Chapter 1: System Development Environment

A. The System Development Environment

1. System

The word System is derived from Greek word Systema, which means an organized relationship between any set of components to achieve some common cause or objective.

A system is "an orderly grouping of interdependent components linked together according to a plan to achieve a specific goal."

Constraints of a System

A system must have three basic constraints −

- A system must have some **structure and behavior** which is designed to achieve a predefined objective.
- **Interconnectivity** and **interdependence** must exist among the system components.
- The **objectives of the organization** have a **higher priority** than the objectives of its subsystems.

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

Properties of a System

A system has the following properties −

a. Organization
- Organization implies structure and order.
- It is the arrangement of components that helps to achieve predetermined objectives.

b. Interaction
- It is defined by the manner in which the components operate with each other.
- For example, in an organization, purchasing department must interact with production department and payroll with personnel department.

c. Interdependence
- Interdependence means how the components of a system depend on one another.
- For proper functioning, the components are coordinated and linked together according to a specified plan. The output of one subsystem is the required by other subsystem as input.

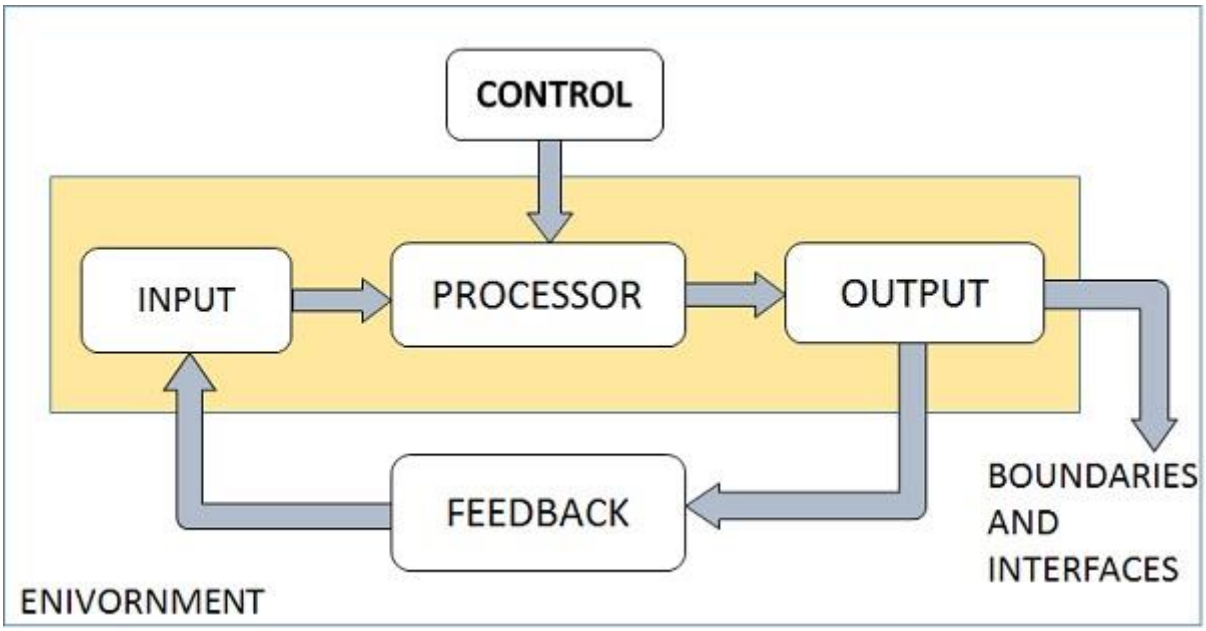
d. Integration
- Integration is concerned with how a system components are connected together.
- It means that the parts of the system work together within the system even if each part performs a unique function.

e. Central Objective

- The objective of system must be central.
- It may be real or stated. It is not uncommon for an organization to state an objective and operate to achieve another.
- The users must know the main objective of a computer application early in the analysis for a successful design and conversion.

Elements of a System

The following diagram shows the elements of a system −



a. Outputs and Inputs

- The main aim of a system is to produce an output which is useful for its user.
- Inputs are the information that enters into the system for processing.
- Output is the outcome of processing.

b. Processor(s)

- The processor is the element of a system that involves the actual transformation of input into output.
- It is the operational component of a system. Processors may modify the input either totally or partially, depending on the output specification.
- As the output specifications change, so does the processing. In some cases, input is also modified to enable the processor for handling the transformation.

c. Control

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

- The behavior of a computer System is controlled by the Operating System and software. In order to keep system in balance, what and how much input is needed is determined by Output Specifications.

d. Feedback

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

e. Environment

- The environment is the "super system" within which an organization operates.
- It is the source of external elements that strike on the system.
- It determines how a system must function. For example, vendors and competitors of organization's environment, may provide constraints that affect the actual performance of the business.

f. Boundaries and Interface

- A system should be defined by its boundaries. Boundaries are the limits that identify its components, processes, and interrelationship when it interfaces with another system.
- Each system has boundaries that determine its sphere of influence and control.
- The knowledge of the boundaries of a given system is crucial in determining the nature of its interface with other systems for successful design.

Types of Systems

The systems can be divided into the following types −

a. Physical or Abstract Systems

- Physical systems are tangible entities. We can touch and feel them.
- Physical System may be static or dynamic in nature. For example, desks and chairs are the physical parts of computer center which are static. A programmed computer is a dynamic system in which programs, data, and applications can change according to the user's needs.
- Abstract systems are non-physical entities or conceptual that may be formulas, representation or model of a real system.

b. Open or Closed Systems

- An open system must interact with its environment. It receives inputs from and delivers outputs to the outside of the system. For example, an information system which must adapt to the changing environmental conditions.
- A closed system does not interact with its environment. It is isolated from environmental influences. A completely closed system is rare in reality.

c. Adaptive and Non Adaptive System

- Adaptive System responds to the change in the environment in a way to improve their performance and to survive. For example, human beings, animals.
- Non Adaptive System is the system which does not respond to the environment. For example, machines.

d. Permanent or Temporary System

- Permanent System persists for long time. For example, business policies.
- Temporary System is made for specified time and after that they are demolished. For example, A DJ system is set up for a program and it is dissembled after the program.

e. Natural and Manufactured System

- Natural systems are created by the nature. For example, solar system, seasonal system.
- Manufactured System is the man-made system. For example, Rockets, dams, trains.

f. Deterministic or Probabilistic System

- Deterministic system operates in a predictable manner and the interaction between system components is known with certainty. For example, two molecules of hydrogen and one molecule of oxygen makes water.
- Probabilistic System shows uncertain behavior. The exact output is not known. For example, Weather forecasting, mail delivery.

g. Social, Human-Machine, Machine System

- Social System is made up of people. For example, social clubs, societies.
- In Human-Machine System, both human and machines are involved to perform a particular task. For example, Computer programming.
- Machine System is where human interference is neglected. All the tasks are performed by the machine. For example, an autonomous robot.

h. Man–Made Information Systems

- It is an interconnected set of information resources to manage data for particular organization, under Direct Management Control (DMC).
- This system includes hardware, software, communication, data, and application for producing information according to the need of an organization.
  Man-made information systems are divided into three types −
- **Formal Information System** − It is based on the flow of information in the form of memos, instructions, etc., from top level to lower levels of management.
- **Informal Information System** − this is employee based system which solves the day to day work related problems.
- **Computer Based System** − this system is directly dependent on the computer for managing business applications. For example, automatic library system, railway reservation system, banking system, etc.

Systems Models

a. Schematic Models

- A schematic model is a 2-D chart that shows system elements and their linkages.
- Different arrows are used to show information flow, material flow, and information feedback.

b. Flow System Models

- A flow system model shows the orderly flow of the material, energy, and information that hold the system together.
- Program Evaluation and Review Technique (PERT), for example, is used to abstract a real world system in model form.

c. Static System Models

- They represent one pair of relationships such as *activity–time* or *cost–quantity*.
- The Gantt chart, for example, gives a static picture of an activity-time relationship.

d. Dynamic System Models

- Business organizations are dynamic systems. A dynamic model approximates the type of organization or application that analysts deal with.
- It shows an ongoing, constantly changing status of the system. It consists of −
    - Inputs that enter the system
    - The processor through which transformation takes place
    - The program(s) required for processing
    - The output(s) that result from processing.

2. A Modern Approach to Systems Analysis and Design

- 1950s: All applications had to be developed in machine language or assembly language. They had to be developed from scratch because due to the absence of software industry.
- 1960s: Smaller, faster, less expensive computers, beginning of the software industry, use in-house development. • 1970s: Realized how expensive to develop customized information system for every application, started development of database management system.
- 1980s: The software industry expended greatly, CASE (computer aided software engineering) tools.
- Started writing application software in OOP languages, graphics were used, developed less software in-house and bought more from software vendors.
- 1990s: focus on system integration, GUI (Graphical user interface) applications, client/server platforms, Internet. • The new century: Web application development, wireless PDAs (personal digital assistants, e.g. pocket PCs), ASP (application service provider).

3. Information System

- In a simplest sense, a system that provides information to people in an organization is called information system (IS).
- Information systems in organizations capture and manage data to produce useful information that supports an organization and its employees, customers, suppliers and partners. So, many organizations consider information system to be the essential one.
- Information systems produce information by using data about significant people, places, and things from within the organization and/or from the external environment to make decisions, control operations, analyze problems, and create new products or services.

