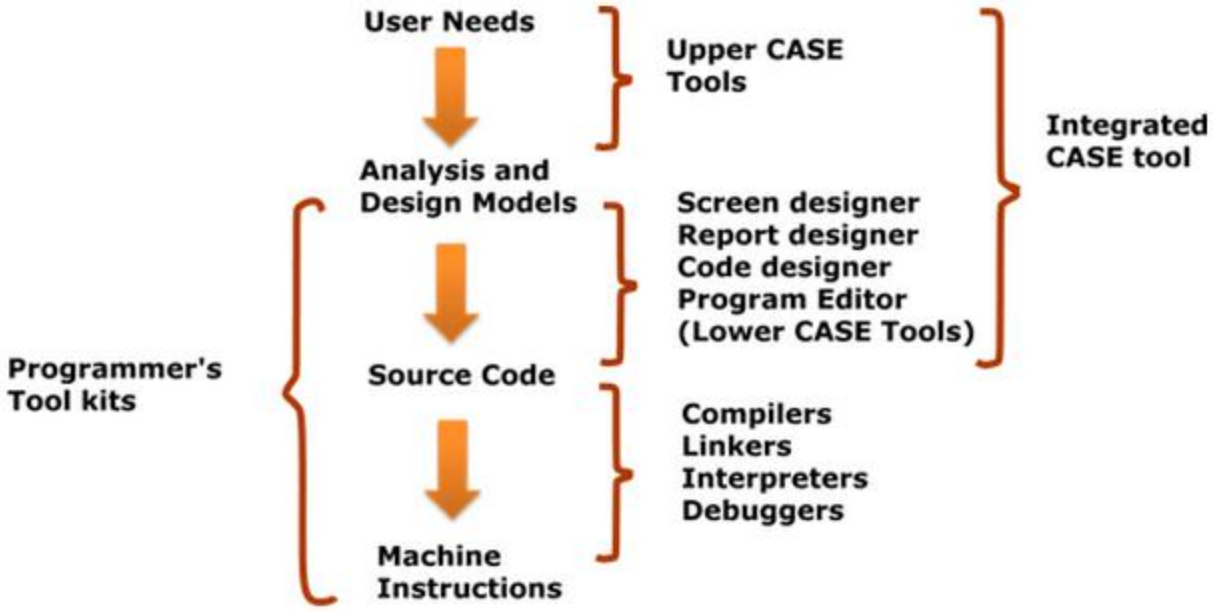
Figure: Relationship between CASE tools and SDLC

Types of CASE:

- Upper CASE:
  It describes tools that automate or support the 'upper' or earliest phases of systems development. It supports the information planning and the project identification and selection, project initiation and planning, analysis and design phases of the systems development life cycle. It has the components like Diagramming tools, Form and report generators, and Analysis tools.

- Lower CASE:
  It describes tools that automate or support the 'lower' or later phases of systems development. It supports the implementation and maintenance phases of the systems development life cycle. It has the component like Code generators.

- I-CASE (Integrated CASE)
  It support the entire SDLC. Also called as Cross life-cycle CASE. It supports activities that occur across multiple phases of the systems development life cycle. It has the component like project management tools.



CASE Tool And Application Development

Example: Commercially available systems provide tools (i.e. computer program packages) for each phase of the system development life cycle. Some of the visual CASE tools are: ***Oracle 2000 Designer, Evergreen EasyCase, Popkin System Architect***, etc. A typical package is ***Visual Analyst*** which has several tools integrated together. A large number of object oriented CASE tools are available in the market. These include: ***Paradigm Plus*** from Protosoft, ***Rational Rose*** from Rational and ***WithClass*** from MicroGold Software. ***Enterprise Architect*** from Parx System is another comprehensive UML analysis and design tool.

CASE Tool Components

Diagramming Tools

- Include capabilities to produce diagrams such as Data Flow Diagrams, Entity Relationship Diagrams, and Class Diagrams etc.
- They have facilities to draw these diagrams and to store the details internally.
- Their ability to change and redraw eliminates an activity that Systems Analysts find both tedious and undesirable.
- Example: SmartDraw, IBMs/DFD.

***Description Tools*:**

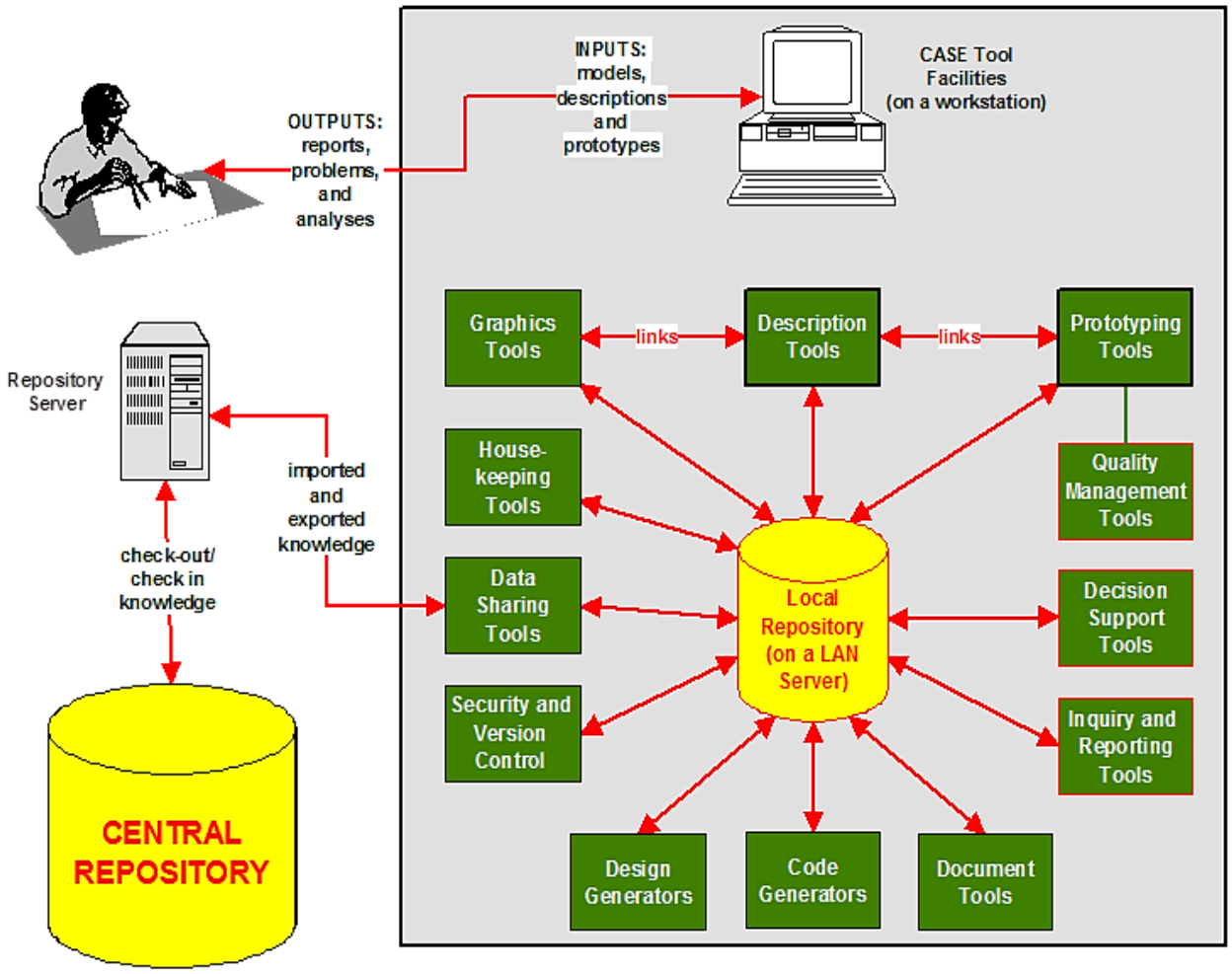
- Used to record, delete, edit, and output non graphical documentation and specifications.

***Prototyping Tools*:**

- Used to construct system components including input, output, and programs.

Quality Management Tools:

- Analyze models descriptions and prototypes for consistency, completeness, or conformance to accepted rules of the methodologies the CASE tools support.



Inquiry and Reporting Tools:

- Extract models, descriptions, and specifications from the repository.

Information Repository Tools:

- Contains the details of system components such as data items, data flows, processors etc.
- They are designed to access information easily,
- Include built-in controls and preserve the accuracy and consistency of system details.

Decision Support Tools:

- Provide information for various decisions that occur during systems development. For example, some CASE tools help systems analysts estimate and analyze feasibility.

Design Generation Tools:

- Automatically generate first draft designs for various system components based on the business requirements recorded in the repository and technology standards provided by the system designer.

Code Generators:

- Automate the preparation of Computer Software or significant portions of the software.
- They incorporate methods that allow the conversion of system specifications into executable source code.
- Code generators use 'standard' program structures which can be tailored and can have 'custom code' added to generate application programs in a third generation language such as COBOL or PL1.

Testing Tools:

- Help the system designers and builders test databases and application programs.

Data Sharing Tools:

- Provide for import and export of repository information to and from other software tools that cannot directly access the repository.

Version Control Tools:

- Maintain the integrity of the repository by preventing unauthorized or inadvertent changes and saves prior versions of various information stored in the repository.

Housekeeping Tools:

- Establish user accounts, privileges, repository subsets, tool defaults, backup and recovery, and other essential facilities

Objectives of CASE

- Improve quality of systems developed
- Increase speed of development and design
- Ease and improve testing process through automated checking
- Improve integration of development activities via common methodologies
- Improve quality and completeness of documentation
- Help standardize the development process
- Improve project management
- Simply program maintenance
- Promote reusability
- Improve software portability

Advantages of CASE

1. Increased Speed
2. Increased Accuracy
3. Reduced Lifetime Maintenance
4. Better Documentation
5. Programming in the hands of non-programmer

Disadvantages of CASE

1. CASE tools do not necessarily prevent people from making bad designs
2. The output from the CASE tool may look nice, but if it was poorly done, it will still be bad
3. If a person is new to using CASE tool, they might need a lot of time to study the concept (i.e. Training is required)
4. CASE is not cheap.

The Role of CASE in Data Modeling

CASE tools play a major role in the following activities:

- Project management
- Data dictionary
- Code generation
- User interface design
- Schema generation
- Creation of meta-data for data warehouse
- Reverse engineering
- Re-engineering
- Document generation
- Version control
- Object Oriented analysis and design
- Software testing
- Data modeling
- Project scheduling
- Cost estimation

Books References:

✓ Hoffer, J.A., George, J.F. and Valacich J.S., "Modern Systems Analysis and Design", 3rd Edition, Pearson Education, 2003.
✓ Joseph S. Valacich, Joey F. George, Jeffrey A. Hoffer, "Essentials of Systems Analysis and Design" - 5th Edition.
✓ K.E. Kendall and J.E. Kendall, "Systems Analysis and Design", 8th Edition.
✓ Jeffrey L. Whitten, Loonnie D. Bentley, 5rd Edition, "Systems Analysis and Design Methods".
✓ Englewood Cliffs, New Jersey, "Systems Analysis and Design".
✓ V. Rajaraman, "Analysis and Design of Information Systems", 2nd Edition, Prentice Hall of India, New Delhi, 2002
✓ Shelly Cashman, "System Analysis and Design" - 9th Edition.

Websites References:

✓ http://www.visual-paradigm.com/product/bpva/tutorials/dfd.jsp
✓ http://users.csc.calpoly.edu/~jdalbey/205/Lectures/HOWTO-ERD.html

Assignments:

(1) List out some of the differences between Flowchart and DFD.
(2) Customer sends enquiry to commercial department; receives quotations from the sales department and places an order. Based on the customer order, the work order is sent to the planning department for planning scheduling and control, in turn, the planning department raises a job order on the "shop floor". On completion, delivery note and invoice are made out costing department also prepares an order wise comparative statement of estimated and actual costs. Draw a DFD diagram of following up to level 2. (T.U. 2067)
(3) What are the management skills needed by system analysts? (T.U. 2067)
(4) Front office of Hotel is responsible for all room reservations, room allocations and final settlement of bills. Any company or person can reserve rooms for their future stay. They have to indicate from what date to what day they need the room. They also have to indicate how many rooms are required. Sometimes the reservations could be cancelled or the dates or number of rooms changed. For reservation, cancellation or modification or rooms, customer receives an acknowledgement from the hotel. Draw a DFD diagram of the following up to level 2 (T.U. 2067)
(5) What are the three relationship types of E-R diagrams? How are these relationships paired to build an E-R diagram? (T.U. 2067)
(6) Design the E-R diagram of the following:
    (a) Customer with draws money from his account.
    (b) Student attends classes. (T.U. 2067)
(7) What are the roles of CASE in data modeling? (T.U. 2068)
(8) Why are good interpersonal communication skills essential for system analysts? (T.U. 2068)
(9) What do you mean by CASE tools? Explain the CASE tools in data modeling. (T.U. 2069)
(10) Draw the DFD for "Student Information System" upto level 2. (T.U. 2069)
(11) What are the key steps for designing the ERD? Explain with example. (T.U. 2069, 2070)
(12) Differentiate between physical DFD and logical DFD. (T.U. 2070)

A Gentle Advice:

Please go through your text books and reference books for detail study!!! Thank you all.

Notes Compiled By:

Bijay Mishra
biizay.blogspot.com
9813911076 or 9841695609

Unit 3 - Structured Methodologies

3. Structured Methodologies **6 Hrs.**

3.1 The Need for a Structured Methodology
3.2 The Role of CASE in Data Modeling
3.3 Advantages and Disadvantages of Modeling
3.4 Data Dictionaries
3.5 Modeling Tools (Structured English, Decision Table, and Decision Tree)

The Need for a Structured Methodology

Structure is the arrangement or interrelationship of the parts as dominated by the whole. *Structured* is to put together systematically, construct and optimize. *Method* is a regular, orderly procedure or way of doing things. *Methodology* is a system of methods.

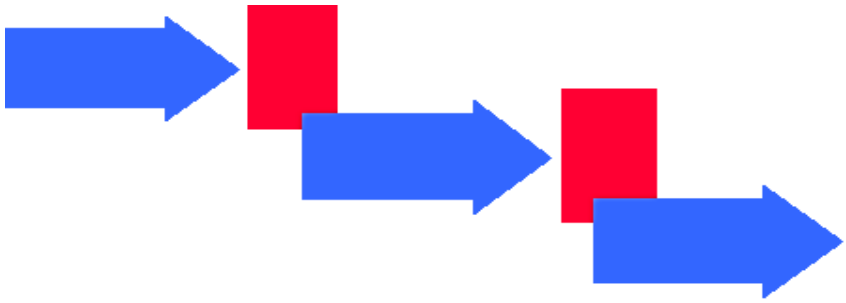


Figure: Structured methodology as an orderly procedure

Structured Systems Analysis and Design Method (SSADM) (originally released as methodology) is a systems approach to the analysis and design of information systems. The SSADM is an open methodology based on the waterfall model used for the analysis and design of information systems.

SSADM is an open standard, and as such is freely available for use by companies or individuals. It has been used for all government information systems development since 1981, when it was first released, and has also been used by many companies in the expectation that its use will result in robust, high-quality information systems. SSADM is still widely used for large scale information systems projects, and many proprietary CASE tools are available that support SSADM techniques.

SSADM follows the waterfall life cycle model starting from the feasibility study to the physical design stage of development. One of the main features of SSADM is the intensive user involvement in the requirements analysis stage. The users are made to sign off each stage as they are completed assuring that requirements are met. The users are provided with clear, easily understandable documentation consisting of various diagrammatic representations of the system. SSADM breaks up a development project into stages, modules, steps and tasks. The first and foremost model developed in SSADM is the data model. It is a part of requirements gathering and consists of well-defined stages, steps and products.

b i i z a y @ g m a i l . c o m

SSADM Techniques

The techniques used in SSADM are logical data modeling, data flow modeling and entity behavior modeling.

1. Logical Data Modeling:
   This involves the process of identifying, modeling and documenting data as a part of system requirements gathering. The data are classified further into entities and relationships.

2. Data Flow Modeling:
   This involves tracking the data flow in an information system. It clearly analyzes the processes, data stores, external entities and data movement.

3. Entity Behavior Modeling:
   This involves identifying and documenting the events influencing each entity and the sequence in which these events happen.

Objective of SSADM

SSADM was developed with the following objectives

- Ensure that projects can successfully continue should a loss of staff occur without a damaging effect on the project
- Develop overall better quality systems
- Improve the way in which projects are controlled and managed
- Allow more effective use of experienced and inexperienced staff and their development
- Make it possible for projects to be supported by computer based tools e.g.
- Computer-aided software engineering systems
- Improve communication between participants in a project so an effective framework is in place

Characteristics of SSADM

Some of the important characteristics of SSADM are:

- Dividing a project into small modules with well-defined objectives
- Useful during requirements specification and system design stage
- Diagrammatic representation and other useful modeling techniques
- Simple and easily understood by clients and developers
- Performing activities in a sequence

Stages of SSADM

The stages of SSADM include:

- Determining feasibility
- Investigating the current environment
- Determining business systems options
- Defining requirements
- Determining technical system options
- Creating the logical design
- Creating the physical design

Each of these stages applies certain techniques and a sequence of analysis. They include conventions and procedures for recording and interpreting the information with the help of diagrams and text.

Advantages of SSADM

1. *Timelines*: Theoretically, SSADM allows one to plan, manage and control a project well. These points are essential to deliver the product on time.
2. *Usability*: Within SSADM special emphasis is put on the analysis of user needs. Simultaneously, the systems model is developed and a comprehensive demand analysis is carried out. Both are tried to see if they are well suited to each other. Respond to changes in the business environment: As in SSADM documentation of the project's progress is taken very seriously, issues like business objectives and business needs are considered while the project is being developed. This offers the possibility to tailor the planning of the project to the actual requirements of the business.
3. *Effective use of skills*: SSADM does not require very special skills and can easily be taught to the staff. Normally, common modeling and diagramming tools are used. Commercial CASE tools are also offered in order to be able to set up SSADM easily.
4. *Better quality*: SSADM reduces the error rate of IS by defining a certain quality level in the beginning and constantly checking the system.
5. *Improvement of productivity*: By encouraging on-time delivery, meeting business requirements, ensuring better quality, using human resources effectively as well as trying to avoid bureaucracy, SSADM improves the overall productivity of the specific project and the company.
6. *Cuts costs*: SSADM separates the logical and the physical systems design. So the system does not have to be implemented again with new hard -or software.

Disadvantages of SSADM

1. SSADM puts special emphasis on the analysis of the system and its documentation. This causes the danger of over-analyzing, which can be very time and cost consuming. Due to various types of description methods, checks of consistence cannot be carried out.
2. Especially with large systems, the outline diagram can become very unclear, because all relevant data flows have to be included.

The Role of CASE in Data Modeling

Please go through Unit 2 for the reference. Everything related to CASE have been explained in detail.

Advantages and Disadvantages of Modeling

For this topic you should give the answer related to:

- Process Modeling (Example: DFD)
- Conceptual Data Modeling (Example: E-R Diagram)
- Logic Modeling (Examples: Structured English, Decision Tables, Decision Trees)

The first two topics have been covered in Unit 2 while the third topic would be covered in this unit. So go through it.

Data Dictionary

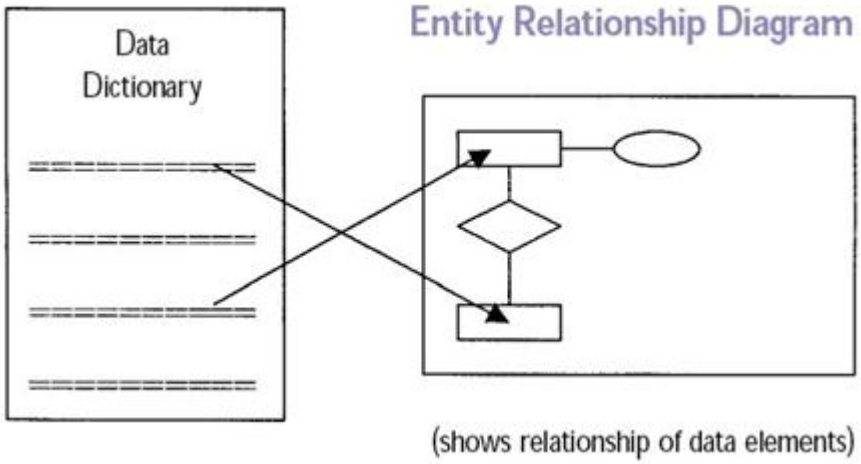
A data dictionary is a table providing a comprehensive description of each field in the database. This commonly includes: field name, data type, data format, field size, description and example.

This is shown in the example below.

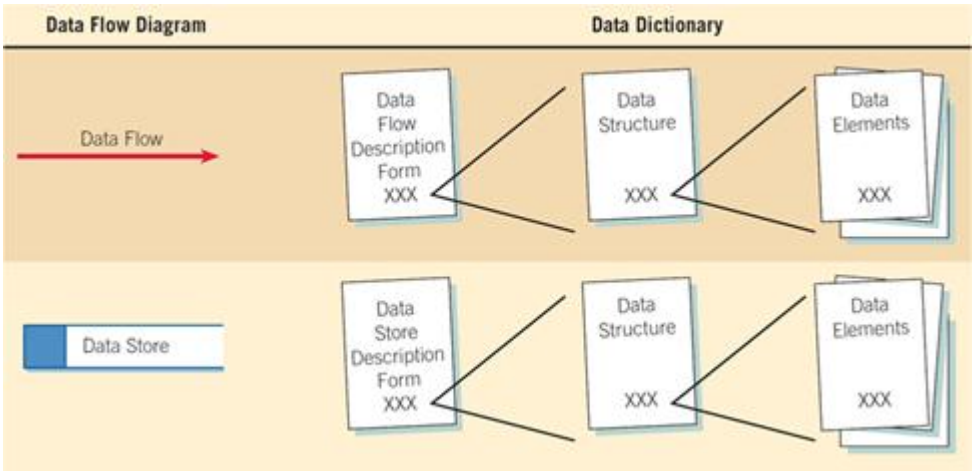
Field name	Data type	Data format	Field Size	Description	Example
UserId	Text	NNNNNNNN	8	Unique eight-digit number represented as text	0001539
FirstName	Text		25	First name of employee	Bijay
Surname	Text		25	Surname of employee	Mishra
DOB	Date	DD/MM/YYYY	10	Date of birth as a short date format	29/08/1981
HourlyPayRate	Currency	#####.##	8	Rate of pay expressed in dollars per hour	34.50
Height	Real	#.##	3	Height in metres, with two decimal places	1.58
FeesPaid	Boolean		1	Y or N for Yes or No	Y

Data dictionary is a repository of data about data (metadata). It contains information about each of the component of DFDS, data stores, processes and data flow. Data Dictionary is an integral part of system specifications, since with it, DFDS are just pictures with no details.

Data dictionary is a structured repository of data. It is a catalogue of all data elements, data structures and processes described in logical DFDs. Although we give descriptive names to the data flows, process and data stores in a DFD, it does not give the details. Hence to keep the details of the contents of data flows, process and data stores we also require a data dictionary. It clearly documents the list of contents of all data flows, processes and data stores.



b i i z a y @ g m a i l . c o m



Before we discuss the importance and contents of a data dictionary, let us understand the meaning of the following terminology:

Data Element: Data element is the smallest unit of data that has some meaning. For example, part code, part name, date of transaction, etc., are data elements.

Data Structure: Data structure is a group of data elements that describe a unit in the system. For example, Part Details is a data structure that contains part code, part name and date of transaction as data elements.

Data Store: Data store is a data structure for collecting data input during processing. For example, Part Register is a data store.

Data Flow: Data flow is a data structure that shows a unit of data in motion. For example, New Part Details is a data flow that moves from an external entity to a process.

The data elements, data stores, data flows and processes are described in a data dictionary as illustrated in Figures (1), (2), (3) and (4)

Part Code Data Element		
Short Description: This element describes the code of a part or subassembly.		
Organisation: B.R. Auto Limited		
Date:		
Aliases (contents): Part Number		
If Discrete		If Continuous
Value	Meaning	Range of
		Value: A001 to Z999
		Typical Value: A001
		Length: 4
		Internal Representation: Character

Figure (1): An Example of Data Element in a Data Dictionary

b i i z a y @ g m a i l . c o m

Data Dictionary

Transactions file Description: Day's Transaction	Data Store
Data Flow in:	Data flow out:
Issue, Return, Receipt	Consolidated transaction of the day
Contents: Part No., Part Name, Date of Transaction, Issue, Receipt, Name of Concerned Person and Posting Status	
	Physical Organisation: Store

Figure (2): An Example of Data Store in a Data Dictionary

New Part Details	Data Flow
Source Ref. Description: Part	
Destn. Ref. Description: Part Register	
Expanded Description: Part Details like: Part code, Part name, Date of Transaction etc. are entered into the Part register.	
Included data structure:	Volume Information
Part register	Volume increases when new part is entered but do not decrease when old part is discarded.

Figure (3): An Example of Data Flow in a Data Dictionary

Description: Maintain Part Register

Process

Input	Logic Summary	Output
New part details	All part details with a unique part code is accepted. Duplicate part code is not accepted	Updated part
Daily Transaction Details	Daily Transactions Details of those part codes are accepted that exist in Part-Registers.	

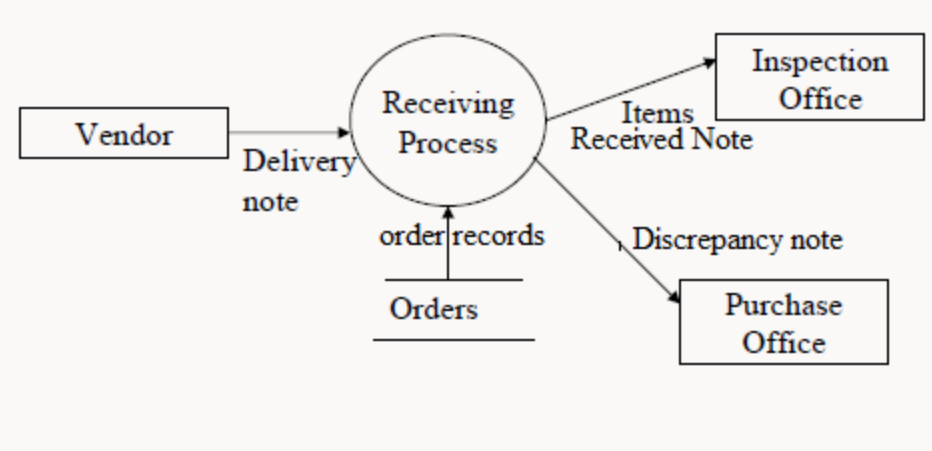
Figure (4): An Example of Process in a Data Dictionary

b i i z a y @ g m a i l . c o m

Data Dictionary Format
Data dictionary gives in detail the characteristics of a data element. Typical characteristics are:

Data name: Should be descriptive and self-explanatory. This will help in documentation and maintenance
Data description: What it represents
Origin: Where the data originates. E.g. input from forms, comes from receiving office, keyed in by user etc.
Destination: Where data will flow and will be used (if any)
Data Type: numeric, alphanumeric, letters (or text), binary (0 or 1; True or False), Integer, Decimal fixed point, real (floating point), currency unit, date
Length: no of columns needed
Limits on value: (if relevant) e.g. upper and lower bounds of value (age>0, <100)
Remarks: (if any)

Consider the DFD below for an example:



Vendor sends items with a delivery note while fulfilling an order (along with the physical items) to a receiving office. Receiving office compares a delivery note against order placed. If there is a discrepancy (i.e. difference), a discrepancy note is sent to purchase office. Actual items received note is sent to the inspection office along with items received.

Data Elements in Data Flow
Delivery note = Order no + Vendor code + Vendor name + Vendor address + item code + item name + delivery date + quantity supplied + units.

Discrepancy note = Order no + Vendor code + Vendor name + Vendor address + item code + item name + delivery date + quantity supplied + units + excess/deficiency + no of days late/early.

Items received note = Delivery note

Order records = order no + vendor code + vendor name + vendor address + item code + item name + order date + quantity ordered + units + delivery period.

Example 1:
Name: Order number
Description: Used to identify order given to vendor
Origin: Part of delivery note from vendor
Destination: Receiving process
Data type: Numeric Integer
Length: 8 digits
Limits on value : >000, <=99999999 Actual value not relevant. Used only as unique identifier
Remarks: It is a key field.

Example 2:
Name: Delivery date
Description: Date item is to be delivered
Origin: Part of delivery note from vendor. Is also in orders data store which is input to receiving process
Destination: Receiving process
Data type: Numeric Integer
Length: 8 digits
Limits on value: Date field in the form DDMMYYYY. Should satisfy constraints of a date in calendar
Remarks: Blank fields not allowed. E.g. 05082004 is OK but not 582004

Purpose of Data Dictionary:
The purpose of data dictionary is to ensure that everyone working on the development, maintenance, or usage of a system uses consistent names and conventions for all its components.

During Analysis: by the analysts when reviewing requirements with users, to serve as reference for the DFDs of the existing system and any proposed improvements

During Design: as reference for DFDs of proposed system; as specifications for programmers, analysts, and data entry personnel who will construct the system

During Coding: to ensure naming consistency amongst programmers, etc.

During Installation: as reference for DFDs when training users and planning conversion strategies

During Usage: as reference for DFDs used to train new users, as reference for programmers, etc. when making system revisions

Importance of Data Dictionary
Data dictionary is an important tool for structured analysis as it offers various advantages. The reason why analyst use data dictionary as follows:

- It is a valuable reference for designing the system. It is used to build the database and write programs during design phase.
- It assists in communicating meanings of different elements, terms and procedures.
- It facilitates analysis in determining additions and changes in the system.
- It helps the analyst to record the details of each element and data structure.
- It is used to locate errors in the system descriptions.
- It is also a useful reference document during implementation of the system.
- To manage the details in the large system.
- To communicate a common meaning for all system elements.
- To document the features of the system.
- To facilitate analysis of details in order to evaluate characteristics and determine where system changes should be made.
- To find errors and omissions in the system.

Advantages of Data Dictionaries

There are a number of advantages of using data dictionary in computer system analysis and design. The main advantages are: consistency, clarity, reusability, completeness, increase in sharing, etc. Some other advantages are:

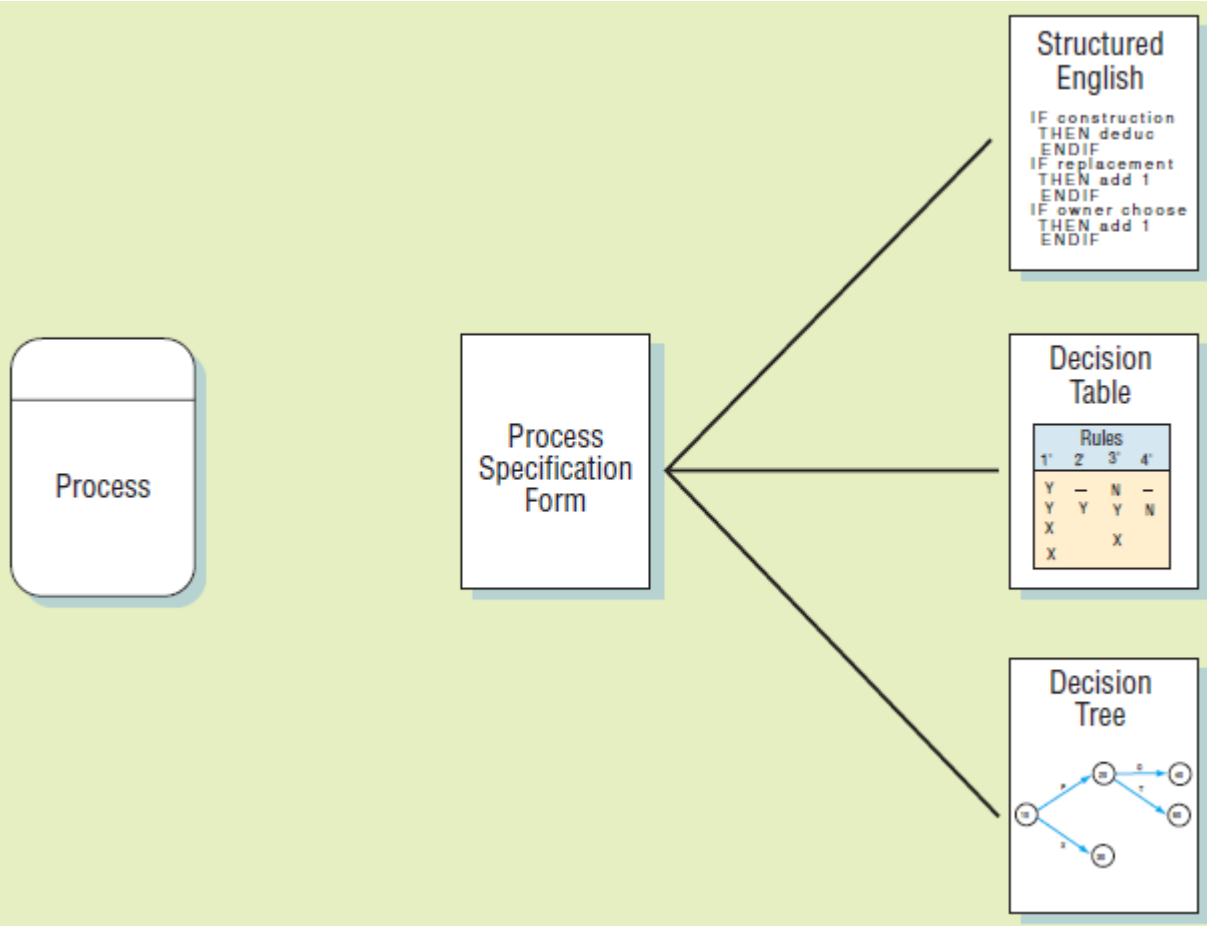
- May be used at high or low level of analysis
- Provides good system documentation at granular level
- Improve documentation and control
- Improves consistency in data use
- Easier data analysis
- Reduce data redundancy
- Simpler Programming
- Reduce errors
- Defines Information items unambiguously
- Improves Knowledge about the data resources

Disadvantages of Data Dictionaries

- Doesn't provide the details about data
- Needs careful planning, defining the exact requirements planning, its contents, testing, implementation and planning.
- For large computer based systems, the data dictionary grows rapidly in size and complexity.
- Difficult to maintain it manually
- It includes not only the installation cost but also the cost of collecting the information keeping it up to date

Modeling Tools

After defining all the data elements and data structures in the data dictionary, the systems analyst begins to describe the processes. Process descriptions are the tools for documenting the procedures and describing the system logic. They contain the logic used to process the input data for getting the output.



Some of the modeling tools (also called as ***process description tools*** or ***logic modeling tools***) used in structured methodologies are:

(a) Structured English
(b) Decision Table
(c) Decision Tree

Structured English

Structured English is used to describe the logic of a process. It is based on the principle of the structured programming. It is created by the merging of the English language with the syntax of the structured programming. It tries to express the verbal statement in the more logical form.

Structured English uses narrative statements to describe a procedure and reserve words for logical formulation. Structured English specification still requires analyst to identify the conditions that occur and alternative actions to be taken.

Entire process can be stated quickly in English like statements. No strict syntaxes, no special symbols or formats are used but simple English words with vocabulary consisting of:

a. Imperative English language verbs
b. Terms defined in data dictionary
c. Reserved or key words for logic formulation

Common keywords used in Structured English

START, BEGIN, END, STOP, DO, WHILE, DO WHILE, FOR, UNTIL, DO UNTIL, REPEAT, END WHILE, END UNTIL, END REPEAT, IF, IF THEN, ELSE, IF ELSE, END IF, THEN, ELSE THEN, ELSE IF, SO, CASE, EQUAL, LT, LE, GT, GE, NOT, TRUE, FALSE, AND, OR, XOR, GET, WRITE, PUT, UPDATE, CLOSE, OPEN, CREATE, DELETE, EXIT, FILE, READ, EOF, EOT

It uses three basic types of statements:

1. *Sequence Structures:* They include a set of instructions that are carried out one after another and do not depend on any condition.
2. *Decision Structures:* They include one or more sets of instructions that are carried out depending upon one or more conditions. They generally use the phrase IF THEN ELSE to carry out different actions.
3. *Iteration Structures:* They include a set of instructions that are repeated until a particular condition occurs. They generally use the phrase DO WHILE ...ENDDO to repeat a set of instructions.

Some of the Statements along with examples are:

```
SELECT <Expression>

CASE 1 IF <condition> THEN <statements>
CASE 2 IF <condition> THEN <statements>
```

Example:

SELECT the dollar amount that applies:
    CASE 1 IF the planned expenditure > $10,000
        THEN send expense_justification to corporate headquarters
    CASE 2 IF the planned expenditure >= $1000 and <= $10,000
        THEN send expense_justification to plant manager
    CASE 3 IF the planned expenditure < $1000
        THEN send expense_justification to area supervisor

```
REPEAT UNTIL < condition>
    <statement (s)>
END- REPEAT UNTIL
```

Example:

REPEAT-UNTIL all invoice line-items are extended
    Multiply quantity_shipped by item_cost to get item_extended_cost
    Add item_extended_cost to total
END-REPEAT-UNTIL

b i i z a y @ g m a i l . c o m

```
IF  <condition>
 THEN<statement (s)>
ELSE
 <statement (s)>
ENDIF
```

Example:

```
IF a customer_order_total is greater than $400
        THEN
        IF customer_balance is > 60 days past due
                THEN hold the customer_order
                send reminder letter
                ELSE process the customer order
                END-IF
        ELSE process the customer order
END-IF
```

```
DO
        <statement (s)>
UNTIL
        <condition>
```

Example:

```
DO
        READ next Inventory Record
        BEGIN IF
                If Quantity in stock is less than reorder point
                 THEN GENERATE Order
        END IF
UNTIL end of file (Inventory)
```

```
DO WHILE <condition>
<statement (s)>
END-DO WHILE
```

Example:

```
DO-WHILE there are overtime_pay_records
        Read a record
        Add overtime_hours to overtime_hours_total
        Add overtime_pay to overtime_pay_total
END-DO-WHILE
```


b i i z a y @ g m a i l . c o m

Example 1: "Compute Discount"
IF: order is from bookstore.
AND IF order is of six copies or more per book title
THEN: discount is 25%.
ELSE (order is for less than six copies per book title.)
SO: no discount is allowed.
ELSE (order is from library or from individuals)
SO- IF order is for 50 or more copies per book title.
THEN: discount is 15%.
ELSE IF order is for 20 to 49 copies per book title.
THEN: discount is 10%.
ELSE IF order is 6 to 9 copies per title.
THEN: discount is 5%.
ELSE (order is less than 6 copies per book title).
THEN: discount is not allowed.

Example 2: A bank will grant loan under the following conditions:

If a customer has an account with the bank and had no loan outstanding, loan will be granted.
If a customer has an account with the bank but some amount is outstanding from previous loans then loan will be granted if special approval is given.
Reject all loan applications in all other cases.

```
APPROVE LOAN
IF customer has a Bank Account THEN
        IF Customer has no dues from previous account THEN
            Allow loan facility
        ELSE
        IF Management Approval is obtained THEN
            Allow loan facility
        ELSE
            Reject
        ENDIF
        ENDIF
              ELSE
              Reject
ENDIF
EXIT
```

Advantages of Structured English

- Clarifying the logic and relationships found in human languages
- Structured English is intended to be used as an effective communication technique for analysts and users.
- It survives the life of the project
- It can be kept in automated format

- It can be made concise, precise, and readable
- It can be tailored to suit the user
- It can be written quickly and naturally
- Structured English can be used to state the rules clearly.
- After activities have been described in Structured English, one can ask another person to review so as to determine the mistake of omission in making the decision process.
- Structured English offers a concise way of summarizing a procedure where decision must and action taken.

Disadvantages of Structured English

- It takes some time to build structured English skills
- It seems to be more formal than it is
- It can scare off the user

Decision Table

The decision table or a logic table is a chart with four sections listing all the logical conditions and actions. In addition the top section permits space for title, date, author, system and comment.

A decision table appears as a matrix of rows and columns that shows conditions and corresponding actions. Decision rules, included in a decision table, states what procedure is to follow when certain condition exist.

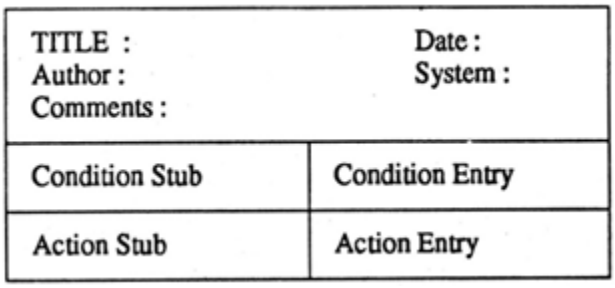


Figure: Format of Decision Table

A decision table is divided into four sections:

1. Condition Stub
2. Condition Entries
3. Action Stub
4. Action Entries

Condition stub identifies total set of relevant test or condition. These conditions require yes or no answers. Combination of these conditions are then identified and expressed as rules or conditions entries. The condition stub contains a list of all the necessary tests in a decision table. In the lower left-hand corner of the decision table we find the action stub where one may note all the processes desired in a given module.

Condition entries provide all possible permutations of yes or no responses related to the condition statement. The upper right corner provides the space for the condition entry - all possible permutations of yes and no responses related to the condition stub. The yes and no possibilities are arranged as a vertical column called rules. Rules are numbered 1, 2, 3 and so on. We can determine the rules in a decision table by the formula:

Number of rules = 2^N = 2N where N represents the number of condition and ^ means exponentiation. Thus a decision table with four conditions has 16 (24 = 2 x 2 x 2 x 2 = 16) rules one with six conditions has 64 rules and eight conditions yield 256 rules.

Action stub list the possible actions which can occur as a result of different condition combinations. Action Stub contains a list of all the processes involved in a decision table.

Action entries show what specific action in the set to take when selected or group of conditions are true. Action entry indicates via dot or X whether something should happen in a decision table.

Steps of Creating a Decision Table

1. Determine conditions and their values.
   The number of conditions becomes the number of rows in the top half of the decision table.

2. Determine possible actions that can be taken
   This becomes the number of rows in the lower half of the decision table.

3. Determine condition alternatives for each condition
   In the simplest form two alternatives (Y or N), in an extended there may be many alternatives.

4. Calculate the maximum number of columns (Rules) in the decision table
   For example, say we have Conditions A, B, and C. Each condition has the following values:

   Condition A: Y, N
   Condition B: 1, 2, 3
   Condition C: Y, N

   In this example, A and C have 2 values, and B has 3 values. Therefore, the number of rule columns is 2 * 3 * 2 = 12.

5. Fill in the condition alternatives
   It is done to ensure we get every possible combination of values in the table.
   Example:
   A table has 3 conditions: Condition A, B, and C. All three conditions have two possible values: Y (yes) and N (no). Draw the condition section and populate the condition alternatives.

Since each condition has 2 possible values, there will be 2 * 2 * 2 = 8 rule columns:

	1	2	3	4	5	6	7	8
Condition A								
Condition B								
Condition C								

To calculate the first row, divide the number of rules by 2. That gives us 4. The first row will have 4 Y's and 4 N's and that should fill the whole row:

	1	2	3	4	5	6	7	8
Condition A	Y	Y	Y	Y	N	N	N	N
Condition B								
Condition C								

For the second row, we divide the value from the previous row (4) by 2: 4/2 = 2. So we now fill in the second row with two Y's, two N's, two Y's, two N's, until the row is full:

	1	2	3	4	5	6	7	8
Condition A	Y	Y	Y	Y	N	N	N	N
Condition B	Y	Y	N	N	Y	Y	N	N
Condition C								

The last row should work out such that the values just alternate, one of each across the row. To make sure, let's get the previous row's value and divide by 2: 2/2 = 1. So we should have the last row filled with one Y, one N, one Y, one N, until the row is full. Our final table now looks like this:

	1	2	3	4	5	6	7	8
Condition A	Y	Y	Y	Y	N	N	N	N
Condition B	Y	Y	N	N	Y	Y	N	N
Condition C	Y	N	Y	N	Y	N	Y	N

6. Fill in the action entry and complete table by inserting an “X” where rules suggest actions
   This involves reading over the process description again and marking an X in columns where the conditions are met.

7. Combine rules where it is apparent

b i i z a y @ g m a i l . c o m

Condition 1:	Y	Y	Condition 1:	Y	
Condition 2:	Y	N	Condition 2:	–	(can be 'Y' or 'N')
Action 1:	**X**	**X**	**Action 1:**	**X**	
Action 2:			Action 2:		

8. Check for impossible situations, contradiction and redundancies.

Conditions And Actions	**Rules**			
	1	**2**	**3**	**4**
Salary > NRs 50,00,000/year	Y	Y	N	N
Salary < NRs 2,000/month	Y	N	Y	N
Action 1				
Action 2				

This is an impossible situation

For example a person cannot earn greater than NRs 50,00,000 per year and less than NRs. 2,000 per month. Contradictions occur when rules suggest different actions but satisfy the same conditions. Contradictions often occur if the dashes (-) are incorrectly inserted into the table.