Types of Information Systems

a. Transaction Processing Systems (TPSs)

- These are the computerized systems that perform and records the daily routine transactions necessary to conduct business. These systems serve the operational level of the organization.
- Some examples include sales order entry, hotel reservation systems, payroll, employee record keeping, and shipping.

- Transaction processing systems are central to a business. TPS failure for a few hours can cause a firm's demise and perhaps other firms linked to it.
- Managers need TPS to monitor the status of internal operations and the firm's relations with external environment.
- TPS are also major producers of information for the other types of systems.
- Online transaction processing systems (OLTPS) is an interactive data processing system that involves a direct connection between TPS programs and users.
- As soon as a single transaction is entered into a computer system, the program interacts immediately with the user for that transaction.
- It is often known as the live system where there is no time lag between data creation and its processing. A good example of this system is online ticket reservation system.

b. Management Information Systems (MISs)

- These are the information systems at the management level of an organization and serve management-level functions like planning, controlling, and decision-making.
- These systems provide reports that are usually generated on a predetermined schedule and appear in prearranged format.
- Typically, these systems use internal data provided by the transaction processing systems. These systems are used for structured decision-making and in some cases for semi-structured decision making as well. Salary analysis and sales reporting are the examples in which MIS can be used.

c. Decision Support Systems (DSSs)

- These systems also serve at the management level of the organization.
- These systems combine data and sophisticated analytical models or data analysis tools to support semistructured and unstructured decision-making.
- These systems use internal information from TPS and MIS, and often information from external sources, such as current stock prices or product prices of competitors.
- DSS have more analytical power than other systems. Contract cost analysis is an example in which DSS can be used.

d. Executive Information Systems (EISs)

- These systems are also called executive support systems (ESSs) and serve the strategic level of the organization.
- These systems are designed to address unstructured decision making through advanced graphics and communication.
- These systems incorporate data about external events such as new tax laws or competitors, but they also draw summarized information from internal MIS and DSS.
- These systems are not designed to solve a specific problem but they provide a generalized computing and telecommunication capacity that can be applied to a changing array of problems.
- 5-year operating plan is an example in which EIS can be used.

e. Expert Systems

- An expert system is an extension of DSS that captures and reproduces the knowledge and expertise of an expert problem solver or decision maker and then simulates the "thinking" or "actions" of that expert. These systems imitate the logic and reasoning of the experts within their respective fields.
- Expert systems are implemented with artificial intelligence (AI) technology that captures, stores, and provides access to the reasoning of the experts.

f. Communication and Collaboration Systems

- These systems enable more effective communications between workers, partners, customers and suppliers to enhance their ability to collaborate.
- These systems use network technology that allows companies to coordinate with other organizations across great distances.
- These systems create new efficiencies and new relationships between an organization, its customers and suppliers, and business partners redefining organizational boundaries.

g. Office Automation Systems

- Office automation (OA) is more than word processing and spreadsheet applications.
- Office automation systems support the wide range of business office activities for improved work flow and communication between workers, regardless of whether or not those workers are located in the same office.
- Office automation functions include word processing, spreadsheet applications, and electronic mails, work group computing, fax processing, work flow management etc.
- Office automation systems can be designed to support both individuals and work groups. Personnel information systems are those designed to meet the needs of a single user.
- They are designed to boost an individual's productivity. Work group information systems, on the other hand, are designed to meet the needs of a work group. They are designed to boost the group's productivity.

Developing Information Systems

- System Development Methodology is a standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.
- Most organizations use a standard set of steps, called a systems development methodology to develop and support their information systems.
- Components of an Information System: People, Network ,Software, Hardware and Data

4. Systems Development Life Cycle (SDLC)

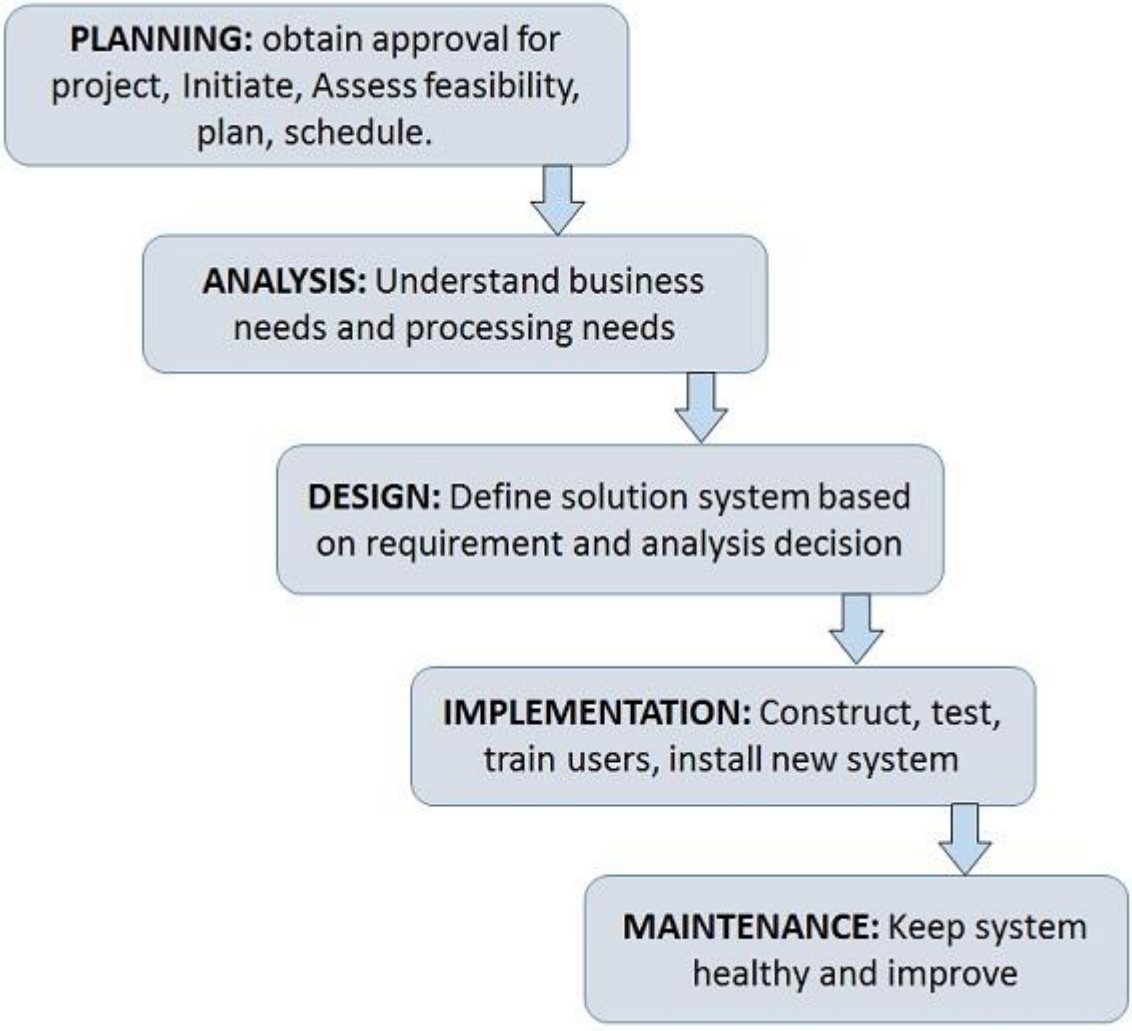
System Development Life Cycle (SDLC) is a conceptual model which includes policies and procedures for developing or altering systems throughout their life cycles.

SDLC is used by analysts to develop an information system. SDLC includes the following activities

- requirements
- design
- implementation
- testing
- deployment
- operations
- maintenance

5. Phases of SDLC

Systems Development Life Cycle is a systematic approach which explicitly breaks down the work into phases that are required to implement either new or modified Information System.



a. Feasibility Study or Planning

- Define the problem and scope of existing system.
- Overview the new system and determine its objectives.
- Confirm project feasibility and produce the project Schedule.
- During this phase, threats, constraints, integration and security of system are also considered.
- A feasibility report for the entire project is created at the end of this phase.

b. Analysis and Specification

- Gather, analyze, and validate the information.
- Define the requirements and prototypes for new system.
- Evaluate the alternatives and prioritize the requirements.
- Examine the information needs of end-user and enhances the system goal.
- A Software Requirement Specification (SRS) document, which specifies the software, hardware, functional, and network requirements of the system is prepared at the end of this phase.

c. System Design

- Includes the design of application, network, databases, user interfaces, and system interfaces.
- Transform the SRS document into logical structure, which contains detailed and complete set of specifications that can be implemented in a programming language.
- Create a contingency, training, maintenance, and operation plan.
- Review the proposed design. Ensure that the final design must meet the requirements stated in SRS document.
- Finally, prepare a design document which will be used during next phases.

d. Implementation

- Implement the design into source code through coding.
- Combine all the modules together into training environment that detects errors and defects.
- A test report which contains errors is prepared through test plan that includes test related tasks such as test case generation, testing criteria, and resource allocation for testing.
- Integrate the information system into its environment and install the new system.

e. Maintenance/Support

- Include all the activities such as phone support or physical on-site support for users that is required once the system is installing.
- Implement the changes that software might undergo over a period of time, or implement any new requirements after the software is deployed at the customer location.
- It also includes handling the residual errors and resolve any issues that may exist in the system even after the testing phase.
- Maintenance and support may be needed for a longer time for large systems and for a short time for smaller systems.