Contradictions could also be a result of discrepant information gathered from different systems users. Redundancy occurs when identical sets of alternatives require the exact same action.

Conditions And Actions	**Rules**						
	1	**2**	**3**	**4**	**5**	**6**	**7**
Condition 1	Y	Y	Y	Y	Y	N	N
Condition 2	Y	Y	Y	N	N	Y	N
Condition 3	-	N	-	-	-	N	Y
Action 1	X			X	X		
Action 2			X			X	
Action 3		X					X

Contradiction **Redundancy**

9. Rearrange the condition and actions if this makes the decision table more understandable.

Conditions And Actions	**Rules**			
	1	2	3	4
Raining Days	Y	Y	N	N
Weekend	Y	N	Y	N
Stay at home		X		X
Go to neighbor's house	X			
Go shopping with friends			X	

b i i z a y @ g m a i l . c o m

Conditions And Actions	Rules 1	2	3
Raining Days	Y	-	N
Weekend	Y	N	Y
Stay at home		X	
Go to neighbor's house	X		
Go shopping with friends			X

Example: Let us consider the following example:
If order is from book store
And if order is for 6 copies
Then discount is 25%
Else (if order is for less than 6 copies)
No discount is allowed
Else (if order is from libraries)
If order is for 50 copies or more
Then discount is 15%
Else if order is for 20 to 49 copies
Then discount is 10%
Else if order is for 6 to 19 copies
Then discount is 5%
Else (order is for less than 6 copies)
No discount is allowed

TITLE:		DATE:					
Author: Comments:		System:					
Condition Stub		Condition entry					
		1	2	3	4	5	6
IF	Customer is bookstore	Y	Y	N	N	N	N
	Order size is 6 or more	Y	N	N	N	N	N
	Customer is library	N	N	Y	Y	Y	Y
	Order size is 50 or more	N	N	Y	N	N	N
	Order size is 20-49	N	N	N	Y	N	N
	Order size is 6-19	N	N	N	N	Y	N
Then	Allow 25% discount	X	-	-	-	-	-
	Allow 15% discount	-	-	X	-	-	-
	Allow 10% discount	-	-	-	X	-	-
	Allow 5% discount	-	-	-	-	X	-
	No discount	-	X	-	-	-	X
	Action Stub	Action Entry					

Figure: Decision Table for Book Order

b i i z a y @ g m a i l . c o m

How can you reduce the size and complexity of a decision table?
To reduce the size and complexity of a decision table, use separate, linked decision tables, use numbers that indicate sequence rather than X's where rules and action stubs intersect, and simplify the table by identifying indifferent conditions.

If two or more combinations result in the same action, then the table can be simplified. Consider the following example:

Condition 1	Y Y
Condition 2	Y Y
Condition 3	Y N
Action 1	X X

The same action occurs whether condition 3 is true or false. As a result, one column can be eliminated from the table as follows:

Condition 1	Y
Condition 2	Y
Condition 3	-
Action 1	X

Advantages of Decision Table

- Simple and easy to understand by non-computer literate users and managers
- Good documentation of rules used in data processing.
- Simple representation of complex decision rules.
- Tabular representation allows systematic validation of specification detection of redundancy, incompleteness & inconsistency of rules
- Algorithms exist to automatically convert decision tables to equivalent computer programs.
- Allows systematic creation of test data
- Help the analysis ensure completeness
- Easy to check for possible errors
- Enhances readability
- Alternatives are shown side by side.
- Cause & effect relationship is shown, thus permitting easier user validation.
- Possible to check that all combinations of conditions have been considered thereby ensuring the completeness.
- Decision tables allow you to check for the completeness, consistency, and redundancy of logic.
- Aids in analysis of structured decisions.

Disadvantages of Decision Table

- *Total sequence*: The total sequence is not clearly shown, i.e., no overall picture is given by decision tables as presented by flowcharts.
- *Logic*: Where the logic of a system is simple, flowcharts nearly always serve the purpose better than a decision table.
- *Scaling*: Decision tables do not scale up well. We need to "factor" large tables into smaller ones to remove redundancy.
- *Familiarity*: All programmers may not be familiar with Decision Tables and therefore flow charts are more common.

Decision Tree

Decision tree turns a decision table into a diagram. It consists of nodes (a square for actions and a circle for conditions) and branches.

A decision tree is a diagram that resembles a tree, with a root on the left hand side and branches representing each decision. It is read from left to right and the actions to be undertaken are recorded down the right hand side of the diagram. The root of the tree, on the left of the diagram is starting point of the decision sequence. The particular branch to be followed depends on the prevalent conditions and decision to be made. Progression from left to right along the particular branch is the result making a series of decisions. Following each decision point is the next set of decision to be considered. The nodes of the tree thus represent conditions and indicate that a determination must be node about which condition exist before next path can be chosen. The right side of the tree lists the action to be taken depending on the sequence of conditions that is followed.

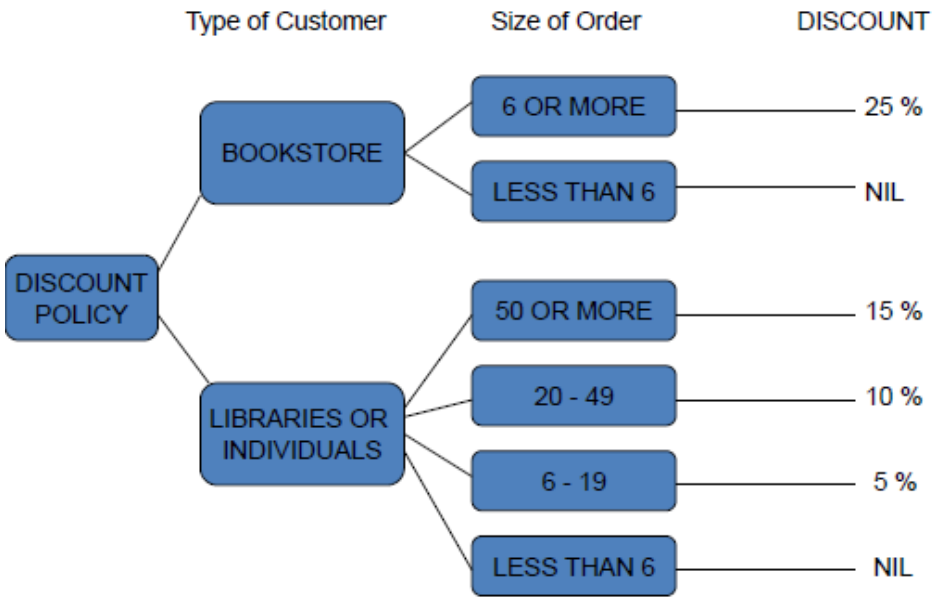


Figure: Decision Tree for Book Order

Developing a decision tree is beneficial to analyst as the need to describe condition and action compels the analyst to formally identify the actual decision that must be taken. A decision tree helps to show the paths that are possible in a design following an action or decision by the user.

b i i z a y @ g m a i l . c o m

Steps to Build a Decision Tree

- Draw the decision tree using squares to represent decisions and circles to represent uncertainty.
- Evaluate the decision tree to make sure all possible outcomes are included.
- Calculate the tree values working from the right side back to the left.
- Calculate the values of uncertain outcome nodes by multiplying the value of the outcomes by their probability (i.e., expected values).

Advantages of Decision Tree

- The order of checking conditions and executing actions is immediately noticeable
- It is easy to understand and interpret
- It can map nicely to a set of business rules
- It can be applied to any real world problems
- It makes no prior assumptions about the data
- It has ability to process both numerical and categorical data
- It can be combined with other decision techniques.
- Worst, best and expected values can be determined for different scenarios

Disadvantages of Decision Tree

- Output attribute must be categorical
- Limited to one output attribute
- Decision tree algorithms are unstable
- Trees created from numeric datasets can be complex
- Calculations can get very complex particularly if many values are uncertain and/or if many outcomes are linked.
- Takes space and consequently a minimum amount of description on the conditions and actions can be written on the trees.

Differences between Decision Table and Decision Tree

Decision tables are used when the process is logically complex involving large number of conditions and alternate solutions. Use Decision tables when there are a large number of conditions to check and logic is complex. A major drawback of a decision tree is the lack of information in its format to tell us what other combinations of conditions to test. This is where the decision table is useful.

Decision Trees are used when conditions to be tested must follow a strict time sequence. Decision trees are used to verify logic and in problems that involve a few complex decisions resulting in limited number of actions. Use Decision trees when sequencing of conditions is important and if there are not many conditions to be tested.

Conditions and actions of decision trees are found on some branches but not on others, which contrasts with decision tables, in which they are all part of the same table. Those conditions and actions that are critical are connected directly to other conditions and actions, whereas those conditions that do not matter are absent. In other words it does not have to be symmetrical.

Compared to decision tables, decision trees are more readily understood by others in the organization. Decision Table can create more queries, it is more of multipath/multiflow. Decision Tree follows single path.

Decision tables are better than decision trees at portraying complex logic, are more compact, and are easier to manipulate. Decision trees are better than decision tables at portraying simple problems and at helping people make decisions in practice.

Summary:

Criteria	Decision Tables	Decision Trees
Portraying complex logic	Best	Worst
Portraying simple rules	Worst	Best
Making decisions	Worst	Best
More compact	Best	Worst
Easier to manipulate	Best	Worst

References:

- ✓ Hoffer, J.A., George, J.F. and Valacich J.S., "Modern Systems Analysis and Design", 3rd Edition, Pearson Education, 2003.
- ✓ Jeffrey L. Whitten, Loonnie D. Bentley, 5rd Edition, "Systems Analysis and Design Methods".
- ✓ Englewood Cliffs, New Jersey, "Systems Analysis and Design".
- ✓ Joseph S. Valacich, Joey F. George, Jeffrey A. Hoffer, "Essentials of Systems Analysis and Design" - 5th Edition.
- ✓ K.E. Kendall and J.E. Kendall, "Systems Analysis and Design", 8th Edition.
- ✓ V. Rajaraman, "Analysis and Design of Information Systems", 2nd Edition, Prentice Hall of India, New Delhi, 2002
- ✓ Shelly Cashman, "System Analysis and Design" - 9th Edition.

Assignments:

(1) What are the needs of structured methodologies?
(2) What effect does data dictionaries causes while developing the system? Give your opinion.
(3) Explain the data dictionaries with example. (T.U. 2067)
(4) Explain the steps of creating a decision table. How can you reduce the size and complexity of a decision table? Explain with example. (T.U. 2067)
(5) Construct a decision table and decision tree that represents a sales person commission. The rules are as follows:
   (a) If fewer than 400 units are sold, then the sales person commission is 2% of total sales.
   (b) If between 400-499 units are sold, then the sales person commission is 3% of total sales

(c) If 500 or more units are sold, and the sales person has been has been employed by the company for 1 year or less, then the sales person commission is 4% of total sales
(d) If 500 or more units are sold, and the sales person has been with the company for more than 1 year, then the sales person commission is 5% of total sales (T.U. 2068)

(6) Explain with example the linkage between DFD, Decision Table, and ERD. (T.U. 2068)
(7) Explain modeling tools. (T.U. 2070)
(8) Develop the decision table and decision tree for the following:
The gatekeeper at ABC Park is given the following instructions for admitting persons to the park:
(a) If the person is under 3 years of age, there is no admission fee.
(b) If the person is under 16 years of age, half of the full admission fee is charged. This admission fee is reduced to quarter of the full admission fee if the person is accompanied by an adult. Note: Reduction is allowed only if the person is under 12 years of age.
(c) If the person is a student and between 16-18 years of age, half of the full admission fee is charged, otherwise full admission fee is charged.
(d) If the person is over 18 years of age, full admission fee is charged.
(e) A discount of 10% is allowed to a person above 16 years of age, if they come in a group of 10 persons or more in size.
(f) There is no student concessions during weekends. On weekdays, person under 12 year of age gets one free ride. (T.U. 2070)

A Gentle Advice:

Please go through your text books and reference books for detail study!!! Thank you all.

Notes Compiled By:

Bijay Mishra
biizay.blogspot.com
9813911076 or 9841695609

Unit 4 - System Analysis

4. System Analysis **8 Hrs.**

4.1 Systems Planning and Initial Investigation
4.2 Information Gathering Techniques (Interview and Questionnaire)
4.3 The Tools of Structured Analysis
4.4 Feasibility Study (Introduction, Steps in Feasibility Study)
4.5 Cost/Benefit Analysis (Direct and Indirect Cost, Tangible and Intangible Cost, Payback Period)

System Analysis

It is the third phase of the SDLC in which the current system is studied and alternative replacement systems are proposed. In this phase, the developer interacts with the user of the system to find out the user requirements and analyses the system to understand the functioning.

Based on this system study, the analyst prepares a model of the desired system. This model is purely based on what the system is required to do. At this stage the implementation details are not taken care of. Only the model of the system is prepared based on the idea that the system is made up of a set of interacting objects. The important elements of the system are emphasized.

System Planning and Initial Investigation

Strategic MIS Planning

Strategic MIS planning determines the basic objectives for the user to achieve the strategies and policies needed to achieve the objectives, and the tactical plans to implement the strategies.

At first, set the MIS objectives and in next step MIS policies are defined and which in turn, are translated into long-range (conceptual), medium-range (Managerial) and short-range (operational) plans for implementation.

Long Range Planning: General Courses of action to achieve strategic objectives.

Medium Range Planning: Action priorities, standards, procedures, and goals to achieve strategic objectives.

Short Range Planning: Performance targets, task schedules to achieve strategic objectives.

Managerial and Operational Planning

Managerial MIS planning integrates strategies with operational plans. It is a process in which specific functional strategies are to be carried out to achieve long-range plans. Examples are: Operating expense budget, Human Resource budget of each computer application etc.

b i i z a y @ g m a i l . c o m

The MIS operating plan requires the heaviest user involvement to define the system's requirements.

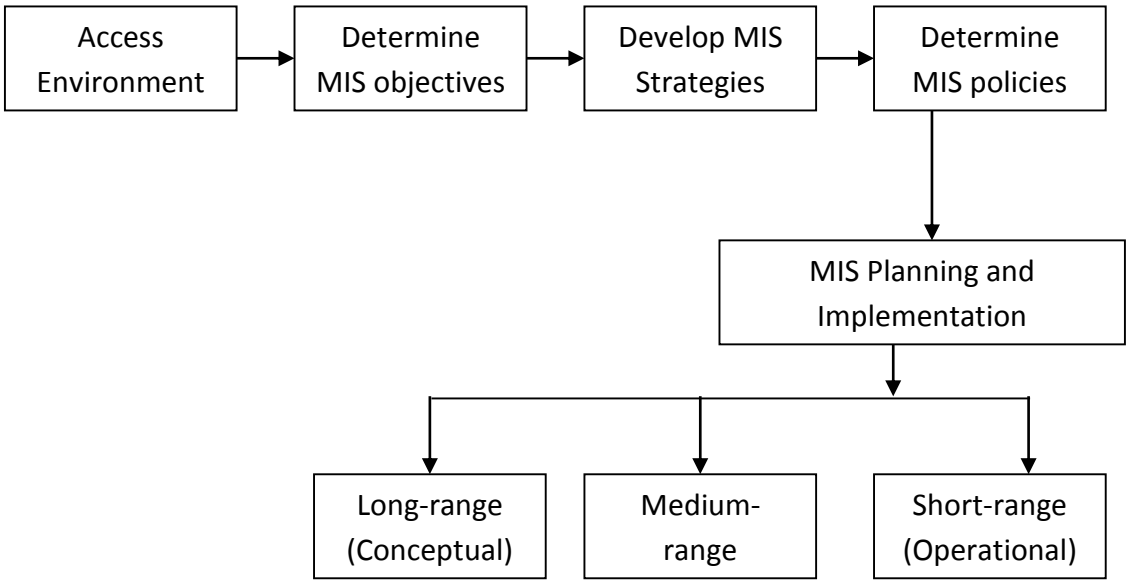


Figure: MIS Strategic Planning - Conceptual Model

Initial Investigation

The identification of a need is a user's request to change, improve or enhance an existing system because there is likely to be a stream of such request standard procedures must be established to deal with them. The initial investigation is one way of handling this. The objective is to determine whether the request is valid and feasible before the recommendation is reached to do nothing, improve or modifying an existing system or viewed a new one.

The user request identifies the need for change and authorizes the initial investigation. It may undergo seven modifications before it becomes a written commitment. Once the request is approved the following activities are carried out background investigation, fact finding and presentation of results, project proposal.

Steps in Initial Investigation

1. Needs Identification
2. Determining user's information requirements
3. Problem definition and project initiation
4. Background Analysis
5. Fact Finding
6. Fact Analysis
7. Determination of Feasibility

1. Need's Identification

The success of system depends on large end of how accurately a problem is defined, thoroughly investigated properly carried out through the choice of solution. User need identification and analysis are concerned with what the user need. The user or the analyst may identify the need

for the candidate or for an enhancement in the existing system. Shared, complete and accurate information requirements are essential in building computer based information system.

2. Determining user's information requirements

Some of the strategies for determining information requirements

There are three key strategies or general approaches for determining information regarding the user's requirements:

1. Asking
2. Getting information from the existing information system
3. Prototyping

a. Asking:

This strategy obtains information with users by simply asking them about the requirements. It assumes a stable system where users are well informed and can overcome biases defining their problems.

There are three key asking methods:

- *Questions* may be open ended or closed. An open ended question allows respondent to formulate to response. It is used when the feelings or ideas are important. In contrast a closed question request one answer from a specific set of responses. It is used when factual responses are known.

- *Brainstorming* is a technique used for generating new ideas and obtaining general information requirements.

- *Group consensus* asks participants for their expectations regarding specific variables.

b. Getting information from the existing information system*:*

Determining information from an existing application has been called the data analysis approach. It simply asks the user what information is correctly received and what other information is required. It realized heavily on the user to articulate information needs. The analyst examines all reports, discusses with the user each piece of information, examined and determines unfulfilled information needs by interviewing the user. The analyst is primarily involved in improving the existing flow of data to the user.

c. Prototyping:

The third strategy for determining user information requirement is used when the user can't establish information needs accurately before the information system is built. The reason could be the lack of an existing model on which to base requirement or a difficulty in visualizing candidate system. In this case, the user needs to anchor on real life systems from which adjustment can be made. Therefore the iterative discovery approach captures an initial set of information requirements and builds a system to meet these requirements. As users gain experience in its use, they request additional requirements or modifications in the system. In essence, information requirements are discovered by using the system, prototyping is suitable in environment where it is difficult to formulate a concrete model for defining information requirements and where the need of the user are evolving.

3. Problem Definition and Project Initiation

The first step in an initial investigation is to define the problem that led to the user request. The problem must be slotted clearly, understood and agreed upon by the user and the analyst. It must state the objectives the user is trying to achieve and the results the user wants to see, emphasis should be on the logical requirement what must be the results of the problems rather than the physical requirements.

4. Background Analysis

Once the project is initiated, the analyst begins to learn about the setting, the existing system, and the physical processes related to the revised system. For example: it is important to understand the structure of bank, who runs it, who reports to safe deposit area, there relationship and accounting in customer service and nature, frequency and level of interaction between the safe deposit and these departments. Therefore the analyst should prepare an organization chart with a list of the function and the people who perform them. In doing so, he/she would have a better feel for the work environment in which safe deposit operates the kinds of customers involved and the procedures employees following conducting business in safe deposit.

5. Fact Finding:

After performing the primary investigation the analyst begins to collect data on the existing system output, input, and cost. The tools used in data collection are

1. Review of written documents
2. On-site observations
3. Interviews and Questionnaires

a. Review of written documents:

When available, all documentation on data carriers (forms, records, reports, manuals etc.) is organized and evaluated. Included in procedures manuals are the requirement of the system which helps in determining to what extend they are met by the present system. Unfortunately, most manuals are not up-to-date or may not be readable. Day-to-day problems may have forced changes, which are not reflected in the manual. Further more people have a tendency to ignore procedures and find shortcuts, as long as the outcome is satisfactory.

Regarding existing forms, the analyst need to find out how they are filled out, how they are to the user, what changes to be made and how they are to be read.

b. On-site observation:

Another fact finding method used by the analyst is the on-site observation or direct observation. The analyst's role is that of an information seeker. One purpose of onsite observation is to get as close as the real system being studied. As an observer the analyst follows a set of rules. While making observation he/she is more likely to listen then talk and to listen with interest when information is passed on. He/she avoids giving advice, doesn't pass moral judgment on what is observed, doesn't argue with the user staff and doesn't show undue friendliness towards one but not others.

On-site observation is the most difficult fact finding technique. It requires entry into the user's area and can cause adverse reaction why the user's staff is not handled properly. The analyst

observes the physical layout of the current system, the location and the movement of people and the workflow.

c. Interviews and Questionnaires:

On-site observation is directed toward describing and understanding events and behavior as they occur. This method is, however, is less effective learning about people's perception, feelings and motivations. The alternative is the personal interview and the questionnaire. In either method, heavy reliance is placed on the interviewee's reports and for information about the job, the present system or experience. The quality of response is judged in the terms of its reliability and validity.

Reliability means that the information gathered is dependable enough to be used for making decisions about the system being studied. Validity means that the questions asked are so worded as to provide the intended information. So the reliability and the validity on the data gathered on the design of the interview or questionnaires and the manner in which each instrument is administered.

In an interview since the analyst (interviewer) and the person interviewed meet face to face, there is an opportunity for greater flexibility in getting information. The interviewer is also in a natural position to observe the subjects and the situations to which they are responding. In contrast the information obtained through a questionnaire is limited to the written response of the subject to predefine question.

6. Fact Analysis

As data are collected, they must be organized and evaluated and conclusion drawn for preparing a report to the user for the final review and approval. Many tools are used for data organization and analysis. These tools are input/ output analysis, decision tables and structured charts.

- *Input/output analysis* identifies elements that are related to the input and output of a given system. *Flow charts* and *dataflow diagrams* are excellent tools for input/output analysis.
- *Decision table* describes the data flow within a system that is generally used as a supplement when complex decision logic can't be represented clearly in a flow chart. As a documented tool, they can provide a simpler form of data analysis than the flow charts. When completed they are an easy to follow communication device between the technical and non-technical personnel. They are verbally oriented to managers, easy to learn and update and continue to function once the logic is developed.
- A *structure chart* is a working tool and an excellent way to keep track of the data collected for a system.

7. Determination of Feasibility

The outcome of the initial investigation is to determine whether an alternative system is feasible. Feasibility study is carried out to select the best system that meets the performance requirements. Approval of document initiates a feasibility study, which leads to the selection of the best candidate system.

Information Gathering

Also known as Requirement Elicitation / Requirement Gathering / Requirement Determination.

Information gathering is an art and a science, the approach and manner, in which information is gathered, requires persons with sensitivity, commonsense and knowledge of what and when to gather and what channels to use in securing information.

The information system designed for an organization must meet the requirements of the end users of the organization. To obtain what an end user expects from the information system the designer must gain complete knowledge of the organization's working. It is important for the analyst to know the information gathering techniques so that no information is overlooked and the nature and functions of an organization are clearly understood.

The main purpose of gathering information is to determine the information requirements of an organization. Information requirements are often not stated precisely by management. It is the analyst's responsibility to prepare a precise systems requirements specifications (SRS), which is easily understood by users, as SRS document is a vital document before starting a project.

Information Sources

- Users of System
- Forms and Documents used in the organization
- Procedure manuals, rule books etc.
- Reports used by the organization
- Existing computer programs (If Any)

Where does information originated?

Information is gathered from two principle sources- personal or written documents from within the organization and from the organization environment.

The primary external sources are:

1. Vendors
2. Government document
3. Newspapers and journals.

The primary internal sources are:

1. Financial report
2. User report
3. Manual report

Information Gathering Tools/Techniques/Methods

The primary information gathering tools are documentations, on-site observations, and review of literature, interviews and questionnaires.

Review of Literature, Procedure and Forms

Search the literature through professional references & procedure manuals, textbooks, company studies, government publications.

On-site observation

It is the process of recognizing & noting people, objects and occurrences to obtain information. The major objective of on-site observation is to get close to the "real" system being studied. The methods used may be natural/contrived, obtrusive/unobtrusive, direct/indirect, and structured/unstructured. The main limitation of observation is the difficulty of observing attitudes and motivation and the many unproductive hours that often are spent in observing one-time activities.

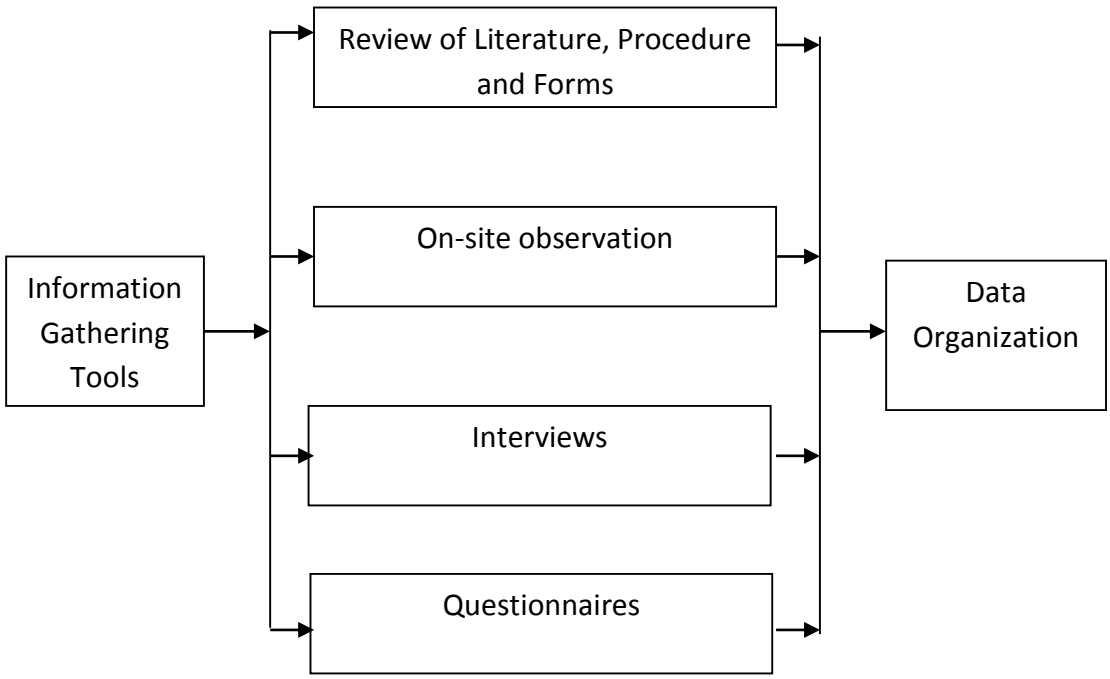


Figure: Information Gathering Methods

Interviewing

An interview is a conversation between two people (the interviewer and the interviewee) where questions are asked by the interviewer to obtain information from the interviewee

Interviewing is one of the primary ways analysts gather information about an information system project. It is the main approach used to analyze large structured systems. During interviewing you will gather facts, opinions, and speculation and observe body language, emotions, and other signs of what people want and how they assess current systems.

Interviews are the formal meetings where the analyst can collect information about the working of the present system and the requirement of any planned replacement.

It can be used for a variety of different purposes as:

1. To collect information about the procedure taking place in an organization.
2. To verify the analyst understanding of system operations with users of all levels.
3. To confirm aspects of a purposed system design.
4. To build confidence in the design of a new information.

Advantages of Interviews:

- The flexibility of interview makes it a superior technique for enquiring into areas which are difficult to be explored using any other technique.
- It provides a better opportunity then the questionnaires to evaluate the validity of the information collected. The interviewer can not only observe what subjects but also study their body language and expression.
- It is an affective technique for getting information about complex subjects and for sensing the feelings underline opinions.
- Generally people prefer being interviewed rather than filling up lengthy questionnaires.
- Interviews gives the analyst an opportunity to motivate the interviewee to respond freely and openly to questions.
- Interviews allow the system analyst to probe (penetrating investigation) for more feedback from the interviewee.
- Interviews permit the system analyst to adapt or revise questions for each individual.
- A good system analyst may be able to obtain information by observing the interviewee's body movements and facial expressions as well as listening to verbal replies to questions.

Disadvantages of Interviews:

- This method requires more time and money for its conduction.
- Time-consuming, it takes long preparation time.
- Success highly dependent on analyst's human relations skills
- May be impractical due to location of interviewees

Steps for Successful Interview

1. Set the stage for interview
2. Establish rapport
3. Asking the questions
4. Obtaining and recording the response
5. Evaluate the outcome of interview.

The above steps are explained in three parts:

Planning the Interview

Before starting the interview the analyst must make a list of people to be interviewed and in what order, plan and note down a list of questions to be asked, plan several interviews with same person-mainly to clarify doubts and interview groups as appropriate. The three crucial aspects of the interview need to be plan by the analyst. These aspects are:-

1. The object of the interview
2. The time and the venue of the interview
3. The authorization for the interview

Conducting the interview

The analyst should start the interview by giving a clear explanation of the purpose of the interview. The analyst should also specify by the subject first selected and assure that the information collected during interview will be treated as confidential.

Concluding the interview
It is important to stick to the agreed time and conclude the interview with a summary of the main discussion point.

Guidelines to a Successful Interview:
There are some guidelines to ensure a successful interview:

- Make a prior appointment with the person to be interviewed and meet him at the allotted time.
- Read background material and go prepared with the checklist.
- State purpose of interview
- Be punctual and pay attention to what user says.
- Do not use computer jargon.
- Obtain both quantitative and qualitative Information.
- Discriminate between essential and desirable requirements.
- State what you understand and get it confirmed.
- Do not prolong interview and summarize the information gathered by you during the interview and verify this with the user

Questionnaires
The questionnaire is considered as a tool for gathering required information. The use of questionnaires is an information-gathering technique that allows systems analysts to study attitudes, beliefs, behavior, and characteristics of several key people in the organization who may be affected by the current and proposed systems. Attitudes are what people in the organization say they want (in a new system, for instance); beliefs are what people think is actually true; behavior is what organizational members do; and characteristics are properties of people or things. In situation when it is not possible for the analyst (because of time, distance or cost) to interview persons involved in a system.

Responses gained through questionnaires (also called surveys) using closed questions can be quantified. Responses to questionnaires using open-ended questions are analyzed and interpreted in other ways. Through the use of questionnaires, the analyst may be seeking to quantify what was found in interviews.

Advantages of Questionnaire
A questionnaire offers the following advantages:

1. It is easier to administer as compared to the interview as economical.
2. A wide geographical area can be covered and a large number of people can be contacted using this method.
3. The logical order of the question and the standardized instruction for giving answers ensure uniformity of questions.
4. The respondents feel free to express themselves in questionnaires then in an interview.
5. With questionnaire the respondent does not feel pressure of answering the question there and then. They get enough time to think about the question and the response.

Disadvantages of Questionnaire
A questionnaire offers the following disadvantage:

1. Many people are not good at writing
2. Low percentage of returns
3. Lack of response from some can bias results
4. Respondents can be unknown
5. Respondents is passive
6. Information collected is less rich

Types of interviews and questionnaires

Interviews can be highly unstructured where neither the questions nor the respective responses are specified in advance and highly unstructured in which the questions and responses are predefined.

1. Unstructured approach

The unstructured interview allows respondents to answer questions at will in their own words. The responses are uninhibited rather than forced. They are self and reveling and personal rather than general or superficial. The analyst assumes the role of an interviewer who encourages the respondent to speak without any hesitation and helps them to express their feelings and opinion. This is best achieved when the subject are willing to share and have no feeling of disapproval. However, this approach is more time consuming.

Advantages

- Provides for greater creativity and spontaneity in interviewing
- Facilitates deeper understanding of the feelings and standing of the interviewee.
- Offers greater flexibility in conducting an overall interview

Disadvantages

- More information of questionable use is gathered
- Takes more time to conduct (so it is costly)
- Requires extensive training & experience for effective results

2. Structured approach

In this, all subjects are asked the identical questions in a set order. This is done to ensure that all subjects are answering identical questions which include the reliability of the responses.

Structured interviews and questionnaires may have questions of either closed or open ended type. An open ended question is one which does not need a specific response while a closed ended question is one in which the responses are given as a set of options.

Advantages

- Easy to administer & evaluate due to standardization
- Requires limited training
- Easy to train new staffs.

Disadvantages

- High initial preparation cost
- Standardization of questions tends to reduce spontaneity
- Mechanizes interviewing which makes it impractical for all interview settings.

How to get information from Interviews?

Interviewers use several types of questions to extract the information they seek during the hiring process. It is important that you identify the question type used as this will help you understand how the question is being asked and how it should be answered.

When developing interview questions, consider what types of question would best give you the information you are seeking. Listed below are different types of interview questions:

Open-ended Questions

Open-ended questions (or statements) are those that leave all possible response options open to the respondent. They are conversational questions with no specific answers in mind. A system analyst will want to use open-ended questions when it is impossible to list effectively all the possible responses to the question. Open-ended questions are basic to any effective interview because they call for interviewees to relate information and ideas that they feel are important. An open question is likely to receive a long answer.

Example:

1. What collaboration strategies lead to good teamwork?
2. What are some of the common data entry errors made in this department?
3. Describe any problems you are currently experiencing with output reports.
4. In your opinion, how helpful are the user manuals for the current system's accounting application?
5. I wonder what would happen if your customers complained even more?
6. What is the biggest problem you have when communicating your information requirements to headquarters? Describe it briefly.

Advantages of Open-ended Questions:

The benefits of using open-ended questions are as follows:

1. Putting the interviewee at ease.
2. Allowing the interviewer to pick up on the interviewee's vocabulary, this reflects his or her education, values, attitudes, and beliefs, etc.
3. Providing richness of detail.
4. Revealing avenues of further questioning that may have gone untapped.
5. Making it more interesting for the interviewee.
6. Allowing more spontaneity.
7. Making phrasing easier for the interviewer.
8. Using them in a pinch if the interviewer is caught unprepared.

Disadvantages of Open-ended Questions:

There are, however, also many drawbacks of open-ended questions:

1. Asking questions that may result in too much irrelevant detail.
2. Possibly losing control of the interview.
3. Allowing responses that may take too much time for the amount of useful information gained.
4. Potentially seeming that the interviewer is unprepared.
5. Possibly giving the impression that the interviewer is on a "fishing expedition" with no real objective for the interview.

Closed-ended Questions

The alternative to open-ended questions is found in the other basic question type: closed questions. Closed questions (or statements) are those that limit or close the response options available to the respondent. They are structured questions with limited range of possible answers. A closed question can be answered with either a single word or a short phrase.

A closed question limits the response available to the interviewee. You may be familiar with closed questions through multiple-choice exams in college. You are given a question and five responses, but you are not allowed to write down your own response and still be counted as having correctly answered the question.

A special kind of closed question is the bipolar question. This type of question limits the interviewee even further by only allowing a choice on either pole, such as:

- ✓ Yes/No
- ✓ Agree/Disagree
- ✓ True/False

A closed question can be answered with either:

- ✓ Multiple Choices
- ✓ Rating a response
- ✓ Ranking items

Example:

1. How old are you? ……
2. Bijay Mishra sir teaches system analysis and design course in BSc.CSIT, BCA, BIT and BE Computer on various colleges. True/False
3. Do you use the Web to provide information to vendors? Yes/No
4. Does your Web site maintain a FAQ page for employees with payroll questions? Yes/No
5. Answer question by circling the appropriate number. My educational background can best be described as:
    - SLC
    - Intermediate /Plus 2
    - Bachelor's Degree
    - Master's Degree
6. Answer question by circling the appropriate number. "When the sales figures are prepared by computer data services they are late."

Never	Rarely	Sometimes	Often	Always
1	2	3	4	5

7. The ecommerce on the Web lacks security.
    [ ] Agree [ ] Disagree
8. Rate the following: The implementation of quality discounts would cause an increase in customer orders.
    ___ Strongly agree.
    ___ Agree.
    ___ No opinion.
    ___ Disagree.
    ___ Strongly disagree.

9. Rank the following transactions according to the amount of time you spend processing them.
   ___ % new customer orders
   ___ % order cancellations
   ___ % order modifications
   ___ % payments

10. Below are the six software packages currently available. Please check the software package(s) you personally use most frequently.
    [ ] Microsoft Excel [ ] Microsoft Access
    [ ] Microsoft PowerPoint [ ] Microsoft Windows Live Mail
    [ ] Oracle SCM [ ] Visible Analyst

How do you rate the following?

	very poor	*Poor*	*OK*	*Good*	*Very good*
a Service	☐	☐	☐	☐	☐
b Cleanliness	☐	☐	☐	☐	☐
c Parking	☐	☐	☐	☐	☐
d Quality of Food	☐	☐	☐	☐	☐
e Choice of Food	☐	☐	☐	☐	☐

Using a scale of 0= Not at all important to 5=Very Important, please rate the following aspects of our service in the restaurant?

	Not at all important 0	1	2	3	4	Very Important 5	No Opinion
Speed of Service	☐	☐	☐	☐	☐	☐	☐
Friendliness of Staff	☐	☐	☐	☐	☐	☐	☐
Helpfulness of Staff	☐	☐	☐	☐	☐	☐	☐
Value for Money	☐	☐	☐	☐	☐	☐	☐

Please rank the following in order of importance from 1 to 4 where 1 is most important to you and 4 is least important to you

Speed of Service ☐
Ease of Parking ☐
Cleanliness ☐
Friendliness of Staff ☐

How would you rate the following?

	Poor	Average	Good
Product	○	○	○
Sales	○	○	○

Rate the performance of your immediate supervisor.

	Excellent	Good	Average	Fair	Poor
Clear Communication of instructions on a regular basis	1	2	3	4	5
Fair and consistent administration of company policies	1	2	3	4	5
Supervisor's overall knowledge & competence	1	2	3	4	5
Willingness to admit and correct mistakes	1	2	3	4	5
Willingness of manager to provide positive recognition	1	2	3	4	5
Willingness to provide appropriate training opportunities	1	2	3	4	5

Department members cooperate to meet goals.

- Strongly Agree
- Agree
- Slightly agree
- Neutral
- Slightly Disagree
- Disagree
- Strongly Disagree

Advantages of Closed-ended Questions:

The benefits of using closed questions of either type include the following:

1. Saving time.
2. Easily comparing interviews.
3. Getting to the point.
4. Keeping control over the interview.
5. Covering lots of ground quickly.
6. Getting to relevant data.

Disadvantages of Closed-ended Questions:

The drawbacks of using closed questions are substantial, however. They include the following:

1. Being boring for the interviewee.
2. Incomplete responses fails to obtain rich detail (because the interviewer supplies the frame of reference for the interviewee).
3. Missing main ideas for the preceding reason.
4. Failing to build rapport between interviewer and interviewee.

Thus, as the interviewer, you must think carefully about the question types you will use.

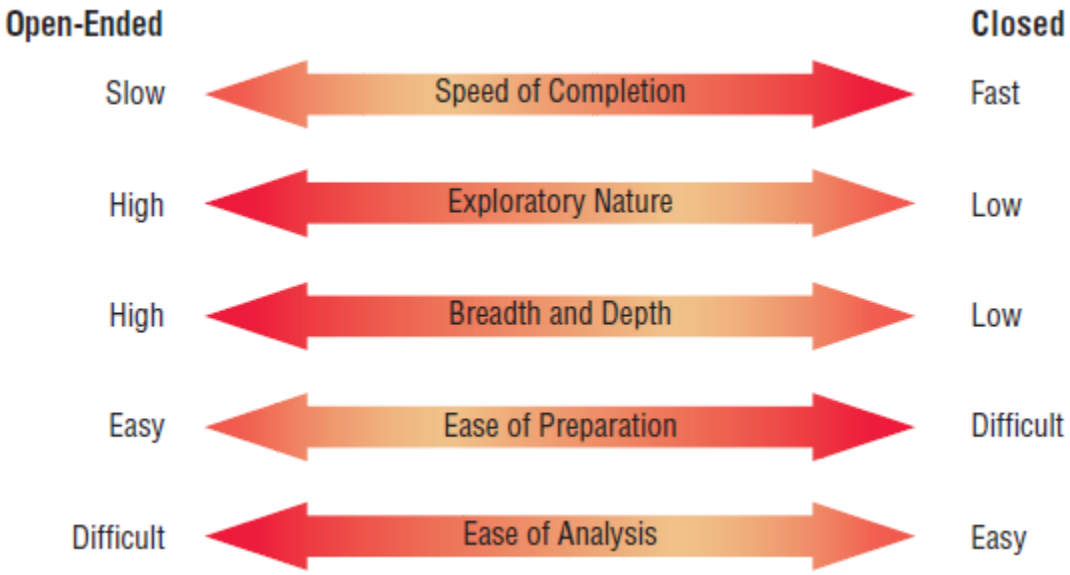


Figure: Differences between open-ended and closed-ended questions.

Probe

A third type of question is the probe or follow-up. Often, we want or need more information than we get when we ask a question during an interview. *Probing* is asking follow-up questions when we do not fully understand a response, when answers are vague or ambiguous or when we want to obtain more specific or in-depth information.

Interviewers should also have several follow-up questions and probe for details that explore all aspects of a given situation or experience. In qualitative interviewing, probes cannot be planned in advance.

Example:

1. When is your birthday? How do you like to celebrate?
2. What are the most frequent problems you experience with computer output?
   A ……..
   B ……..
   C ……..
   Of the problems you listed above, what is the single most troublesome? Why?
3. Is that relevant to the main question? How is "what you are saying related to what I asked"?
4. Sorry, I don't understand. Could you help by giving an example?
5. Which would you prefer of A or B? Why?
6. What about the length of time the test takes? Does that matter?
7. Have you ever smoked marijuana? If yes, about how many times have you smoked it?

The Tools of Structured Analysis

Structured analysis is a set of techniques and graphical tools that allow the analyst to develop a new kind of system specification that are easily understandable to the user. The traditional approach focuses on the cost benefits and feasibility analysis, project management, hardware and software selection and personal consideration. In contrast, structured analysis considers new goals and structured tools for analysis.

Structured analysis is a development method for analysis of an existing system. It is a set of techniques that allow the analyst to design the proposed system. The main purpose of structured analysis is to completely understand the current system.

The new goals specify the following:
1. Use graphics wherever possible to help communicate pattern with the user.
2. Differentiate between logical and physical system.
3. Build a logical system model to familiarize the user with system characteristic and interrelationship before implementation.

The structured tool focuses on the following tools:
   a. Dataflow Diagrams
   b. ER diagrams
   c. Data Dictionary
   d. Structured English
   e. Decision Trees
   f. Decision Tables

The objective is to build a new document called system specification.

Data Flow Diagram (DFD)
Data flow diagrams are widely used graphic tools for describing the movement of data within or outside the system. As a DFD consists of a series of bubbles joined by lines, it is also known as a 'bubble chart'.

Note: *See notes provided by me on "Chapter 2 - Modeling Tools for Systems Analyst" for Data Flow Diagram.*

ER Diagram (ERD)
An entity-relationship diagram (or, E-R diagram) is a detailed, logical, and graphical representation of the data for an organization or business area. It is a graphical representation of an E-R model. ERD illustrate the logical structure of databases.

Note: *See notes provided by me on "Chapter 2 - Modeling Tools for Systems Analyst" for Entity Relationship Diagram.*

Data Dictionary (DD)
Data dictionary is an organized list of terms and their definitions for all the data elements and data structures that are pertinent to the system. It stores the names along with their descriptions of all data used in a system.

Note: *See notes provided by me on "Chapter 3 - Structured Methodologies" for Data Dictionary.*

Structured English
It is a tool for sharpening the narrative for described processing logic and procedures. It is based on the principle of the structured programming. It is created by the merging of the

English language with the syntax of the structured programming. It tries to express the verbal statement in the more logical form.

Note: *See notes provided by me on "Chapter 3 - Structured Methodologies" for Structured English.*

Decision Table

A decision table appears as a matrix of rows and columns that shows conditions and corresponding actions. Decision rules, included in a decision table, states what procedure is to follow when certain condition exist. This method has been used in analysis of business functions such as investing control, sales analysis, credit analysis and transportation control and routing.

Note: *See notes provided by me on "Chapter 3 - Structured Methodologies" for Decision Table.*

Decision Tree

It is a graphical technique that shows a decision situation as a linked series of nodes and branches. Decision tree turns a decision table into a diagram. A decision tree is a diagram that resembles a tree, with a root on the left hand side and branches representing each decision. It is read from left to right and the actions to be undertaken are recorded down the right hand side of the diagram.

Note: *See notes provided by me on "Chapter 3 - Structured Methodologies" for Decision Tree.*

Feasibility Study

Once the number of projects has been narrowed according to the criteria discussed previously, it is still necessary to determine if the selected projects are feasible. Our definition of feasibility goes much deeper than common usage of the term, because systems projects feasibility is assessed in three principal ways: operationally, technically, and economically. The feasibility study is not a full-blown systems study. Rather, the feasibility study is used to gather broad data for the members of management that in turn enables them to make a decision on whether to proceed with a systems study.

Data for the feasibility study can be gathered through interviews. The kind of interview required is directly related to the problem or opportunity being suggested. The systems analyst typically interviews those requesting help and those directly concerned with the decision-making process, typically management. Although it is important to address the correct problem, the systems analyst should not spend too much time doing feasibility studies, because many projects will be requested and only a few can or should be executed.

Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of the existing business or proposed venture, opportunities and threats as presented by the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest term, the two criteria to judge feasibility are cost required and value to be attained. As such, a well-designed feasibility study should provide a historical background of the business or project, description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal

requirements and tax obligations. Generally, feasibility studies precede technical development and project implementation.

Objectives of Feasibility Study

The main objectives of feasibility study are:

1. To identify the deficiencies in the current system.
2. To determine objectives of the proposed system.
3. To acquire a sense of scope of the system.
4. To identify the responsible users.
5. To determine whether it is feasible to develop the new system.

The feasibility study must be highly time-compressed, encompassing several activities in a short Span of time. After an analyst determines reasonable objectives for a project, the analyst needs to determine if it is possible for the organization and its members to see the project through to completion. Generally, the process of feasibility assessment is effective in screening out projects that are inconsistent with the business's objectives, technically impossible, or economically without merit.

Although it is painstaking, studying feasibility is worthwhile because it saves businesses and systems analysts time and money. In order for an analyst to recommend further development, a project must show that it is feasible in all three of the following ways: technically, economically, and operationally.

Technical Feasibility

Technical feasibility analysis questions whether the hardware, software, and other technologies needed for the project to exist in the organization or it has to be acquired or it needs to be developed?

In technical feasibility the following issues are taken into consideration.

- Whether the required technology is available or not
- Whether the required resources are available
  - Manpower- programmers, testers & debuggers
  - Software and hardware

Once the technical feasibility is established, it is important to consider the monetary factors also. Since it might happen that developing a particular system may be technically possible but it may require huge investments and benefits may be less. For evaluating this, economic feasibility of the proposed system is carried out.

Example: If a project requires coding in a new programming language and none of the personnel knows the language, the organization must consider the feasibility of training existing personnel or hiring consultants. If these options prove too costly or take too long, the project may not be technically feasible.

Economic Feasibility

For any system if the expected benefits equal or exceed the expected costs, the system can be judged to be economically feasible. In economic feasibility, cost benefit analysis is done in

which expected costs and benefits are evaluated. Economic analysis is used for evaluating the effectiveness of the proposed system. In economic feasibility, the most important is cost-benefit analysis. As the name suggests, it is an analysis of the costs to be incurred in the system and benefits derivable out of the system.

The concerned business must be able to see the value of the investment it is pondering before committing to an entire systems study. If short-term costs are not overshadowed by long-term gains or produce no immediate reduction in operating costs, the system is not economically feasible and the project should not proceed any further.

Operational Feasibility

Operational feasibility is dependent on the human resources available for the project and involves projecting whether the system will operate and be used once it is installed.

Operational feasibility is mainly concerned with issues like whether the system will be used if it is developed and implemented. Whether there will be resistance from users that will affect the possible application benefits?

The essential questions that help in testing the operational feasibility of a system are as follows:

- "Does management support the project?
- "Are the users not happy with current business practices? Will it reduce the time (operation) considerably? If yes, then they will welcome the change and the new system.
- "Have the users been involved in the planning and development of the project? Early involvement reduces the probability of resistance towards the new system.
- "Will the proposed system really benefit the organization? Does the overall response increase? Will accessibility of information be lost? Will the system effect the customers in considerable way?

Some other feasibility that needs to be kept under consideration are:

Social Feasibility

Social feasibility is a determination of whether a proposed project will be acceptable to the people or not. This determination typically examines the probability of the project being accepted by the group directly affected by the proposed system change. The effect of the project on the social status of the project participants must be assessed to ensure compatibility. It should be recognized that workers in certain industries may have certain status symbols within the society.

Legal Feasibility

It determines whether the proposed system conflicts with legal requirements. It is a determination of whether a proposed project infringes on known Acts, Statutes, as well as any pending legislation. Although in some instances the project might appear sound, on closer investigation it may be found to infringe on several legal areas. Example, a data processing system must comply with the local Data Protection Acts. It includes study concerning contracts, liability, violations, and legal other traps frequently unknown to the technical staff. Whatever the system, it has to be legally approved before we implement the system.

Management Feasibility
It is a determination of whether a proposed project will be acceptable to management. If management does not accept a project or gives a negligible support to it, the analyst will tend to view the project as a non-feasible one.

Schedule Feasibility (or Time Feasibility)
This involves questions such as how much time is available to build the new system, when it can be built (i.e. during holidays), whether it interferes with normal business operation, etc. A project will fail if it takes too long to be completed before it is useful. Typically this means estimating how long the system will take to develop, and if it can be completed in a given time period using some methods like payback period. Schedule feasibility is a measure of how reasonable the project timetable is. Given our technical expertise, are the project deadlines reasonable? Some projects are initiated with specific deadlines. You need to determine whether the deadlines are mandatory or desirable.

Market and Real Estate Feasibility
Market feasibility study typically involves testing geographic locations for a real estate development project, and usually involves parcels of real estate land. Developers often conduct market studies to determine the best location within a jurisdiction, and to test alternative land uses for given parcels. Jurisdictions often require developers to complete feasibility studies before they will approve a permit application for retail, commercial, industrial, manufacturing, housing, office or mixed-use project. Market feasibility takes into account the importance of the business in the selected area.

Resource Feasibility
This involves questions such as how much time is available to build the new system, when it can be built, whether it interferes with normal business operations, type and amount of resources required, dependencies, etc.

Cultural Feasibility
In this stage, the project's alternatives are evaluated for their impact on the local and general culture. Cultural feasibility deals with the compatibility of the proposed project with the cultural setup of the project environment. In labor-intensive projects, planned functions must be integrated with the local cultural practices and beliefs. For example, religious beliefs may influence what an individual is willing to do or not do.

Political Feasibility
A politically feasible project may be referred to as a "politically correct project." Political considerations often dictate direction for a proposed project. This is particularly true for large projects with national visibility that may have significant government inputs and political implications. For example, political necessity may be a source of support for a project regardless of the project's merits.

On the other hand, worthy projects may face insurmountable opposition simply because of political factors. Political feasibility analysis requires an evaluation of the compatibility of project goals with the prevailing goals of the political system.

Steps in Feasibility Analysis

Feasibility analysis is carried out in following steps:

1. Form a Project Team and Appoint a Project Leader: First of all project management group of the organization forms separate teams for independent projects. Each project team comprises of one or more systems analysts and programmers with a project leader. The project leader is responsible for planning and managing the development activities of the system.

2. Start Preliminary Investigation: The systems analyst of each project team starts preliminary investigations through different fact finding techniques.

3. Prepare the Current Systems Flowchart: After preliminary investigations, the analysts prepare the systems flowchart of the current system. These charts describe the general working of the system in a graphical way.

4. Describe the Deficiencies in the Current System: On studying the systems flowcharts, the analysts identify and describe the deficiencies in the current system.

5. Determine Objectives of the Proposed System: The major objectives of the proposed systems are listed by each analyst and are discussed with the project leader.

6. Prepare the Proposed Systems Flowchart: After determining the major objectives of the proposed system, the analysts prepare their systems flowcharts. Systems flowcharts of the proposed system are compared with those of current system in order to ensure that they meet the objectives.

7. Determine the Technical Feasibility: The existing computer systems (hardware and software) of the concerned department are identified and their technical specifications are noted down. The analysts decide whether the existing systems are sufficient for the technical requirements of the proposed system or not.

8. Determine the Economic Feasibility: The analysts determine the costs and benefits of proposed system in order to ensure that the project is economically feasible.

9. Determine the Operational Feasibility: After determining the economic feasibility, the analysts identify the responsible users of the system and hence determine the operational feasibility of the project.

10. Presentation of Feasibility Analysis: During the feasibility study, the analysts also keep on preparing the feasibility study report. At the end of feasibility analysis, the feasibility analysis report is given to the management along with the oral presentation.

Feasibility Report

After the feasibility study, a document is prepared that is known as “Feasibility Study Report”. Besides this report, the analyst also gives the oral presentation of feasibility study to the management.

Feasibility report is a formal document for management use and is prepared by systems analyst during or after feasibility study. This report generally contains the following sections:

(a) *Cover letter*: It formally presents the report with a brief description of the project problem along with recommendations to be considered.

(b) *Table of Contents*: It lists the sections of feasibility study report along with their page numbers.

(c) *Overview*: It presents the overview of the project problem along with the purpose and scope of the project.

(d) *Description of Existing System*: A brief description of the existing system along with its deficiencies are presented in this section.

(e) *System Requirements*: The system requirements, which are either derived from the existing system or from the discussion with the users, are presented in this section.

(f) *Description of Proposed System*: It presents a general description of the proposed system, highlighting its role in solving the problem. A description of output reports to be generated by the system is also presented in the desired formats.

(g) *Development Plan*: It presents a detailed plan with starting and completion dates for different phases of SDLC. A complementary plan is also needed for hardware and software evaluation, purchase and installation.

(h) *Technical Feasibility Findings*: It presents the findings of technical feasibility study along with recommendations.

(i) *Costs and Benefits*: The detailed findings of cost and benefits analysis are presented in this section. The savings and benefits are highlighted to justify the economic feasibility of the project.

(j) *Operational Feasibility Findings*: It presents the findings of operational feasibility along with the human resource requirements to implement the system.

(k) *Alternatives Considered/Rejected*: The different alternatives that an analyst usually considers and rejects during feasibility study, should also be included in the feasibility study report. These alternatives are required to be discussed because they show, how the suggested system is the best alternative to solve the problem.

(l) *Recommendations and Conclusions*: The benefits and savings are summarized and it is recommended whether the management should decide to proceed with the project or abort the project.

(m) *Appendixes*: In the last section of feasibility study report, all memos, documents and data compiled during study are enclosed for reference.

Oral Presentations

Although, feasibility study report is the best written presentation of the proposed system, it is expected that the analyst gives an oral presentation to the management and end users. During oral presentation, many issues can be clarified and new ideas from the users can be picked up by the analyst.

Cost/Benefit Analysis (CBA)

Costs and benefits of the proposed computer system must always be considered together, because they are interrelated and often interdependent. The costs and benefits are used to determine whether a project is economically feasible. Although the systems analyst is trying to propose a system that fulfills various information requirements, decisions to continue with the proposed system will be based on a cost-benefit analysis, not on information requirements.

The sum value of costs of items needed to implement the system is known as the ***cost*** of the system. The sum value of the savings made is known as the ***benefits*** of the new system. We can define benefit as: Profit or Benefit = Income – Costs.

Since cost plays quite an important role in deciding the new system, it must be identified and estimated properly.

The benefits of a project include four types:

1. Cost-savings benefits
2. Cost-avoidance benefits
3. Improved-service-level benefits
4. Improved Information benefits

Cost-savings benefits

Cost-savings benefits lead to reduction in administrative and operational costs. A reduction in the size of the clerical staff used in the support of an administrative activity is an example of a cost-saving benefit.

Cost-avoidance benefits

Cost-avoidance benefits are those, which eliminate future administrating and operational costs. No need to hire additional staff in future to handle an administrative activity is an example of a cost-avoidance benefit.

Improved-service-level benefits

Improved-service-level benefits are those where the performance of a system is improved by a new computer-based method.

Improved Information benefits

Improved-information-benefit is where computer based methods lead to better information for decision-making. For example, a system that reports the most-improved fifty customers as measured by an increase in sales is an improved-information. This information makes it easier to provide better service to major customers.

The cost of a project include five types:
In developing the cost estimates for a system, we need to consider several cost elements. Among them we need to consider:

1. Hardware Costs
2. Personnel Costs
3. Facility Costs
4. Operating Costs
5. Supply Costs

Hardware Costs
Hardware costs are related to actual purchase or lease of the computer and peripherals. (Example: Printer, Hard Disk, etc.)

Personnel Costs
Personnel costs are staff salaries and benefits paid for those involved in developing the system. E.g. Health Insurance, Vacation, Sick pay etc.

Facility Costs
It includes facilities like wiring, flooring, acoustics, lightening, air conditioning etc.

Operating Costs
It includes all costs associated with the day-to-day operation of a system.

Supply Costs
It includes variable costs that increase with increased use of paper, ribbons, disk, etc.

Classification of Costs and Benefits

The costs associated with the system are expenses, outlays or losses arising from development and using a system. But the benefits are the advantages received from installing and using this system.

Costs and benefits can be classified as follows:
(a) Tangible or intangible
(b) Fixed or variable
(c) Direct or indirect

Tangible or Intangible Costs and Benefits

Tangible Cost:
Tangibility refers to the ease with which costs or benefits can be measure. Therefore tangible cost and benefits can be measured. An outlay of cash for any specific item or activity is referred to as a tangible cost. These costs are known and can be estimated quite accurately. Hardware costs, salaries for professionals, software cost are all tangible costs. Included tangible costs are the cost of equipment such as computers and terminals, the cost of resources, the cost of systems analysts' time, the cost of programmers' time, and other employees' salaries. These costs are usually well established or can be discovered quite easily, and are the costs that will require a cash outlay of the business.

Intangible Cost:
Costs that are known to exist but their financial value cannot be exactly measured are referred to as intangible costs. The estimate is only an approximation. It is difficult to fix exact intangible costs. For example, employee morale problem because of installing new system is an intangible cost. How much moral of an employee has been affected cannot be exactly measured in terms of financial values. The cost of breakdown of an online system during banking hours will cause the bank lose deposits, loss of customer goodwill etc. Intangible costs include losing a competitive edge, losing the reputation for being first with an innovation or the leader in a field, declining company image due to increased customer dissatisfaction, and ineffective decision making due to untimely or inaccessible information.

Tangible Benefits:
Benefits are often more difficult to specify exactly than costs. For example, suppliers can easily quote the cost of purchasing a terminal but it is difficult for them to tell specific benefits or financial advantages for using it in a system. Tangible benefits such as improved response time, completing jobs in fewer hours or producing error free reports are quantifiable. Examples of few other tangible benefits are an increase in the speed of processing, access to otherwise inaccessible information, access to information on a more timely basis than was possible before, the advantage of the computer's superior calculating power, and decreases in the amount of employee time needed to complete specific tasks.

Intangible Benefits:
Intangible benefits such as more satisfied customers or an improved corporate image because of using new system are not easily quantified. Intangible benefits include improving the decision-making process, enhancing accuracy, becoming more competitive in customer service, maintaining a good business image, and increasing job satisfaction for employees by eliminating tedious tasks.

Tangible Benefits	Intangible Benefits
Increased Sales	Increased Market Share
Reductions in Staff	Increased Brand Recognition
Reductions in Inventory	Higher Quality Products
Reductions in IT Costs	Improved Customer Service
Better Supplier Prices	Better Supplier Relations

Figure: Example of Tangible and Intangible Benefits

Direct or Indirect Costs and Benefits

Direct Costs:
Direct costs are those which are directly associated with a system. Direct costs are having rupee value associated with it. They are- applied directly to the operator. For example, the purchase of pen drive for NRs. 2,000/- is a direct cost because we can associate the pen drive with money spent.

b i i z a y @ g m a i l . c o m

Indirect Costs:
Indirect costs result from the operations that are not directly associated with the system. They are often referred to as overhead expenses. For example, cost of space to install a system, insurance, maintenance of computer center, heat, light and air-conditioning are all indirect costs, but it is difficult to calculate the proportion of each attributable to a specific activity such as a report.

Direct Benefits
Direct benefits also can be specifically attributable to a given project. For example, if the proposed system that can handle more transactions say 25% more than the present system then it is direct benefit.

Indirect Benefits:
Indirect benefits are realized as a by-product of another system. For example, a system that tracks sales-calls on customers provides an indirect marketing benefit by giving additional information about competition. In this case, competition information becomes an indirect benefit although its work in terms of money cannot be exactly measured.

Fixed or Variable Costs and Benefits

Fixed Costs:
Some costs and benefits remain constant, regardless of how a system is used. Fixed costs are considered as sunk costs. They don't change. Once encountered, they will not recur. For example, the purchase of equipment for a computer center is called as fixed cost as it remains constant whether in equipment is being used extensively or not. Similarly, the insurance, and purchase of software, etc.

Variable Costs:
In contrast, variable costs are incurred on a regular basis. Recurring period may be weekly or monthly depending upon the system. They are generally proportional to world volume and continue as long as the system is in operation. For example, the cost of computer forms vary in proportion to the amount of processing or the length of the reports desired.

Fixed Benefits:
Fixed benefits also remain constant by using a new system. They also don’t change. If 20% of staff member are reduced, we can call it a fixed benefit.

Variable Benefits:
Variable benefits, on the other hand, are realized on a regular basis. For example the library information system that saves two minutes in providing information about particular book whether it is issued or not, to the borrower compared with the manual system. The amount of time saved varies with the information given to the number of borrowers.

Comparing Costs/Benefits Analysis and Method of Evaluation for CBA

There are many well-known techniques for comparing the costs and benefits of the proposed system. They include:

1. Net Benefit Analysis
2. Present Value Analysis
3. Net Present Value
4. Payback Period
5. Break Even Analysis
6. Cash Flow Analysis

All these techniques provide straightforward ways of yielding information to decision makers about the worthiness of the proposed system.

Break-Even Analysis (BEA)

By comparing costs alone, the systems analyst can use break-even analysis to determine the break-even capacity of the proposed information system. The point at which the total costs of the current system and the proposed system intersect represents the break-even point, the point where it becomes profitable for the business to get the new information system.

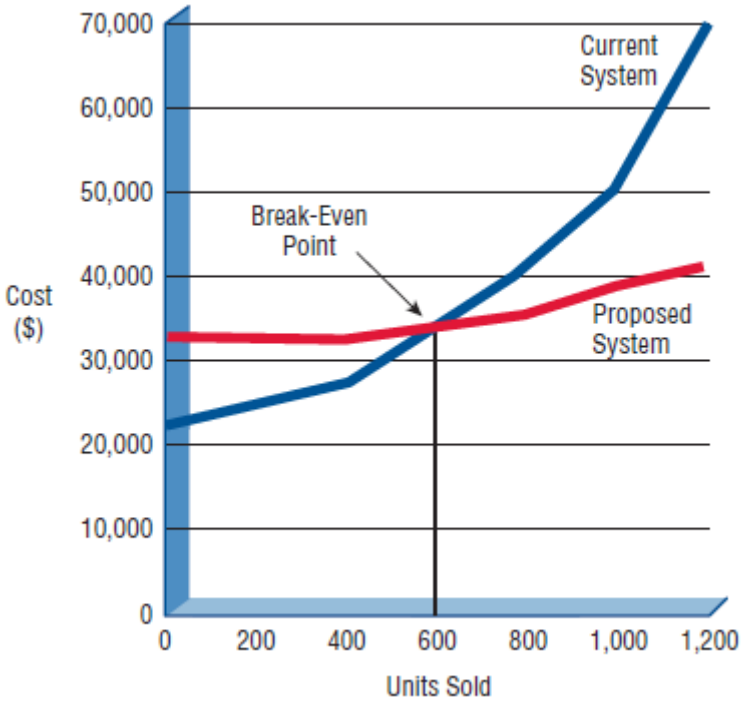


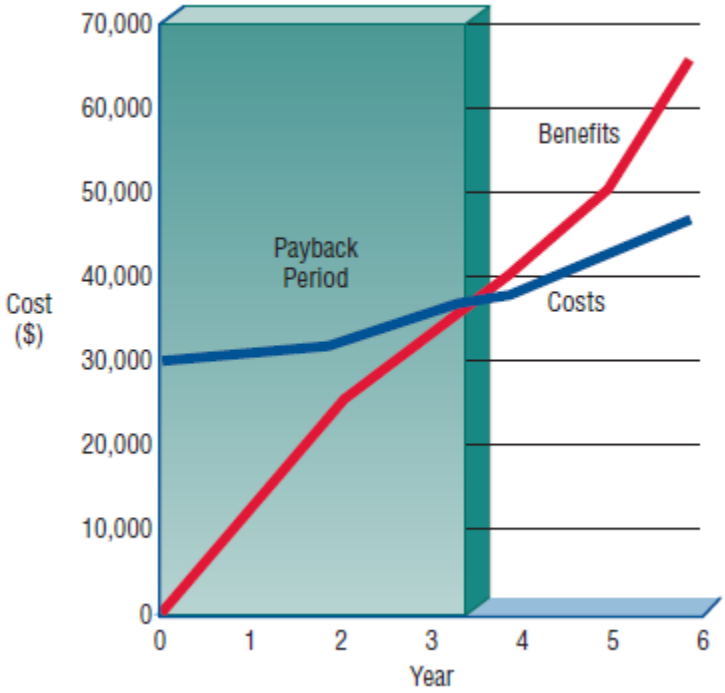
Figure: Break-even analysis for the proposed inventory system.

Total costs include the costs that recur during operation of the system plus the developmental costs that occur only once (one-time costs of installing a new system), that is, the tangible costs that were just discussed. Figure above is an example of a break-even analysis on a small store that maintains inventory using a manual system. As volume rises, the costs of the manual system rise at an increasing rate. A new computer system would cost a substantial sum up front, but the incremental costs for higher volume would be rather small. The graph shows that the computer system would be cost effective if the business sold about 600 units per week.

b i i z a y @ g m a i l . c o m

Break-even analysis is useful when a business is growing and volume is a key variable in costs. One disadvantage of break-even analysis is that benefits are assumed to remain the same, regardless of which system is in place. From our study of tangible and intangible benefits, we know that is clearly not the case.

Break-even analysis can also determine how long it will take for the benefits of the system to pay back the costs of developing it.



Year	Cost ($)	Cumulative Costs ($)	Benefits ($)	Cumulative Benefits ($)
0	30,000	30,000	0	0
1	1,000	31,000	12,000	12,000
2	2,000	33,000	12,000	24,000
3	2,000	35,000	8,000	32,000
4	3,000	38,000	8,000	40,000
5	4,000	42,000	10,000	50,000
6	4,000	46,000	15,000	65,000

Figure: Break-even analysis showing a payback period of three and a half years.

Cash-Flow Analysis (CFA)

Cash-flow analysis examines the direction, size, and pattern of cash flow that is associated with the proposed information system. If you are proposing the replacement of an old information system with a new one and if the new information system will not be generating any additional cash for the business, only cash outlays are associated with the project. If that is the case, the new system cannot be justified on the basis of new revenues generated and must be examined closely for other tangible benefits if it is to be pursued further.

b i i z a y @ g m a i l . c o m

	Year 1				Year 2
	Quarter 1	Quarter 2	Quarter 3	Quarter 4	Quarter 1
Revenue	$5,000	$20,000	$24,960	$31,270	$39,020
Costs					
Software development	10,000	5,000			
Personnel	8,000	8,400	8,800	9,260	9,700
Training	3,000	6,000			
Equipment lease	4,000	4,000	4,000	4,000	4,000
Supplies	1,000	2,000	2,370	2,990	3,730
Maintenance	0	2,000	2,200	2,420	2,660
Total Costs	26,000	27,400	17,370	18,670	20,090
Cash Flow	–21,000	–7,400	7,590	12,600	18,930
Cumulative Cash Flow	–21,000	–28,400	–20,810	–8,210	10,720

Figure: Cash-flow analysis for the computerized mail-addressing system.

Figure above shows a cash-flow analysis for a small company that is providing a mailing service to other small companies in the city. Revenue projections are that only $5,000 will be generated in the first quarter, but after the second quarter, revenue will grow at a steady rate. Costs will be large in the first two quarters and then level off.

Cash-flow analysis is used to determine when a company will begin to make a profit (in this case, it is in the third quarter, with a cash flow of $7,590) and when it will be "out of the red," that is, when revenue has made up for the initial investment (in the first quarter of the second year, when accumulated cash flow changes from a negative amount to a positive $10,720).

The proposed system should have increased revenues along with cash outlays. Then the size of the cash flow must be analyzed along with the patterns of cash flow associated with the purchase of the new system. You must ask when cash outlays and revenues will occur, not only for the initial purchase but also over the life of the information system. In Figure below, system costs total $272,000 over six years and benefits total $280,700. Therefore, we might conclude that benefits outweigh the costs. Benefits only started to surpass costs after the fourth year, however, and dollars in the sixth year will not be equivalent to dollars in the first year.

	Year						
	1	2	3	4	5	6	Total
Costs	$40,000	42,000	44,100	46,300	48,600	51,000	272,000
Benefits	$25,000	31,200	39,000	48,700	60,800	76,000	280,700

Figure: Without considering present value, the benefits appear to outweigh the costs.

b i i z a y @ g m a i l . c o m

Present Value Analysis (PVA)
First, the project benefits are estimated for each year from the time of system implementation. Then, the present values of these savings are computed depending on the interest rate. If the project cost exceeds the present value, then the project is not worthwhile.

The Idea of present value analysis is to determine how much money it is worthwhile investing now in order to receive a given return in some years' time. The answer depends on the interest rate used in the evaluation calculations.

Present value analysis helps the systems analyst to present to business decision makers the time value of the investment in the information system as well as the cash flow. Present value is a way to assess all the economic outlays and revenues of the information system over its economic life, and to compare costs today with future costs and today's benefits with future benefits. It is only a relative measure of a project's return on investment and the formula for present value (PV) is:

$$\frac{\text{Amount}}{(1 + i)^n}$$

Where,
PV = present value
i = interest rate
n = no of years

***Example 1**:*
Let us suppose, we have, interest rate =10%, cost NRs. 3000

Year	Benefit	Discount Factor $[1/(1+r)^1]$	Present value of benefit
1	600	0.909	600 x 0.909 = 545.4
2	900	0.082	900 x0.082 = 743.4
3	1,500	0.751	1,500 x 0.751 = 1126.5
4	1,500	0.683	1,500 x 0.683 = 1024.5
5	1,800	0.621	1,800 x 0.621 = 1117.8
Total	6,300	3.046	3,557.6

Since the present value (PV) of benefits is more than the cost (i.e. NRs. 3000), the project is worthwhile.

Example 2:
Present value of NRs. 3000 invested at 15% interest at the end of 5th year is calculated as

$PV = 3000/(1 + 0.15)^5$
$= 1491.53$

Table below shows present value analysis for 5 years

b i i z a y @ g m a i l . c o m

Year	Estimation Future Value	Present Value	Cumulative Present Value of Benefits
1	3000	2608.69	2608.69
2	3000	2268.43	4877.12
3	3000	1972.54	6949.66
4	3000	1715.25	8564.91
5	3000	1491.53	10056.44

Payback Period (PBP)

It defines the time required to recover the money spent on a project by summing up the net benefit (difference between the cost and benefit) for each year. It is a method to determine the time required to recover the money spent on a project.

Example:

Original investment (cost) = NRs. 12,000

Year	Return Benefits	
	Project X	Project Y
1	3,000	3,000
2	3,000	3,500
3	3,000	4,000
4	3,000	4,500
5	3,000	5,000

Solution:

Year	Return Benefits		Cumulative benefits	
	Project X	Project Y	Project X	Project Y
1	3,000	3,000	3,000	3,000
2	3,000	3,500	6,000	6,500
3	3,000	4,000	9,000	10,500
4	3,000	4,500	12,000 ←	15,000
5	3,000	5,000	15,000	20,000

Therefore, the cumulative benefit of project X at year 4 is equal to the cost (i.e. NRs. 12,000) and that of project Y lies between year 3 and year 4. So, the payback period is 4 years for the project X and for project Y is calculated below:

Payback Period = (Amt not covered in lower yr / Difference in cumulative amts. of HY & LY) + LY
= (1500 / 4500) + 3
= 3.33 yr.

Net Benefit Analysis (NBA)

Net benefit analysis is easy to calculate, interpret and easy to present. It doesn't account for time value of money and doesn't discount future cash flow. It can be calculated by:

Net Benefit = Total Cost - Total Benefits

b i i z a y @ g m a i l . c o m

NBA is so simple to calculate. First find out the components of benefit (due to an economic decision/ economic project/ investment), then measure and value the benefit in different components and then aggregate them (add them suitably because there could be benefits of different types/ components flowing over a period of time). The same thing applies to all cost components: identify them, measure and value them and then aggregate them suitably. Once the values of Total benefits and Total costs are computed, it is simple to arrive at net benefit as the difference between the two.

Net Present Value (NPV)

NPV is a relative measure of a project's return on investment. It is expressed as a percentage of the investment. It is relatively easy to calculate and only accounts for time value of money. The NPV is equal to is the total present value of the benefits minus the total present value of the costs.

In theory, any project with a positive NPV is economically feasible because the project will produce a larger return than would be achieved by investing the same amount of money in a discount rate investment.

NPV = Benefits - Costs
% = Net Present Value/Investments

Example 1:

Suppose total benefits is $80,000 and costs are $50,000
Then Net Present Value = $(80000 - 50000)
= $30000

Therefore, % = 30000/80000
= 0.375

Example 2:

PROJECT A:

	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Total
Benefits:	3,000	28,000	31,000	34,000	36,000	39,000	42,000	
Present Value Factor (12%):	1.000	0.893	0.797	0.712	0.636	0.567	0.507	
Present Value:	3,000	25,004	24,707	24,208	22,896	22,113	21,294	143,222
Costs:	60,000	17,000	18,500	19,200	21,000	22,000	23,300	
Present Value Factor (12%):	1.000	0.893	0.797	0.712	0.636	0.567	0.507	
Present Value:	60,000	15,181	14,745	13,670	13,356	12,474	11,813	141,239
Net Present Value:								**1,983**

net present value of Project A

b i i z a y @ g m a i l . c o m

PROJECT B:

	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Total
Benefits:	——	6,000	26,000	54,000	70,000	82,000	92,000	
Present Value Factor (12%):	——	0.893	0.797	0.712	0.636	0.567	0.507	
Present Value:	——	5,358	20,722	38,448	44,520	46,494	46,644	202,186
Costs:	80,000	40,000	25,000	22,000	24,000	26,500	30,000	
Present Value Factor (12%):	1.000	0.893	0.797	0.712	0.636	0.567	0.507	
Present Value:	80,000	35,720	19,925	15,664	15,264	15,026	15,210	196,809
Net Present Value:					net present value of Project B			**5,377**

Figure: Net present value analysis for Project A and Project B.

The above figure shows that Project B is a better investment than Project A because it has a higher net present value.

Note:

1. If NPV > 0; It means the investment would add value to the firm, Then the project may be accepted.
2. If NPV < 0; It means the investment would subtract value from the firm, Then the project should be rejected.
3. If NPV = 0; It means the investment would neither gain nor lose value for the firm, Then We should be indifferent in the decision whether to accept or reject the project. This project adds no monetary value. Decision should be based on other criteria, e.g. strategic positioning or other factors not explicitly included in the calculation.

Guidelines for Analysis

The use of the methods discussed in the preceding subsections depends on the methods employed and accepted in the organization itself. For general guidelines, however, it is safe to say the following:

1. Use break-even analysis if the project needs to be justified in terms of cost, not benefits, or if benefits do not substantially improve with the proposed system.
2. Use payback when the improved tangible benefits form a convincing argument for the proposed system.
3. Use cash-flow analysis when the project is expensive relative to the size of the company or when the business would be significantly affected by a large drain (even if temporary) on funds.
4. Use present value analysis when the payback period is long or when the cost of borrowing money is high.

Whichever method is chosen, it is important to remember that cost-benefit analysis should be approached systematically, in a way that can be explained and justified to managers, who will eventually decide whether to commit resources

References:

- ✓ Hoffer, J.A., George, J.F. and Valacich J.S., "Modern Systems Analysis and Design", 3rd Edition, Pearson Education, 2003.
- ✓ K.E. Kendall and J.E. Kendall, "Systems Analysis and Design", 5th Edition, Pearson Education, 2003.
- ✓ V. Rajaraman, "Analysis and Design of Information Systems", 2nd Edition, Prentice Hall of India, New Delhi, 2002
- ✓ Elias M. Awad, "Systems Analysis and Design", Galgotia.
- ✓ Jeffrey L. Whitten, Loonnie D. Bentley, 5rd Edition, "Systems Analysis and Design Methods".
- ✓ Englewood Cliffs, New Jersey, "Systems Analysis and Design".
- ✓ Joseph S. Valacich, Joey F. George, Jeffrey A. Hoffer, "Essentials of Systems Analysis and Design" - 5th Edition.
- ✓ Shelly Cashman, "System Analysis and Design" - 9th Edition.

Assignments:

(1) What kinds of information should be sought in interviews?
(2) Explain the cost-benefit analysis with example. (T.U. 2067)
(3) Describe the commonly used methods for performing economic cost-benefit analysis. (T.U. 2067)
(4) Explain the steps in feasibility analysis. (T.U. 2068, 2069)
(5) Explain the payback period with an example. (T.U. 2068)
(6) Why feasibility analysis is necessary before designing a system? (T.U. 2070)
(7) Explain with example the tangible and intangible benefits. (T.U. 2070)

A Gentle Advice:

Please go through your text books and reference books for detail study!!! Thank you all.

Notes Compiled By:

Bijay Mishra
biizay.blogspot.com
9813911076 or 9841695609

Unit 5 - System Design

5. System Design **8 Hrs.**

5.1 The Process and Stages of Systems Design

5.2 Designing Forms and Reports (Process of Formatting Forms and Reports)

5.3 Designing Databases (The Process, Normalization till 3NF, Transferring ERD to Relations)

5.4 File Organization and Database Design (Introduction, Organization of Records in Files)

System Design

The most creative and challenging phase of the system life cycle is system design. The term design describes both a final system and a process by which it is developed. It refers to the technical specifications (analogous to the engineer's blueprints) that will be applied in implementing the candidate system. It also includes the constructions of programs and program testing. The key question here is: How should the problem be solved? Information system provides solution to a business problem. It is in the design phase that the user requirements finalized in the analysis space are translated into practical based of achieving them.

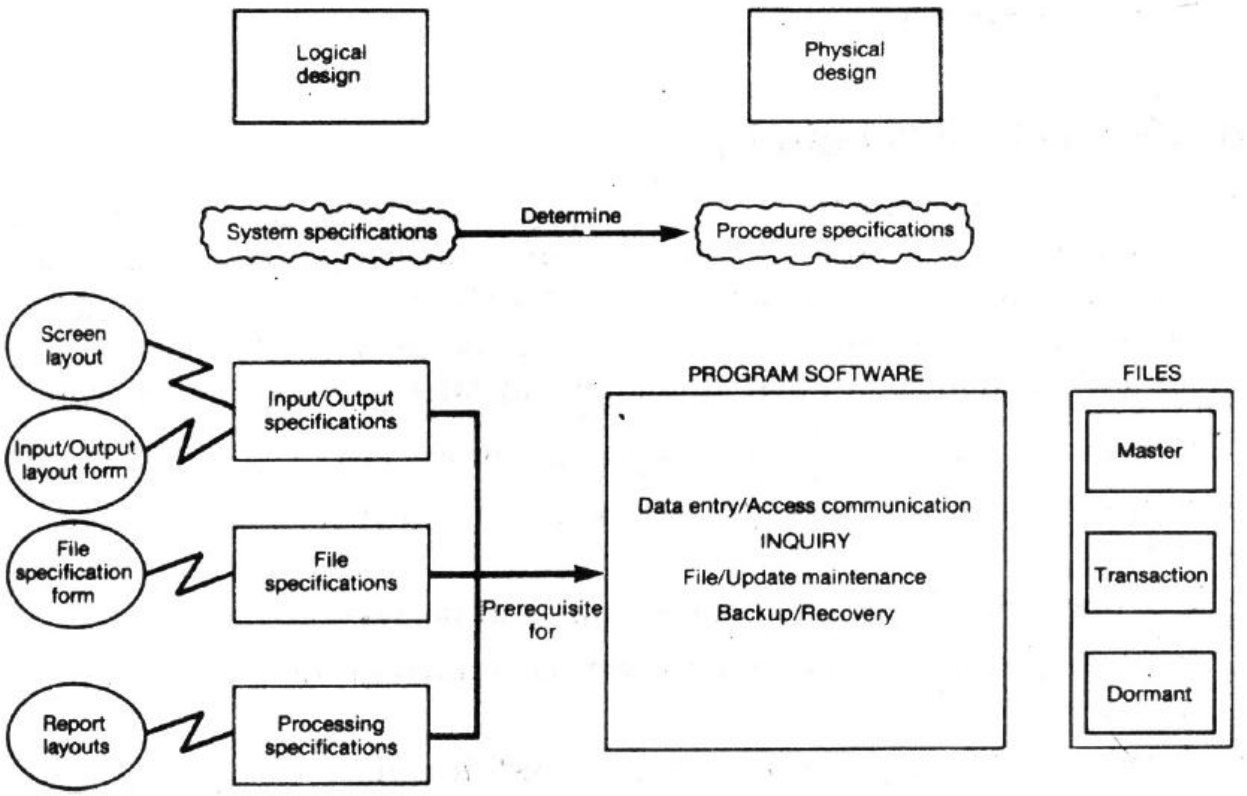


Figure: Overview of System Design

The Process and Stages of Systems Design

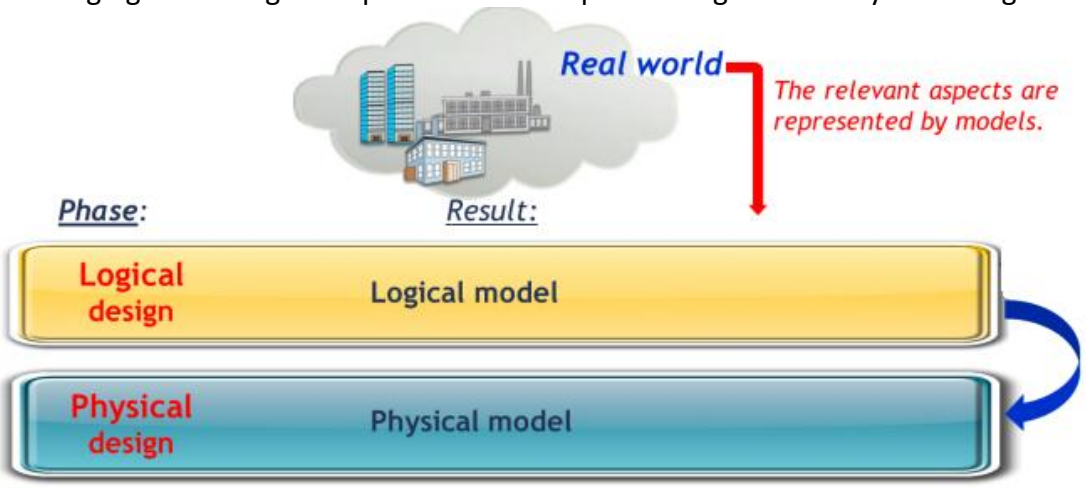
The design phase focuses on the detailed implementation of the system recommended in the feasibility study. Emphasis is on translating performance specifications into design

b i i z a y @ g m a i l . c o m

specifications. The design phase is a transition from a user-oriented document (system proposal) to a document oriented to the programmers or data base personnel.

Logical and Physical System Design

Systems design goes through two phases of development: Logical and Physical Design.



For a candidate system, logical system design describes the inputs (source), outputs (destination), data bases (data stores), and procedures (data flows)—all in a format that meets the user's requirements. When analysts prepare the logical system design, they specify the user needs at a level of detail that virtually determines the information flow into and out of the system and the required data resources.

In a logical design, a specification of the principal features of the new system, which satisfies the system's objectives are produced by the designer. Logical design results in the blueprint of the new system. So the part of the design process that is independent of any specific hardware or software platform is referred to as logical design.

The structured system design tools that can be used to provide a logical representation of data flow analysis and data processing analysis in a computer based information system are: Data Dictionary, Data Flow Diagrams, Decision Table, ER Diagram, Structured English, and Decision Tree.

Following logical design is physical design. The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified/authenticated, how it is processed, and how it is displayed as in physical design, the following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing Requirements,
5. System control and backup or recovery.

The physical portion of systems design can generally be broken down into three sub-tasks:

a. User Interface Design
b. Data Design
c. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor.

Logical Design	Physical Design
The part of the design process that is independent of any specific hardware or software platform is referred to as logical design.	The design that relates to the actual input and output processes of the system is known as physical design.
The logical designing describes the structure & characteristics or features, like output, input, files, database & procedures.	The physical design is an actual software & a working system. There will be constraints like Hardware, Software, Cost, Time & Interfaces.
Logical design precedes physical design.	Physical design succeeds logical design.
Logical design should accurately render a visual representation of the activities and data relevant to a particular business. Example: ERD, Business process diagrams, etc.	Physical design involves the actual design of a database according to the requirements that were established during logical design. Example: Creating tables and columns, based on entities and attributes that were defined during logical design.
The logical design is highly stable, it will only change if the business changes (due to mergers, acquisitions, diversification, new products/services, etc.)	The physical design is more dynamic, and is ultimately driven by changes in technology.
Logical refers to abstract information processing aspects-what is achieved not how it performed e.g. create an entity, request information, make payment.	Physical refers to the actual implementation in terms of hardware, software or data e.g. type of customer details.
The logical design considers which elements must be present for the system to meet its objectives.	The final physical design considers how the logical design can be implemented with the current technology available.
The logical design is more conceptual and abstract than the physical design. In the logical design, you look at the logical relationships among the objects.	In the physical design, you look at the most effective way of storing and retrieving the objects.

b i i z a y @ g m a i l . c o m

Designing Forms and Reports

Form is a business document that contains some predefined 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. The form is a tool with a message; it is the physical carrier of data-of information. It also can constitute authority for action. Each form is a request for action. It provides information for making decisions and improving operations.

Report is a business document that contains only predefined data. It is a passive document for reading or viewing data. It typically contains data from many database records or transactions.

Classification of Forms

A printed form is generally classified by what it does in the system. There are three primary classifications:

1. Memory Form
2. Action Form
3. Report Form

An ***action form*** requests the user to do something-get action. (Examples are purchase orders and shop orders).

A ***memory form*** is a record of historical data that remains in a file, is used for reference, and serves as control on key details. (Examples are inventory records, stock ledger, purchase records, and bond registers).

A ***report form*** guides supervisors and other administrators in their activities. It provides data on a project or a job. (Examples are Balance Sheet, Trial Balance, Profit/Loss statements)

Form Design

Data provide the basis for information systems. Without data there is no system, but data must be provided in the right form for input and the information produced must be in a format acceptable by the user.

Form design follows analyzing forms, evaluating present documents, and creating new or improved forms. Since the purpose of a form is to communicate effectively through forms design, there are several requirements- Include instructions for completing the form. Minimize the amount of handwriting. Data to be entered (keyed) should be sequenced top-to-bottom and left-to-right. Use designs based on known metaphors.

Form Design make forms easy to fill out: to reduce errors, speed completion, facilitate data entry, form flow (top to bottom, left to right), form sections: logical grouping of info, form captions: Captions tell the person completing the form what to put on a blank line, space, or box. Ensure that forms meet the intended purpose: effectiveness (specialty forms), assure accurate completion: (row & column totals), keep forms attractive: uncluttered, enough space to fill, fonts and line weights to separate categories, ask each item of data only once. The seven sections of a form are: Heading, Identification and access, Instructions, Body, Signature and verification, Totals, Comments. Captions may be line caption, putting the caption on the same

line or below the line, boxed caption, providing a box for data instead of a line, vertical check off, lining up choices or alternatives vertically, horizontal check off, lining up choices or alternatives horizontally.

Requirements of Forms Design

Forms design follows analyzing forms, evaluating present documents, and creating new or improved forms. Bear in mind that detailed analysis occurs only after the problem definition stage and the beginning of designing the candidate system. Since the purpose of a form is to communicate effectively through forms design, there are several major requirements:

1. *Identification and wording.* The form title must clearly identify its purpose. Columns and rows should be labeled to avoid confusion. The form should also be identified by form name or code number to make it easy to reorder.

2. *Maximum readability and use.* The form must be easy to use and fill out. It should be legible, intelligible, and uncomplicated. Ample writing space must be provided for inserting data. This means analyzing for adequate space and balancing the overall form's layout, administration, and use.

3. *Physical factors.* The form's composition, color, layout (margins, space, etc.), and paper stock should lend themselves to easy reading. Pages should be numbered when multipage reports are being generated for the user.

4. *Order of data items.* The data requested should reflect a logical sequence. Related data should be in adjacent positions. Data copied from source documents should be in the same sequence on both forms. Much of this design takes place in the forms analysis phase.

5. *Ease of data entry.* If used for data entry, the form should have field positions indicated under each column of data and should have some indication of where decimal points are (use broken vertical lines).

6. *Size and arrangement.* The form must be easily stored and filed. It should provide for signatures. Important items must be in a prominent location on the form.

7. *Use of instructions.* The instructions that accompany a form should clearly show how it is used and handled.

8. *Efficiency considerations.* The form must be cost effective. This means eliminating unnecessary data and facilitating reading lines across the form. To illustrate, if a poorly designed form causes 10 supervisors to waste 30 seconds each, then 5 minutes are lost because of the form. If (the firm uses 10,000 of these forms per year, then 833 hours of lost time could have been saved by a better forms design.

9. *Type of report.* Forms design should also consider whether the content is executive summary, intermediate managerial information, or supporting- data. The user requirements for each type often determine the final form design.

Report Design

Report Design Principles and Design Issues are:

- Page Size - Today the page sizes of choice are standard (8½" x 11") and legal (8½" x 14").