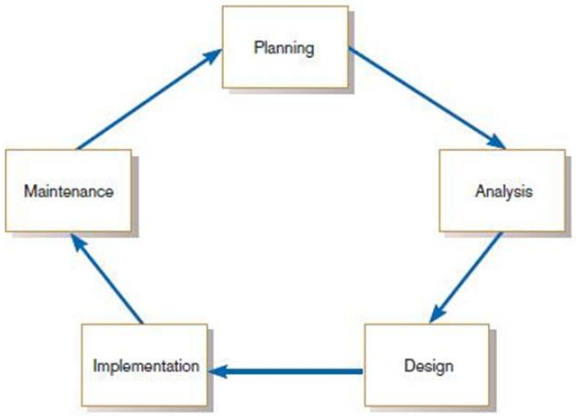


Figure
The systems development life cycle

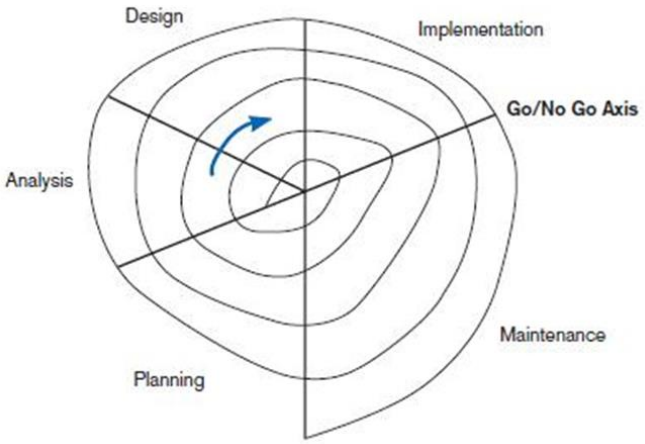


Figure
Evolutionary model

6. The Heart of the Systems Development Process

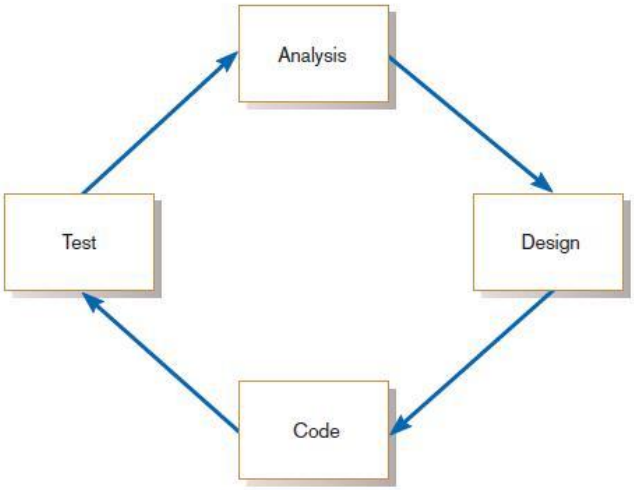


Figure
The analysis–design–code–test loop

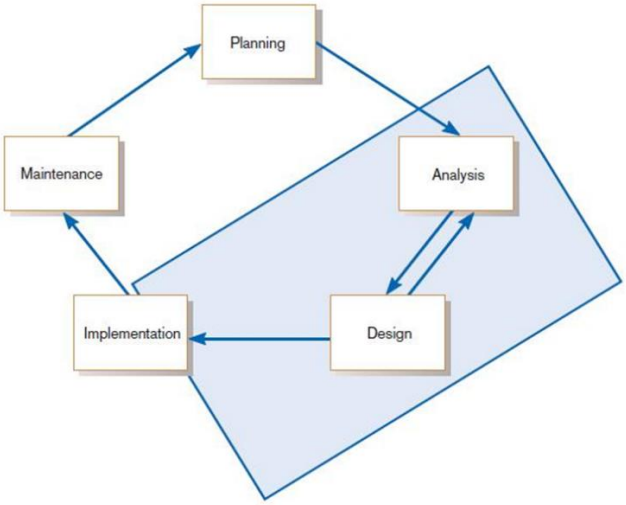
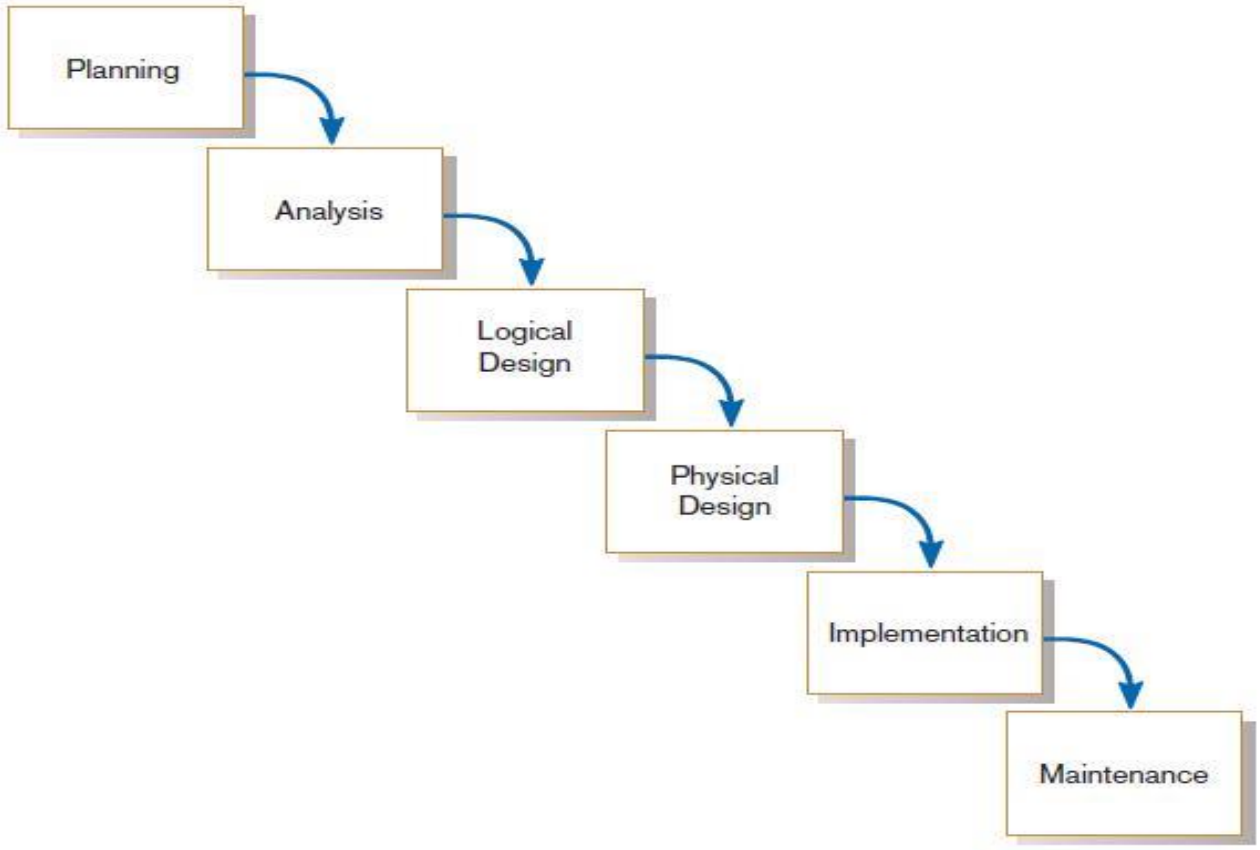


Figure
The heart of systems development

- Current practice combines analysis, design, and implementation into a single iterative and parallel process of activities

7. Traditional Waterfall SDLC

- It is a sequential model that divides software development into pre-defined phases.
- Each phase must be completed before the next phase can begin with no overlap between the phases.
- Each phase is designed for performing specific activity during the SDLC phase.
- It was introduced in 1970 by Winston Royce.



Waterfall Methodology can be used when:

- Requirements are not changing frequently
- Application is not complicated and big
- Project is short
- Requirement is clear
- Environment is stable
- Technology and tools used are not dynamic and is stable
- Resources are available and trained

Advantages	Dis-Advantages
• Before the next phase of development, each phase must be completed	• Error can be fixed only during the phase
• Suited for smaller projects where requirements are well defined	• It is not desirable for complex project where requirement changes frequently
• They should perform quality assurance test (Verification and Validation) before completing each stage	• Testing period comes quite late in the developmental process
• Elaborate documentation is done at every phase of the software's development cycle	• Documentation occupies a lot of time of developers and testers
• Project is completely dependent on project team with minimum client intervention	• Clients valuable feedback cannot be included with ongoing development phase
• Any changes in software is made during the process of the development	• Small changes or errors that arise in the completed software may cause a lot of problems