- Page Orientation- Portrait orientation is often preferred because it is oriented the way we orient most books and reports; however, landscape is often necessitated for tabular reports because more columns can be printed.

- Page Headings- At minimum, page headers should include a recognizable report title, date and time, and page numbers.

- Report Legends - A legend is an explanation of abbreviations, colors, or codes used in a report. In a printed report, a legend can be printed on only the first or last page. On a display screen, a legend can be made available as a pop-up dialogue box.

- Column Headings - Column headings should be short and descriptive. Avoid abbreviations or include a Report Legend.

- Heading Alignments - Alignment should be tested with users for preferences with a special emphasis on the risk of misinterpretation of the information.

- Column Spacing - If columns are too close, users may not properly differentiate between the columns. If they are too far apart, the user may have difficulty following a single row. Rule of thumb is to use 3-5 spaces between each.

- Row Headings - The first one or two columns should identify data that differentiates each row. Rows should be sequenced in a fashion that supports their use. Frequently rows are sorted on a numerical key or alphabetically.

- Formatting - Data is often stored without formatting characters to save storage space. Outputs should reformat data to match the users' norms.

- Control Breaks - Groups of rows should be logically grouped in the report. The transition from one group to the next is called a control break and is frequently followed by subtotals for the group.

- End of Report - The end of a report should be clearly indicated to ensure that users have the entire report.

Steps in designing output reports with a Computer-Aided software tool

-Determine the need for the report.
-Determine the users.
-Determine the data items to be included.
-Estimate the size of the report.
-Determine the report title.
-Number the pages of the report.

-Include the preparation date on the report.
-Label each column of data appropriately.
-Define variable data.
-Review prototype reports.

Types of Forms

1. *Flat forms*
   - Single copy form
   - Prepared manually or by machine
   - Printed on any grade of papers
   - Easiest form to design, print and reproduce
   - Have low-volume use
   - Least expensive

2. *Unit-set/Snap-out Forms*
   - Have original copy & several copy with one-time carbon papers interleaved between them.
   - Carbon paper is 3/8" shorter than copies.
   - Copies are perforated.
   - Forms can be easily snapped out after completion.

Figure: *Unit-set/Snap-out Forms*

3. *Continuous strip/Fanfold Forms*
   - Multiple unit forms joined together in a continuous strip with perforation between each pair of form
   - "One-time carbon" is interleaved between copies, which are stacked in fanfold arrangement.
   - Least expensive construction for large-volume use.
   - Computer printouts are invariably produced on them.
   - Virtually is part of system design.

b i i z a y @ g m a i l . c o m

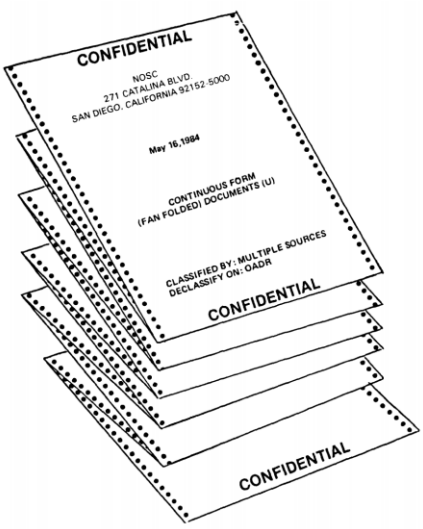


Figure: *Continuous strip/Fanfold Forms*

4. *NCR (No Carbon Required) Form*
   - Several copies can be made by pressing a chemical undercoating on the top sheet into a clayline coating on the top of the 2nd sheet.
   - Writing pressure forms an image by the coating material.
   - Application is in sales book, checkbook, inventory tickets and deposit slips.
   - It offers cleaner and longer-lasting copies than carbon-interleaved forms.
   - Cost is 25% more than carbon- interleaved forms.

Figure: *NCR (No Carbon Required) Form*

Forms Control

Businesses often have a forms specialist who controls forms. Basic duties for controlling forms:

- Make sure that each form fulfils a specific purpose.
- Prevent duplication of the information that is collected and of the forms that collect it.
- Deciding how to get forms reproduced in the most economical way.
- Establishing stock control and inventory procedures that make forms available when needed at the lowest possible cost.
- A unique form number and revision date should be included on each form.

Common Types of Reports

- Scheduled: produced at predefined time intervals for routine information needs
- Key-indicator: provide summary of critical information on regular basis
- Exception: highlights data outside of normal operating ranges
- Drill-down: provide details behind summary of key-indicator or exception reports
- Ad-hoc: respond to unplanned requests for non-routine information needs

The Process of Designing Database

The database design process can be divided into six steps.

1. ***Requirement Analysis:***
   The very first step in designing a database application is to understand what data is to be stored in the database, what application must be built in top of it, and what operations are most frequent and subject to performance requirements. In other words, we must find out what the users want from the database.
2. ***Conceptual Database Design:***
   The information gathered in the requirements analysis step is used to develop a high level description of the data to be stored in the database, along with the constraints known to hold over this data. The ER model is one of several high-level or semantic, data models used in database design.
3. ***Logical Database Design:***
   We must choose a database to convert the conceptual database design into a database schema in the data model of the chosen DBMS. Normally we will consider the Relational DBMS and therefore, the task in the logical design step is to convert an ER schema into a relational database schema.
4. ***Schema Refinement:***
   This step is to analyze the collection of relations in our relational database schema to identify potential problems, and refine it.
5. ***Physical Database Design:***
   This step may simply involve building indexes on some table and clustering some tables or it may involve substantial redesign of parts of database schema obtained from the earlier steps.
6. ***Application and Security Design:***
   Any software project that involves a DBMS must consider aspects of the application that go beyond the database itself. We must describe the role of each entity (users, user group, departments) in every process that is reflected in some application task, as part of a complete workflow for the task. A DBMS provides several mechanisms to assist in this step.

Note: The ER model is most relevant to the first three steps.

Normalization

Normalization is the process of creating table designs by assigning specific fields or attributes to each table in the database. A table design specifies the fields and identifies the primary key in a particular table or file. Normalization involves applying a set of rules that can help you identify and correct inherent problems and complexities in your table designs. The concept of normalization is based on the work of Edgar Codd, a British computer scientist who formulated the basic principles of relational database design.

The normalization process typically involves four stages: un-normalized design, first normal form, second normal form, and third normal form. The three normal forms constitute a progression in which third normal form represents the best design. Most business-related databases must be designed in third normal form.

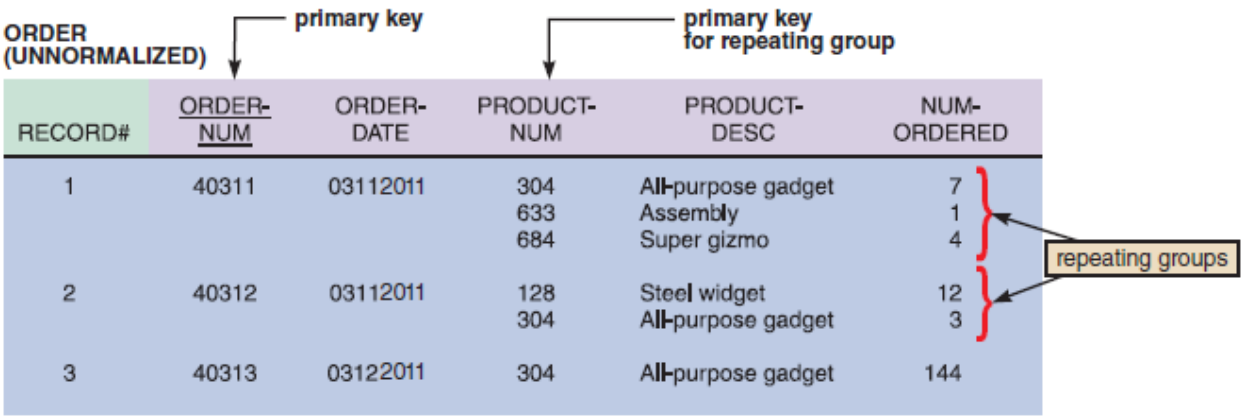
Un-normalized Designs

A table design that contains a repeating group is called un-normalized. The standard notation method for representing an un-normalized design is to enclose the repeating group of fields within a second set of parentheses.

An example of an un-normalized table would look like this:
NAME (FIELD 1, FIELD 2, FIELD 3, (REPEATING FIELD 1, REPEATING FIELD 2))

During data design, you must be able to recognize a repeating group of fields. A repeating group is a set of one or more fields that can occur any number of times in a single record, with each occurrence having different values.

Now review the un-normalized ORDER table design shown in figure below:



RECORD#	ORDER-NUM	ORDER-DATE	PRODUCT-NUM	PRODUCT-DESC	NUM-ORDERED
1	40311	03112011	304	All-purpose gadget	7
			633	Assembly	1
			684	Super gizmo	4
2	40312	03112011	128	Steel widget	12
			304	All-purpose gadget	3
3	40313	03122011	304	All-purpose gadget	144

Figure: ORDER Table

Here, the records 1 and 2 have repeating groups because they contain several products. ORDER-NUM is the primary key for the ORDER table, and PRODUCT-NUM serves as a primary key for the repeating group. Because it contains a repeating group, the ORDER table design is un-normalized.

Following the notation guidelines, you can describe the design as follows:
ORDER (ORDER-NUM, ORDER-DATE, (PRODUCT-NUM, PRODUCT-DESC, NUM-ORDERED))

The notation indicates that the ORDER table design contains five fields, which are listed within the outer parentheses. The ORDER-NUM field is underlined to show that it is the primary key. The PRODUCT-NUM, PRODUCT-DESC, and NUM-ORDERED fields are enclosed within an inner set of parentheses to indicate that they are fields within a repeating group. Notice that PRODUCT-NUM also is underlined because it acts as the primary key of the repeating group.

First Normal Form

A table is in first normal form (1NF) if it does not contain a repeating group. To convert an un-normalized design to 1NF, you must expand the table's primary key to include the primary key of the repeating group.

For example, in the ORDER table shown in previous figure, the repeating group consists of three fields: PRODUCT-NUM, PRODUCT-DESC, and NUM-ORDERED. Of the three fields, only PRODUCT-NUM can be a primary key because it uniquely identifies each instance of the repeating group.

The product description cannot be a primary key because it might or might not be unique. For example, a company might sell a large number of parts with the same descriptive name, such as washer, relying on a coded part number to identify uniquely each washer size.

When you expand the primary key of ORDER table to include PRODUCT-NUM, you eliminate the repeating group and the ORDER table is now in 1NF, as shown:

ORDER (ORDER-NUM, ORDER-DATE, PRODUCT-NUM, PRODUCT-DESC, NUM-ORDERED)

The figure below shows the ORDER table in 1NF.

ORDER IN 1NF — combination primary key

RECORD#	ORDER-NUM	ORDER-DATE	PRODUCT-NUM	PRODUCT-DESC	NUM-ORDERED
1	40311	03112011	304	All-purpose gadget	7
2	40311	03112011	633	Assembly	1
3	40311	03112011	684	Super gizmo	4
4	40312	03112011	128	Steel widget	12
5	40312	03112011	304	All-purpose gadget	3
6	40313	03122011	304	All-purpose gadget	144

repeating groups have been eliminated

Figure: The ORDER table as it appears in 1NF

Here, the repeating groups have been eliminated. Notice that the repeating group for order 40311 has become three separate records, and the repeating group for order 40312 has become two separate records. The 1NF primary key is a combination of ORDER-NUM and PRODUCT-NUM, which uniquely identifies each record.

Therefore, when a table is in 1NF, each record stores data about a single instance of a specific order and a specific product.

Second Normal Form

A table design is in second normal form (2NF) if it is in 1NF and if all fields that are not part of the primary key are functionally dependent on the entire primary key. If any field in a 1NF table depends on only one of the fields in a combination primary key, then the table is not in 2NF.

Notice that if a 1NF design has a primary key that consists of only one field, the problem of partial dependence does not arise — because the entire primary key is a single field. Therefore, a 1NF table with a single-field primary key is automatically in 2NF.

A standard process exists for converting a table from 1NF to 2NF. The objective is to break the original table into two or more new tables and reassign the fields so that each non-key field will depend on the entire primary key in its table.

Steps:
1. First, create and name a separate table for each field in the existing primary key. For example, (See in the figure above of 1NF), the ORDER table's primary key has two fields, ORDER-NUM and PRODUCT-NUM, so you must create two tables. The ellipsis (…) indicates that fields will be assigned later.

The result is:
ORDER (ORDER-NUM, ……..)
PRODUCT (PRODUCT-NUM, ……..)

2. Next, create a new table for each possible combination of the original primary key fields. In the figure as that of 1NF example, you would create and name a new table with a combination primary key of ORDER-NUM and PRODUCT-NUM. This table describes individual lines in an order, so it is named ORDER-LINE, as shown:

ORDER-LINE (ORDER-NUM, PRODUCT-NUM, …….)

3. Finally, study the three tables and place each field with its appropriate primary key, which is the minimal key on which it functionally depends. When you finish placing all the fields, remove any table that did not have any additional fields assigned to it. The remaining tables are the 2NF version of your original table. In the figure as that of 1NF example, the three tables would be shown as:

ORDER (ORDER-NUM, ORDER-DATE)
PRODUCT (PRODUCT-NUM, PRODUCT-DESC)
ORDER-LINE (ORDER-NUM, PRODUCT-NUM, NUM-ORDERED)

The figure below shows the 2NF table designs. By following the steps, you have converted the original 1NF table into three 2NF tables.

ORDER IN 2NF

primary key

RECORD#	ORDER-NUM	ORDER-DATE
1	40311	03112011
2	40312	03112011
3	40313	03122011

b i i z a y @ g m a i l . c o m

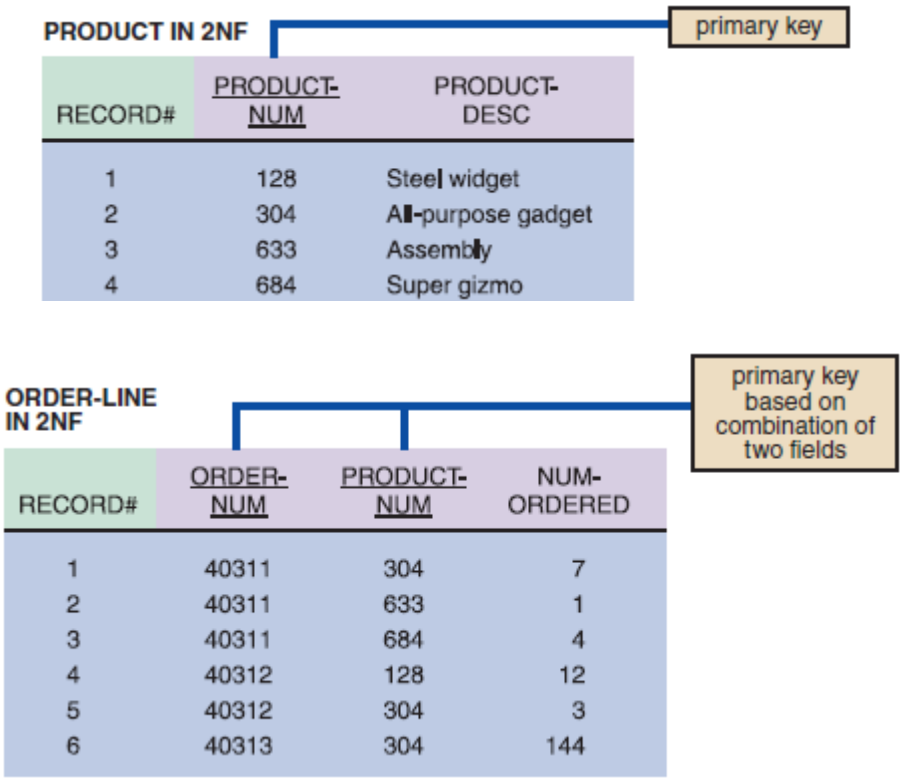


Figure: ORDER, PRODUCT, and ORDER-LINE tables in 2NF

Here, all fields are functionally dependent on the primary key.

Third Normal Form

A popular rule of thumb is that a design is in 3NF if every non-key field depends on the key, the whole key, and nothing but the key. A table design is in third normal form (3NF) if it is in 2NF and if no non-key field is dependent on another non-key field. Remember that a non-key field is a field that is not a candidate key for the primary key.

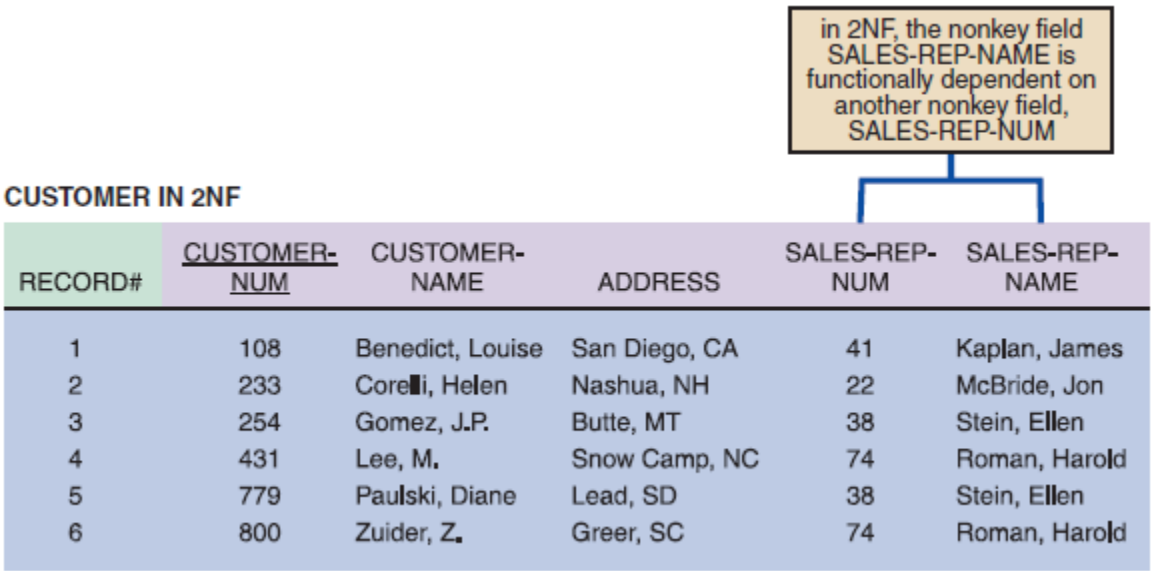


Figure: 2NF design for the CUSTOMER table

b i i z a y @ g m a i l . c o m

Consider the following CUSTOMER table design, as shown in figure above:
CUSTOMER (CUSTOMER-NUM, CUSTOMER-NAME, ADDRESS, SALES-REP-NUM, SALES-REP-NAME)

Here, the table is in 1NF because it has no repeating groups. The design also is in 2NF because the primary key is a single field.

Here, the CUSTOMER example is not in 3NF because one non-key field, SALES-REP-NAME, depends on another non-key field, SALES-REP-NUM.

To convert the table to 3NF, you must remove all fields from the 2NF table that depend on another non-key field and place them in a new table that uses the non-key field as a primary key.

Therefore, to reach 3NF, you must remove SALES-REP-NAME and place it into a new table that uses SALES-REP-NUM as the primary key.

The figure below shows the third normal form producing two separate tables:

CUSTOMER (CUSTOMER-NUM, CUSTOMER-NAME, ADDRESS, SALES-REP-NUM)
SALES-REP (SALES-REP-NUM, SALES-REP-NAME)

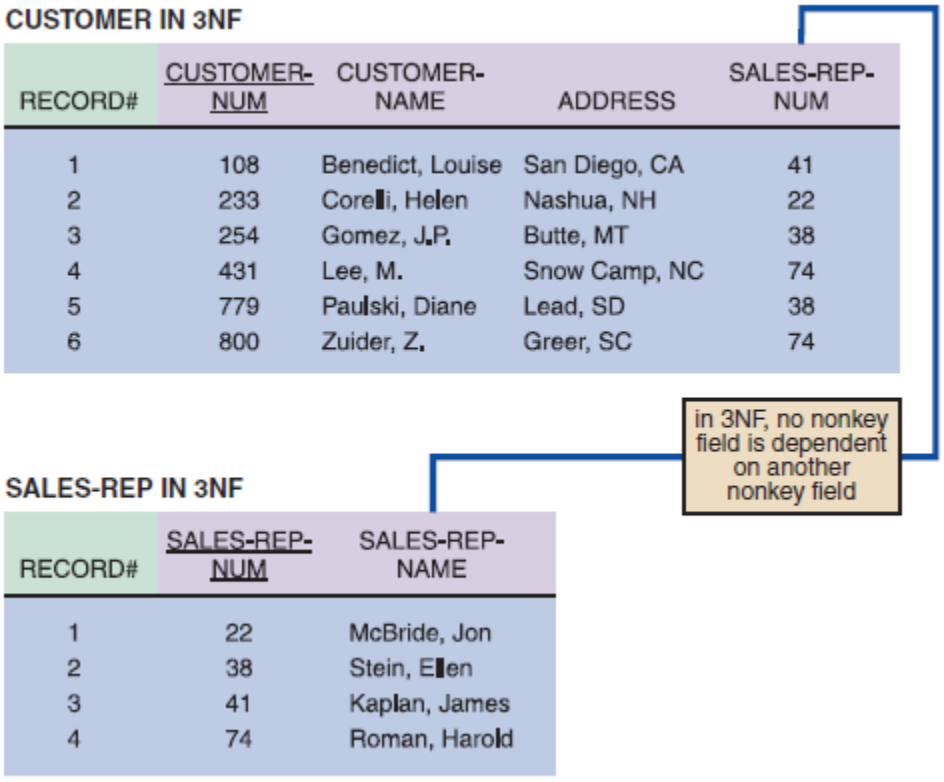


Figure: When the CUSTOMER table is transformed from 2NF to 3NF, the result is two tables: CUSTOMER and SALES-REP

b i i z a y @ g m a i l . c o m

Transferring ER Diagram to Relations

It is useful to transform the conceptual data model into a set of normalized relations. The steps for transforming ER Diagrams into relations are:

1. Represent entities.
2. Represent relationships.
3. Normalize the relations.
4. Merge the relations.

Representing Entities

Each regular entity is transformed into a relation. The identifier of the entity type becomes the primary key of the corresponding relation. The primary key must satisfy the following two conditions.

- ✓ The value of the key must uniquely identify every row in the relation.
- ✓ The key should be non-redundant.

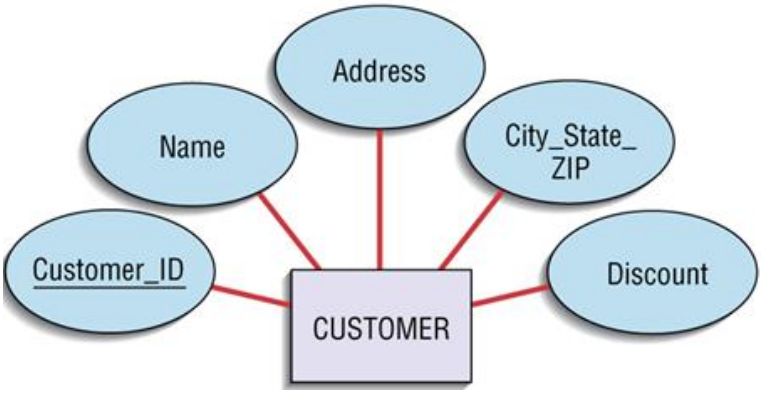


Figure: Transforming an entity type to a relation - ER Diagram

CUSTOMER

Customer_ID	Name	Address	City_State_ZIP	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%

Figure: Transforming an entity type to a relation – Relation

Represent Relationships

Binary 1:N Relationships

- ✓ Add the primary key attribute (or attributes) of the entity on the one side of the relationship as a foreign key in the relation on the right side.
- ✓ The one side migrates to the many side.

b i i z a y @ g m a i l . c o m

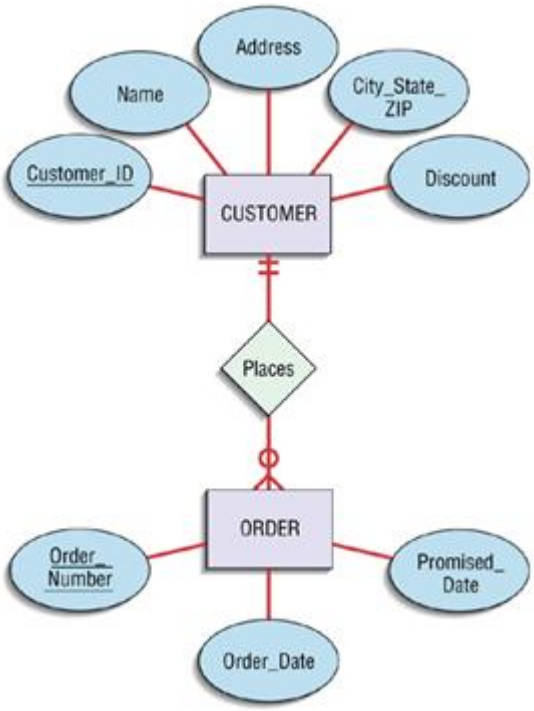


Figure: Representing a 1:N relationship – ER Diagram

CUSTOMER

Customer_ID	Name	Address	City_State_ZIP	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%

ORDER

Order_Number	Order_Date	Promised_Date	Customer_ID
57194	3/15/0X	3/28/0X	6390
63725	3/17/0X	4/01/0X	1273
80149	3/14/0X	3/24/0X	6390

Figure: Representing a 1:N relationship – Relation

Binary or Unary 1:1

Three possible options

- ✓ Add the primary key of A as a foreign key of B.
- ✓ Add the primary key of B as a foreign key of A.
- ✓ Both of the above.

Binary and Higher M:N relationships

- ✓ Create another relation and include primary keys of all relations as primary key of new relation.

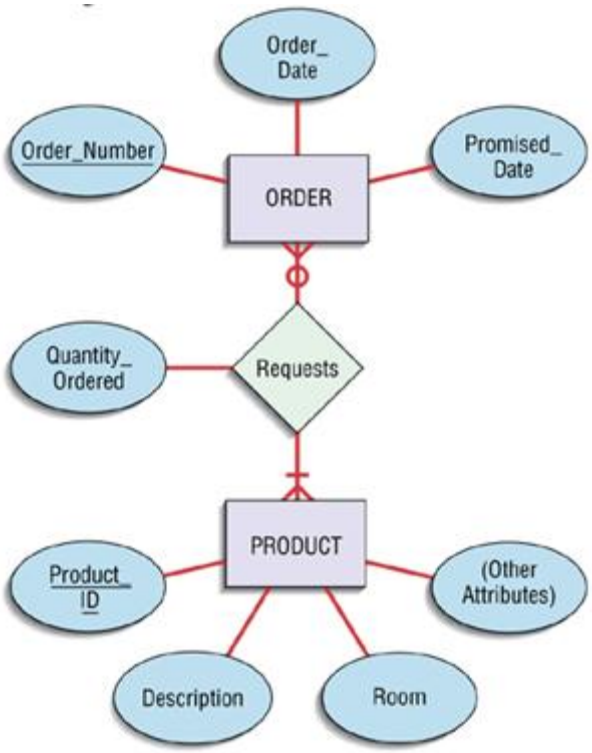


Figure: Representing a M:N relationship – ER Diagram

ORDER

Order_Number	Order_Date	Promised_Date
61384	2/17/2002	3/01/2002
62009	2/13/2002	2/27/2002
62807	2/15/2002	3/01/2002

ORDER LINE

Order_Number	Product_ID	Quantity_ Ordered
61384	M128	2
61384	A261	1

PRODUCT

Product_ID	Description	(Other Attributes)
M128	Bookcase	—
A261	Wall unit	—
R149	Cabinet	—

Figure: Representing a M:N relationship – Relations

Unary 1:N Relationships

- ✓ Relationship between instances of a single entity type
- ✓ Utilize a recursive foreign key
    - A foreign key in a relation that references the primary key values of that same relation.

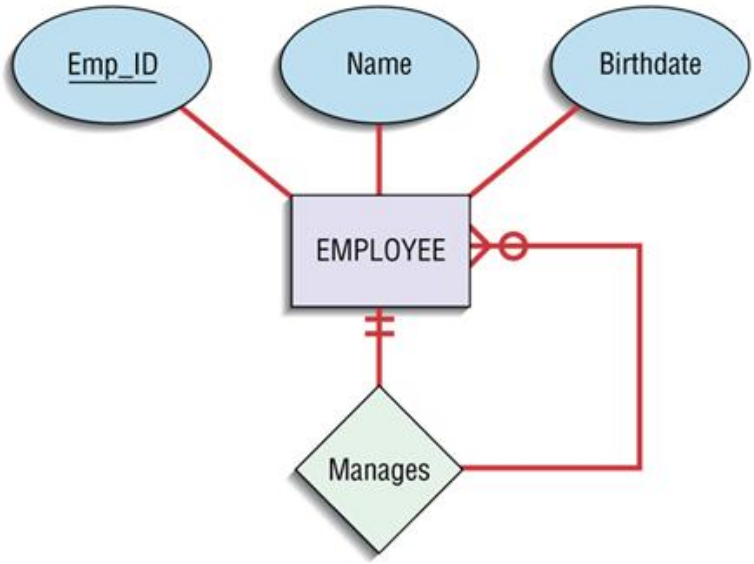


Figure: Two unary relations – EMPLOYEE with Manages relationship (1:N)

EMPLOYEE (Emp_ID, Name, Birthdate, Manager_ID)

Unary M:N Relationships

- ✓ Create a separate relation.
- ✓ Primary key of new relation is a composite of two attributes that both take their values from the same primary key.

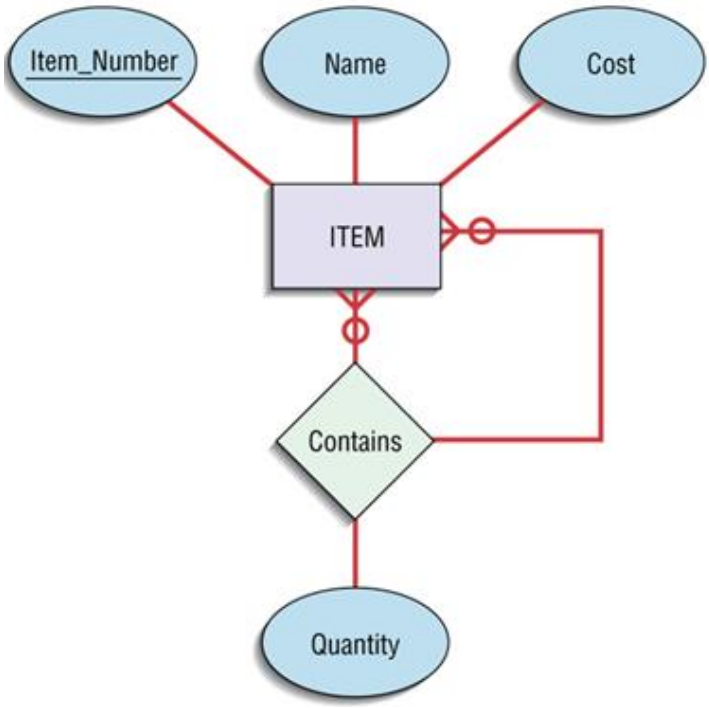


Figure: Two unary relations – Bill-of-materials structure (M:N)

ITEM (Item_Number, Name, Cost)
ITEMCOMPONENT (Item_Number, Component_Number, Quatity)

E-R to Relational Transformation

E-R Structure	Relational Representation
Regular entity	Create a relation with primary key and nonkey attributes.
Weak entity	Create a relation with a composite primary key (which includes the primary key of the entity on which this weak entity depends) and nonkey attributes.
Binary or unary 1:1 relationship	Place the primary key of either entity in the relation for the other entity or do this for both entities.
Binary 1:*N* relationship	Place the primary key of the entity on the one side of the relationship as a foreign key in the relation for the entity on the many side.
Binary or unary *M:N* relationship or associative entity	Create a relation with a composite primary key using the primary keys of the related entities, plus any non-key attributes of the relationship or associative entity.
Binary or unary *M:N* relationship or associative entity with additional key(s)	Create a relation with a composite primary key using the primary keys of the related entities and additional primary key attributes associated with the relationship or associative entity, plus any nonkey attributes of the relationship or associative entity.
Binary or unary *M:N* relationship or associative entity with its own key	Create a relation with the primary key associated with the relationship or associative entity, plus any nonkey attributes of the relationship or associative entity and the primary keys of the related entities (as nonkey attributes).

Merging Relations

Purpose is to remove redundant relations. View Integration Problems or merging relations is the last step in the logical database design and prior to physical file and database design.

An example of merging relations

EMPLOYEE1 (Emp_ID, Name, Address, Phone)
EMPLOYEE2 (Emp_ID, Name, Address, Jobcode)

Since these two relations have same primary key (Emp_ID) and describes same entity, they should be merged into one relation.
EMPLOYEE (Emp_ID, Name, Address, Phone, Jobcode)

b i i z a y @ g m a i l . c o m

Problems that arise while merging relations:

Synonyms

- ✓ Two different names used for the same attribute
- ✓ Example: Emp_ID and Emp_No may be the synonyms
- ✓ When merging, get agreement from users on a single, standard name

Example:
STUDENT1 (S_id, Name)
STUDENT2 (Reg_No, Name, Address)

Here, both the S_id and Reg_no are synonyms for a person's social security no and are identical attributes. Merging the two relations would produce:

STUDENT (SSN, Name, Address)

Homonyms

- ✓ A single attribute name that is used for two or more different attributes
- ✓ Example; the term account may be fixed account, savings account, loan account, etc
- ✓ Resolved by creating a new name

Example:
STUDENT1 (S_id, Name, Address)
STUDENT2 (S_id, Name, Address, Phone_no, DOB)

Here, the attribute address in STUDENT1 refers to the home address and in STUDENT2 refers to the college address. To resolve the conflict, we need to create a new attribute names as:

STUDENT (S_id, Name, Phone_no, DOB, Home_address, College_address)

Dependencies between non-keys

- ✓ Dependencies between non-keys may result when two 3NF relations are merged to form a single relation

Example:
STUDENT1 (S_id, Major)
STUDENT2 (S_id, Adviser)

Above relation is merged into:
STUDENT (S_id, Major, Adviser)

Suppose each major has exactly one adviser. In this case, adviser is functionally dependent on major:

Major →Adviser

If the above dependency exists, then the STUDENT is in 2NF but not 3NF, because it contains functional dependency between non-keys. So, create a 3NF relation with new relation as:

STUDENT (S-id, Major)
MAJORADVISER (Major, Adviser)

Class/Subclass

- ✓ May be hidden in user views or relations

Example:
PATIENT1 (P_id, Name, Address, Date_treated)
PATIENT2 (P_id, Room_no)

Here, the relation can be merged but what if the patients are of two types: Inpatients and outpatients. Inpatient occupies Room_no while the outpatient contains attribute Date_treated.

In this situation you should create class/subclass relationships for these entities:
PATIENT (P_id, Name, Address)
INPATIENT (P_id, Room_no)
OUTPATIENT (P_id, Date_treated)

File Organization and Database Design

Databases are not merely a collection of files. Rather, a database is a central source of data meant to be shared by many users for a variety of applications. The heart of a database is the database management system (DBMS), which allows the creation, modification, and updating of the database; the retrieval of data; and the generation of reports and displays. The person who ensures that the database meets its objectives is called the database administrator.

File Structure

- Bytes: set of 8 bits that represents a character.
- Data Item (Element): one or more bytes are combined to form a data item. It is also known as field.
- Record: collection of data items or fields.
- File: Collection of related records.
- Database: It is a set of interrelated files for real time processing.
- Database Management System (DBMS): It is a software that creates and manipulates the databases.

Types of Files:

1. Master files
2. Transaction files
3. Look-up files
4. Audit files
5. History files
6. Document files

Master File
Master files store core information that is important to the business and, more specifically, to the application, such as order information or customer mailing information. They usually are kept for long periods of time, and new records are appended to the end of the file as new orders or new customers are captured by the system. If changes need to be made to existing records, programs must be written to update the old information.

Transaction File
A transaction file holds information that can be used to update a master file. The transaction file can be destroyed after changes are added, or the file may be saved in case the transactions need to be accessed again in the future. Customer address changes, for one, would be stored in a transaction file until a program is run that updates the customer address master file with the new information.

Look-up File
Look-up files contain static values, such as a list of valid zip codes or the names of the U.S. states. Typically, the list is used for validation. For example, if a customer's mailing address is entered into a master file, the state name is validated against a look-up file that contains U.S. states to make sure that the operator entered the value correctly.

Audit File
An audit file records "before" and "after" images of data as it is altered so that an audit can be performed if the integrity of the data is questioned. For control purposes, a company might need to store information about how data changes over time. For example, as human resource clerks change employee salaries in a human resource system, the system should record the person who made the changes to the salary amount, the date, and the actual change that was made.

History File
Sometimes files become so large that they are unwieldy, and much of the information in the file is no longer used. The history file (or archive file) stores past transactions (e.g., old customers, past orders) that are no longer needed by system users. Typically the file is stored off-line, yet it can be accessed on an as-needed basis.

Document File
It consists of historical data for review without overhead of regenerating document.

File Organization
File organization refers to the relationship of the key of the record to the physical location of that record in the computer file. File organization may be either ***physical file*** or a ***logical file***.

A physical file is a physical unit, such as magnetic tape or a disk. A logical file on the other hand is a complete set of records for a specific application or purpose. A logical file may occupy a part of physical file or may extend over more than one physical file.

The objectives of computer based file organization are:

(1) High throughput for processing transactions
(2) Efficient use of storage space
(3) Protection from failures or data loss
(4) Minimizing need for reorganization
(5) Accommodating growth
(6) Security from unauthorized use
(7) Ease of file creation and maintenance
(8) Efficient means of retrieving information.

There are five methods of organizing files. These are:-

1. Sequential Organization
2. Indexed Sequential Organization
3. Hashed File Organization
4. Inverted List Organization
5. Direct-Access organization

Sequential Organization

Here the records are arranged in the ascending or descending order or chronological order of a key field which may be numeric or both. Since the records are ordered by a key field, there is no storage location identification.

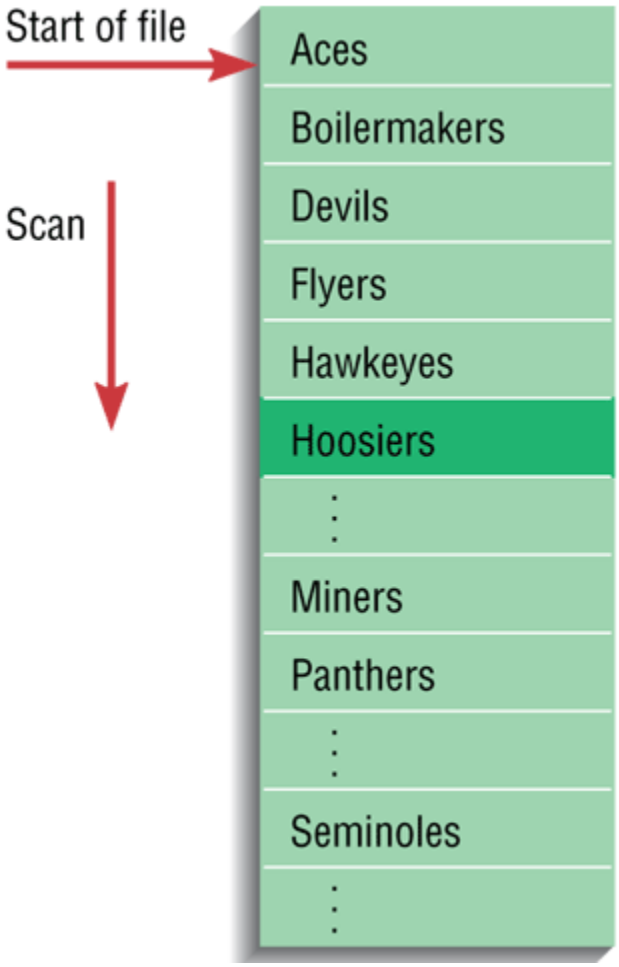


Figure: Sequential File Organization

b i i z a y @ g m a i l . c o m

Sequential file organization is used in applications like payroll management where the file is to be processed in entirety, i.e. each record is processed. Here, to have an access to a particular record, each record must be examined until we get the desired record. Sequential files are normally created and stored on magnetic tape using batch processing method.

Advantages:

- Simple to understand.
- Easy to maintain and organize
- Loading a record requires only the record key.
- Relatively inexpensive I/O media and devices can be used.
- Easy to reconstruct the files.
- The proportion of file records to be processed is high.
- Best use of storage space
- Easy to program

Disadvantages:

- Entire file must be processed, to get specific information.
- Very low activity rate stored.
- Transactions must be stored and placed in sequence prior to processing.
- Data redundancy is high, as same data can be stored at different places with different keys.
- Impossible to handle random enquiries.
- Records cannot be added at the middle of the file

Indexed Sequential Organization

Here the records are stored sequentially on a direct access device i.e. magnetic disk and the data is accessible randomly and sequentially. It covers the positive aspects of both sequential and direct access files. The type of file organization is suitable for both batch processing and online processing.

Here, the records are organized in sequence foe efficient processing of large batch jobs but an index is also used to speed up access to the records. Indexing permit access to selected records without searching the entire file.

Index

A table used to determine the location of rows in a file that satisfy some condition.

Secondary Index

Index based upon a combination of fields for which more than one row may have same combination of values.

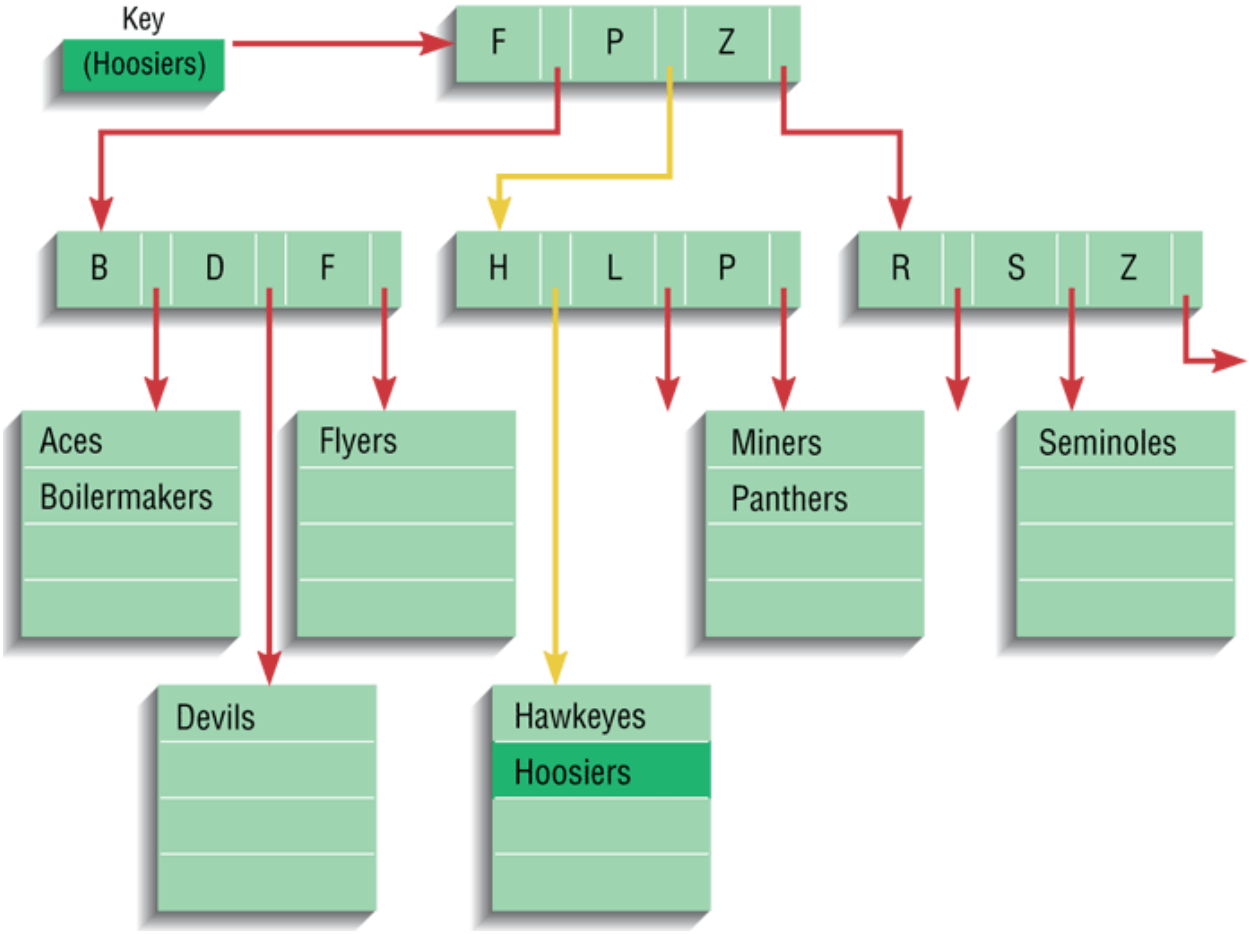


Figure: Indexed Sequential Organization

Advantages:

- Permits efficient and economic use of sequential processing technique when the activity rate is high.
- Permits quick access to records, in a relatively efficient way when this activity is a fraction of the work load.
- Indexed sequential organization reduces the magnitude of the sequential search and provides quick access for sequential and direct processing.
- Records can be inserted in the middle of the file.

Disadvantages:

- Slow retrieval, when compared to other methods.
- Does not use the storage space efficiently.
- Hardware and software used are relatively expensive.
- It takes longer to search the index for data access or retrieval.
- Unique keys are required
- Periodic reorganization is required.

Guidelines for Choosing Indexes

- Specify a unique index for the primary key of each table.
- Specify an index for foreign keys.
- Specify an index for non-key fields that are referenced in qualification, sorting and grouping commands for the purpose of retrieving data.

Hashed File Organization

Hashing is the most common form of purely random access to a file or database. A hash function is computed on some attribute of each record. It specifies the location of the record to be placed. In hashed file organization the address for each row is determined using an algorithm.

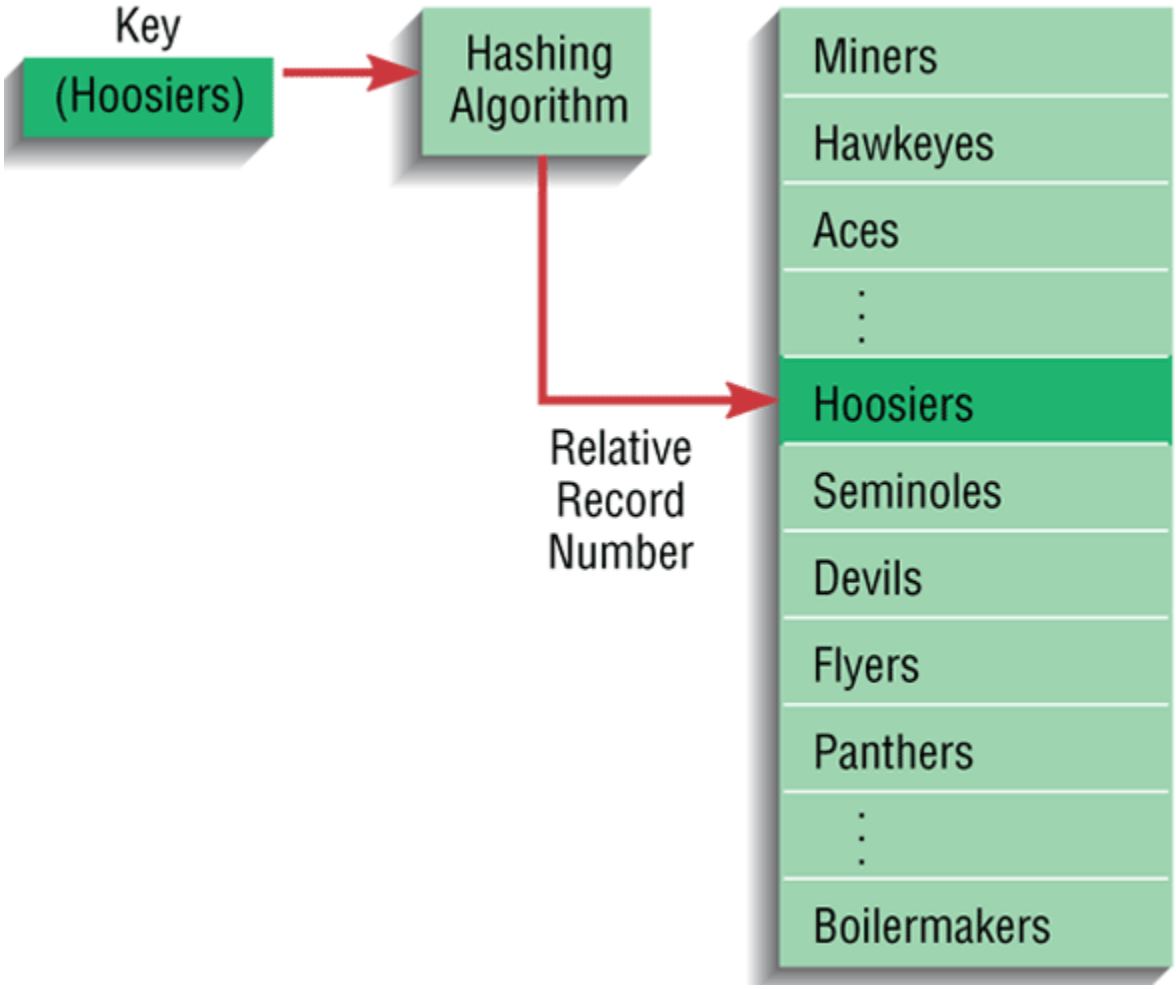


Figure: Hashed File Organization

Advantages:

Hash is a good storage structure in the following situations:

- When tuples are retrieve based on an exact match on the hash field value, particularly if the access order is random. For example, if the STUDENT relation is hashed on Name then retrieval of the tuple with Name equal to "Ram" is efficient.

Disadvantages:

Hash is not a good storage structure in the following situations:

- When tuples are retrieved based on a range of values for the hash field. For example, retrieve all students whose name begins with the "R".

- When tuples are retrieved based on a range of values for the hash field. For example, if STUDENT relation has hash filed Roll Number and the query is to retrieve all students with roll numbers in the range of 3000-5000.
- When tuples are retrieved based on a field other than the hash field. For example, if the STUDENT relation is hashed on Roll Number, then hashing cannot be used to search for a tuple based on the Class attribute.

- When tuples are retrieved based on only part of the hash field. For example, if the STUDENT relation is hashed on Roll Number and Class, then hashing cannot be used to search for a tuple based on the class attribute alone.

- When the hash field frequently updated. When a hash field updated, the DBMS must deleted the entire tuple and possible relocate it to a new address (if the has function results in a new address). Thus, frequent updating of the hash field impacts performance.

Inverted List Organization

Like the indexed-sequential storage method, the inverted list organization maintains an index. The two methods differ, however, in the index level and record storage. The indexed-sequential method has a multiple index for a given key, whereas the inverted list method has a single index for each key type. In an inverted list, records are not necessarily stored in a particular sequence. They are placed in the data storage area, but indexes are updated for the record keys and location.

Example: Data for the flight reservation system.

The flight number, description and the departure time are as given as keys. In the data location area, no particular sequence is followed. If a passenger needs information about the Houston flight, the agent requests the record with Houston flight. The DBMS carries a sequential search to find the required record. The output will then be that the flight number is 170 departing at 10.10 A.M and flight number 169 departing at 8.15 A.M.

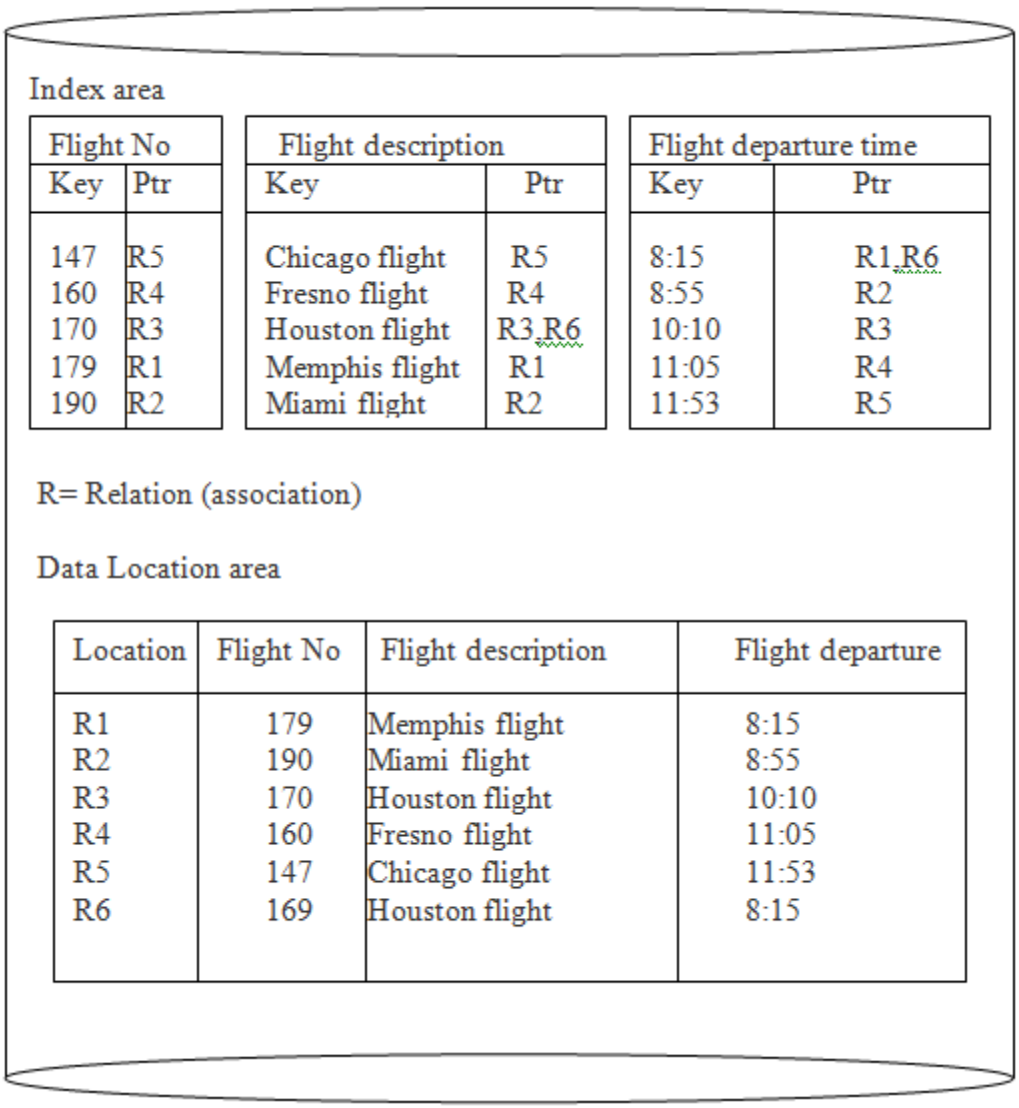
If the passenger searches for information about a Houston flight that departs at 8.15, then the DBMS searches the table and retrievesR3 and R6. Then it checks the flight departure time and retrieves R6 standing for flight number 169.

Advantages:

- It is best for applications that request specific data on multiple keys.
- They are ideal for static files because additions and deletions cause expensive pointer updating.
- Records are not necessarily stored in a particular sequence.

Disadvantages:

- Loss of “locality” so to improve this, keep the List File in main memory



Direct-Access Organization

Files in his type are stored in direct access storage devices such as magnetic disk, using an identifying key. The identifying key relates to tits actual storage position in the file. The computer can directly locate the key to find the desired record without having to search through any other record first. Here the records are stored randomly, hence the name random file. It uses online system where the response and updating are fast.

Advantages:

- Records can be immediately accessed for updating.
- Several files can be simultaneously updated during transaction processing.
- Transaction need not be sorted.
- Existing records can be amended or modified.
- Very easy to handle random enquiries.
- Most suitable for interactive online applications.
- Records can be inserted or updated in the middle of the file.
- Better control over record allocation.

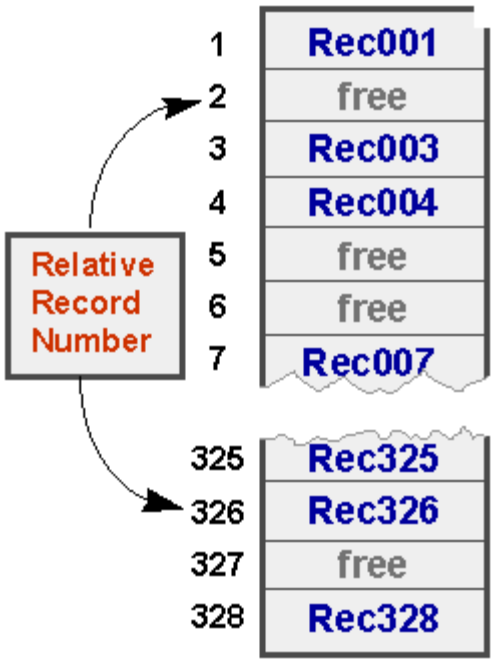


Figure: Direct-Access Organization

Disadvantages:

- Data may be accidentally erased or over written unless special precautions are taken.
- Risk of loss of accuracy and breach of security. Special backup and reconstruction procedures must be established.
- Les efficient use of storage space.
- Expensive hardware and software are required.
- High complexity in programming.
- File updating is more difficult when compared to that of sequential method.
- Calculating address required for processing.
- Impossible to process variable length records.

References:

- ✓ Hoffer, J.A., George, J.F. and Valacich J.S., "Modern Systems Analysis and Design", 3rd Edition, Pearson Education, 2003.
- ✓ K.E. Kendall and J.E. Kendall, "Systems Analysis and Design", 5th Edition, Pearson Education, 2003.
- ✓ Jeffrey L. Whitten, Loonnie D. Bentley, 5rd Edition, "Systems Analysis and Design Methods".
- ✓ Englewood Cliffs, New Jersey, "Systems Analysis and Design".
- ✓ V. Rajaraman, "Analysis and Design of Information Systems", 2nd Edition, Prentice Hall of India, New Delhi, 2002
- ✓ E.Yourdon "Modern Structured Analysis", Prentice Hall of India, 1996.
- ✓ Elias M Awad, "Systems Analysis and Design", Galgotia.
- ✓ Joseph S. Valacich, Joey F. George, Jeffrey A. Hoffer, "Essentials of Systems Analysis and Design" - 5th Edition.
- ✓ Shelly Cashman, "System Analysis and Design" - 9th Edition.

Assignments:

(1) What are the components of a system design?
(2) Explain the desirable features of a good quality design.
(3) How can you transform an E-R diagram into relations? Explain with suitable example. (T.U. 2067)
(4) What is the difference between a 2NF and 3NF relations? (T.U. 2067)
(5) Explain the six types of files used in information systems. (T.U. 2067)
(6) What do you mean by file organization? Explain with example. (T.U. 2067, 2068)
(7) What is the normalization of a relation? Explain with example. (T.U. 2067, 2069)
(8) What are the process for designing the forms and reports? (T.U. 2068)
(9) What do you mean by database normalization? Why is it important? (T.U. 2070)

A Gentle Advice:

Please go through your text books and reference books for detail study!!! Thank you all.

Notes Compiled By:

Bijay Mishra
biizay.blogspot.com
9813911076 or 9841695609

Unit 6 - System Implementation

6. System Implementation **8 Hrs.**

6.1 System Implementation (Introduction, Process of Coding, Testing and Installation)
6.2 The Process of Documenting the System, Training and Supporting Users
6.3 Software application Testing, Types of Testing
6.4 Quality Assurance
6.5 Maintenance: Adaptive, Corrective, Preventive, and Perfective
6.6 The Process of Maintaining Information Systems
6.7 Cost of Maintenance
6.8 Managing Maintenance
6.9 Measuring Effectiveness of Maintenance
6.10 Hardware/Software Selection and Computer Contract
6.11 Project Scheduling and Software

System Implementation:

Implementation is the process of building properly working system, install it in the organization, replacing old system and working methods, and finalize system and user documentation and training end prepared to support the system to assists users.

The purpose of implementation is to build a properly working system and to install it in the organization, replacing the old system and work methods as well as finalizing all system and user documentation, thoroughly training users and others to effectively use the new system, and preparing support systems to assist users as they encounter difficulties. Because implementation is such an important phase of system development, all the activities that are to be done during this phase are to be planned.

System implementation is made up of many activities. The six major activities are:

1. Coding
2. Testing
3. Installation
4. Documentation
5. Training
6. Support

Objective/Purpose of Implementation:

The objectives of implementation are:

1. Converting system design specification into working and reliable software and hardware.
2. Documenting the work that has been accomplished.
3. Providing help for current and future users and system caretakers.
4. Testing the software that has been developed.

b i i z a y @ g m a i l . c o m

The Process of Coding, Testing, and Installation

Coding:

Coding is the process where, physical design specifications created by the analysis team are turned into working computer code by program team. It means actually writing code or monitoring coding done by programmers to ensure that programs meet design specifications.

Coding Deliverables: Code, Program Documentation

Testing:

Testing is the process of finding and fixing bugs and errors. Tests are performed using various strategies. Testing performed in parallel with coding. It involves using test data and scenarios to verify that each component and the whole system work under normal and abnormal circumstances.

Testing Deliverables: Test Scenarios (test plan) and test data, Results of programs and system testing

Installation:

It is the process during which the current system is replaced by new system. It includes installing the new system in organizational sites as well as dealing with personal and organizational resistance to the change that the new system causes.

Installation Deliverables: User guides, User training plan, Installation and conversion plan

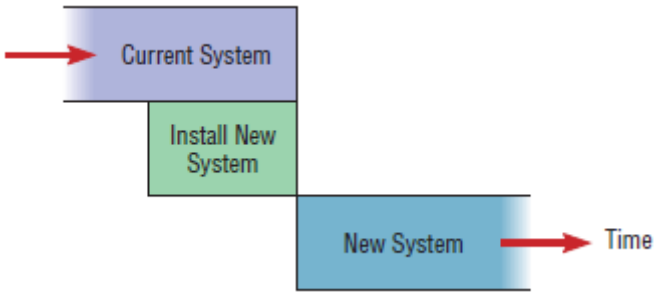
Types of Installation:

Four installation strategies:

- ✓ Direct Installation
- ✓ Parallel Installation
- ✓ Single-location installation
- ✓ Phased Installation

Direct Installation

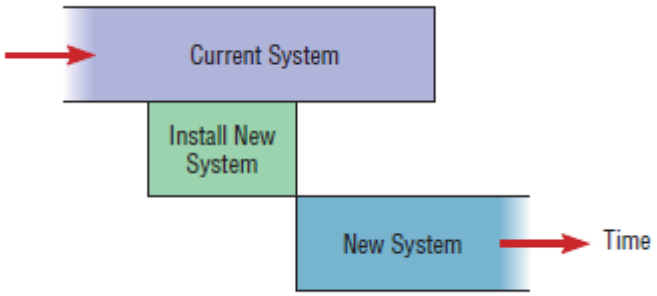
Changing over from the old information system to a new one by turning off the old system when the new one is turned on



Parallel Installation

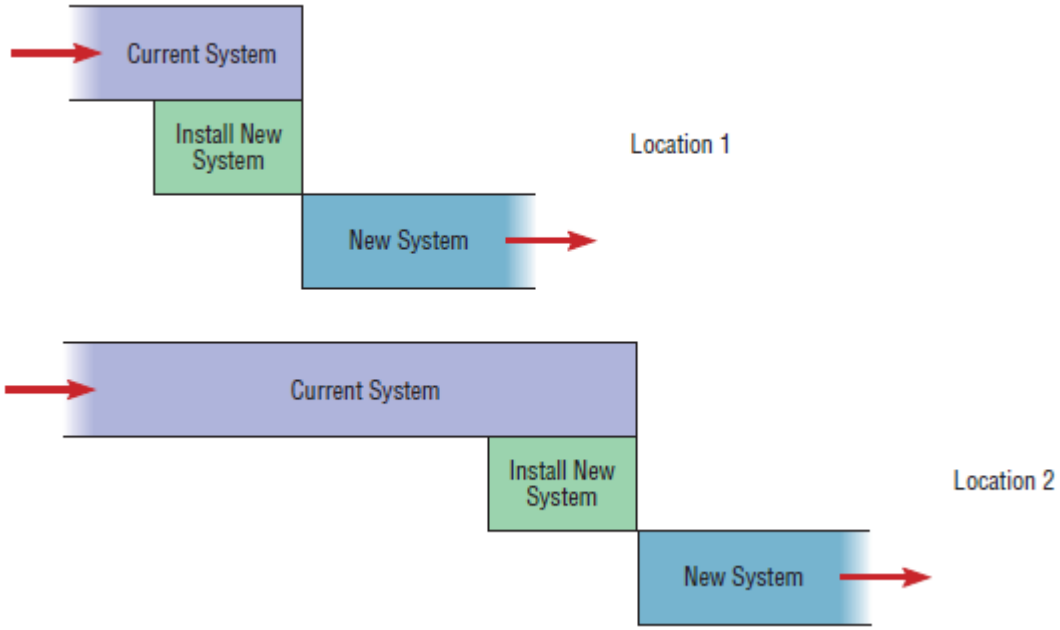
Running the old information system and the new one at the same time until management decides the old system can be turned off.

b i i z a y @ g m a i l . c o m



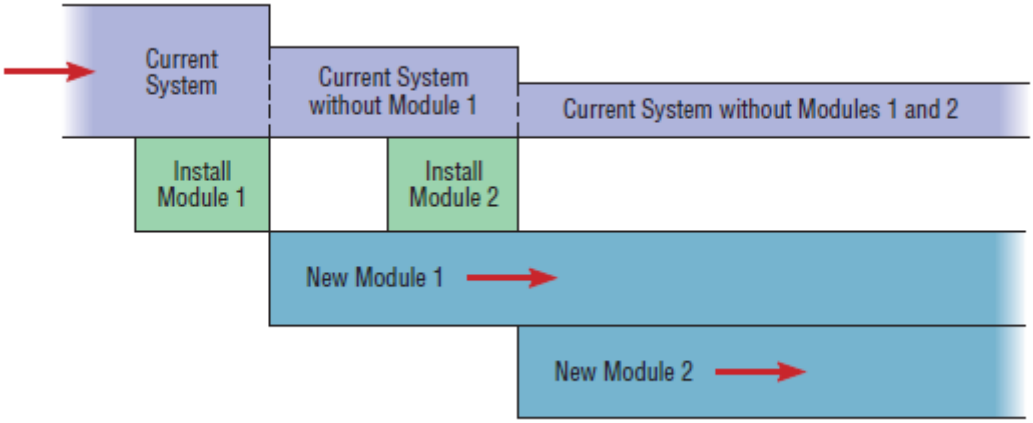
Single location installation

Trying out an information system at one site and using the experience to decide if and how the new system should be deployed throughout the organization



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.



b i i z a y @ g m a i l . c o m

The Process of Documenting the System, Training Users, and Supporting Users

Documentation:

It is the process of converting software specification with the hard copy format it acts as references maintained for user. It includes reviewing all project dictionary or CASE repository entries for completeness as well as finalizing all user documentation, such as user guides, reference cards and tutorials.

Two audiences for final documentation:

- Information systems personnel (System Builders, internal users) who will maintain the system throughout its productive life.
- People who will use the system (System Users) as part of their daily lives.

Documentation Deliverables: System Documentation, User Documentation

A. *System documentation*

It is detailed information about a system's design specifications, its internal workings and its functionality. The intended audience: maintenance programmers. System documentation is of two types:

i. *Internal documentation*

System documentation that is part of the program source code or is generated at compile time

ii. *External documentation*

System documentation that includes the outcome of structured diagramming techniques such as data-flow and entity-relationship diagrams

B. *User Documentation*

Written or other visual information about an application system, how it works, and how to use it

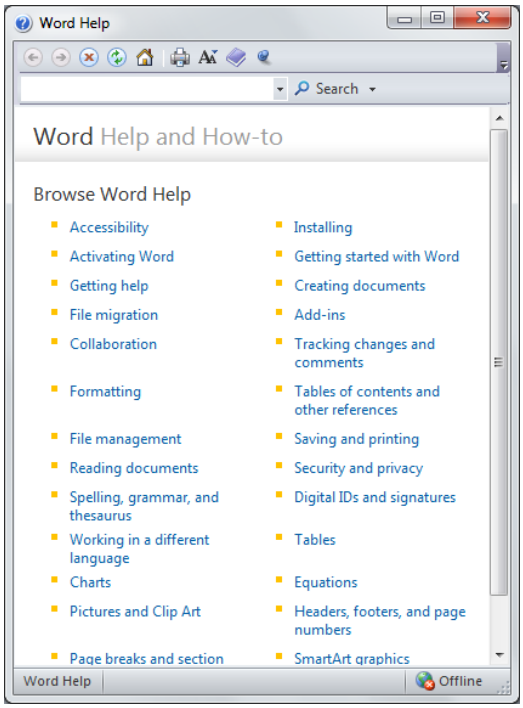


Figure: User documentation is typically in the form of online help (Microsoft Word 2010)

b i i z a y @ g m a i l . c o m

User Training:
It is the process of providing necessary skills and knowledge to user. It may include a variety of human and computer-assisted sessions as well as tools to explain the purpose and use of the system.

The training can be:

- Application specific or
- General for operating system and off the shelf software.

User Training Deliverables: Classes, Tutorials, User training modules, training materials, and computer based training aids

User Support:
It is the process of building of mechanism which acts as guidelines for the user in case of any problem arises. It ensures that users can obtain the assistance they need as questions and problems arise.

User Support Deliverables: Help desk, online help, Bulletin boards and other support mechanisms

Software Application Testing

Once coding has begun, the testing process can begin and proceed in parallel. As each program module is produced, it can be tested individually, then as part of a larger program, and then as part of a larger system. We should emphasize that although testing is done during implementation, you must begin planning for testing earlier in the project. Planning involves determining what needs to be tested and collecting test data. These activities are often done during the analysis phase, because testing requirements are related to system requirements.

Testing software begins earlier in the systems development life cycle, even though many of the actual testing activities are carried out during implementation. During analysis, you develop an overall test plan. During design, you develop a unit test plan, an integration test plan, and a system test plan. During implementation, these various plans are put into effect, and the actual testing is performed.

The purpose of these written test plans is to improve communication among all the people involved in testing the application software. The plan specifies what each person's role will be during testing. The test plans also serve as checklists you can use to determine whether all testing steps have been completed. The overall test plan is not just a single document but is a collection of documents. Each of the component documents represents a complete test plan for one part of the system or for a particular type of test.

Some organizations have specially trained personnel who supervise and support testing. Testing managers are responsible for developing test plans, establishing testing standards, integrating testing and development activities in the life cycle, and ensuring that test plans are completed. Testing specialists help develop test plans, create test cases and scenarios, execute the actual tests, and analyze and report test results.

Seven Different Types of Tests

Software application testing is an umbrella term that covers several types of tests. Tests can be done with or without executing the code, and they may be manual or automated. Using this framework, we can categorize types of tests as shown in table below:

	Manual	Automated
Static (Without code execution)	Inspection	Syntax checking
Dynamic (With code execution)	Walkthrough Desk checking	Unit test Integration test System test

Table: A categorization of test types

Inspection:

Inspections are formal group activities in which participants manually examine code for occurrences of well-known errors. Syntax, grammar, and some other routine errors can be checked in early stages of coding by automated inspection software, so manual inspection checks are used for more subtle errors. Code inspection participants compare the code they are examining to a checklist of well-known errors for that particular language.

Walkthrough:

In walkthrough, the correctness of the models produced is checked and the errors detected are notified for amendments. Unlike inspections, what the code does is an important question in a walkthrough. Using structured walkthroughs is an effective method of detecting errors in code. Structured walkthroughs can be used to review many systems development deliverables, including design specifications and code. Whereas specification walkthroughs tend to be formal reviews, code walkthroughs tend to be informal. Informality makes programmers less apprehensive of criticism and, thus, helps increase the frequency of walkthroughs. Code walkthroughs should be done frequently when the pieces of work reviewed are relatively small and before the work is formally tested.

Desk Checking:

Desk checking is an informal process where the programmer or someone else who understands the logic of the program works through the code with a paper and pencil. The programmer executes each instruction, using test cases that may or may not be written down. In one sense, the reviewer acts as the computer, mentally checking each step and its results for the entire set of computer instructions.

Syntax Checking:

Syntax checking is typically done by a compiler. Errors in syntax are uncovered, but the code is not executed.

Unit Test:

Unit testing is sometime called as module or functional testing. In unit testing, each module (roughly a section of code that performs a single function) is tested alone in an attempt to discover any errors that may exist in the module's code.

Integration Test:
It implies the process of bringing together all of the modules that a program comprises of for testing purpose. Modules are typically integrated in a top-down, incremental fashion.

Integration testing is gradual. First you test the highest level, or coordinating module, and only one of its subordinate modules. The process assumes a typical structure for a program, with one highest-level, or main, module and various subordinate modules referenced from the main module. Each subordinate module may have a set of modules subordinate to it, and so on, similar to an organization chart. Next, you continue testing subsequent modules at the same level until all subordinate to the highest-level module have been successfully tested together.

Once the program has been tested successfully with the high-level module and all of its immediate subordinate modules, you add modules from the next level one at a time. Again, you move forward only when tests are successfully completed. If an error occurs, the process stops, the error is identified and corrected, and the test is redone. The process repeats until the entire program—all modules at all levels—is successfully integrated and tested with no errors.

System Test:
It implies the bringing together of all the programs that a system comprises of for testing purpose. Programs are typically integrated in a top-down, incremental fashion.

System testing is a similar process, but instead of integrating modules into programs for testing, you integrate programs into systems. System testing follows the same incremental logic that integration testing does. Under both integration and system testing, not only do individual modules and programs get tested many times, so do the interfaces between modules and programs.

Acceptance Testing

Once the system tests have been satisfactorily completed, the system is ready for acceptance testing, which is testing the system in the environment where it will eventually be used. Acceptance refers to the fact that users typically sign off on the system and "accept" it once they are satisfied with it. The purpose of acceptance testing is for users to determine whether the system meets their requirements. The extent of acceptance testing will vary with the organization and with the system in question.

The most complete acceptance testing will include ***alpha testing***, (also called mock client testing), where simulated but typical data are used for system testing; ***beta testing***, in which live data are used in the users' real working environment; and a system audit conducted by the organization's internal auditors or by members of the quality assurance group.

Alpha Testing
During alpha testing, the entire system is implemented in a test environment to discover whether the system is overtly destructive to itself or to the rest of the environment. The types of tests performed during alpha testing include the following:

Recovery Testing
It forces the software (or environment) to fail in order to verify that recovery is properly performed.

Security Testing
It verifies that protection mechanisms built into the system will protect it from improper penetration.

Stress Testing
It tries to break the system (e.g., what happens when a record is written to the database with incomplete information or what happens under extreme online transaction loads or with a large number of concurrent users).

Performance Testing
It determines how the system performs on the range of possible environments in which it may be used (e.g., different hardware configurations, networks, operating systems); often the goal is to have the system perform with similar response time and other performance measures in each environment.

Beta Testing
In beta testing, a subset of the intended users runs the system in their own environments using their own data. The intent of the beta test is to determine whether the software, documentation, technical support, and training activities work as intended. In essence, beta testing can be viewed as a rehearsal of the installation phase.

Quality Assurance

Quality assurance (QA) means carrying out timely checks to ensure that the system being developed meets the quality standards specified. It is necessary to carry out quality assurance checks to make sure that there are no errors in the system and that the system developed meets the original user requirements.

Quality assurance is the process of verifying or determining whether products or services meet or exceed customer assumptions. It is a process driven approach with specific step to define and attain goals. This process considers design, development, production, and service. Quality assurance is the validation of the output at each stage or phase against the statement of requirements.

SQA (Software Quality Assurance) consists of auditing and reporting functions of the software in order to provide management with gaining insight of the software and confidence that the product is meeting the goal. Conformance to software requirements is the foundation from which software quality is measured.

Analysts use four levels of quality assurance: testing, verification, validation, and certification.

Validation: "Are we building the right product?"
Verification: "Are we building the product right?"

Certification: Software certification is an endorsement of the correctness of the program, an issue that is rising in importance for information systems applications.

Maintenance

Maintenance is the changes made to a system to fix or enhance its functionality. Software maintenance is a very broad activity that includes error corrections, enhancements of capabilities, deletion of obsolete capabilities, and optimization.

The deliverables and outcomes from the process of maintenance are the development of a new version of the software and new versions of all design documents created or modified during the maintenance effort.

Types of Maintenance

There are four types of maintenance:

Corrective maintenance

Changes made to a system to repair flaws in its design, coding, or implementation

Adaptive maintenance

Changes made to a system to evolve its functionality to changing business needs or technologies

Perfective maintenance

Changes made to a system to add new features or to improve performance

Preventive maintenance

Changes made to a system to avoid possible future problems

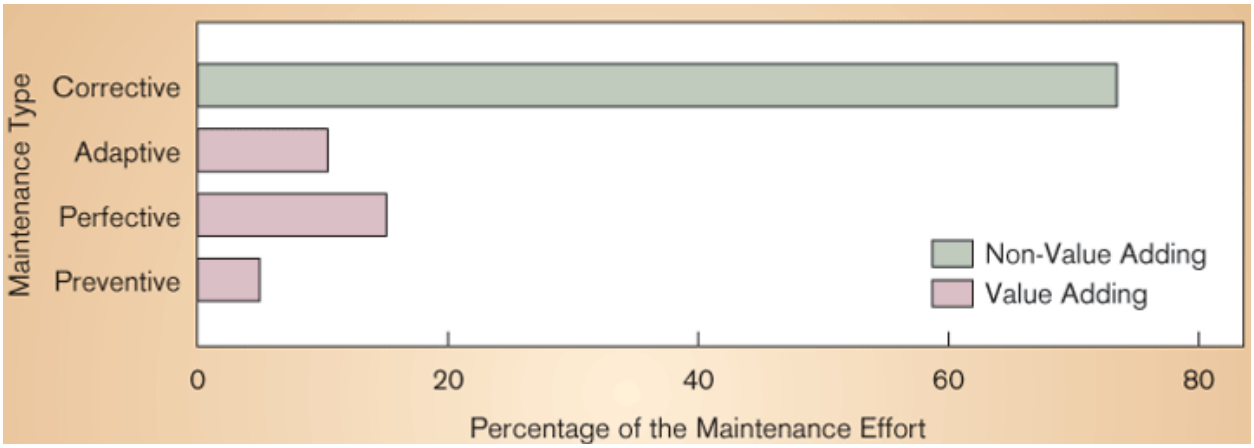


Figure: Maintenance Types and their Corresponding Efforts

By far, most maintenance is corrective, and therefore urgent and non-value adding.

Problems during Maintenance

Some of the problems that arises during maintenance are:

- Often the program is written by another person or group of persons.
- Often the program is changed by person who did not understand it clearly.
- Program listings are not structured.
- High staff turnover.
- Information gap.
- Systems are not designed for change.

b i i z a y @ g m a i l . c o m

The Process of Maintaining Information Systems

It is the process of returning to the beginning of the SDLC and repeating development steps focusing on system change until the change is implemented. Maintenance is the longest phase in the SDLC. Maintenance is like a mini-SDLC.

Once a request is received, analysis must be conducted to gain an understanding of the scope of the request. It must be determined how the request will affect the current system and the duration of such a project. As with the initial development of a system, the size of a maintenance request can be analyzed for risk and feasibility. Next, a change request can be transformed into a formal design change, which can then be fed into the maintenance implementation phase. Thus, many similarities exist between the SDLC and the activities within the maintenance process.

Four major activities are involved in maintenance:

(1) Obtaining maintenance requests
(2) Transforming requests into changes
(3) Designing changes
(4) Implementing changes

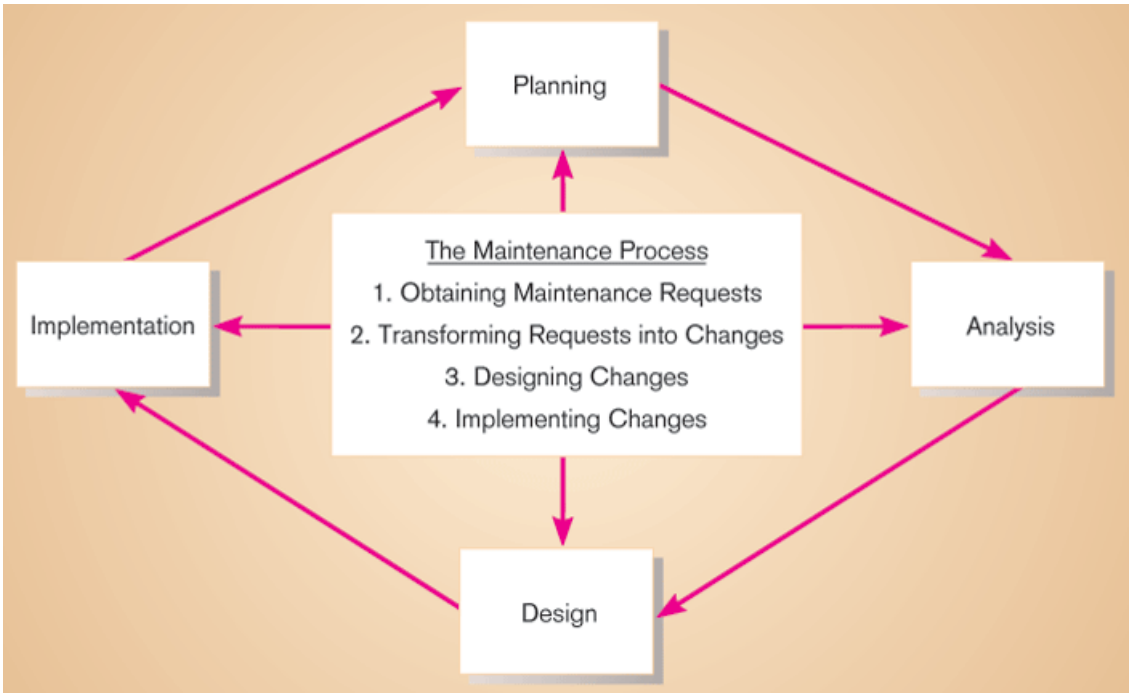


Figure: The Maintenance Process

The figure shows that the first phase of the SDLC - systems planning and selection - is analogous to the maintenance process of obtaining a maintenance request (step 1). The SDLC phase systems analysis is analogous to the maintenance process of transforming requests into a specific system change (step 2). The systems design phase of the SDLC, of course, equates to the designing changes process (step 3). Finally, the SDLC phase implementation and operation equates to implementing changes (step 4). This similarity between the maintenance process and the SDLC is no accident. The concepts and techniques used to develop a system initially are also used to maintain it.

The Cost of Maintenance

Information systems maintenance costs are a significant expenditure. For some organizations, as much 60-80 percent of their information systems budget is allocated to maintenance activities.

Maintainability is the ease with which software can be understood, corrected, adapted, and enhanced. Systems with low maintainability result in uncontrollable maintenance expenses.

Numerous factors influence the maintainability of a system. These factors, or cost elements, determine the extent to which a system has high or low maintainability. Of these factors, three are most significant: number of latent defects, number of customers, and documentation quality. The others—personnel, tools, and software structure—have noticeable, but less, influence.

The factors that influence system maintainability are:

- Latent defects
- Number of customers for a given system
- Quality of system documentation
- Maintenance personnel
- Tools
- Well-structured programs

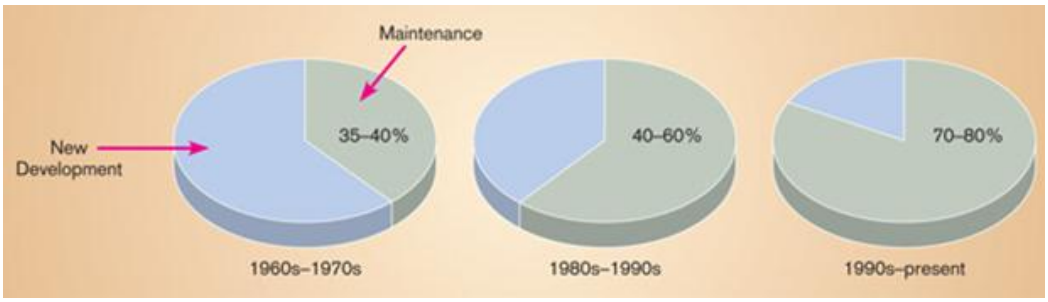
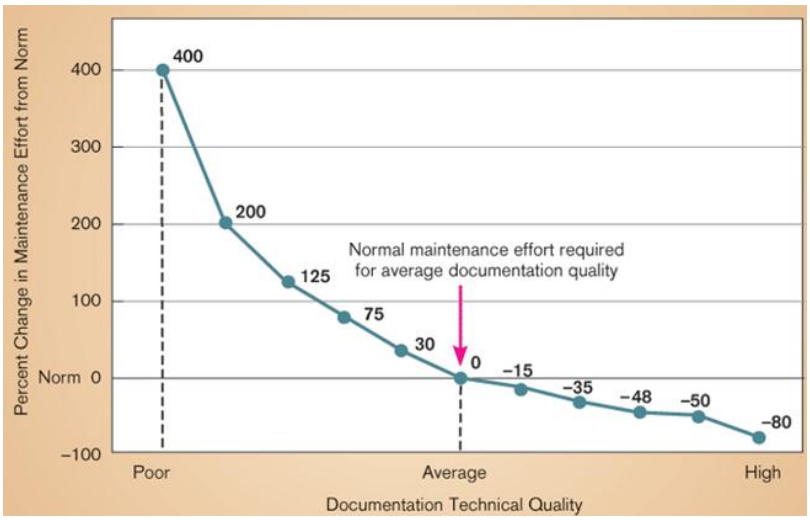


Fig: New development versus maintenance as a percent of software budget



A well-documented system is easier to understand, and therefore easier to maintain.

b i i z a y @ g m a i l . c o m

Managing Maintenance

Maintenance can be managed through four ways:

(a) Managing Maintenance Personnel
(b) Measuring Maintenance Effectiveness
(c) Controlling Maintenance Request
(d) Configuration Management

Managing Maintenance Personnel

Number of people working in maintenance has surpassed number working in development. Maintenance work is often viewed negatively by IS personnel. Organizations have historically have rewarded people involved in new development better than maintenance personnel. Organizations often rotate personnel in and out of maintenance roles in order to lessen negative feelings about maintenance. Three possible organizational structures exist:

Separate

Maintenance group consists of different personnel than development group.

Combined

Developers also maintain systems.

Functional

Maintenance personnel work within the functional business unit.

Maintenance Organization Type	Advantages	Disadvantages
Separate	Improved system and documentation quality	Ignorance of critical undocumented information
Combined	Maintenance group knows all about system	Less emphasis on good documentation
Functional	Personnel have vested interest	Limited job mobility and human or technical resources

Table: Advantages and Disadvantages of Different Maintenance Organization Type

Measuring Maintenance Effectiveness

Because maintenance can be so costly, it is important to measure its effectiveness. To measure effectiveness, you must measure these factors:

- ✓ Number of failures
- ✓ Time between each failure
- ✓ Type of failure

Measuring the number of and time between failures will provide you with the basis to calculate a widely-used measure of system quality. This measure is referred to as the mean time between failures (MTBF).

As its name implies, the MTBF measure shows the average length of time between the identification of one system failure until the next. Over time, you should expect the MTBF value to increase rapidly after a few months of use (and corrective maintenance) of the system. If the MTBF does not rapidly increase over time, it will be a signal to management that major problems exist within the system that are not being adequately resolved through the maintenance process.

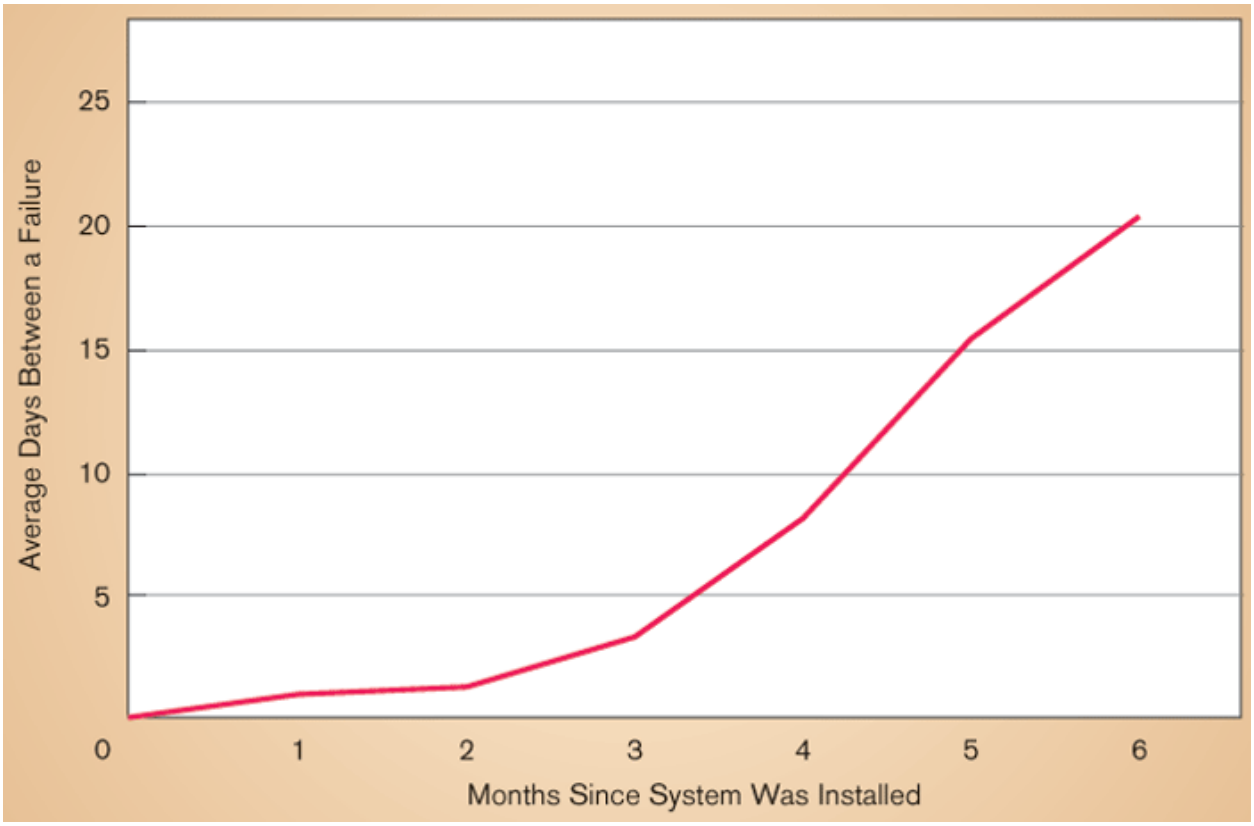


Figure: How the mean time between failures should change over time

Controlling Maintenance Requests

Another maintenance activity is managing maintenance requests. From a management perspective, a key issue is deciding which requests to perform and which to ignore. Because some requests will be more critical than others, some method of prioritizing requests must be determined.