8. Approaches for Improving System Development

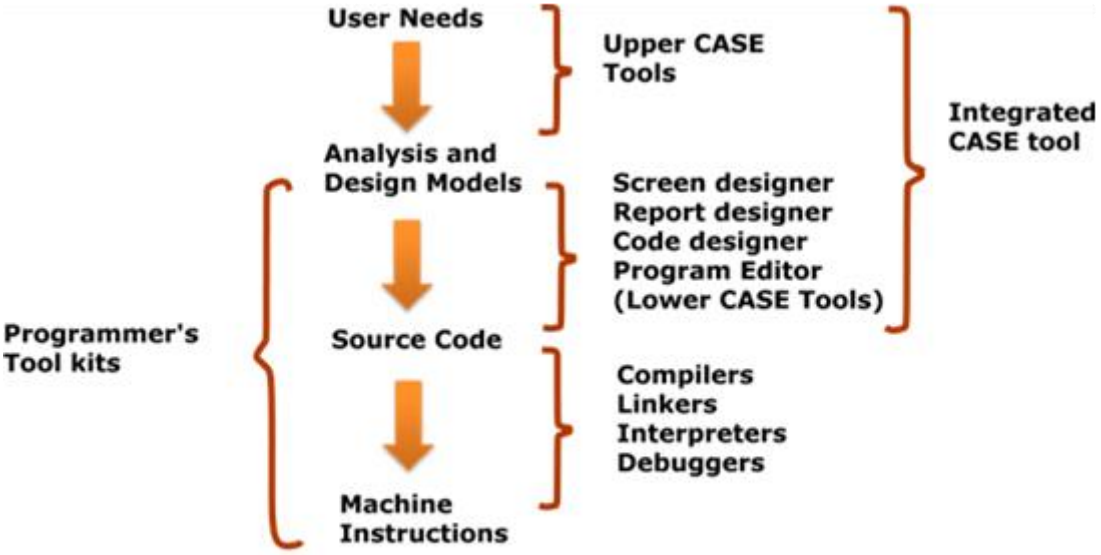
- Attempts to make systems development less of an art and more of a science are usually referred to as systems engineering or software engineering

CASE Tools

- Full form: Computer-Aided Software Engineering tools
- Diagramming tools/program enable graphical representation and automate the SDLC activities.
- Computer displays and report generators help prototype how systems "look and feel".
- Documentation generators standardize technical and user documentation.
- Code generators enable automatic generation of programs and database code directly from design documents, diagrams, forms, and reports.
- Supports a wide variety of SDLC activities: project identification and selection, project initiation and planning, analysis, design, and implementation and maintenance.

Components of CASE Tools

CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:



CASE Tool And Application Development

a. Central Repository

CASE tools require a central repository, which can serve as a source of common, integrated and consistent information.

Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored.

Central repository also serves as data dictionary.

b. Upper Case Tools

Upper CASE tools are used in planning, analysis and design stages of SDLC.

c. Lower Case Tools

Lower CASE tools are used in implementation, testing and maintenance. Integrated Case Tools - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

Types of CASE tools

Diagram tools

These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts.

Process Modeling Tools

Process modeling is method to create software process model, which is used to develop the software. Process modeling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, EPF Composer

Project Management Tools

These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers have to strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. For example, Creative Pro Office, Trac Project, Basecamp.

Documentation Tools

Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project. Documentation tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. For example, Doxygen, DrExplain,

Adobe RoboHelp for documentation.

Analysis Tools

These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. For example, Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis.

Design Tools

These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design

Configuration Management Tools

An instance of software is released under one version. Configuration Management tools deal with –

- Version and revision management
- Baseline configuration management
- Change control management

CASE tools help in this by automatic tracking, version management and release management. For

Example, Fossil, Git, Accu REV.

Change Control Tools

These tools are considered as a part of configuration management tools. They deal with changes made to the software after its baseline is fixed or when the software is first released. CASE tools automate change

tracking, file management, code management and more. It also helps in enforcing change policy of the organization.

Programming Tools

These tools consist of programming environments like IDE (Integrated Development Environment), in-built Modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse.

Prototyping Tools

Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspect of actual product.

Prototyping CASE tools essentially come with graphical libraries. They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing information. In addition, they provide simulation of software prototype. For example, Serena prototype composer, Mockup Builder.

Web Development Tools

These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. For example, Fontello, Adobe Edge Inspect, Foundation 3, Brackets.

Quality Assurance Tools

Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of quality as per organization standards. QA tools consist of configuration and change control tools and software testing tools. For example, SoapTest, AppsWatch, JMeter.

Maintenance Tools

Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, Bugzilla for defect tracking, HP Quality Center.

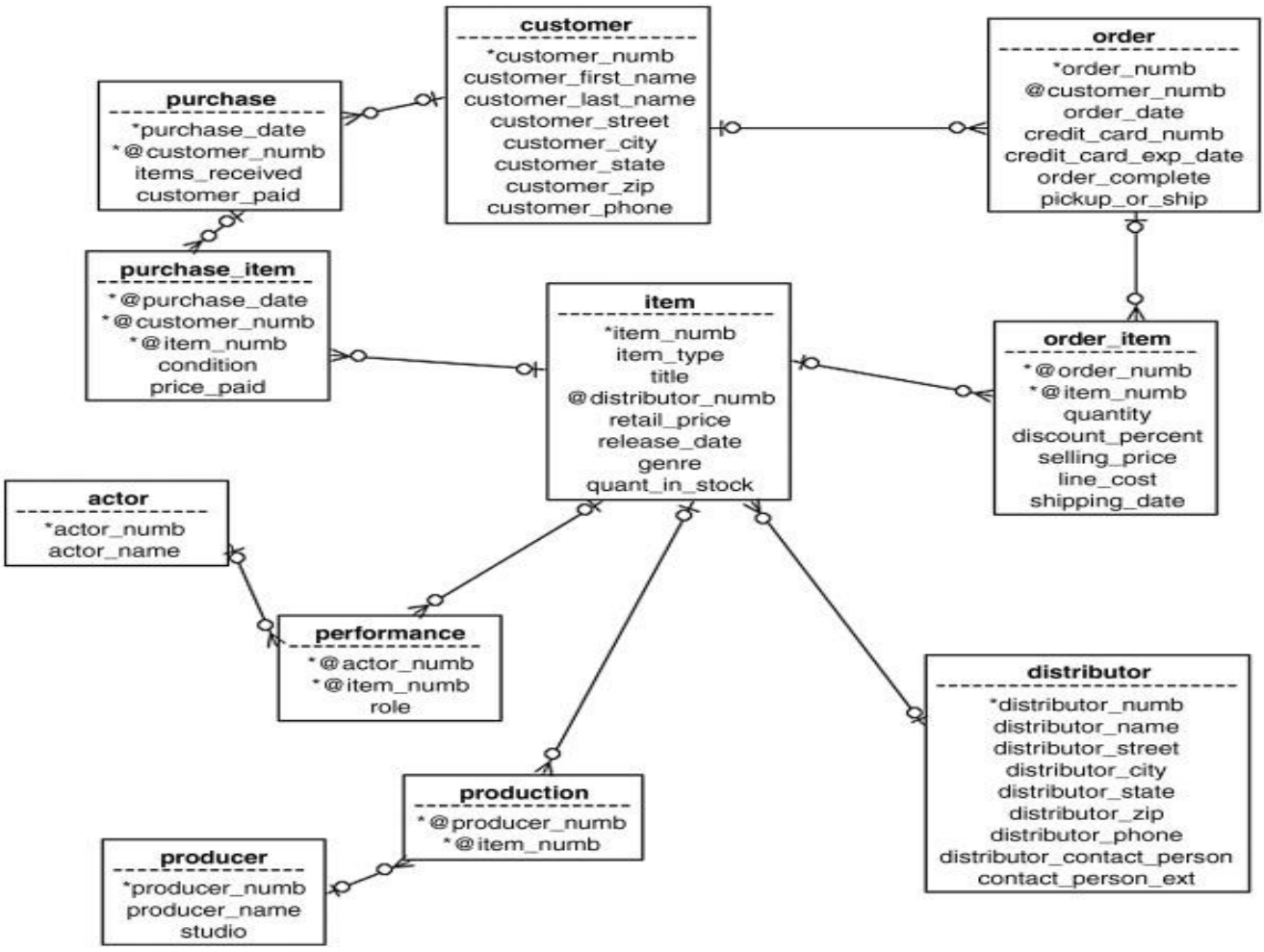


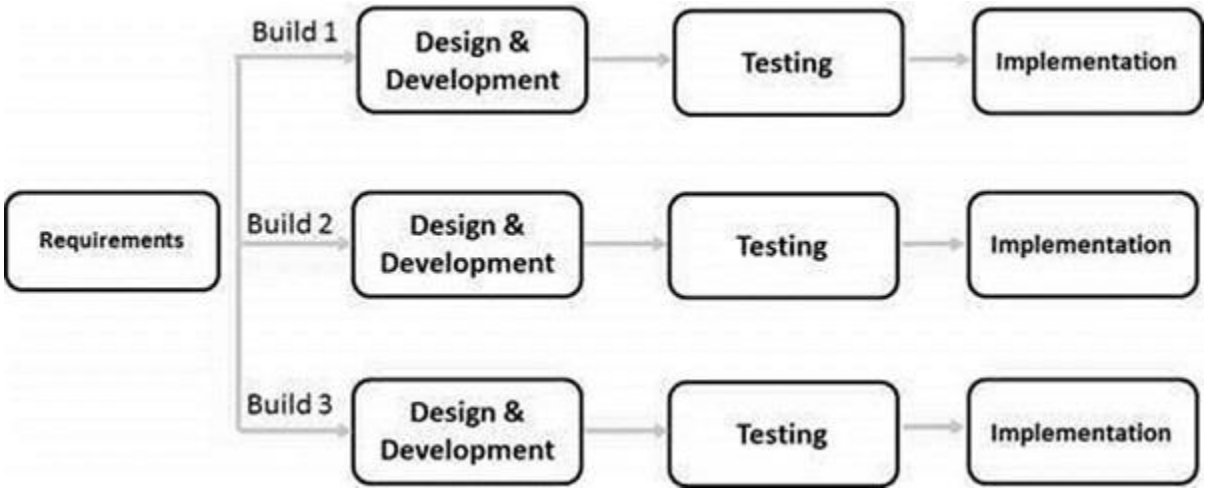
TABLE 1-2 Examples of CASE Usage within the SDLC

SDLC Phase	Key Activities	CASE Tool Usage
Project identification and selection	Display and structure high-level organizational information	Diagramming and matrix tools to create and structure information
Project initiation and planning	Develop project scope and feasibility	Repository and documentation generators to develop project plans
Analysis	Determine and structure system requirements	Diagramming to create process, logic, and data models
Logical and physical design	Create new system designs	Form and report generators to prototype designs; analysis and documentation generators to define specifications
Implementation	Translate designs into an information system	Code generators and analysis, form and report generators to develop system; documentation generators to develop system and user documentation
Maintenance	Evolve information system	All tools are used (repeat life cycle)

9. SDLC Models

Iterative/Incremental Model:

- Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented.
- At each iteration, design modifications are made and new functional capabilities are added.
- The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).