The figure below shows a flowchart that suggests one possible method for dealing with maintenance change requests.

b i i z a y @ g m a i l . c o m

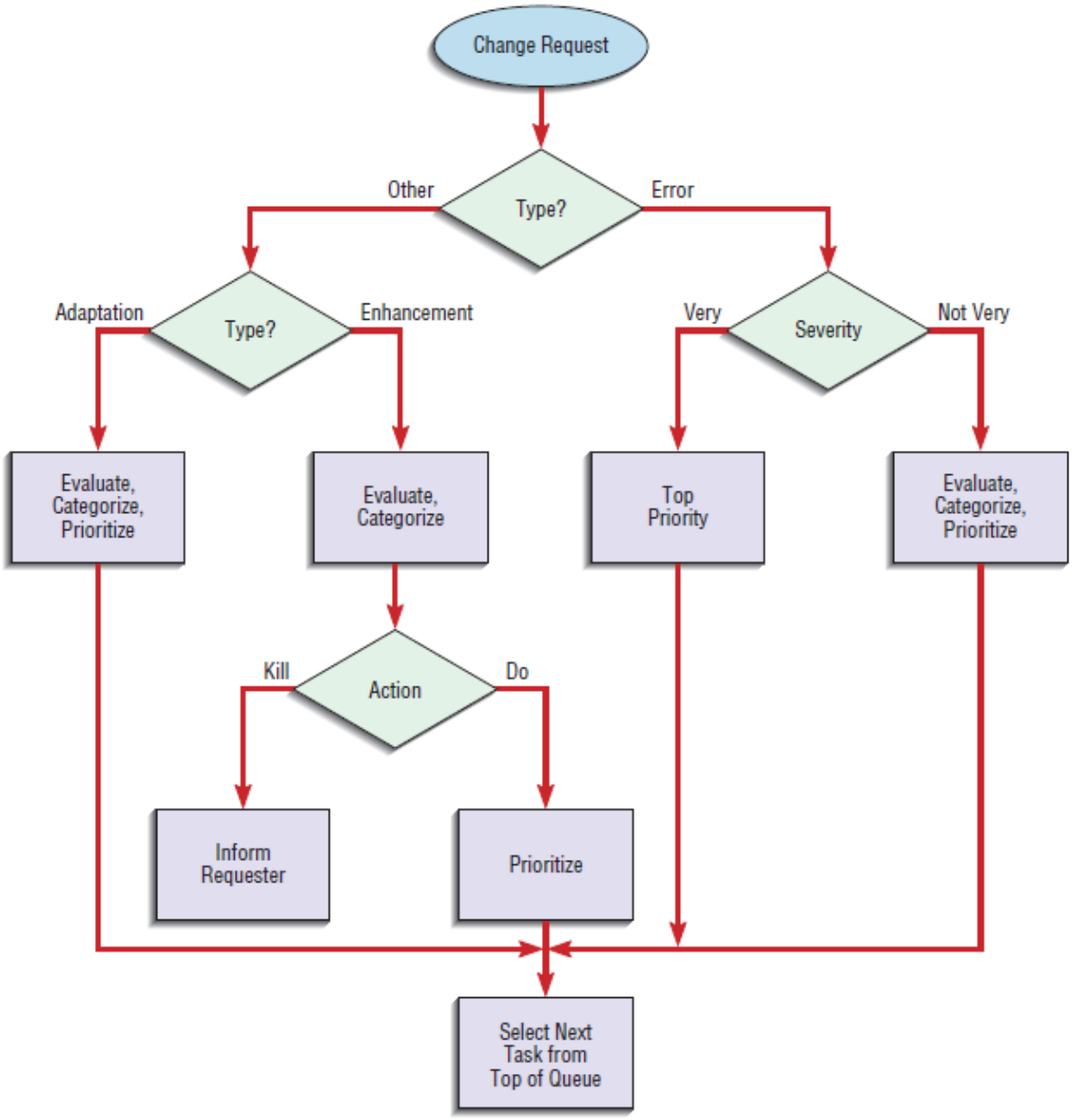


Figure: Flowchart showing how to control maintenance requests

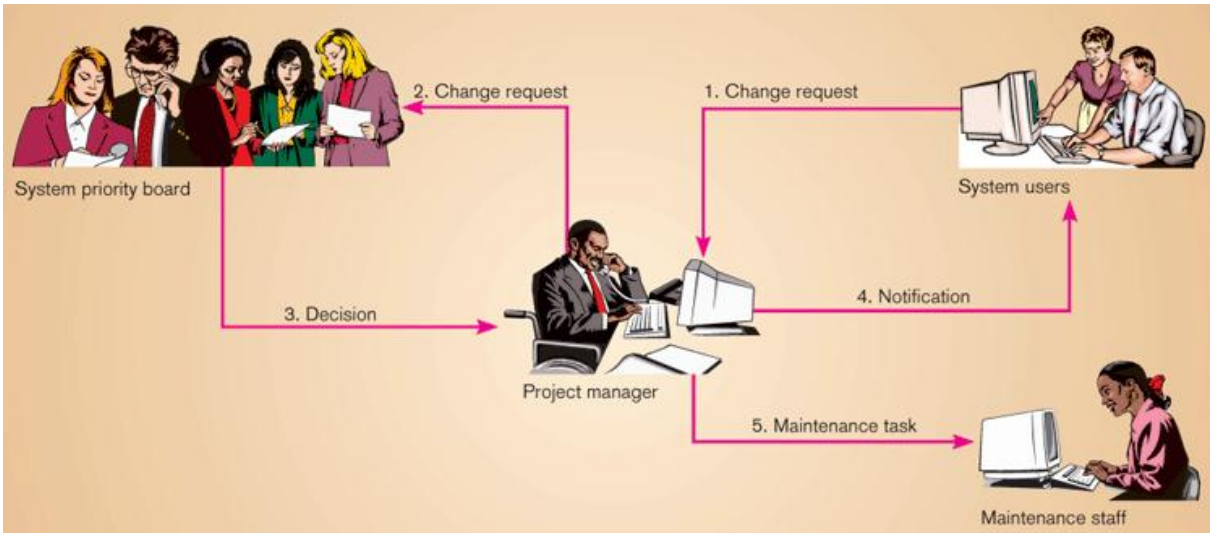


Figure: The Flow of a Maintenance Request

b i i z a y @ g m a i l . c o m

Configuration Management

A final aspect of managing maintenance is configuration management, which is the process of ensuring that only authorized changes are made to a system. Once a system has been implemented and installed, the programming code used to construct the system represents the baseline modules of the system.

Some of the configuration techniques are:

- ✓ *Baseline modules*
  Software modules that have been tested, documented, and approved to be included in the most recently created version of a system

- ✓ *System librarian*
  A person responsible for controlling the checking out and checking in of baseline modules when a system is being developed or maintained

- ✓ *Build routines*
  Guidelines that list the instructions to construct an executable system from the baseline source code

Configuration Management Tools:

They are the special software systems for managing system configuration. They control access to modules in the system library. Historical changes can be traced and previous versions can be reconstructed. Two types of control exists:

Revision control: modules labeled as frozen (unchangeable) or floating (checked out by programmer for modification)

Source code control: extend revision control to all interrelated modules

Hardware/Software Selection and the Computer Contract

Once a decision has been reached to install an in-house computing device, the next step is to prepare a list of specifications of the proposed system so that suitable vendors would be invited for meeting the specific requirements. The tender specifications are prepared as per norms of approved feasibility report. Main technical parameters of the various units of the required hardware objectives of the project and implementation schedule are also included in the tender specifications. Vendors may also be asked to quote separately in respect of leasing' and 'buying' options. In addition to this, the vendors may be required to furnish the details of the infrastructure which the customer will have to arrange.

Tender Evaluations

It is often seen that requirements as indicated by the customer do not match with the offer made by individual vendors where the specification given by the vendors are far below the essential requirements of the customers, such offers may be rejected straightway from the purview of short listing. Marginal shortfalls may be considered on merits. However, in case of additional features in the offers which could be categorized as 'desirable', it becomes necessary to assign appropriate weights to such features, in order to bring all the bids on equal footing.

The additional features include quantifiable differences are:

1. One-time costs as well as in the continually running and maintenance costs.
2. Equipment characteristics such as storage capacities, speeds of various units of computing device.
3. In-built spare capacity as well as capability of the system to support additional peripherals.
4. Additional support to be provided by the vendor.

Costing Factor

Cost consideration is quite important factor in computer acquisitions. Costs are of two types:

1. One-time costs such as post of site preparation (space, false ceilings, special floorings, electrical fittings, air- conditioning etc.)
2. Continued running and maintenance costs of the entire installations.

Equipment Characteristics

Hardware device which provides higher transfer rates (that is arrange number of bytes passing between various functional units per unit of time), or has large storage capacity or in case of printers, if the printed characters per line are quite high, then such additional characteristics get entitled to weightage to the extent of their practical utility to the buying organization. Appropriate weightage can also be given to such characteristics as high mean-time-between-failures (MTBF), compatibility with the equipment, peripherals etc. available in the market

Potential for Growth

Following features can be considered in this category:

1. Potentiality of the system to grow beyond the currently specified capacity by adding on certain components,
2. Potentiality of growth within a particular family of computer models,
3. Capacity of the system to handle a large variety of peripherals,
4. Ability of the system to handle additional workloads after considering the peak hour load.

Vendor Support

The features to be given weightage under the vendor support include:

- ✓ Hardware maintenance facilities offered.
- ✓ Training facilities provided.
- ✓ Assistance to be provided in software development
- ✓ Back-up facilities provided by vendor in case of system failure
- ✓ Comparative delivery periods offered by different vendors.

Service Suppliers

Outside computer services are commonly used by small firms or first time users. Also called 'Services' they include the following:

1. Computer manufacturers supply services such as system design, programming, education and training and hardware maintenance.
2. Service bureaus run "bread and butter" applications for small firms. Larger firms contract for specialized applications or for running jobs during peak volume periods. The primary services are programming, file and system conversions, system design and user training.

3. Facilities management (FM) furnishes specialists to manage a user installed computer on the user's premises. In some cases service is limited to developing application programs. The user runs the system but calls on the service organization for developmental work and maintenance.

Evaluation and Validation

The evaluation phase ranks vendor proposals and determines the one best suited to the user's needs. It looks into items such as price, availability, and technical support. System validation ensures that the vendors can in fact match his/her claims, especially system performance. True validation is verified by having each system demonstrated.

Role of Consultant

For a small firm, an analysis of competitive bids can be confusing. For this reason, the user may wish to contract on outside consultant to do the job. Consultants provide expertise and an objective opinion. A recent survey found, however, that 50 percent of respondent users had an unfavorable experience with the consultants they hired, and 25 percent said they would never hire another consultant. With such findings, a decision to use consultants should be based on careful selection and planning. A rule of thumb is that the larger the acquisition the more serious should be the consideration of using professional help.

Although the payoffs from using consulting services can be dramatic, the costs are also high for many small companies that are exploring system acquisition, for them, consulting services may be totally out of reach. During 1984, the average rates of consultants were $600 - $1,800 a day, not including travel and out-of-pocket expenses.

The past decade has seen the growth of internal management consultant terms in large organizations as opposed to external consulting teams. The figure below outlines the cases where an external or internal consultant is appropriate.

External Consultant	Internal Consultant
Full-time internal consultant is not needed or is beyond the budget of the organization.	An outside consultant is too costly; internal consultants can be much chapter.
Extra help on a project is needed for a short time, and internal person cannot afford the time.	A fast decision necessitates using an internal consultant.
The internal staff does not possess the expertise or broad knowledge needed for a specific situation.	An external consultant often does not understand the nature of the internal problem.
The political nature of the problem requires an objective, neutral opinion.	An internal consultant already exists who has an objective and technical understanding.
An outside opinion is desired in addition to that of the internal consultant.	An inside opinion is desired in addition to that of the external consultant.

Table: Pros and Cons of Using Consultants

Criteria for Vendor's Selection

Mandatory requirement is that, if a vendor fails to meet them, he would be screened out without any reason. The desirable characteristics would surely be little bit difficult to evaluate because he may offer several alternatives in lieu of them. The criteria of vendor selection may be listed in descending order by importance as below:

Economic Factors

- ✓ Cost comparisons
- ✓ Return on investment
- ✓ Acquisition method

Hardware Factors

- ✓ Hardware performance and its reliability
- ✓ Facilities for back up facilities
- ✓ Provision for back up facilities
- ✓ Firmness of delivery data
- ✓ Compatibility with existing systems
- ✓ Expandability

Software Factors

- ✓ Performance of software and its price
- ✓ Efficiency and reliability of available software
- ✓ Programming languages available
- ✓ Availability of well documented package programs
- ✓ Firmness of delivery date for a promised software
- ✓ Ease of use and modification as per user requirements
- ✓ Portability and its capacity to interface with the environment.

Service Factors

- ✓ Facilities provided by the manufacturer for detecting errors in the new programs
- ✓ Providing of good training facilities
- ✓ Assistance in software development and conversion facilities provided
- ✓ Maintenance terms and quality

Reputation of Manufacturer

- ✓ Financial stability
- ✓ Past history for keeping promises
- ✓ Three criteria may have to be further sub-divided particularly for hardware performance and support services.

Software Selection

Software selection is a critical aspect of system development. As mentioned earlier, the search starts with the software, followed by the hardware. There are two ways of acquiring software: custom-made or "off the-shelf packages. Today's trend is toward purchasing packages, which represent roughly 10 percent of what is a cost to develop the same in house.

In addition to reduced cost, there are other advantages:

1. A good package can get the system running in a matter of days rather than the weeks or months required for "home-grown" packages.
2. MIS personnel are released for other projects.
3. Packages are generally reliable and perform according to stated documentation.
4. Minimum risks are usually associated with large-scale systems and programming efforts.
5. Delays in completing software projects in house often occur because programmers quit 'n midstream.
6. It is difficult to predict the cost of "home-grown" software.
7. The user has a chance of seeing how well the package performs before purchasing it.

There are drawbacks, however, to software packages:

1. The package may not meet user requirements adequately.
2. Extensive modification of a package usually results in loss of the vendor support.
3. The methodology for package evaluation and selection is often poorly defined. The result is a haphazard review based on a faulty process or questionable selection criteria.
4. For first-time software package users, the overall expectation from package is often unclear and ill defined.

It can be seen, then, that the quality of a software package cannot be determined by price alone. A systematic review is crucial.

Criteria for Software Selection

Prior to selecting the software, the project team must set up criteria for selection. Selection criteria fall into the categories described here.

Reliability

It is the probability that the software will execute for specified time period without a failure, weighted by the cost to the user of each failure encountered. It relates to the ease of recovery and ability to give consistent results. Reliability is particularly important to the profession.

A hardware failure is based lately on random failures whereas software reliability is based on predestined errors. Although reliable software is a desirable goal, limited progress has been made toward improving it in the last decade. The fact of unreliable software had led to the practice of securing maintenance agreements after the package is in operation. In a sense, unreliability is rewarded.

Software reliability brings up the concept of modularity, or the ease with which a package can be modified. This depends on whether the package was originally designed as a package or was retrofitted after its original development for single installation use. A package with a high degree modularity has the capacity to operate in many machine configurations and perhaps across manufacturers' product lines.

With modularity comes expandability, which emphasizes the sensitivity of a software package to handle an increased volume of transactions or to integrate with other programs. The following questions should be considered:

1. Is there room for expanding the master file?
2. How easily can additional fields, records, and files be added?

3. How much of the system becomes unusable when a part of it fails'.
4. Are there errors a user can make that will bring down the system?
5. What are the recovery capabilities?

Functionality
It is a definition of the facilities, performance, and the other factors that the user requires in the finished product. All such information comes from the user. The following are key questions to consider:

1. Do the input transactions, files, and reports contain the necessary data elements?
2. Are all the necessary computations and processing performed according to specifications?

Capacity
Capacity refers to the capability of the software package to handle the user's requirements for size of files, number of data elements volume of transactions and reports, and number of occurrences of data elements. All limitations should be checked.

Flexibility
It is a measure of the effort required to modify an operational program. One feature of flexibility is adaptability, which is a measure of the ease of extending the product.

Usability
This criterion refers to the effort required to operate, prepare the input, and interpret the output of a program. Additional points to be considered are portability and understandability. Portability refers to the ability of the software to be used on different hardware and operating systems. Understandability means that the purpose of the product is clear to the evaluator and that the package is clearly and simply written, is free of jargon, and contains sufficient references to readily available documents so that the reader can comprehend advanced contents that the reader can comprehend advanced contents.

Security
It is a measure of the likelihood that a system's user can accidentally or intentionally access or destroy unauthorized data. A key question is: How well can one control access of software or data files? Control provides system integrity.

Performance
It is a measure of the capacity of the software package to do what it is expected to do. This criterion focuses on throughput, or how effectively a package performs under peak loads. Each package should the evaluated for acceptance on the user's system.

The language in which a package is written and the operating system and additional performance considerations. If we plan to modify or extend a package, it is easier if it is written in a language that is commonly known to programmers. Likewise, if the package runs only under a disk operating system and the installation is under a full operating system, then either the system downgraded to handle the package as is. In either case, the change could be costly and counterproductive.

Serviceability

This criterion focuses on documentation and vendor support. Complete documentation is critical for software enhancement. It includes a narrative description of the system, system logic and logic flowcharts, input-output and file descriptions and layouts, and operator instructions. Vendor support assures the user adequate technical support for software installation, enhancements, and maintenance. The user should determine how much on-site technical assistance is provided by the vendor, especially during the first few weeks after the installation. The user expects on-site training and support as part of most commercial packages. It is vital to inquire about the amount of training provided. The user may require training at several levels—clerical, operations, programming, and management.

Ownership

Who owns the software once it is "sold" to the user? Most of the standard license agreement forms essentially lease the software to the user for an indefinite time. The user does not "own" it, which means that the source code is inaccessible for modification, except by the vendor. Many users enter into an escrow arrangement whereby the vendor deposits the source code with a third-party escrow agent who agrees to release the code to the user if the vendor goes out of business or is unable to perform the services specified in the license.

In acquiring software, several questions should be asked:

1. What rights to the software is the user buying?
2. Can the user sell or modify the software?
3. If the vendor is modifying the package especially for the user, can the vendor sell it to others within the same industry the user is in?
4. What restrictions are there to copying the software or documentation?"

Minimal Costs

Cost is a major consideration in deciding between in-house and vendor software. Cost-conscious users consider the following points:

1. Development and conversion costs.
2. Delivery schedule.
3. Cost and frequency of software modifications.
4. Usable life span of the package.

A summary of software criteria is presented as below:

Criterion	**Meaning**
1. Reliability	Gives consistent results
2. Functionality	Functions to standards
3. Capacity	Satisfies volume requirements
4. Flexibility	Adapts to changing needs
5. Usability	Easy to operate and understand-usually-friendly
6. Security	Maintains integrity and prevents unauthorized access
7. Performance	Capacity to deliver as expected
8. Serviceability	Good documentation and vendor support
9. Ownership	Right to modify and share use of package
10. Minimal costs	Affordable for intended application.

Project Scheduling and Software

During project planning phase, an agreed-upon baseline is established for the project schedule. This schedule baseline will be used as a starting point against which performance on the project will be measured. It is one of many tools the Project Manager can use during Project Execution and control to determine if the project is on track.

Project Team members use the communications mechanisms documented in the Communications Plan to provide feedback to the Project Manager on their progress. Generally team members document the time spent on tasks and provides estimates of the time required to complete them. The Manager uses this information to update the Project Schedule. In some areas there may be formal time tracking systems that are used to track project activity.

After updating the Project Schedule, the Project Manager must take the time to review the status of the project. Some questions that the Project Manager should be able to answer by examining the project schedule include:

1. Is the project on track?
2. Are there any issues that are becoming evident that need to be addressed now?
3. Which tasks are taking more time than estimated? Less time?
4. If a task is late, what is the effect on subsequent tasks?
5. What is the next deliverable to be produced and when is it scheduled to be complete?
6. What is the amount of effort expended so far and how much is remaining?
7. Are any Project Team members over-allocated or under allocated?
8. How much of the time allocated has been expended to date and what is the time required to complete the project?
9. Most project scheduling tools provide the ability to produce reports to display a variety of useful information. It is recommended that the Project Manager experiment with all available reports to find those that are most useful for reporting information to the Project Team, Customer, and Project Sponsor and/or Project Director.
10. When updating the Project Schedule, it is very important that the Project Manager maintain the integrity of the current schedule. Each version of the schedule should be archived. By creating a new copy of the schedule whenever it is updated, the Project Manager will never lose the running history of the project and will also have a copy of every schedule for audit purposes.

References:

- ✓ Hoffer, J.A., George, J.F. and Valacich J.S., “Modern Systems Analysis and Design”, 3rd Edition, Pearson Education, 2003.
- ✓ K.E. Kendall and J.E. Kendall, “Systems Analysis and Design”, 5th Edition, Pearson Education, 2003.
- ✓ V. Rajaraman, “Analysis and Design of Information Systems”, 2nd Edition, Prentice Hall of India, New Delhi, 2002
- ✓ Joseph S. Valacich, Joey F. George, Jeffrey A. Hoffer, “Essentials of Systems Analysis and Design” - 5th Edition.
- ✓ Shelly Cashman, “System Analysis and Design” - 9th Edition.
- ✓ Jeffrey L. Whitten, Loonnie D. Bentley, 5rd Edition, “Systems Analysis and Design Methods”.
- ✓ Englewood Cliffs, New Jersey, “Systems Analysis and Design”.

Assignments:

(1) Explain the steps in the maintenance process and contrast them with the phase of the systems development life cycle. (T.U 2067)
(2) What is the role of software application testing? (T.U 2067)
(3) Explain the factors that influence the cost of maintenance. (T.U 2067)
(4) What managerial issues can be better understood by measuring maintenance effectiveness? Explain. (T.U 2067)
(5) Differentiate between system documentation and user documentation. (T.U 2068, 2069)
(6) Explain the different types of maintenance. (T.U 2068, 2069)
(7) Explain the process of maintaining the information system with example. (T.U 2069)
(8) What are the two important things to remember about testing the system? (T.U 2069)
(9) What do you mean by quality assurance? Explain with example. (T.U 2070)
(10) Compare between corrective, adaptive, preventive, and perfective maintenance. (T.U 2070)
(11) What are the main deliverable from testing and installation? (T.U 2070)

A Gentle Advice:

Please go through your text books and reference books for detail study!!! Thank you all.

Notes Compiled By:

Bijay Mishra
biizay.blogspot.com
9813911076 or 9841695609

Unit 7 - Object-Oriented Analysis and Design

7. Object-Oriented Analysis and Design **6 Hrs.**

- **7.1 Introduction**
- **7.2 Object-Oriented Development Life Cycle**
- **7.3 The Unified Modeling Language**
- **7.4 Use-Case Modeling**
- **7.5 Object Modeling: Class Diagrams, Object Diagram**
- **7.6 Dynamic Modeling: State Diagrams**
- **7.7 Dynamic Modeling: Sequence Diagrams**
- **7.8 Analysis versus Design**

Introduction

OOAD techniques are best suited to projects that will implement systems using emerging object technologies to construct, manage and assemble those objects into useful computer applications. OOAD is centered on a technique referred to as *Object Modeling*. Object Modeling is a technique for identifying objects within the system environment and identifying the relation between those objects.

Consists of two parts:
1) Object Oriented Analysis
2) Object Oriented Design

Object-Oriented Analysis (OOA)

OOA is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain. It is concerned with developing an object-oriented model of the problem domain

Object-Oriented Design (OOD)

OOD is an approach used to specify the software solution in terms of collaborating objects, their attributes and their methods. It creates abstractions and mechanisms necessary to meet behavioral requirements determined during analysis. It also provides an object-oriented model of a software system to implement the identified requirements.

Object-Oriented Development Life Cycle (OODLC)

In contrast to traditional SDLC, the Object-Oriented System Development Life Cycle is more like an Onion than a waterfall.

In the early stages (or core) of development, the model we build is abstract, focusing on the external qualities of application system. As the model evolves, it becomes more and more detailed, shifting the focus on how the system will be build and how it should function. The emphasis in modeling should be an analysis and design, focusing on front-end conceptual issue, rather than back end implementation issues, which unnecessarily restrict design choice.

Object oriented cycle is like an onion, evolving from abstract to detailed, from external qualities to system architecture and algorithms.

The object oriented development life cycle consists of progressively developing an object representation through three phases - analysis, design, and implementation.

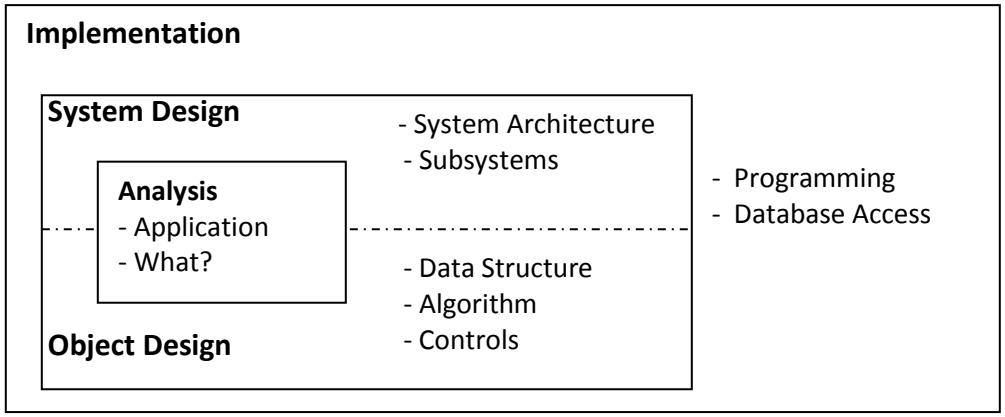


Figure: Phases of Object-Oriented System Development Life Cycle

The Process of Object-Oriented Analysis and Design

Analysis Phase

In the analysis phase, a model of the real world application is developed showing its important properties. It abstracts concepts from the application domain and describes what the intended system must do rather than how it will be done. The model specifies the functional behavior of the system, independent of any implementation details.

Design Phase

The requirement model is translated into a more detailed model that will be essential up to the implementation of the system. In the design phase, we define how the application-oriented analysis model will realized in the implementation environment. All strategic design decision – such as how the DBMS is to be incorporated, how process communication and error handling are to be achieved, what component libraries are to be reused or made. These decisions are incorporated into the design model that adapts to the implementation environment.

- System Design
  During System Design, one proposes an overall system architecture, which organizes the system into components called sub systems and provides the context to make decisions such as identifying concurrency, allocation of sub system to processors and task, handling access to global resources, selecting the implementation of control in software and more.

- Object Design
  During Object Design, one builds a design model by adding implementation details such as restructuring classes for efficiency, internal data structures and algorithms to implement each class, implementation of control, implementation of associations and packaging into physical modules.

Implementation Phase

The design phase is followed by the implementation phase. In this phase, one implements the design using a programming language and / or a database management system.

Benefits of Object Oriented Modeling

1. The ability to tackle more challenging problem domains.
2. Improved communication among users, analysts, designers and programmers.
3. Increased consistency among analysis, design and programming activities.
4. Explicit representation of commonality among system components.
5. Robustness of systems.
6. Reusability of analysis, design and programming results.
7. Increased consistency among all the models developed during Object-Oriented analysis design and programming.

The Unified Modeling Language

UML is a language for specifying, visualizing, constructing and documenting all the artifacts of software systems as well as for business modeling. It is largely based on the object-oriented paradigm and is an essential tool for developing robust and maintainable software systems.

The UML allows representing multiple views of a system by using a variety of graphical diagrams. The underlying model integrates those views so that the system can be analyzed and implemented in a complete and consistent fashion. UML includes many diagrams that represent different perspectives of a system.

Structure Diagrams (represent the static application structure)

1. Class diagram
2. Object diagram
3. Package diagram
4. Component diagram
5. Deployment diagram
6. Composite structure

Behavior Diagrams (represent different aspect of dynamic behavior)

1. Use Case diagram
2. Sequence diagram
3. Collaboration diagram
4. Activity diagram
5. State diagram
6. Communication diagram
7. Timing diagram

Use Case Modeling

Use Case modeling is used for analyzing the functional requirements of a system. It focuses on what the system does or the new system should do. A use case model is developed in the analysis phase of the OODLC. A Use Case model describes what a system does without describing how the system does. It reflects the view at the system from the perspective of a user outside of the system. A Use Case model partitions system functionality into behaviors (i.e. Use Cases) that are significant to the users of the system (i.e. Actors).

Actor

- An Actor is an external entity exists outside of the system and interacts with the system.
- It is someone or something that exchanges information with the system in specific way.
- Because actors are outside the system, we do not need to describe them in detail.
- The advantage of identifying actors is that it helps us to identify the Use Cases they carry out.
- An actor can be a human, another system, hardware devices – anything with which the system interacts or exchanges information.
- There is deference between an actor and a user. A user is anyone who uses the system. An actor on the other hand represents a role that a user can play.
- An actor is a type or class of users, a user is a specific instance of an actor class playing the actor's role.
- Notation for actor is:

Use Case

- A Use Case represents a sequence of related action initiated by an actor to accomplish a specific goal; it represents a specific way of using the system.
- It describes system functions from the perspective of external users.
- Notation for use case is:

Boundary

- A boundary is the dividing line between the system and its environment.
- Use cases are within the boundary.
- Actors are outside of the boundary.
- Notation for boundary is:

Connection/Communication Association

- A connection is an association between an actor and a use case.
- Depicts a usage relationship
- Connection does not indicate data flow
- Notation for connection is:

For identifying Use Cases, Jacobson (1992) recommends to ask following questions:-

- What are main task performed by each actor?
- Will the actor read or update any information in the system?
- Will the actor have to inform the system about changes outside the system?
- Does the actor have to be informed about unexpected changes?

Developing Use-Case Diagrams

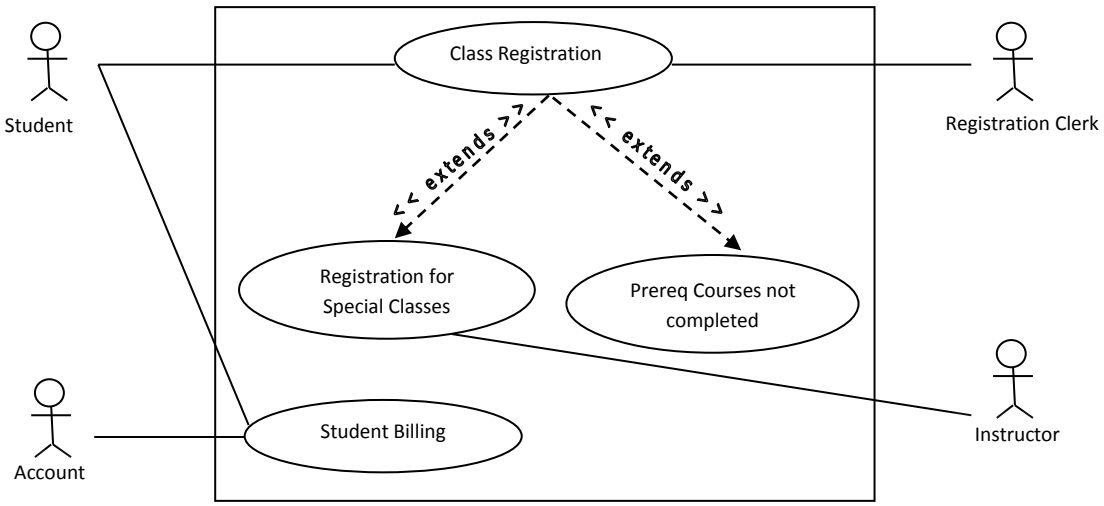


Figure: Use-Case diagram for a University Registration System

Relationship between Use Cases

A Use-Case may participate in relationship with other Use-Cases. Two types of relationships; extend and include.

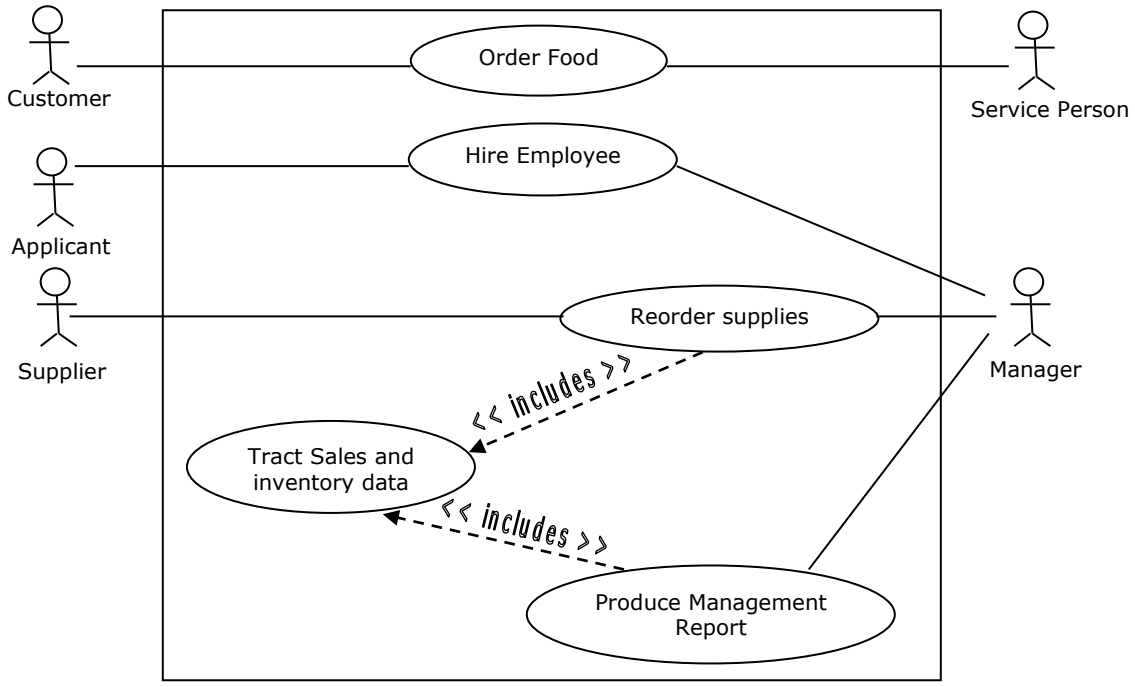
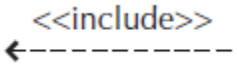


Figure: Use-Case diagram for a Burger's System

b i i z a y @ g m a i l . c o m

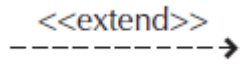
Include Relationship

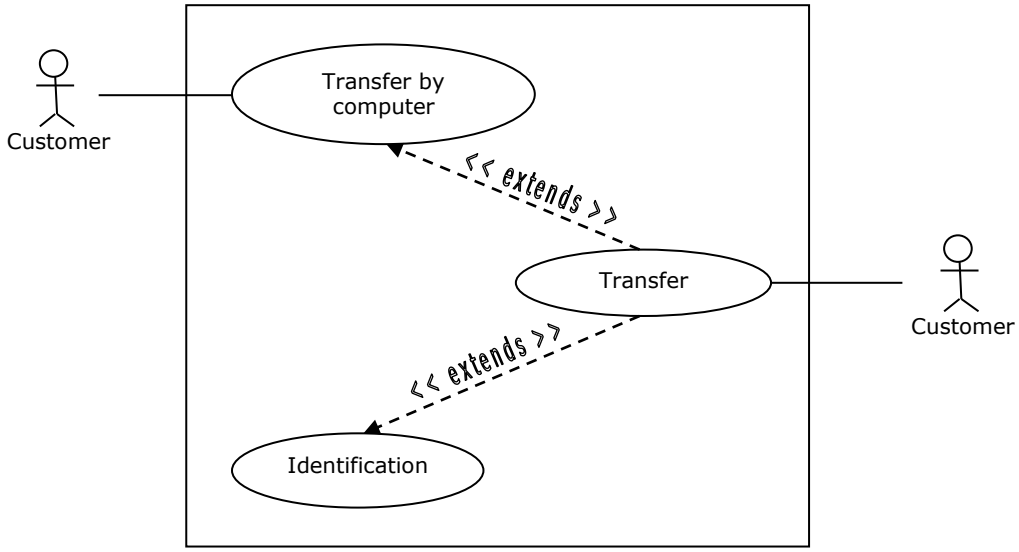
<<include>> shown as a dashed line pointing toward the Use-Case that is being used. Include denotes that one Use-Case uses other while executing. The arrow is drawn from the base use case to the included use case.



Extend Relationship

<<extend>> shown as a dashed arrow pointing toward extended Use-Case extends a Use-Case by adding new behavior or actions. The arrow is drawn from the extension use case to the base use case.





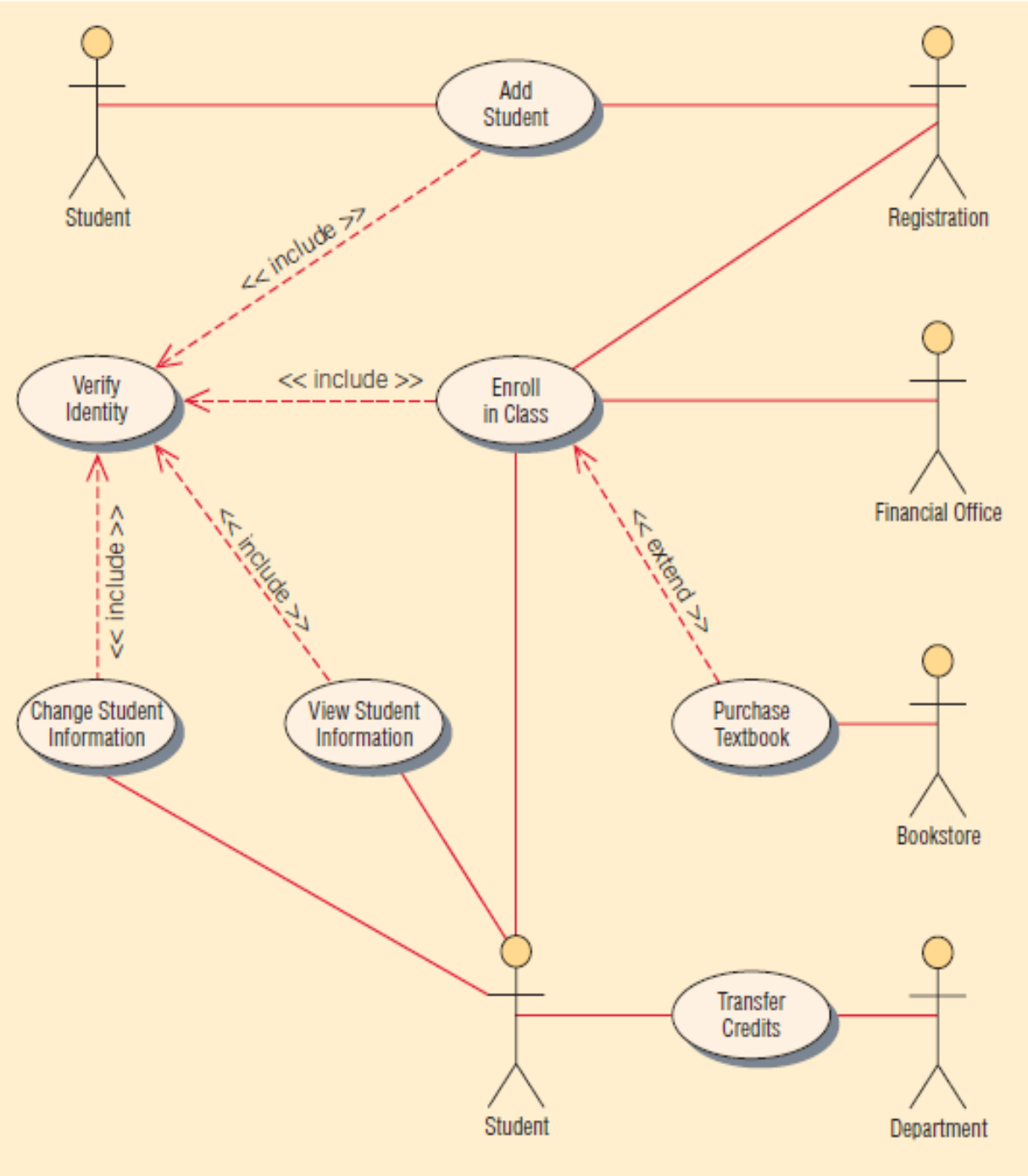
A Use Case diagram shows all the Use Cases in the system but does not describe how these use cases are carried out by actors. The contents of a use case are described in plain text. While describing a use case, focus on its external behavior i.e. how it interacts with actors rather than how the use case is performed inside the system.

If you want to model an extension to, or a variation of a complete Use-Case that exists in its own right, employ 'extend'.

If you need to factor the common behavior among two or more Use Cases into a single generalized Use Case, employ 'include'.

The common behavior described by the generalized Use Case must be inserted into a specialized Use Case to complete it.

Example: A use case example of student enrollment.



Example:

An Auto rental company wants to develop an automated system that would handle car reservations, customer billing and car auctions. Usually, a customer reserves a car, picks it up and then returns it after a certain period of time. At the time of pickup, the customer has the option to buy or waive collision insurance on the car. When the car is returned, the customer receives a bill and pays the specified amount. In addition to renting out cars, every six months or so, the auto rental company auctions the cars that have accumulated over 20,000 miles.

b i i z a y @ g m a i l . c o m

Draw a Use-Case diagram for capturing the requirements of the system to be developed.

Include an abstract Use Case for capturing the common behavior among any two use cases. Extend the diagram to capture corporate billing, where corporate customers are not billed directly; rather, the corporations they work for are billed and payments are made sometime later.

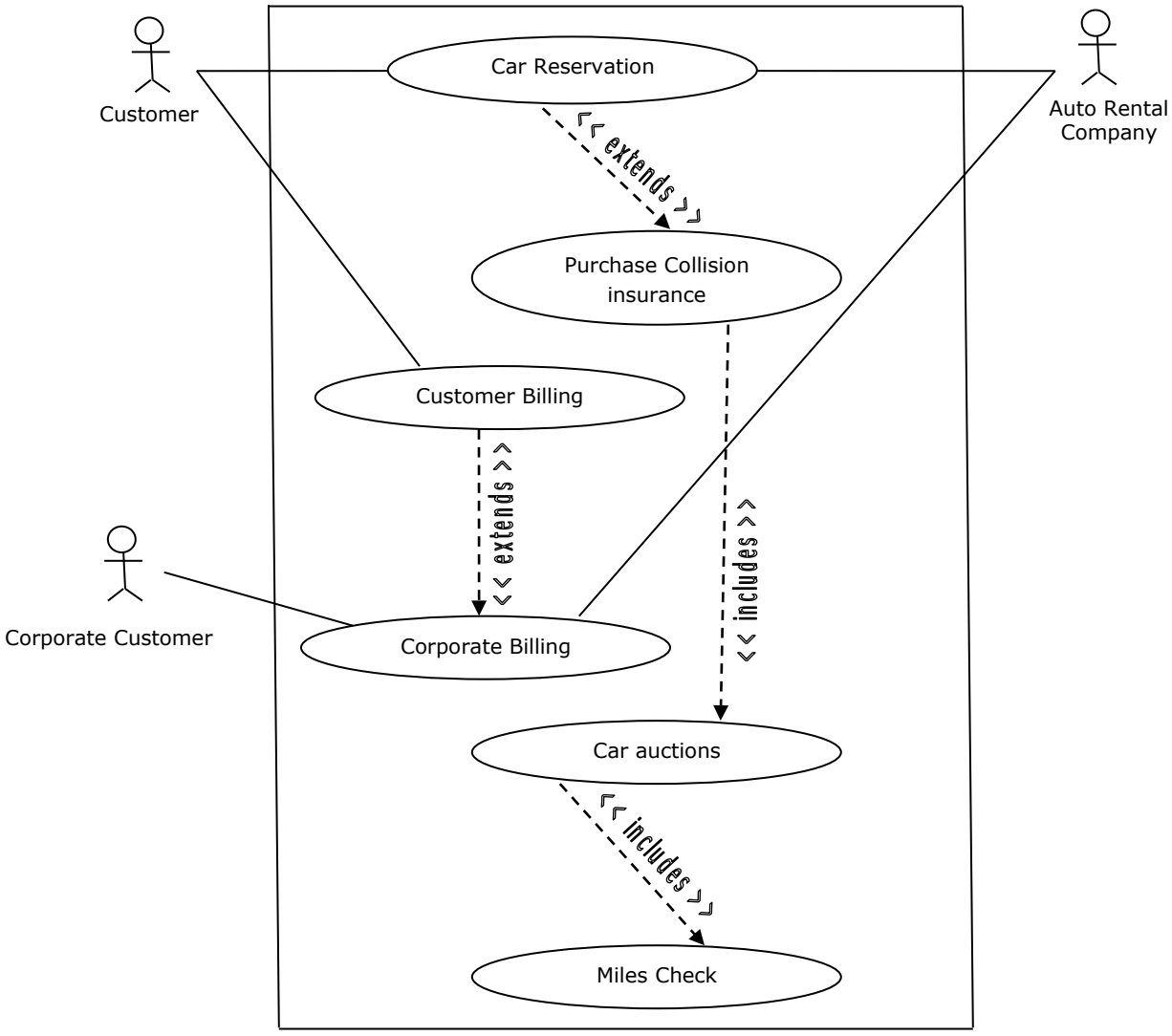


Figure: Use Case diagram: A diagram that depicts the Use Cases and action for a system

b i i z a y @ g m a i l . c o m

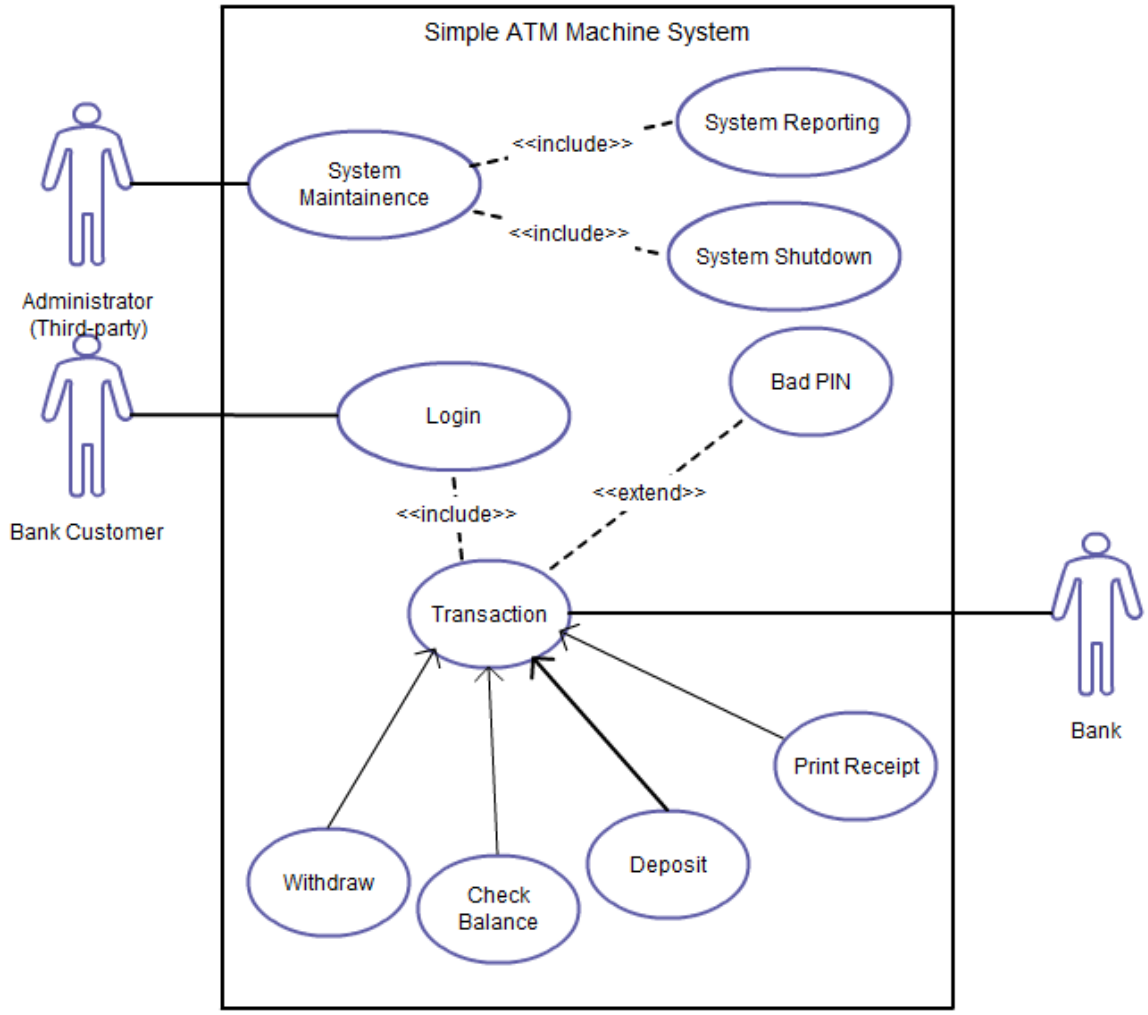


Figure: Use case diagram of ATM Machine System

Object Modeling: Class Diagrams

Object

An object is an entity that has a well-defined role in the application domain, and has state, behaviors and identity. An object is a concept, abstraction or thing that makes sense in a application context. An object could be a tangible visible entity (e.g. person, place or thing) it could be a concept or event (e.g. department, performance, marriage) or it could be an artifact of the design process (e.g. user interface, controller, scheduler etc.). An object has a state and exhibits behavior through operations that can examine or affect its state.

State/Information

The state of an object encompasses its properties (attributes and relationships) and the values those properties have. An object state is determined by its unique identity, attribute values, links to other objects and the state representing its current conditions.

b i i z a y @ g m a i l . c o m

Behavior

The behavior of an object represents how an object acts and reacts. An object's behavior depends on its state and the operation being performed.

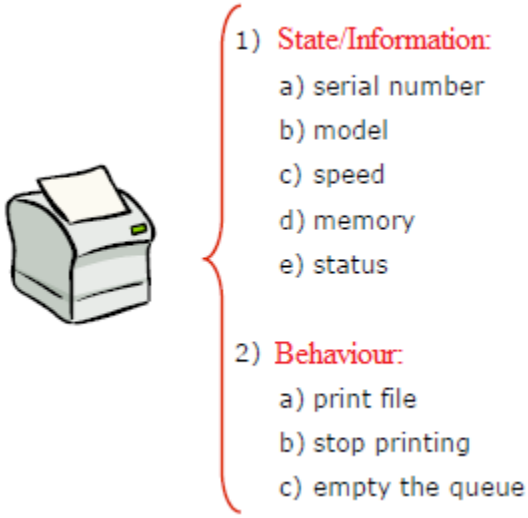


Figure: Object Definition Example

Class

A Class is a set of object that share common structure and a common behavior. Also called as object class. Examples: agency, citizen, car, etc. Objects are created using class definitions as templates.

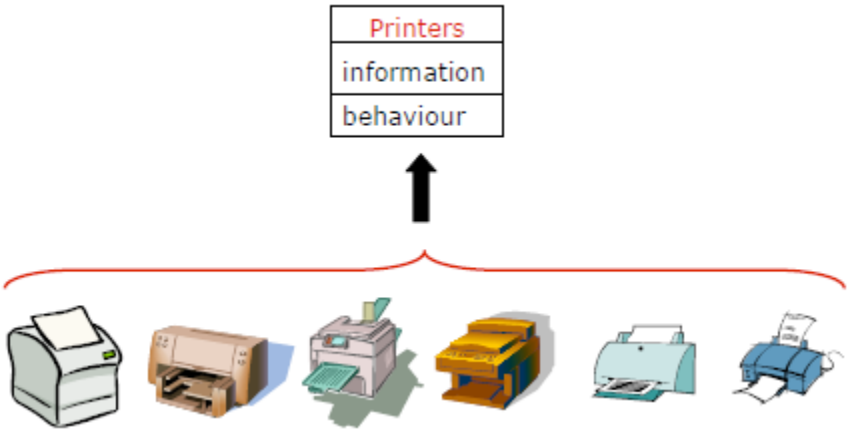


Figure: Class Example

Class Diagram

A Class diagram is used to model the static design view of the system. A class diagram shows the static aspect of a system: *classes, their internal structure* and the *relationship* in which they participate.

b i i z a y @ g m a i l . c o m

In UML, a class is represented by a rectangle with three compartments separated by horizontal lines. The class name appears in the top compartment, the list of attributes in the middle compartment, and the list of operations in the bottom compartment of the box.

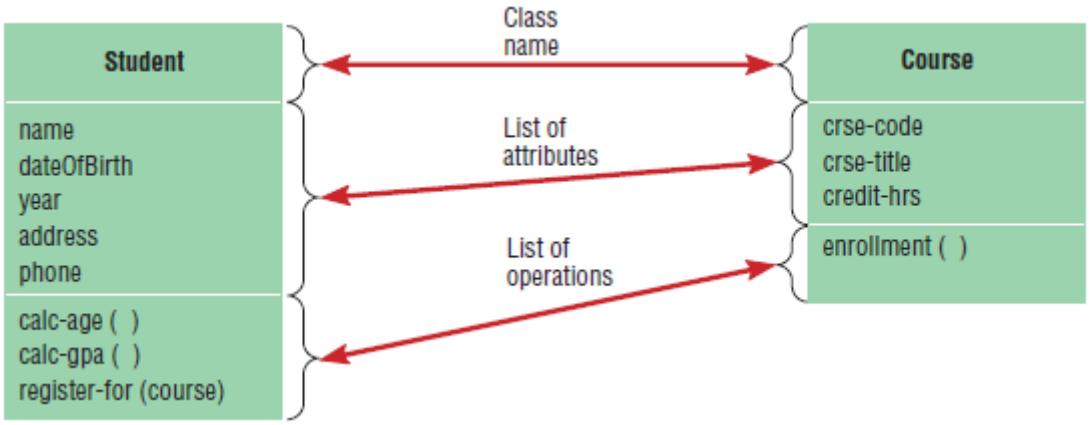


Figure: Class diagram showing two classes

Object Diagram

An object diagram shows a set of objects and their relationships. It is also known as *instance diagram*. It is a graph of instances that are compatible with a given class diagram. It shows the static snapshots of element instances found in class diagrams.

In an object diagram, an object is represented as a rectangle with two compartments. The names of the object and its class are underlined and shown on top compartment using the following syntax:

ObjectName: ClassName

Figure: Object diagram with two instances.

Attribute

An attribute is a named property of a class that describes a range of values that instances of the property may hold. Attribute represents some property of the thing you are modeling that is all object of that class. An attribute has a type and defines the type of its instances. Only the object is able to change the values of its own attributes. The set of attribute values defines the state of the object.

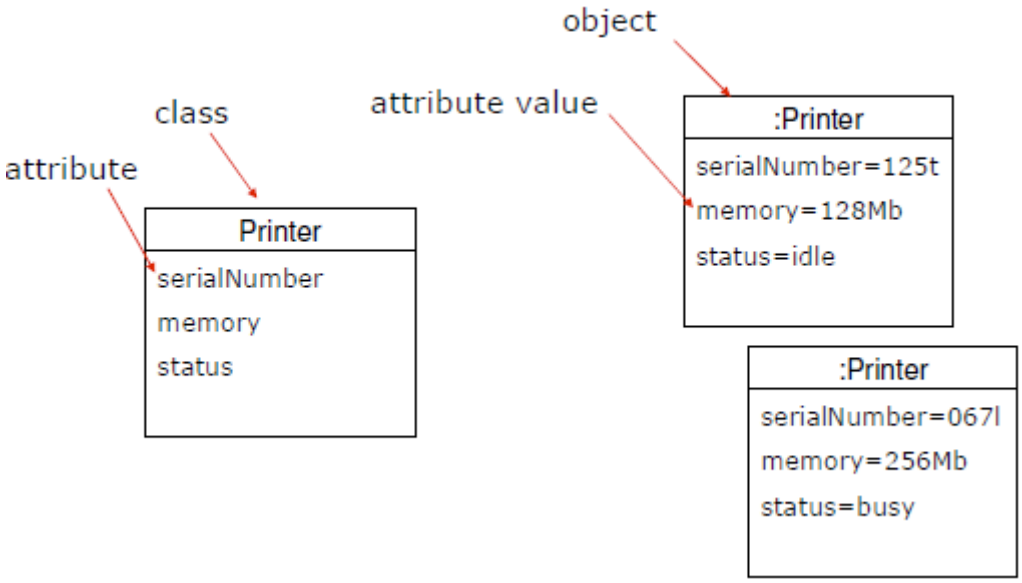


Figure: Attribute Examples

Operation/Methods/Message

An operation is a function or service that is provided by all the instances of a class. An operation is simply an action that one object performs upon another in order to get a response. It is only through such operations that other objects can access or manipulate the information stored in an object. The operations thus provide an external interface to a class without showing its internal structure or how its operations are implemented.

An operation could be:

1. A question - does not change the values of the attributes
2. A command - may change the values of the attributes

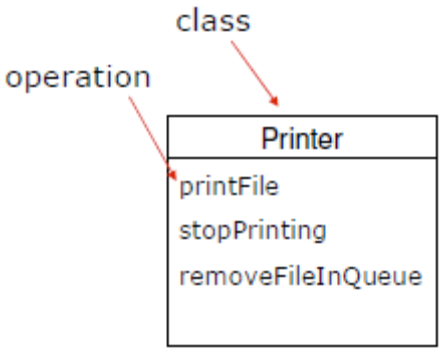


Figure: Operation Example

Encapsulation

The technique of hiding the internal implementation details of an object from its external view is known as encapsulation. It separates implementation from users/clients. A client depends on the interface.

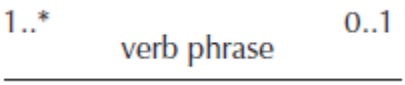
Figure: Encapsulation Example.

Associations

Association is a relationship between object classes. It is a structural relationship that objects of one thing are connected to the objects of another. It is analogous to relationship in the ER Model.

An association is represented as a solid line between participating classes, possibly directed, labelled and with adornments (multiplicity and role names).

Example:



The degree of an association may be one (Unary), two (binary) or higher (n-ary).

Association Role

- ✓ The end of an association where it connects to a class
- ✓ Each role has multiplicity, which indicates how many objects participate in a given association relationship

Figure: Unary Associations

Multiplicity

Multiplicity indicates how many objects participate in a given relationship.

b i i z a y @ g m a i l . c o m

Exactly one	1
Zero or more	0..*
One or more	1..*
Zero or one	0..1
Specified range	2..4
Multiple, disjoint ranges	1..3, 5

Department	1	Boss	A department has one and only one boss.
Employee	0..*	Child	An employee has zero to many children.
Boss	1..*	Employee	A boss is responsible for one or more employees.
Employee	0..1	Spouse	An employee can be married to zero or no spouse.
Employee	2..4	Vacation	An employee can take between two to four vacations each year.
Employee	1..3, 5	Committee	An employee is a member of one to three or five committees.

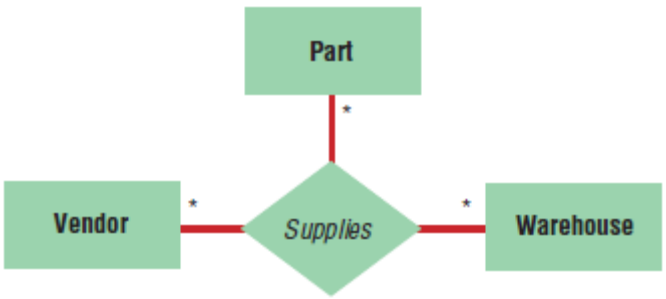


Figure: Ternary Associations

b i i z a y @ g m a i l . c o m

Figure: Binary Associations

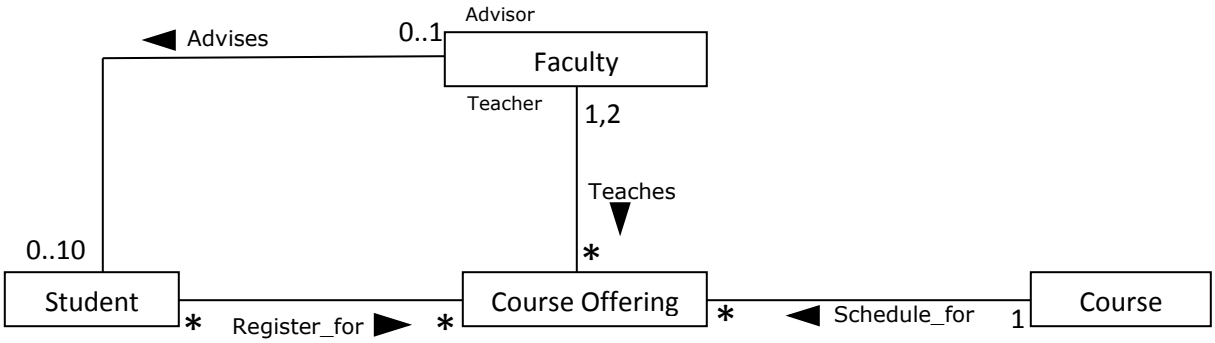


Figure: n-ary Associations

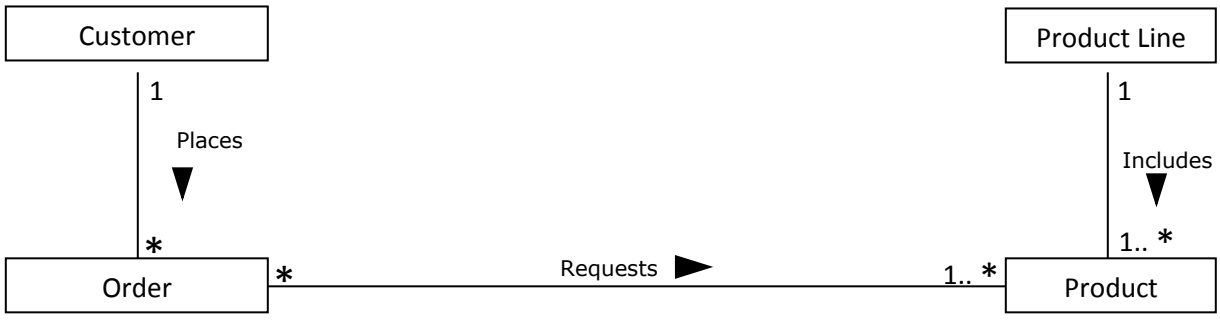


Figure: Class diagram for "Customer Order"

b i i z a y @ g m a i l . c o m

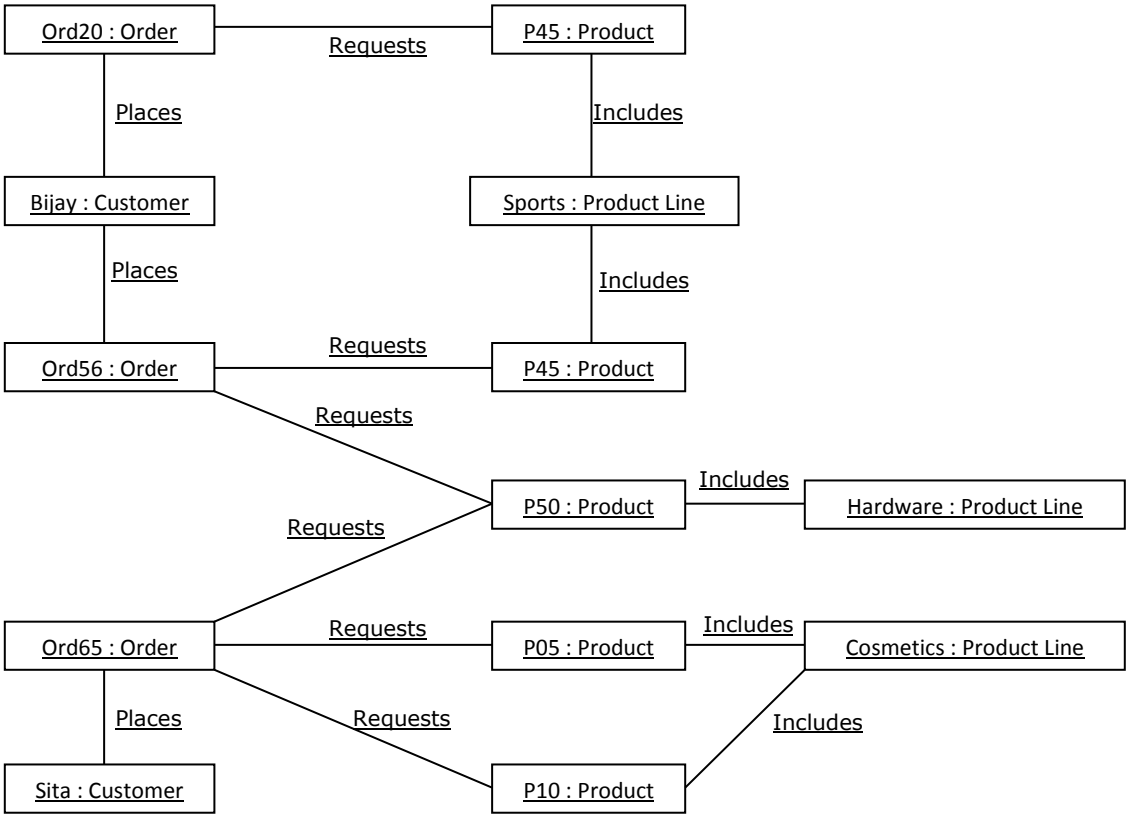


Figure: Object diagram for "Customer Order"

Association Classes

Association class is an association that has attributes or operations of its own, or that participates in relationships with other classes. It is similar to an associative entity in ER modeling. In an association between two classes, the association itself might have some properties. In such cases, the association is represented as a separate association class. An association which has attributes or operations of its own participates in relationships with other classes.

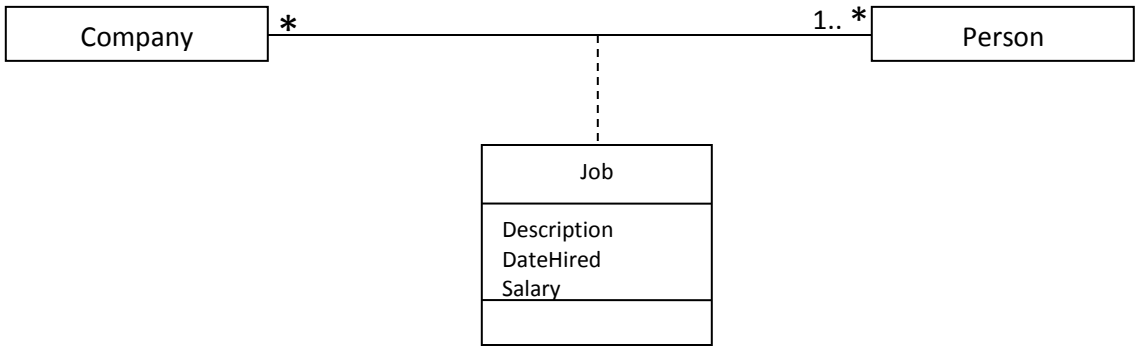


Figure: Class diagram showing an association class "Job"

b i i z a y @ g m a i l . c o m

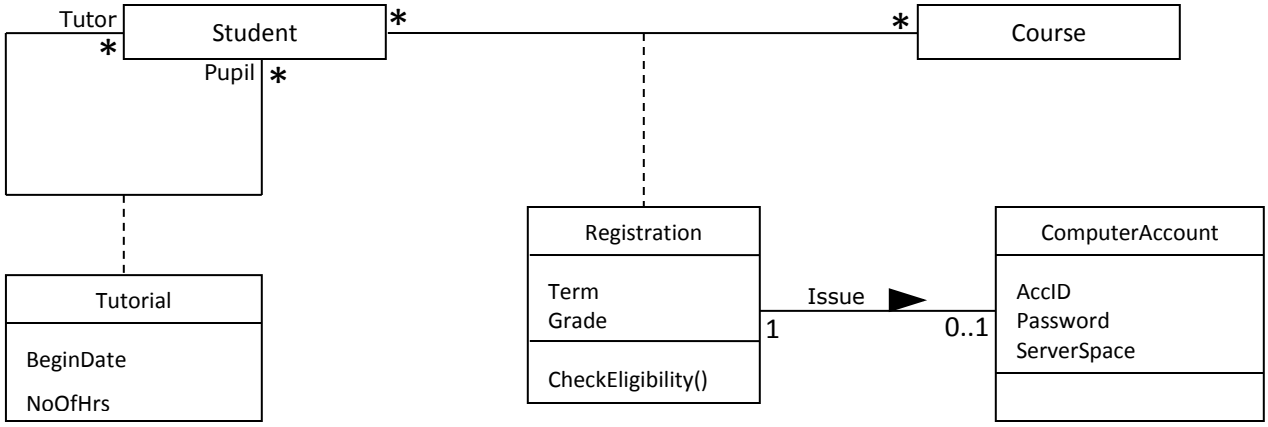


Figure: Class diagram showing two association classes

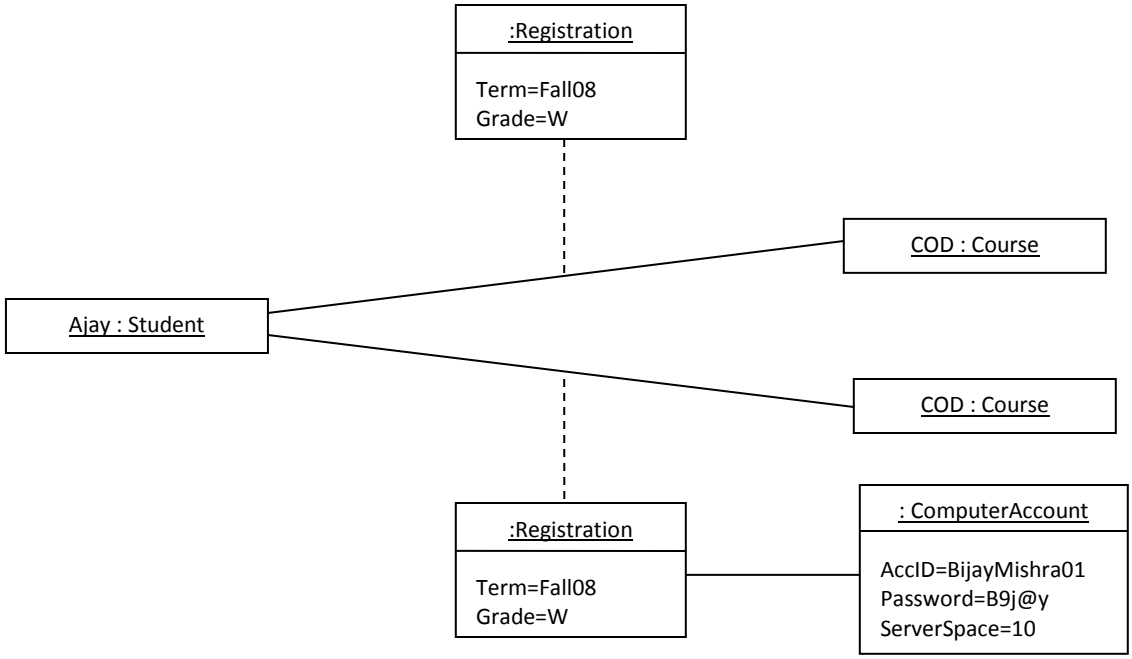
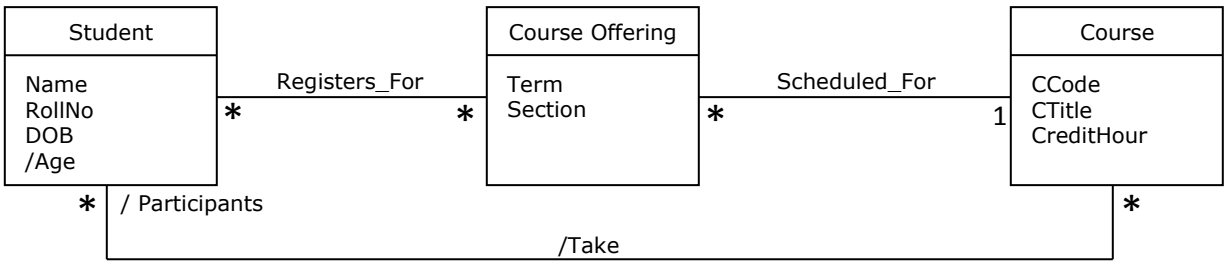


Figure: Object diagram for above class diagram

Representing derived attributes, derived associations and derived roles

Derived attributes, derived associations and derived in roles are labeled in a class diagram with a prefix /. Derived means one that can be computed from other attribute, association, roles.

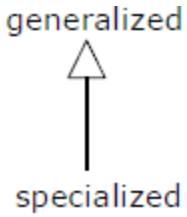


b i i z a y @ g m a i l . c o m

Generalization

Generalization is a relationship between general thing (superclass/parent) and a more specific kind of thing (subclass/child). It is equivalent to "kind-of" or "type-of" relationship. Generalization enables inheritance. It is not an association. Generalization implies that an instance of a subclass is also an instance of its super class.

Generalization a relationship in which objects of a specialized element (child) are substitutable for objects of a generalized element (parent). The child elements share the structure/behavior of the parent. Generalization is rendered graphically as a solid line with hollow arrowhead pointing to the parent.



Specialization is the process of creating more specialized subclasses from an existing class

Superclass

Super-Class is a class that contains the features common to two or more classes. It is a class that is composed of several generalized subclasses.

Subclass

Sub-Class is a class that contains at least the features of its superclass. It is a class that has been generalized.

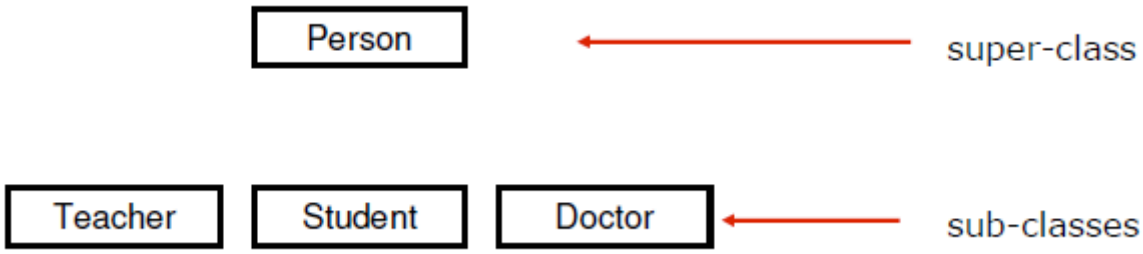


Figure: Super/Sub-Class Example

Abstract Class

Abstract class is a class that has no direct instances, but whose descendants may have direct instances. It is a class that lacks a complete implementation – provides operations without implementing some methods. It cannot be used to create objects.

Concrete Class

Concrete class is a class that can have direct instances. It has methods for all operations.

Discriminator

Discriminator is an attribute that defines sub-classes. It shows which property of an object class is being abstracted by a generalization relationship.

Inheritance

Inheritance is an implementation concept. A property that a subclass inherits the features from its superclass is called inheritance.

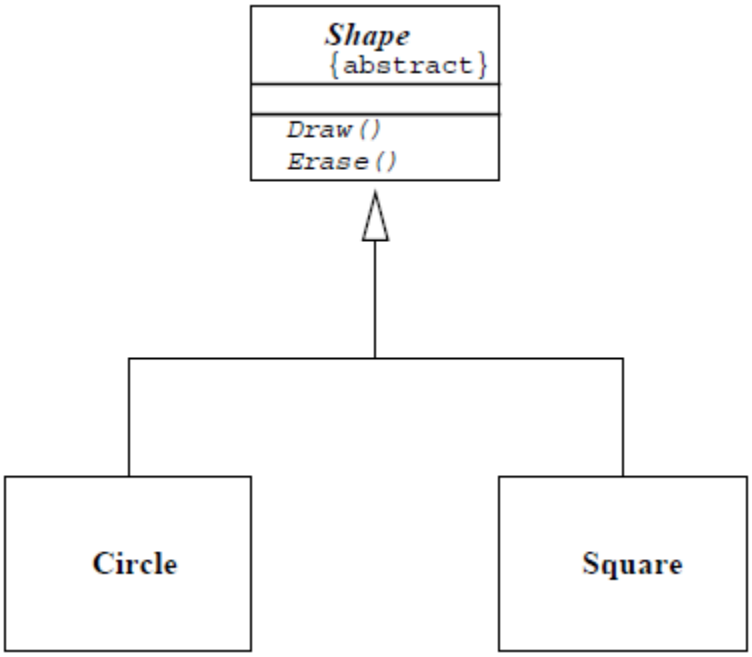


Figure: Inheritance Example

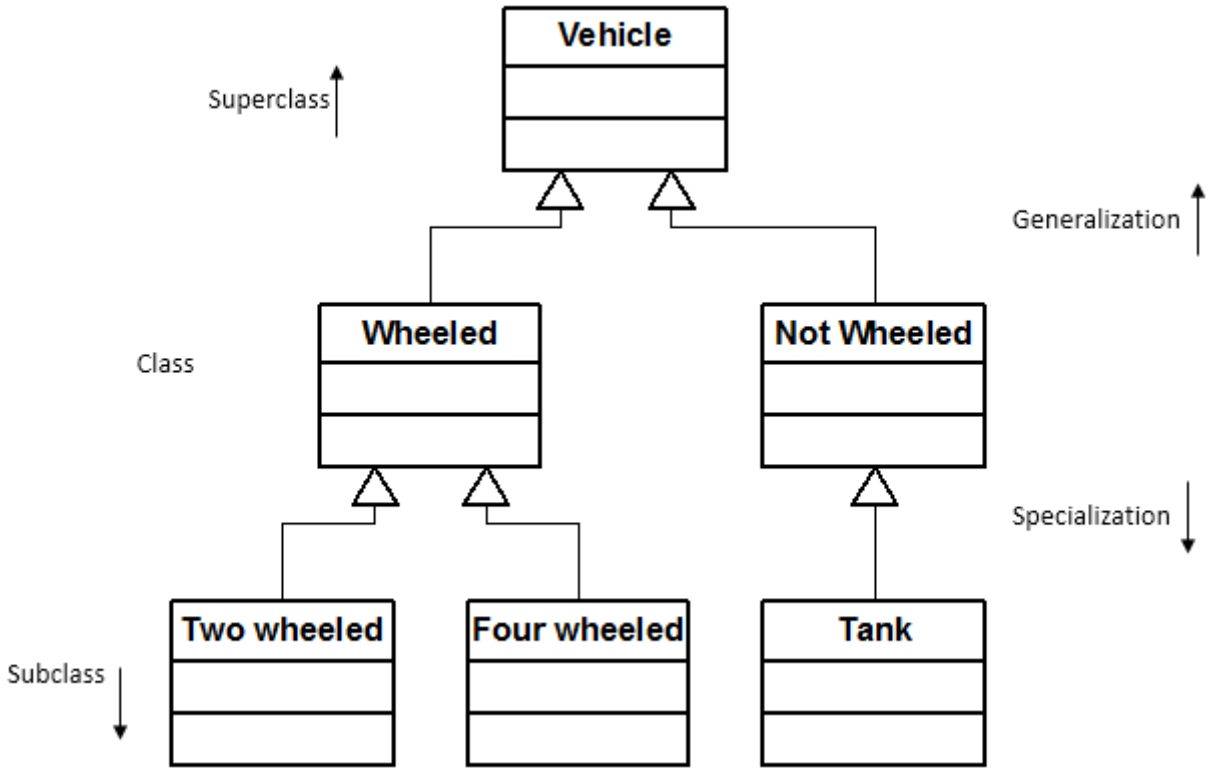


Figure: Example of Inheritance

b i i z a y @ g m a i l . c o m

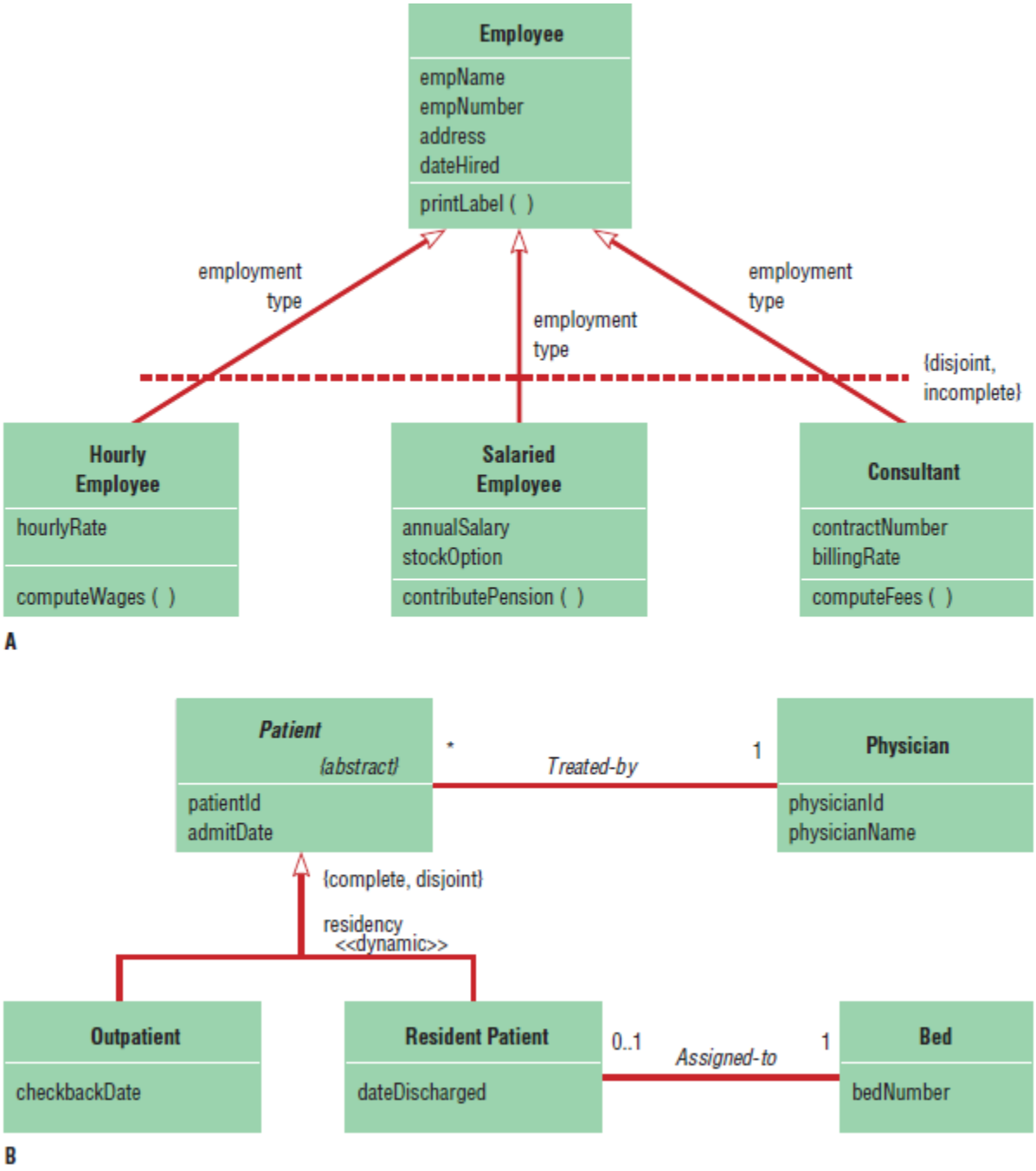


Figure: Examples of generalization, inheritance, and constraints: (A) Employee superclass with three subclasses, (B) Abstract patient class with two concrete subclasses.

b i i z a y @ g m a i l . c o m

Overriding Inheritance

Overriding is the process of replacing a method/operation inherited from a super class by a more specific implementation of that method in a sub class.

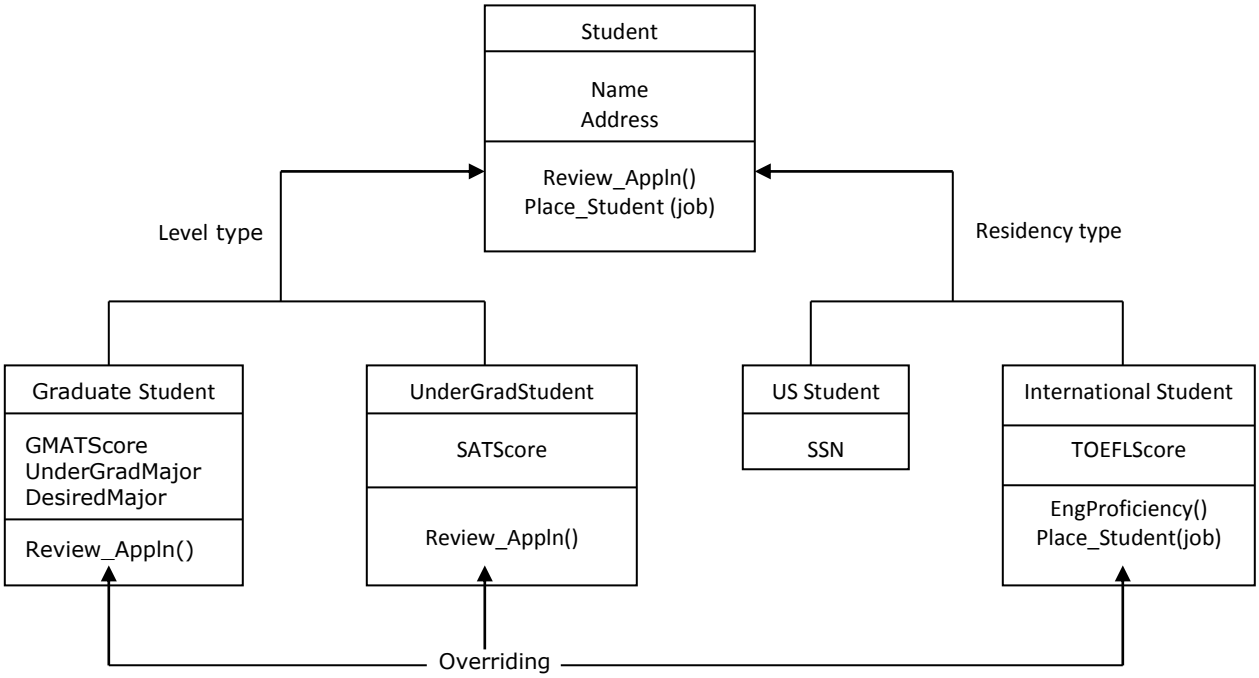


Figure: Overriding Inheritance

Multiple Inheritance

In multiple inheritance a class inherits from more than one super class; an object may be instance of more than one class.

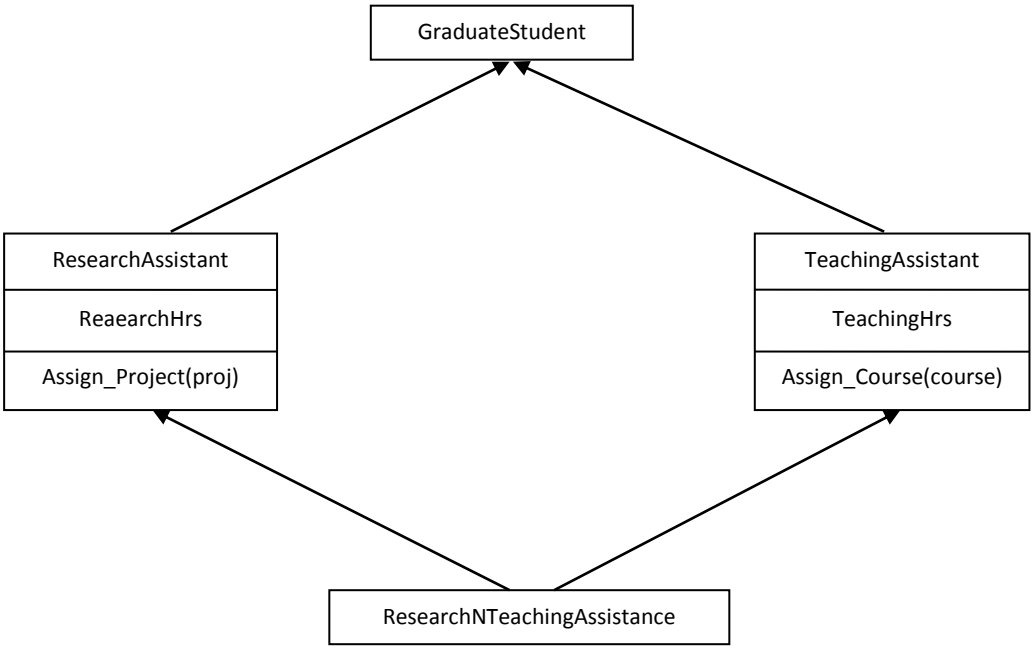


Figure: Multiple Inheritance

b i i z a y @ g m a i l . c o m

Polymorphism

Polymorphism is an ability for two or more objects to respond to the same request, each in its own way. It can dynamically choose the method for an operation at run-time or service-time. In other words, it is many ways of doing the same thing! The same operation may apply to two or more classes in different ways. Operations could be defined and implemented in the superclass, but re-implemented methods are in unique sub-classes.