Advantages

- Some working functionality can be developed quickly and early in the life cycle.
- Results are obtained early and periodically.
- Parallel development can be planned.
- Progress can be measured.
- Less costly to change the scope/requirements.
- Testing and debugging during smaller iteration is easy.
- Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
- Easier to manage risk - High risk part is done first.
- With every increment, operational product is delivered.
- Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
- Risk analysis is better.
- It supports changing requirements.
- Initial Operating time is less.
- Better suited for large and mission-critical projects.
- During the life cycle, software is produced early which facilitates customer evaluation and feedback.

Disadvantages

- More resources may be required.
- Although cost of change is lesser, but it is not very suitable for changing requirements.
- More management attention is required.
- System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
- Defining increments may require definition of the complete system.
- Not suitable for smaller projects.
- Management complexity is more.
- End of project may not be known which risk is.
- Highly skilled resources are required for risk analysis.
- Projects progress is highly dependent upon the risk analysis phase.

Spiral Model

The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product, i.e. learning with maturity which involves minimum risk for the customer as well as the development firms.

The following pointers explain the typical uses of a Spiral Model −

- When there is a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which is usually the case.
- Requirements are complex and need evaluation to get clarity.
- New product line which should be released in phases to get enough customer feedback.
- Significant changes are expected in the product during the development cycle.

Advantages

- Changing requirements can be accommodated.
- Allows extensive use of prototypes.
- Requirements can be captured more accurately.
- Users see the system early.
- Development can be divided into smaller parts and the risky parts can be developed earlier which helps in better risk management.

Disadvantages

- Management is more complex.
- End of the project may not be known early.
- Not suitable for small or low risk projects and could be expensive for small projects.
- Process is complex

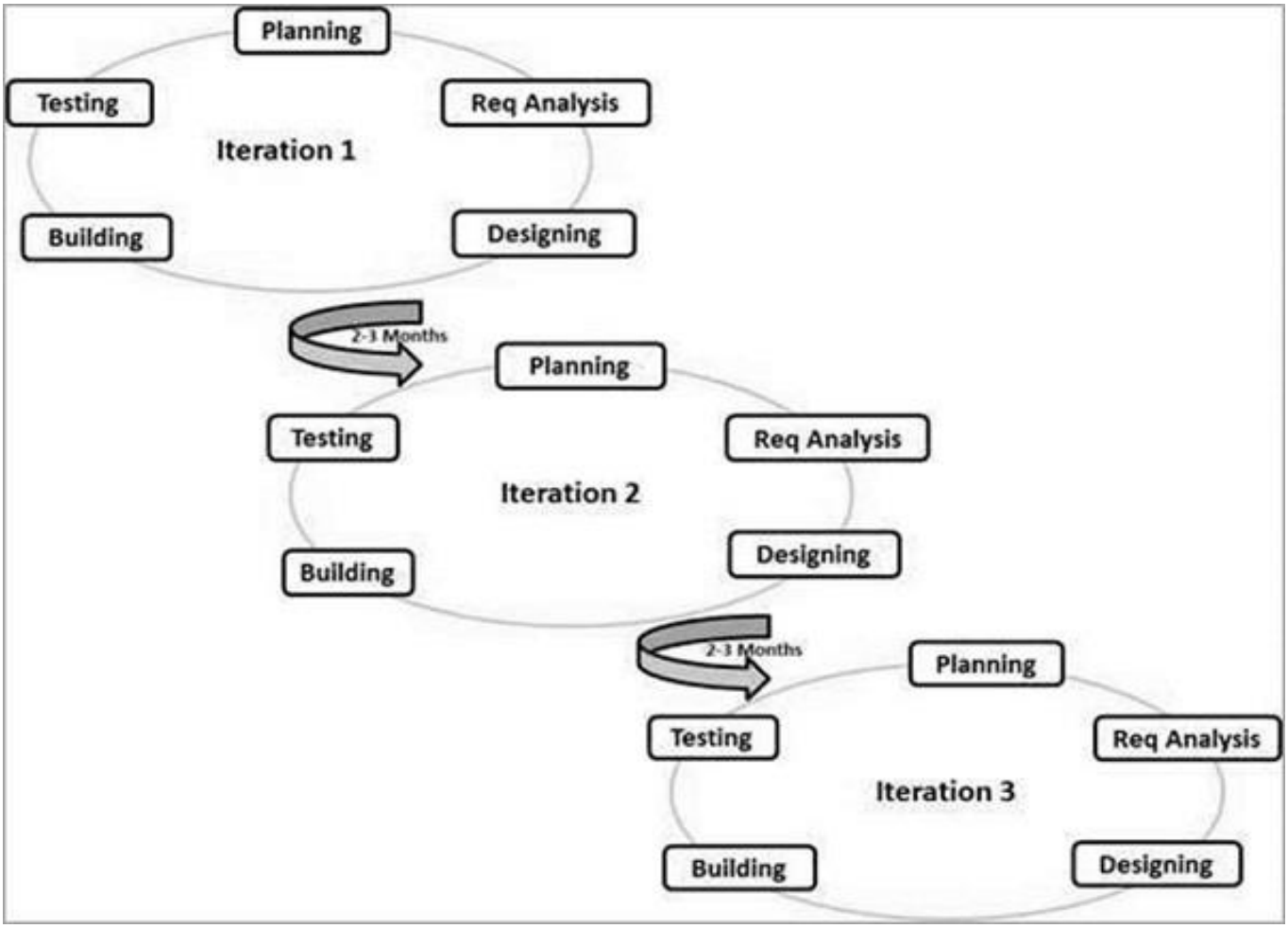
- Spiral may go on indefinitely.
- Large number of intermediate stages requires excessive documentation.

Agile Model

- Agile means able to move quickly and easily.
- Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.
- Agile Methods break the product into small incremental builds.
- These builds are provided in iterations. Each iteration typically lasts from about one to three weeks.

Every iteration involves cross functional teams working simultaneously on various areas like −

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.



Advantages:

- Is a very realistic approach to software development?
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required.
- Easy to manage.
- Gives flexibility to developers.

Disadvantages

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is a very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

When to Use

- Unpredictable or dynamic requirements
- Responsible and motivated developers
- Customers who understand the process and will get involved