Example:

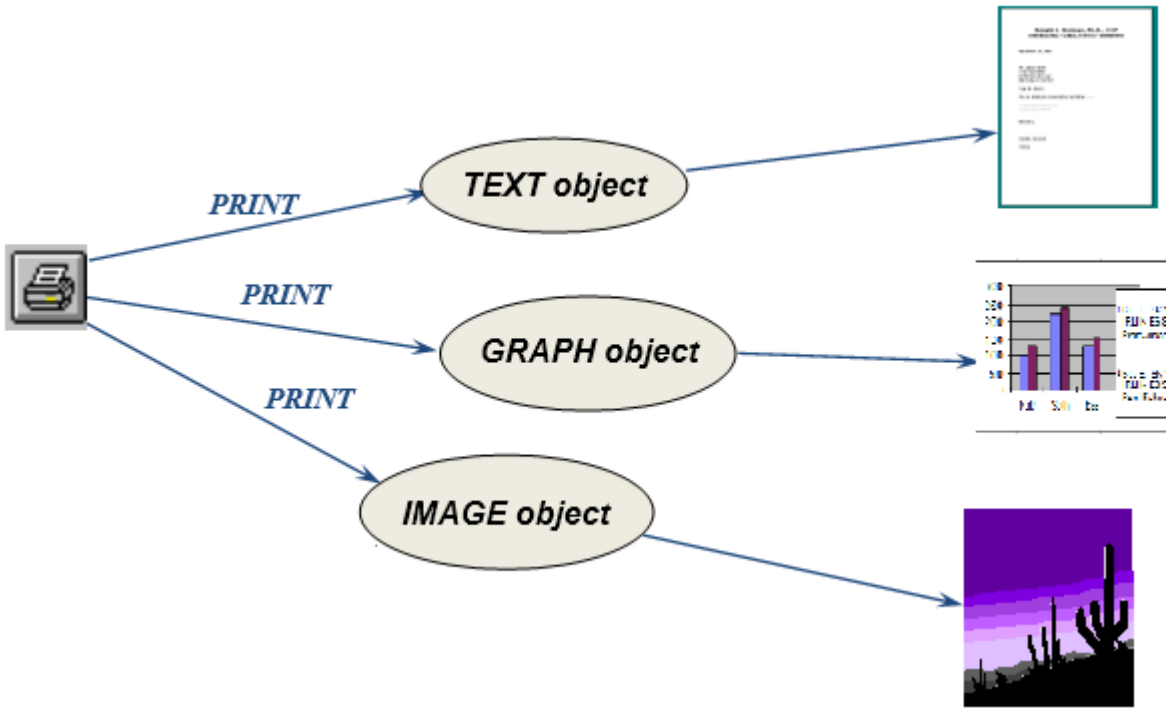
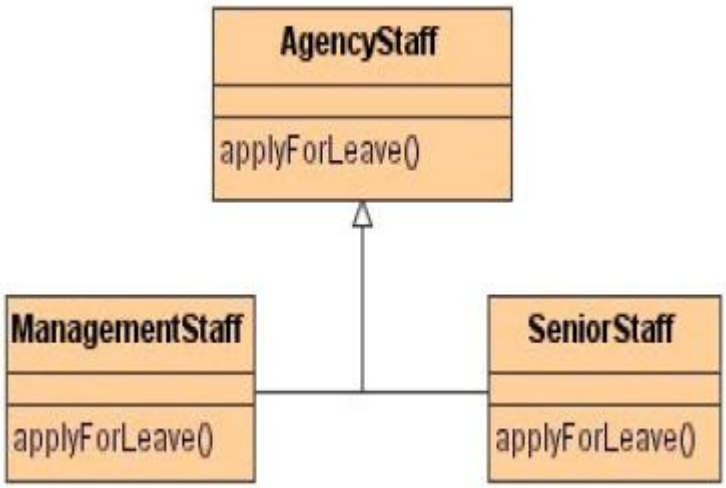


Figure: Polymorphism Example

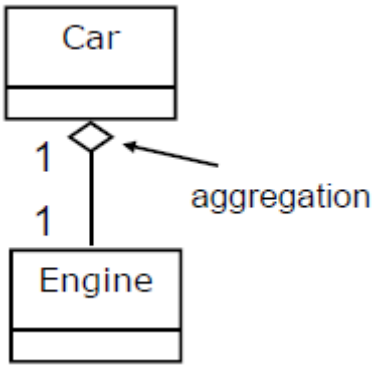


Example: Management-staff and agency-staff can apply for leave, but possibly in different ways.

b i i z a y @ g m a i l . c o m

Aggregation

Aggregation is "a part-of" relationship between a component object (part) and an aggregate object (whole). In aggregation, objects are assembled to create a more complex object. It defines a single point of control for participating objects. The aggregate object coordinates its parts. It is represented by a hollow diamond.



Example: Aggregation Example

Example: Personal computer is composed of CPU, Monitor, Keyboard, Hard Disk, etc. This means that, a CPU is a part of a personal computer. Similarly, Monitor is a part of a personal computer.

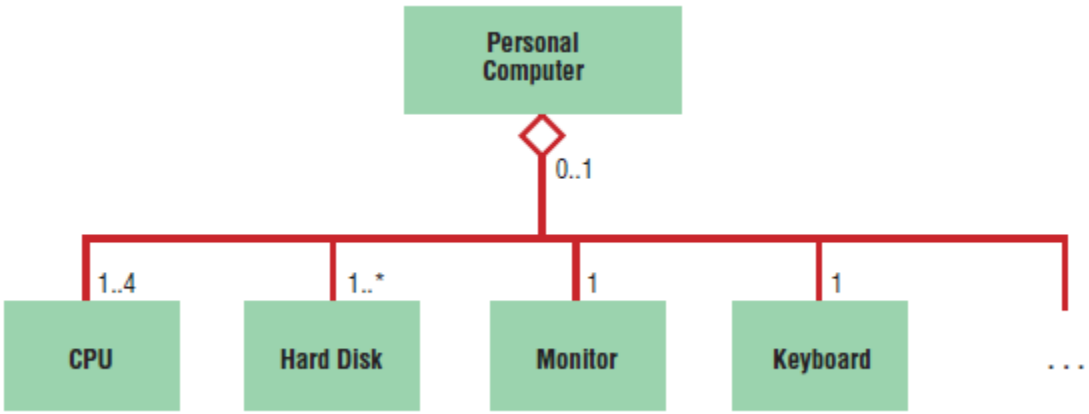


Figure: Example of aggregation.

Composition

Composition represents a stronger form of aggregation. In composition, a part belongs to only one whole object. In other word, composition is "is entirely made of" relation. In composition, parts cannot exist on their own. It is represented by a solid filled diamond.

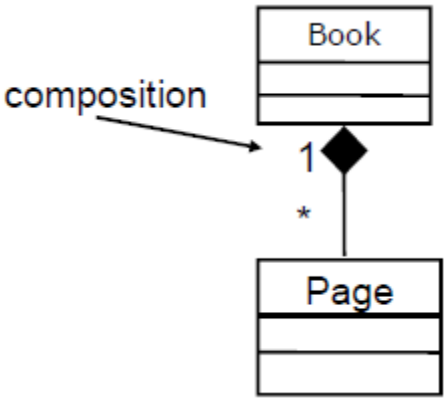


Figure: Composition Example

b i i z a y @ g m a i l . c o m

Example:

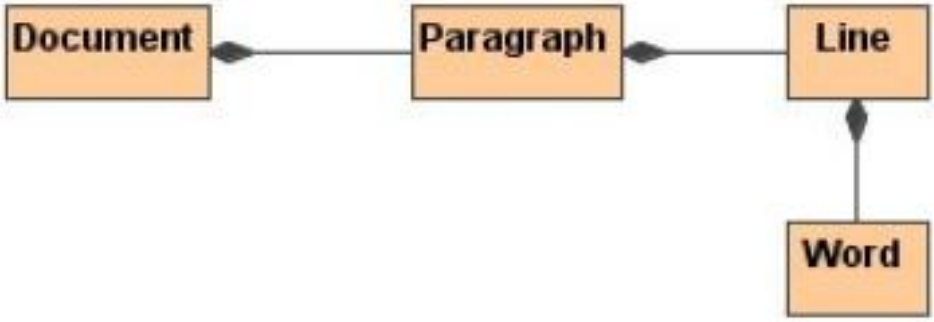


Figure: Composition Example

A word cannot exist if it is not part of a line. If a paragraph is removed, all lines of the paragraph are removed, and all words belonging to that lines are removed.

Example:

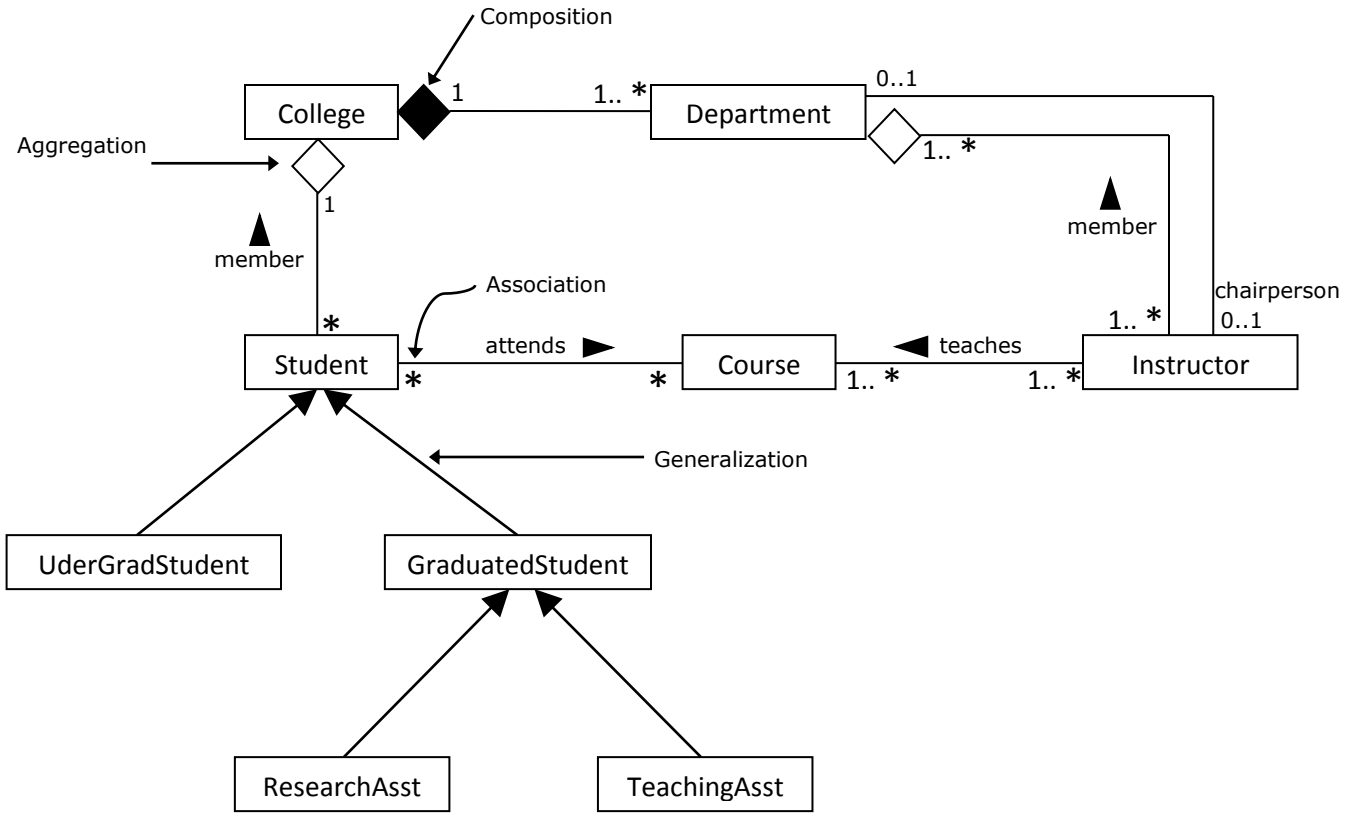


Figure**:** Class Diagram for a College System.

A department is a part of only one college. When whole collapses and if the part also collapses then it is referred to as composition (Example: College and Department). If whole collapses and the part still exist, then it is referred to as aggregation (Example: College and Student).

b i i z a y @ g m a i l . c o m

Example:

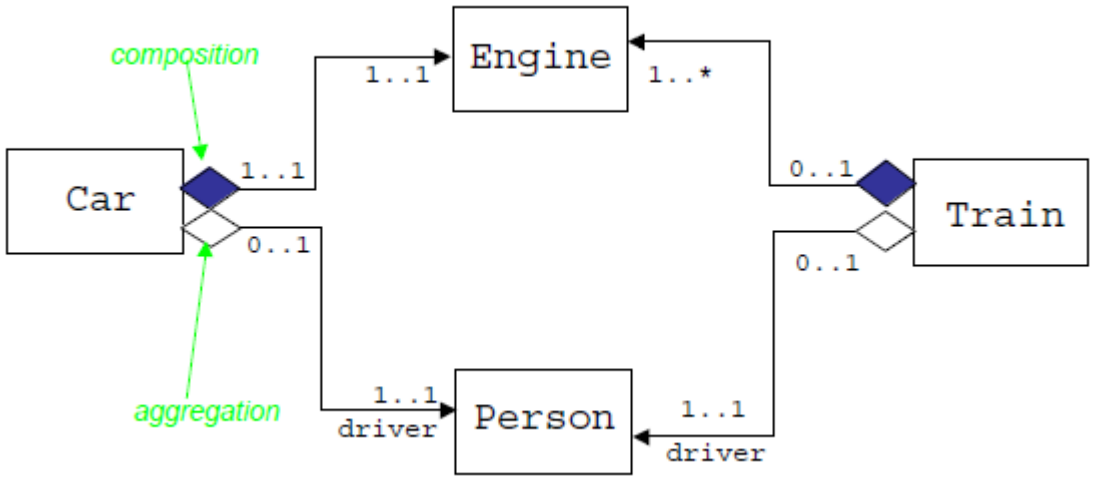


Figure: Composition and Aggregation Example

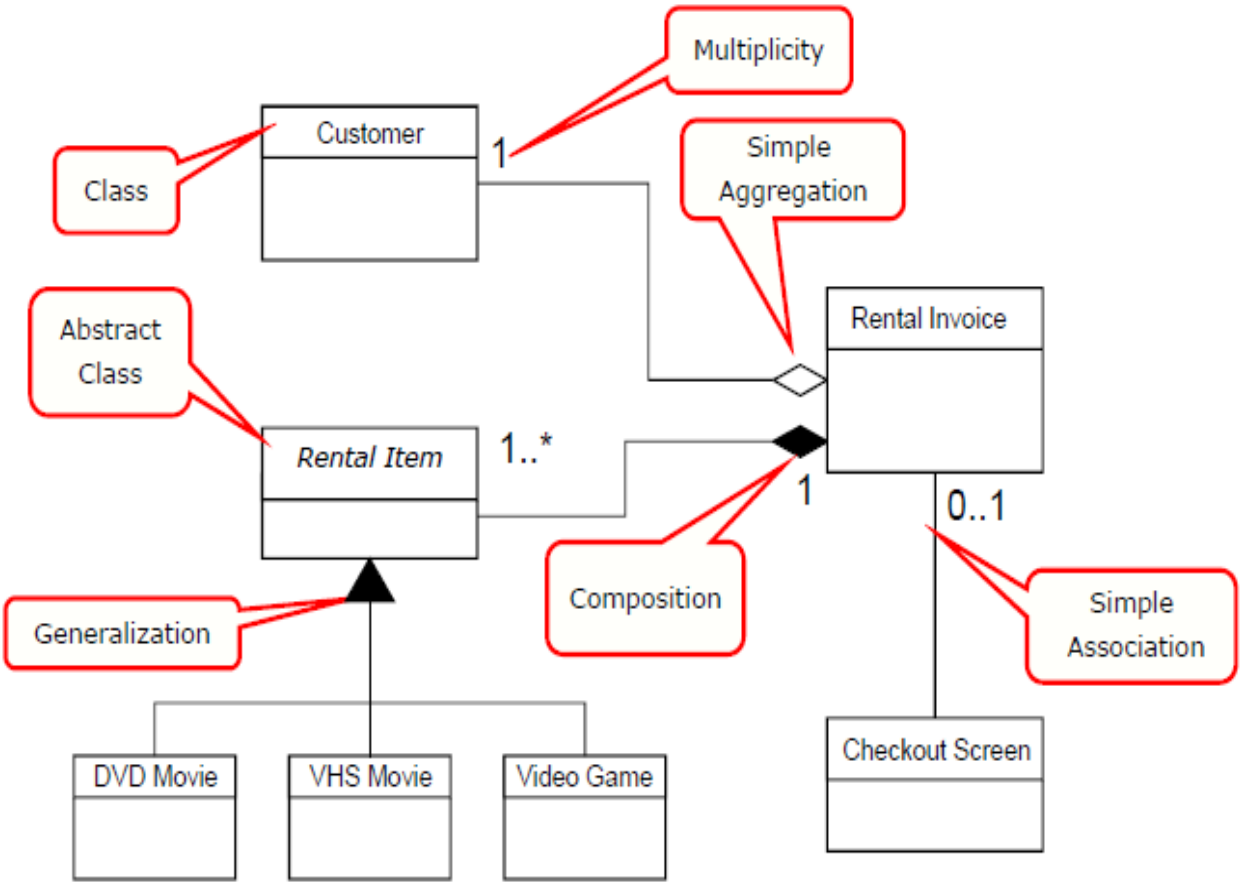


Figure: An Example of Class Diagram for "Video Rental System"

b i i z a y @ g m a i l . c o m

Advantages of Class Diagram:

- It discovers related data and attributes
- It gets a quick picture of the important entities in a system
- It spots dependencies between one class/object and another
- It forces the programmer to think out the structure of his/her classes and how they will interact with each other before actually writing any code. This may lead to a more robust application.
- It provides a blueprint for maintenance programmers to get an overview of how the application is structured before examining the actual code. This may reduce maintenance time.

Disadvantages of Class Diagram:

- It discovers algorithmic (not data-driven) behavior
- It focuses on a static view of the system.
- The programmer may need to learn UML to build the class diagram in the first place.
- The time spent building the class diagram may add to overall development time.
- If the class diagram is overcomplicated, then it may be difficult to correlate with the actual code.

Dynamic Modeling: State Diagrams

A state diagram depicts the various state transitions or changes an object can experience during its lifetime along with events that causes those transitions. These diagrams are used to model the dynamic aspects of a system from the perspective of state transitions.

A state is a condition or situation in the life of an object during which it satisfies some condition, performs some activity, or waits for some event. The state changes when the object receives some even: the object is said to undergo a state transition. The state of an object depends on its attribute values and links to other objects.

An event is something that takes place at certain point in time. Events trigger state transitions. (e.g. - A customer places an order; a student registers for a class etc.). In determining the state of an object, only those attributes that affect its gross behavior are considered to be important. State diagram shows how the object transitions from an initial state to other states. When certain events occur or when certain conditions are satisfied.

State

- A condition during the life of an object during which it satisfies some conditions, performs some actions or waits for some events
- Shown as a rectangle with rounded corners

Figure: A State

b i i z a y @ g m a i l . c o m

Figure: Initial State

Figure: Final State

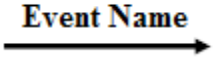
State Transition

- The changes in the attribute of an object or in the links an object has with other objects
- Shown as a solid arrow
- Diagrammed with a guard condition and action

Figure: State Transition

Event

- Something that takes place at a certain point in time



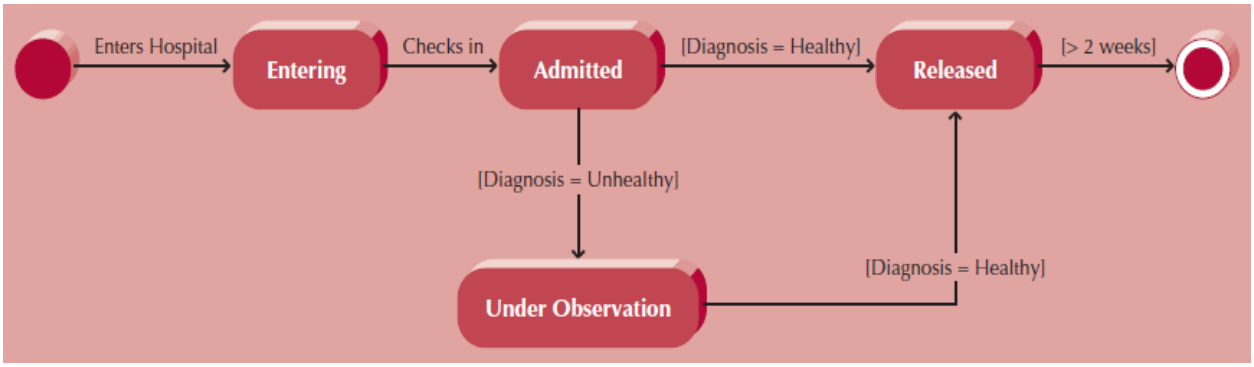


Figure: State Diagram for a Hospital Patient

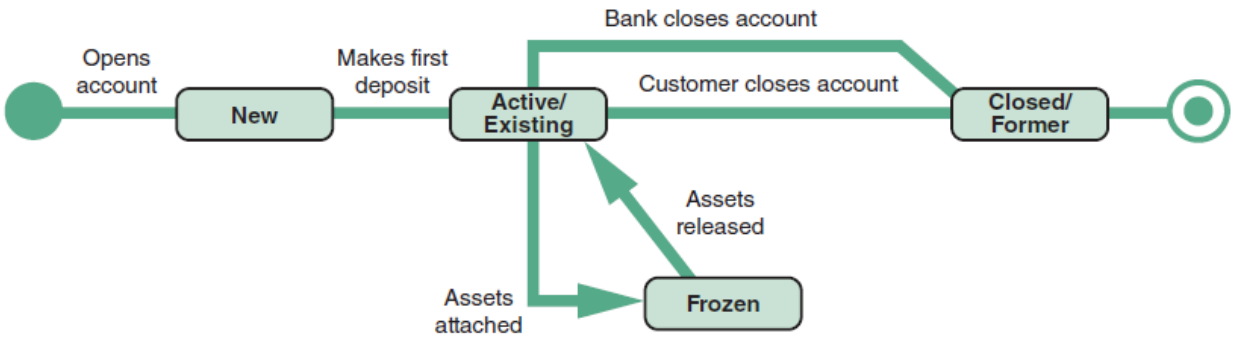


Figure: An example of a state transition diagram for a bank account.

b i i z a y @ g m a i l . c o m

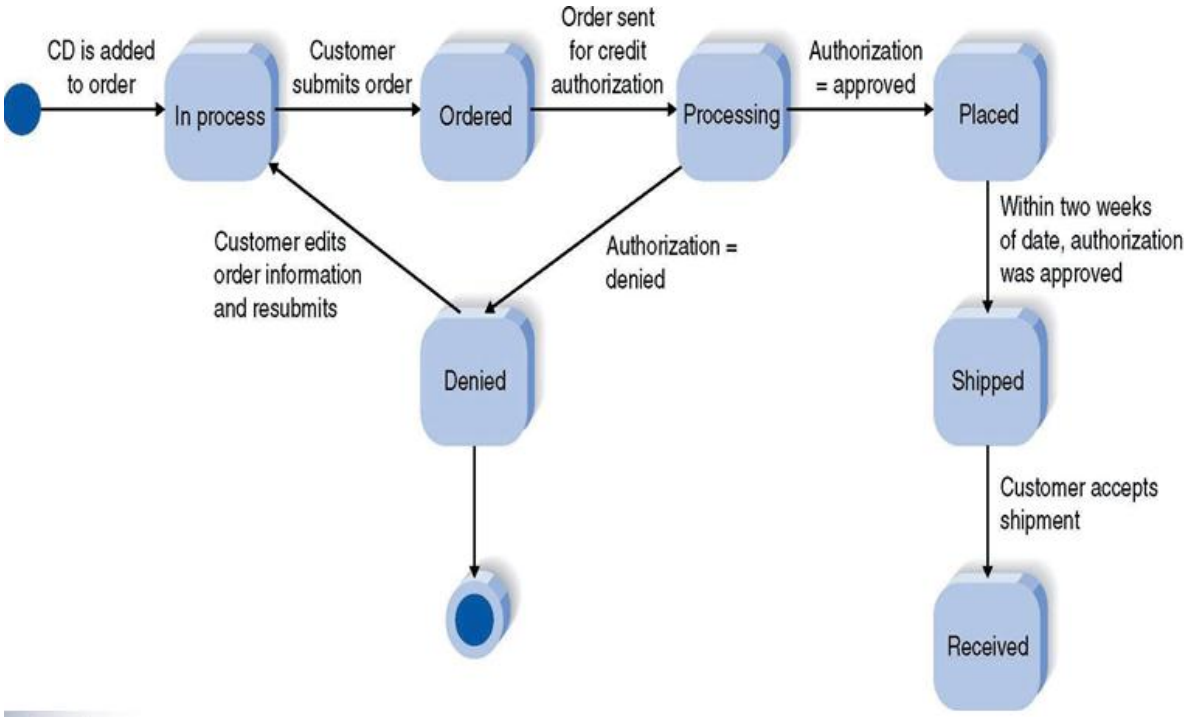


Figure: State diagram for ordering CDs

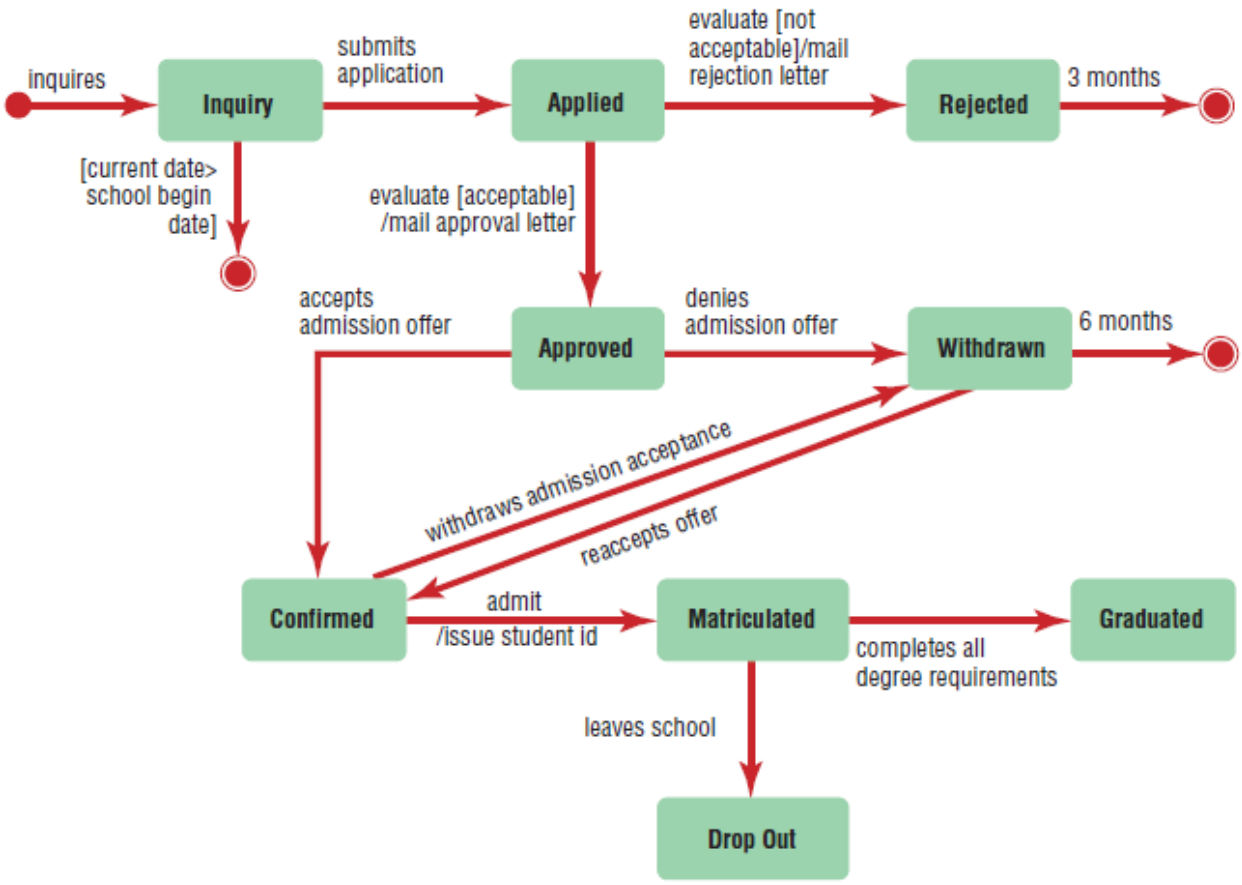


Figure: State diagram for the Student object.

b i i z a y @ g m a i l . c o m

A transition is labeled with a guard condition and/or an action, separated with a forward slash / State diagram: a model of the states of a *single object* and the events that cause the object to change from one state to another

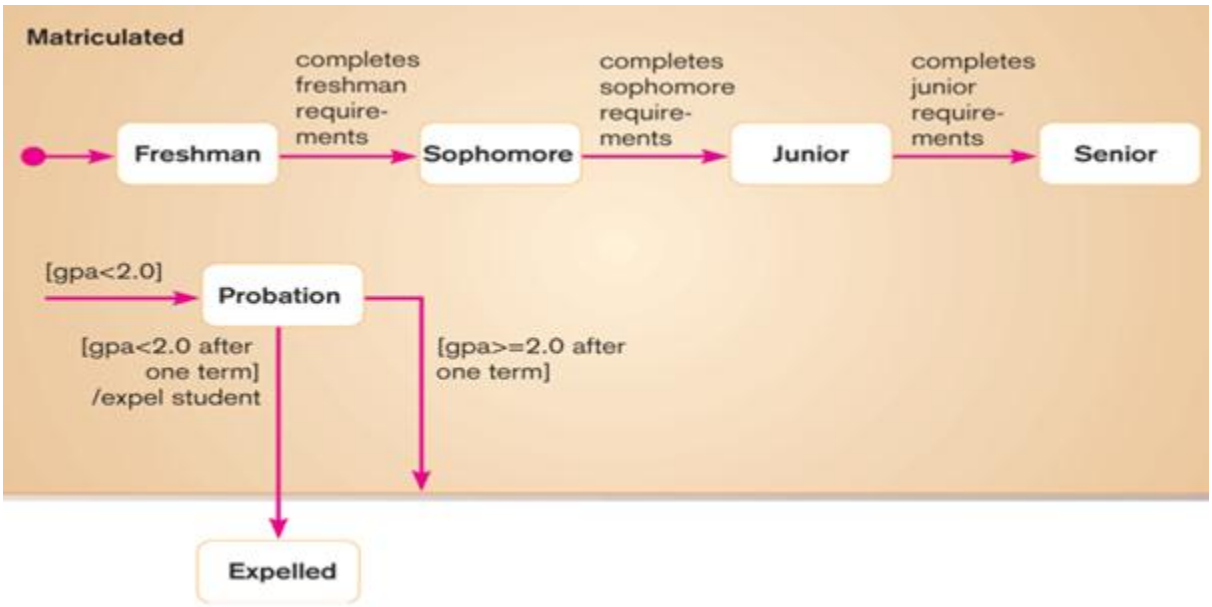


Figure: A nested state diagram for Matriculated state of Student object

A state can be expanded into substates using nested state diagrams, similar to expansion of processes in different levels of DFDs.

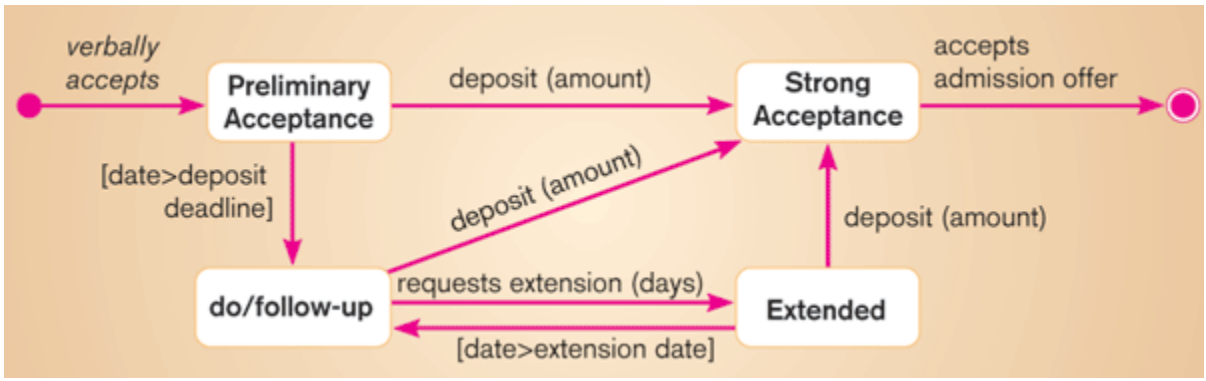


Figure: Nested state diagram for the accepts admission offer event

An event can be expanded into using nested state diagrams, and may involve substates and subtransitions and events.

Advantages of State Diagram:

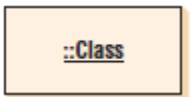
- It conveys or summarizes complex concepts or mechanisms more easily than other techniques.
- It maps the official "states" or statuses that an entity can have, which is to say from initial creation to final disposal.
- It specifies a sequence of states or statuses that an entity goes through during its lifetime.

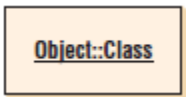
Dynamic Modeling: Sequence Diagramming

A sequence diagram depicts the interactions among objects during a certain period of time. Each sequence diagram shows only the interactions pertinent to a specific use case. This diagram shows a participating objects and the interaction among those objects – arranged in time sequence – by the message the exchange with one another. In a sequence diagram, vertical axis represents time and horizontal axis represents the various participating objects. Time increases as we go down the vertical axis.

Elements of a Sequence Diagram:

Class/Objects: A class is identified by a rectangle with the name inside. Classes that send or receive messages are shown at the top of the sequence diagram.



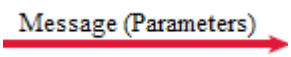


Lifeline: The lifeline represents the time during which the object above it is able to interact with the other objects in the use case. Each object is shown as a vertical dashed line called *lifeline*; the lifeline represents the object's existence over a certain period of time.

An X marks the end of the lifeline.

Activation/Focus: An activation/focus is identified by a narrow vertical shape that covers the lifeline. The focus indicates when an object sends or receives a message. A thin rectangle superimposed on the lifeline of an object, represents an activation of the object; activation shows the period during which the object performs an operation.

Messages: Objects communicate with one another by passing messages. A message is identified by a line showing direction that runs between two objects. The label shows the name of the message and can include additional information about the contents.



b i i z a y @ g m a i l . c o m

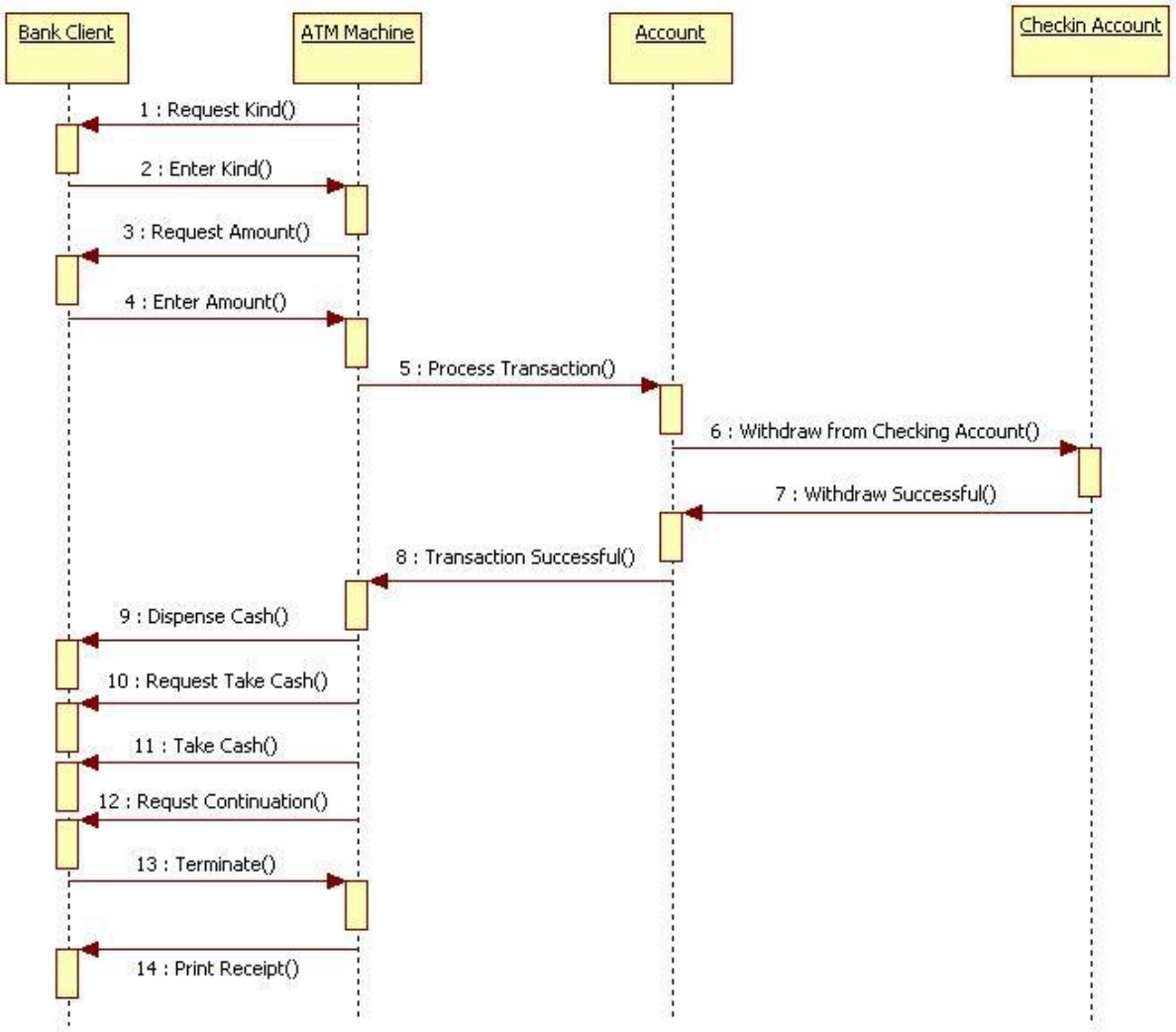


Figure: An Example of Sequence Diagram for ATM

A sequence diagram is constructed by referring to the description of use cases, which outlines the sequence of actions that are necessary to perform a specific task.

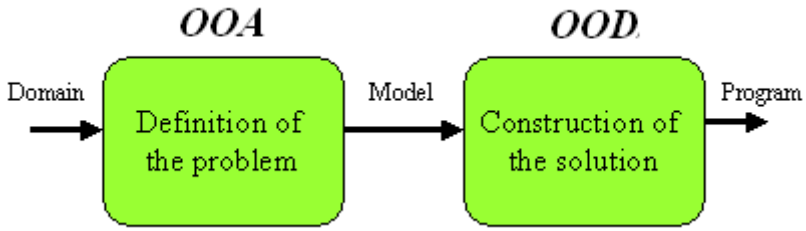
Advantages of Sequence Diagram:

- It helps you discover architectural, interface and logic problems early.
- They are valuable collaboration tools during design meetings because they allow you to discuss the design in concrete terms.
- They can be used to document the dynamic view of the system design at various levels of abstraction.

Analysis versus Design

- Object Oriented Analysis (OOA) is done with no concern of how it will be implemented while Object Oriented Design (OOD) considers implementation issues
- OOA emphasizes an investigation of the problem and requirements, rather than a solution.

- Instead of using functional decomposition of the system, the OOA approach focuses on identifying objects and their activities
- OOD emphasizes a conceptual solution (in software and hardware) that fulfills the requirements, rather than its implementation.
- Useful analysis and design have been summarized in the phrase do the right thing (analysis), and does the thing right (design).
- Analysis is what it means, working out the problem. Design is what the word means - designing the system.
- OOA aims to model the problem domain, the problem to be solved, by developing an OO system while OOD is an activity of looking for logical solutions to solve a problem by using encapsulated entities called objects.
- OOA focuses on what the system does, OOD on how the system does it



Object-oriented analysis (OOA) looks at the problem domain, with the aim of producing a conceptual model of the information that exists in the area being analyzed. Analysis models do not consider any implementation constraints that might exist, such as concurrency, distribution, persistence, or how the system is to be built. Implementation constraints are dealt during object-oriented design (OOD). Analysis is done before the Design

The result of OOA is a description of what the system is functionally required to do, in the form of a conceptual model. That will typically be presented as a set of use cases, one or more UML class diagrams, and a number of interaction diagrams. It may also include some kind of user interface mock-up. The purpose of object oriented analysis is to develop a model that describes computer software as it works to satisfy a set of customer defined requirements.

Object-oriented design (OOD) transforms the conceptual model produced in object-oriented analysis to take account of the constraints imposed by the chosen architecture and any non-functional - technological or environmental - constraints, such as transaction throughput, response time, run-time platform, development environment, or programming language.

Some of the analysis models are:
a) use case modeling
b) class diagrams
c) object diagrams
d) interaction diagrams

Some of the design models are:
a) package diagrams
b) sequence diagrams
c) activity diagrams
d) state diagrams

OOAD vs Structured Approach

Structured Approach (Traditional SDLC)	Object-Oriented Approach
Logical steps of SDLC – Planning, Analysis, Design, Implementation, and Maintenance.	No logical steps of SDLC - repeating all phases during each iteration
Few models to deal with	Many models to deal with
Focuses mainly on the business aspects of a system and deals with other components such as user interface, network architecture, processing architecture separately	Starts with the business aspects of a system but deals with other components such as user interface, network architecture, processing architecture together as the analyst moves from the requirements models to design models
Top-down approach of understanding a problem through process models. Data definition is typically evolved from the process models	Bottom-up approach of understanding a problem through process definition and data definition
A small and consistent vocabulary to follow through the life cycle	A large and changing vocabulary during various phases of the life cycle
Process models are the focus of understanding user requirements	Use cases are the focus of understanding user requirements
Easy to extract design models from the analysis models	Design models depend on many analysis models
Easy to extract design information for systems components such as user-interface, application programs, and database or files	Very complex to extract design information for systems components such as user-interface, application programs, and database or files
Business logic of a process is described by the process description using structured English, decision tables, and decision trees	Business logic of a process is described by use case scenario, use case description, interaction diagram, and activity diagram
Does not dictate any model for development environment	Three-layer approach to systems design is closely analogous to development environment
Identification of input and output to the system is simple and they are extracted from the input/output data-flows in the data-flow diagram. Data structures associated with the data-flows are used to describe the data	Input and output information are scattered in many sequence diagrams as input/output messages. Separate classes describe the data
All data necessary for designing and developing a system is found in a single repository called data dictionary	There is no single repository for data; they are scattered with class definitions and use case descriptions

Development of programs through subroutines and functions are left to the programmers at the implementation phase	Development of programs is conceived at the requirements phase through defining classes, methods, and messages, and continues through the design phase
Detailed programming knowledge is not necessary for successful analysis and design	Detailed programming knowledge is necessary for successful analysis and design
Appropriate for developing documents and then start programming; hence programming can be outsource	UML iterative approach requires continuous development and testing; hence programming cannot be outsourced
Easy to manage systems development project as tasks are defined in phases, through output documents, and especially the hardest part - programming, which is defined through program modules in a top-down fashion	Not easy to manage systems development project as models require continuous revisiting, and the hardest part which is programming that depends on packages - are complex; thus task duration is hard to quantify

Table: Comparison Between the two Approaches of Systems Development

References:

- ✓ Hoffer, J.A., George, J.F. and Valacich J.S., "Modern Systems Analysis and Design", 3rd Edition, Pearson Education, 2003.
- ✓ K.E. Kendall and J.E. Kendall, "Systems Analysis and Design", 5th Edition, Pearson Education, 2003.
- ✓ V. Rajaraman, "Analysis and Design of Information Systems", 2nd Edition, Prentice Hall of India, New Delhi, 2002
- ✓ Joseph S. Valacich, Joey F. George, Jeffrey A. Hoffer, "Essentials of Systems Analysis and Design" - 5th Edition.
- ✓ Shelly Cashman, "System Analysis and Design" - 9th Edition.

Assignments:

(1) What is object oriented analysis and design? Compare it with the traditional system development life cycle.
(2) Differentiate between state diagrams and sequence diagrams in object oriented analysis and design. (T.U. 2067)
(3) Explain the unified modeling language (UML) with example. (T.U. 2068, 2069)
(4) Differentiate between object modeling and dynamic modeling. (T.U. 2070)

A Gentle Advice:

Please go through your text books and reference books for detail study!!! Thank you all.

Notes Compiled By:

Bijay Mishra
biizay.blogspot.com
9813911076 or 9841695609