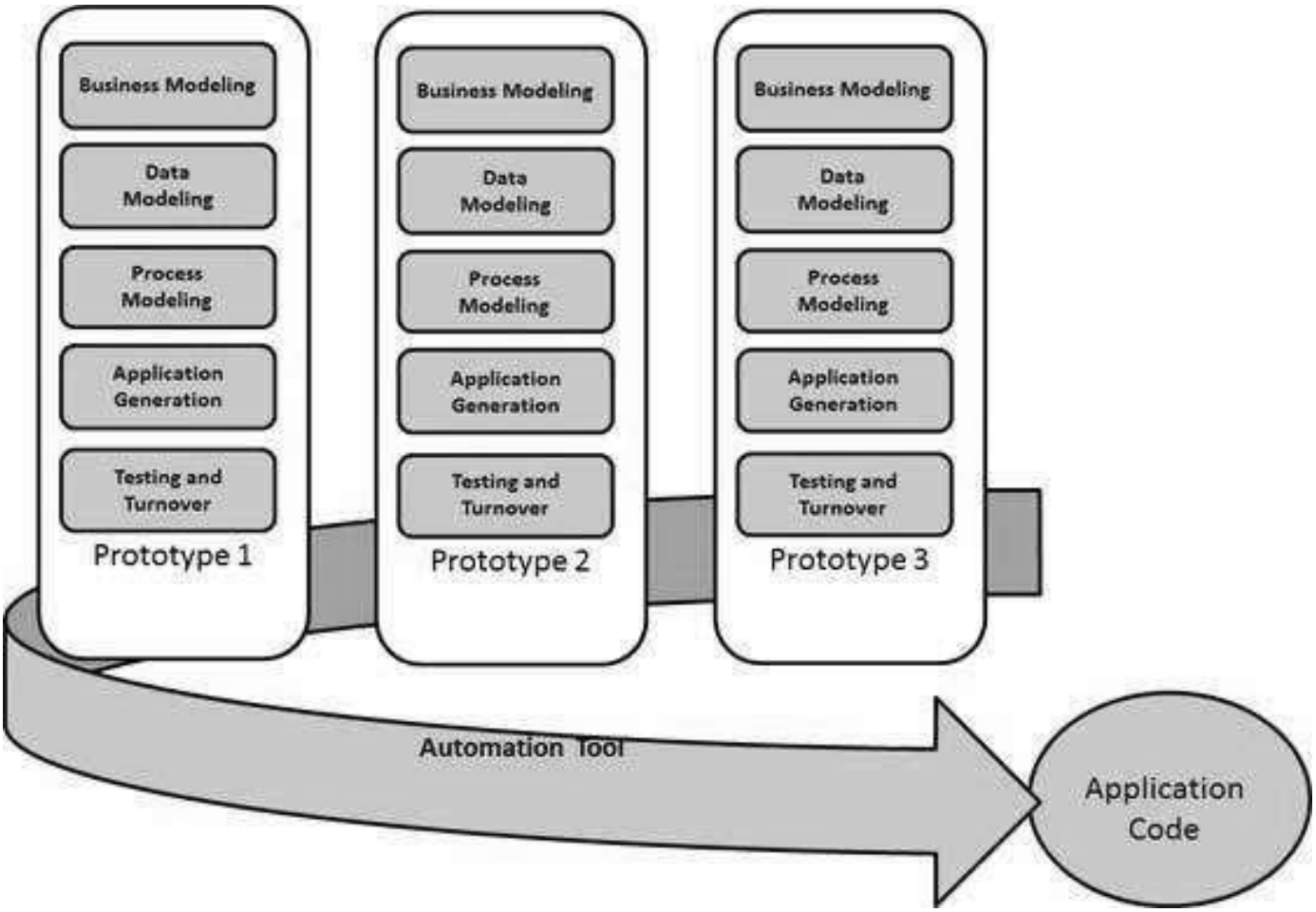
RAD Model

The **RAD (Rapid Application Development)** model is based on prototyping and iterative development with no specific planning involved.

The process of writing the software itself involves the planning required for developing the product.

Rapid Application Development focuses on

- Gathering customer requirements through workshops or focus groups,
- early testing of the prototypes by the customer using iterative concept,
- reuse of the existing prototypes (components),
- Continuous integration and rapid delivery.



Advantages:

- Changing requirements can be accommodated.
- Progress can be measured.
- Iteration time can be short with use of powerful RAD tools.
- Productivity with fewer people in a short time.
- Reduced development time.
- Increases reusability of components.
- Quick initial reviews occur.
- Encourages customer feedback.
- Integration from very beginning solves a lot of integration issues.

Disadvantages:

- Dependency on technically strong team members for identifying business requirements.
- Only system that can be modularized can be built using RAD.
- Requires highly skilled developers/designers.
- High dependency on modelling skills.
- Inapplicable to cheaper projects as cost of Modelling and automated code generation is very high.
- Management complexity is more.

- Suitable for systems that are component based and scalable.
- Requires user involvement throughout the life cycle.
- Suitable for project requiring shorter development times.

When to Use:

- RAD should be used only when a system can be modularized to be delivered in an incremental manner.
- It should be used if there is a high availability of designers for Modelling.
- It should be used only if the budget permits use of automated code generating tools.
- RAD SDLC model should be chosen only if domain experts are available with relevant business knowledge.
- Should be used where the requirements change during the project and working prototypes are to be presented to customer in small iterations of 2-3 months.

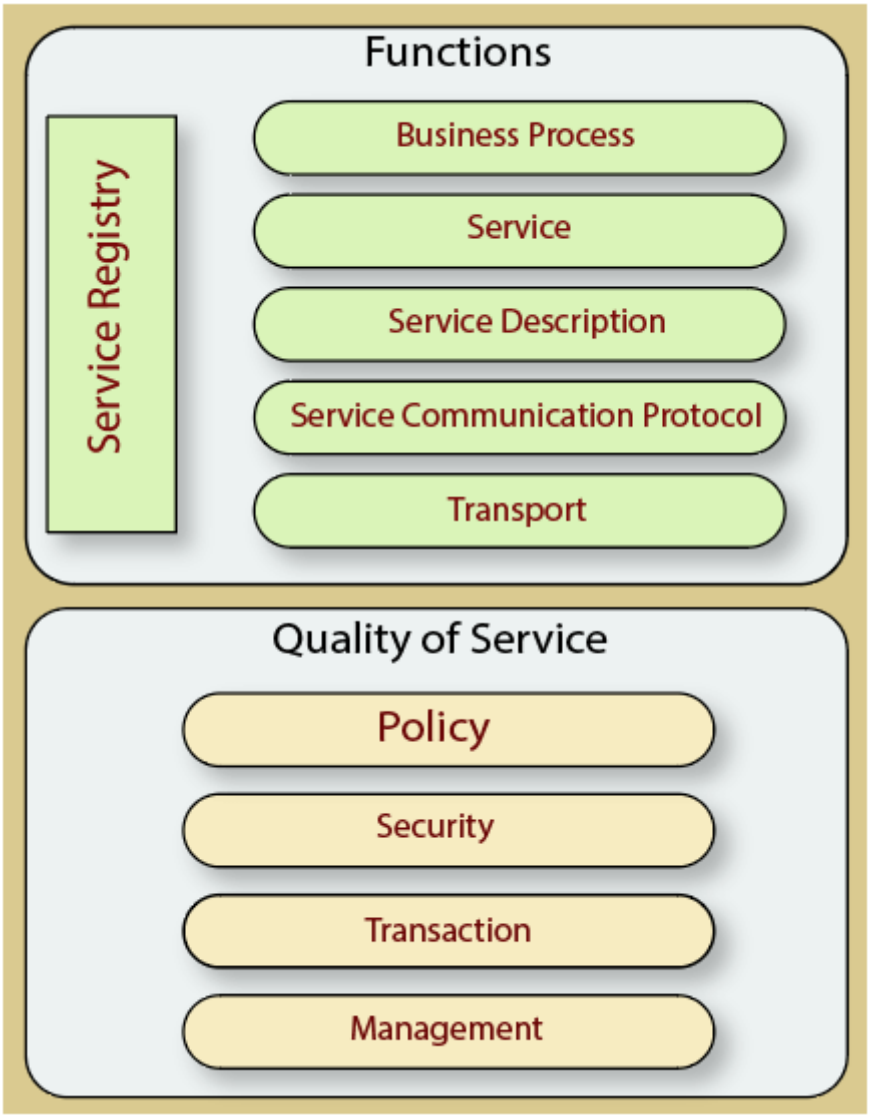
Service-Oriented Architecture (SOA)

- A service is a well-defined, self-contained function that represents a unit of functionality.
- A Service-Oriented Architecture or SOA is a design pattern which is designed to build distributed systems that deliver services to other applications through the protocol.
- Each service executes one action. Each service can be used in other application within the organization or even in other organizations.
- It is only a concept and not limited to any programming language or platform.

Characteristics:

- They are loosely coupled.
- They support interoperability.
- They are location-transparent
- They are self-contained.

Components of service-oriented architecture



Functional aspects

- Transport - It transports the service requests from the service consumer to the service provider and service responses from the service provider to the service consumer.
- Service Communication Protocol - It allows the service provider and the service consumer to communicate with each other.
- Service Description - It describes the service and data required to invoke it.
- Service - It is an actual service.
- Business Process - It represents the group of services called in a particular sequence associated with the particular rules to meet the business requirements.
- Service Registry - It contains the description of data which is used by service providers to publish their services.

Quality of Service aspects

- Policy - It represents the set of protocols according to which a service provider make and provide the services to consumers.
- Security - It represents the set of protocols required for identification and authorization.
- Transaction - It provides the surety of consistent result. This means, if we use the group of services to complete a business function, either all must complete or none of the complete.
- Management - It defines the set of attributes used to manage the services.

Advantages of SOA

- Easy to integrate - In a service-oriented architecture, the integration is a service specification that provides implementation transparency.
- Manage Complexity - Due to service specification, the complexities get isolated, and integration becomes more manageable.
- Platform Independence - The services are platform-independent as they can communicate with other applications through a common language.
- Loose coupling - It facilitates to implement services without impacting other applications or services.
- Parallel Development - As SOA follows layer-based architecture, it provides parallel development.
- Available - The SOA services are easily available to any requester.
- Reliable - As services are small in size, it is easier to test and debug them.

Extreme Programming (XP)

Extreme programming (XP) is one of the most important software development frameworks of Agile models.

It is used to improve software quality and responsiveness to customer requirements.

The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.

- its short cycles
- incremental planning approach
- focus on automated tests written by programmers and customers to monitor the development process.
- reliance on an evolutionary approach to development that lasts throughout the lifetime of the system.
- use of two-person programming teams.
- Planning, analysis, design, and construction are all fused into a single phase of activity.

Working model

- Under this approach, coding and testing are intimately related parts of the same process. The programmers who write the code also develop the tests. The emphasis is on testing those things that can break or go wrong, not on testing everything.
- Code is tested very soon after it is written. The overall philosophy behind eXtreme Programming is that the code will be integrated into the system it is being developed for and tested within a few hours after it has been written. If all the tests run successfully, then development proceeds. If not, the code is reworked until the tests are successful.

Advantages:

- More (and better) communication among developers,
- Higher levels of productivity,
- Higher-quality code, and
- Reinforcement of the other practices in eXtreme Programming, such as the code and test discipline
- Test based approach to requirements and quality assurance.

Disadvantage:

- It is not for everyone and is not applicable to every project.
- Programming pairs is costly
- Skilled/specialized manpower is needed.

Object-Oriented Analysis and Design (OOAD)

Based on objects rather than data or processes

Object: a structure encapsulating attributes and behaviors of a real-world entity

Object class: a logical grouping of objects sharing the same attributes and behaviors

Inheritance: hierarchical arrangement of classes enables subclasses to inherit properties of super classes

- The object-oriented approach combines data and processes (called methods) into single entities called objects.
- Objects usually correspond to the real things an information system deals with, such as customers, suppliers, contracts, and rental agreements.
- Putting data and processes together in one place recognizes the fact that there are a limited number of operations for any given data structure, and the object-oriented approach makes sense even though typical systems development keeps data and processes independent of each other.
- The goal of OOAD is to make systems elements more reusable, thus improving system quality and the productivity of systems analysis and design.

Advantages	Disadvantages
Focuses on data rather than the procedures as in Structured Analysis.	Functionality is restricted within objects. This may pose a problem for systems which are intrinsically procedural or computational in nature.
The principles of encapsulation and data hiding help the developer to develop systems that cannot be tampered by other parts of the system.	It cannot identify which objects would generate an optimal system design.
The principles of encapsulation and data hiding help the developer to develop systems that cannot be tampered by other parts of the system.	The object-oriented models do not easily show the communications between the objects in the system.
It allows effective management of software complexity by the virtue of modularity.	All the interfaces between the objects cannot be represented in a single diagram.
It can be upgraded from small to large systems at a greater ease than in systems following structured analysis.	

B. The Origin of Software

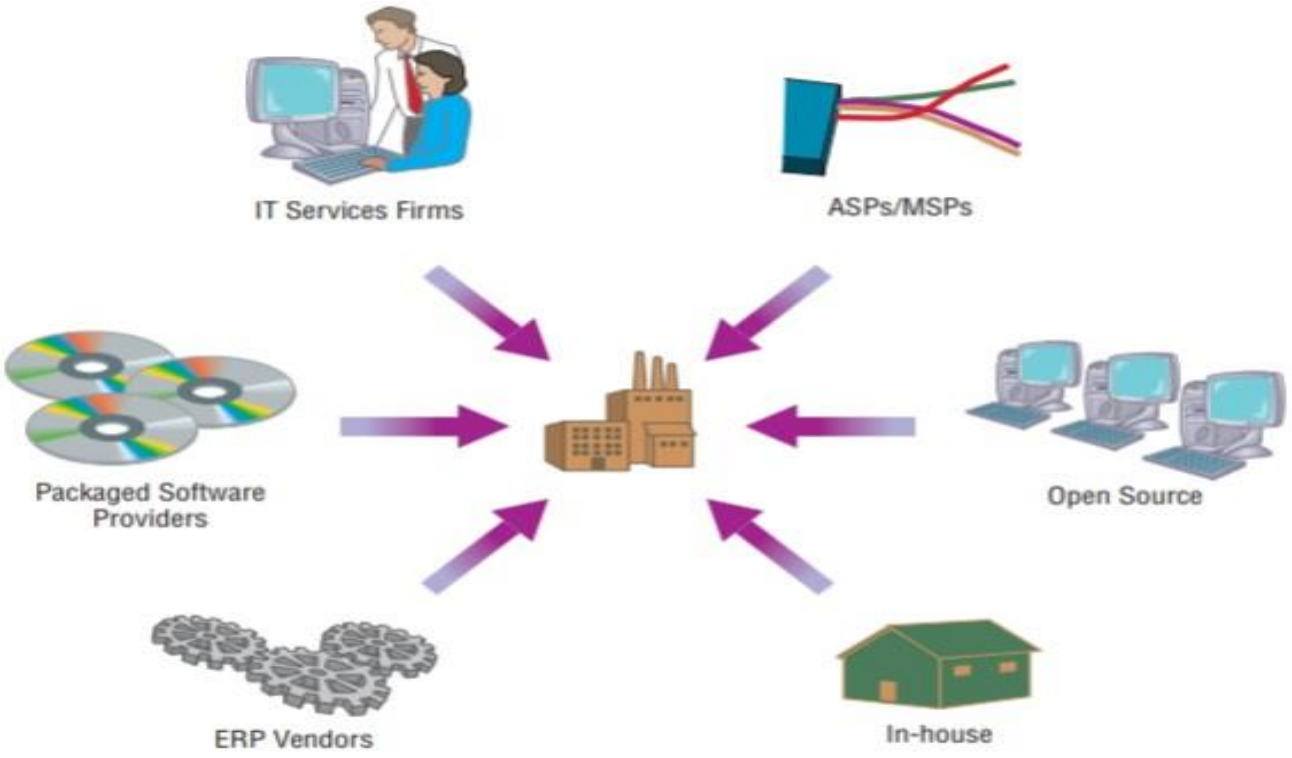
1. Introduction

- If you wanted to write application software, you did it in-house, and you wrote the software from scratch. Today there are many different sources of software and firms that produce software, rather than in the information systems department of a corporation.
- But for those of you who do go on to work in a corporate information systems department, the focus is no longer exclusively on in-house development.

2. Software Acquisition (assets/achievement)

- Internal corporate information systems departments now spend a smaller and smaller proportion of their time and effort on developing systems from scratch.
- Companies continue to spend relatively little time and money on traditional software development and maintenance.

- Instead, they invest in packaged software, open-source software, and outsourced services.



Outsourcing:

If one organization develops or runs a computer application for another organization, that practice is called outsourcing. Outsourcing includes a spectrum of working arrangements. At one extreme is having a firm develop and run your application on its computers—all you do is supply input and take output. A common example of such an arrangement is a company that runs payroll applications for clients so that clients do not have to develop an independent in-house payroll system. Instead, they simply provide employee payroll information to the company, and, for a fee, the company returns completed paychecks, payroll accounting reports, and tax and other statements for employees. For many organizations, payroll is a very cost-effective operation when outsourced in this way. Another example of outsourcing would be if you hired a company to run your applications at your site on your computers. In some cases, an organization employing such an arrangement will dissolve some or all of its information systems (IS) unit and fire all of it IS employees. Often the company brought in to run the organization's computing will rehire many of the organization's original IS unit employees.

The reasons behind outsourcing are:

- freeing up internal resources,
- increasing the revenue potential of the organization,
- reducing time to market,
- increasing process efficiencies, and
- Outsourcing noncore activities.

E.g. biggest outsourcing companies are IBM and EDS.

Sources of Software:

We can group the sources of software into six major categories:

- information technology services firms,
- packaged software producers,
- enterprise-wide solutions,
- cloud computing vendors,
- open-source software, and
- in-house developers

Information Technology Services Firms:

- If a company needs an information system but does not have the expertise or the personnel to develop the system in-house the company will likely consult an information technology services firm.
- IT services firms help companies develop custom information systems for internal use, or they develop, host, and run applications for customers, or they provide other services.
- That many of the leading software companies in the world specialize in services, which include custom systems development.
- These firms employ people with expertise in the development of information systems.
- Their consultants may also have expertise in a given business area.
- For example, consultants who work with banks understand financial institutions as well as information systems.
- Consultants use many of the same methodologies, techniques, and tools that companies use to develop systems in-house.

Packaged Software Producers:

- The growth of the software industry has been phenomenal since its beginnings in the mid-1960s.
- Some of the largest computer companies in the world are companies that produce software exclusively.
- A good example is Microsoft, probably the best-known software company in the world. Almost 87 percent of Microsoft's revenue comes from its software sales, mostly for its Windows operating systems and its personal productivity software, the Microsoft Office Suite.
- Oracle is exclusively a software company known primarily for its database software, but Oracle also makes enterprise systems.
- Software companies develop what are sometimes called prepackaged or off-the-shelf systems.
- Software companies develop software to run on many different computer platforms, from microcomputers to large mainframes.

Enterprise Solutions Software:

- Many firms have chosen complete software solutions, called enterprise solutions or enterprise resource planning (ERP) systems, to support their operations and business processes.

- These ERP software solutions consist of a series of integrated modules. Each module supports an individual, traditional business function, such as accounting, distribution, manufacturing, or human resources.
- The difference between the modules and traditional approaches is that the modules are integrated to focus on business processes rather than on business functional areas.
- For example, a series of modules will support the entire order entry process, from receiving an order, to adjusting inventory, to shipping to billing, to after-the-sale service.
- The traditional approach would use different systems in different functional areas of the business, such as a billing system in accounting and an inventory system in the warehouse.
- Using enterprise software solutions, a firm can integrate all parts of a business process in a unified information system.

The benefits of the enterprise solutions approach

- Include a single repository of data for all aspects of a business process and the flexibility of the modules.
- A single repository ensures more consistent and accurate data, as well as less maintenance.
- The modules are flexible because additional modules can be added as needed once the basic system is in place.
- Added modules are immediately integrated into the existing system.

However, there are disadvantages to enterprise solutions software.

- The systems are very complex, so implementation can take a long time to complete.
- Organizations typically do not have the necessary expertise in-house to implement the systems, so they must rely on consultants or employees of the software vendor, which can be very expensive.
- In some cases, organizations must change how they do business in order to benefit from a migration to enterprise solutions.

Cloud Computing:

- Another method for organizations to obtain applications is to rent them or license them from third-party providers who run the applications at remote sites.
- Users have access to the applications through the Internet or through virtual private networks.
- The application provider buys, installs, maintains, and upgrades the applications.
- Users pay on a per-use basis or they license the software, typically month to month. Although this practice has been known by many different names over the years, today it is called cloud computing.
- Cloud computing refers to the provision of applications over the Internet, where customers do not have to invest in the hardware and software resources needed to run and maintain the applications.
- A well-known example of cloud computing is Google Apps, where users can share and create documents, spreadsheets, and presentations. Another well-known example is Amazon web services, Azur, which provides customer relationship management software online.

Benefits:

- freeing internal IT staff,
- gaining access to applications faster than via internal development
- Achieving lower cost access to corporate-quality applications.

Open-Source Software:

- Open-source software is unlike the other types of software you have read about so far.
- Open-source software is different because it is freely available, not just the final product but the source code itself.
- It is also different because it is developed by a community of interested people instead of by employees of a particular company.
- Open-source software performs the same functions as commercial software, such as operating systems, e-mail, database systems, web browsers, and so on.
- Some of the most well-known and popular open-source software names are Linux, an operating system; mySQL, a database system; and Firefox, a web browser. Open source also applies to software components and objects.
- Open source is developed and maintained by communities of people, and sometimes these communities can be very large.

In-House Development:

- We have talked about several different types of external organizations that serve as sources of software, but in-house development remains an option.
- In-house development has become a progressively smaller piece of all systems development work that takes place in and for organizations. Internal corporate information systems departments now spend a smaller and smaller proportion of their time and effort on developing systems from scratch.
- In-house development can lead to a larger maintenance burden than other development methods, such as packaged applications.

E.g. Bank application, etc.

3. REUSE:

- Reuse is the use of previously written software resources in new applications.
- Because so many bits and pieces of applications are relatively generic across applications, it seems intuitive that great savings can be achieved in many areas if those generic bits and pieces do not have to be written a new each time they are needed.
- Reuse should increase programmer productivity because being able to use existing software for some functions means they can perform more work in the same amount of time.
- Reuse should also decrease development time, minimizing schedule overruns. Because existing pieces of software have already been tested, reusing them should also result in higher-quality software with lower defect rates, decreasing maintenance costs.

Advantages:

- Less effort,
- Time-saving,
- Reduce cost,
- Increase software productivity,
- Utilize fewer resources,
- Leeds to a better quality software.

C. Managing the Information System Project

1. Introduction:

In this chapter, we focus on the systems analyst's role as project manager of an information systems project. Throughout the SDLC, the project manager is responsible for initiating, planning, executing, and closing down the systems development project.

Project management is arguably the most important aspect of an information systems development project. Effective project management helps to ensure that systems development projects meet customer expectations and are delivered within budget and time constraints.

The project management process involves four phases:

- Initiating the project
- Planning the project
- Executing the project
- Closing down the project

a. Initiating a project:

During project initiation, the project manager performs several activities to assess the size, scope, and complexity of the project and to establish procedures to support subsequent activities. The types of activities that will perform when initiating a project are summarized below:

Establishing the project initiation team:

This activity involves organizing project team members to assist in accomplishing the project initiation activities.

Establishing a relationship with the customer:

A thorough understanding of your customer builds stronger partnerships and higher levels of trust

Establishing the project initiation plan:

This step defines the activities required to organize the initiation team while it is working to define the goals and scope of the project.

Establishing management procedures:

Successful projects require the development of effective management procedures.

Establishing the project management environment and project workbook:

The focus of this activity is to collect and organize the tools that you will use while managing the project and to construct the project workbook. Diagrams, charts, and system descriptions provide much of the project workbook contents. Thus, the project workbook serves as a repository for all project correspondence, inputs, outputs, deliverables, procedures, and standards established by the project team.

Developing the project charter:

The project charter is a short (typically one page), high-level document prepared for the customer that describes what the project will deliver and outlines many of the key elements of the project.

b. Planning the project

Research has found a positive relationship between effective project planning and positive project outcomes. Project planning involves defining clear, discrete activities and the work needed to complete each activity within a single project. It often requires you to make numerous assumptions about the availability of resources such as hardware, software, and personnel.

Describing project scope, alternatives, and feasibility:

The purpose of this activity is to understand the content and complexity of the project.

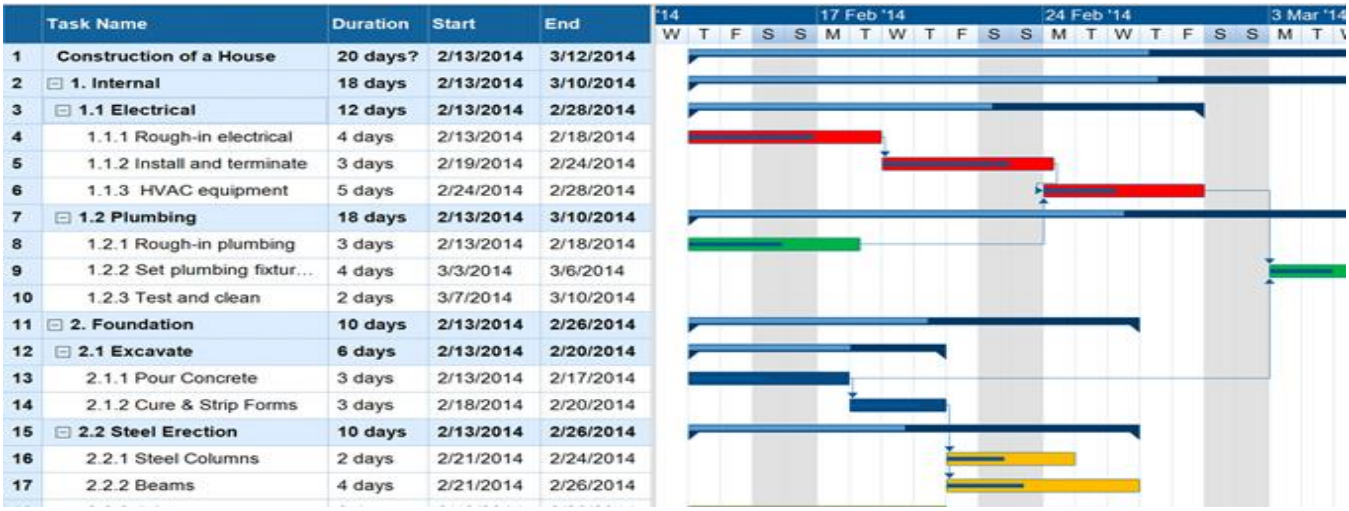
During this activity, you should reach agreement on the following questions:

- What problem or opportunity does the project address?
- What are the quantifiable results to be achieved?
- What needs to be done?
- How will success be measured?
- How will we know when we are finished?

After defining the scope of the project, your next objective is to identify and document general alternative solutions for the current business problem or opportunity.

Dividing the project into manageable tasks:

- This is a critical activity during the project planning process. Here, you must divide the entire project into manageable tasks and then logically order them to ensure a smooth evolution between tasks.
- For example, suppose that you are working on a new development project and need to collect system requirements by interviewing users of the new system and reviewing reports they currently use to do their job. A work breakdown for these activities is represented in a Gantt chart.
- A Gantt chart is a graphical representation of a project that shows each task as a horizontal bar whose length is proportional to its time for completion.



	Task Name	Duration	Start	End
1	Construction of a House	20 days?	2/13/2014	3/12/2014
2	1. Internal	18 days	2/13/2014	3/10/2014
3	1.1 Electrical	12 days	2/13/2014	2/28/2014
4	1.1.1 Rough-in electrical	4 days	2/13/2014	2/18/2014
5	1.1.2 Install and terminate	3 days	2/19/2014	2/24/2014
6	1.1.3 HVAC equipment	5 days	2/24/2014	2/28/2014
7	1.2 Plumbing	18 days	2/13/2014	3/10/2014
8	1.2.1 Rough-in plumbing	3 days	2/13/2014	2/18/2014
9	1.2.2 Set plumbing fixtur...	4 days	3/3/2014	3/6/2014
10	1.2.3 Test and clean	2 days	3/7/2014	3/10/2014
11	2. Foundation	10 days	2/13/2014	2/26/2014
12	2.1 Excavate	6 days	2/13/2014	2/20/2014
13	2.1.1 Pour Concrete	3 days	2/13/2014	2/17/2014
14	2.1.2 Cure & Strip Forms	3 days	2/18/2014	2/20/2014
15	2.2 Steel Erection	10 days	2/13/2014	2/26/2014
16	2.2.1 Steel Columns	2 days	2/21/2014	2/24/2014
17	2.2.2 Beams	4 days	2/21/2014	2/26/2014

Estimating resources and creating a resource plan:

The goal of this activity is to estimate resource requirements for each project activity and to use this information to create a project resource plan. The resource plan helps assemble and deploy resources in the most effective manner.

For example, you would not want to bring additional programmers onto the project at a rate faster than you could prepare work for them. Project managers use a variety of tools to assist in making estimates of project size and costs. The most widely used method is called COCOMO (constructive cost model), COCOMO predict human resource requirements for basic, intermediate, and very complex systems.

Developing a preliminary schedule:

- During this activity, you use the information on tasks and resource availability to assign time estimates to each activity in the work breakdown structure.
- These time estimates will enable you to create target starting and ending dates for the project.
- Target dates can be revisited and modified until a schedule is produced that is acceptable to the customer. Determining an acceptable schedule may require that you find additional or different resources or that the scope of the project be changed.
- The schedule may be represented as a Gantt chart or as a network diagram

Developing a communication plan:

The goal of this activity is to outline the communication procedures among management, project team members, and the customer.

The communication plan includes

- when and how written and oral reports will be provided by the team,
- how team members will coordinate work,
- what messages will be sent to announce the project to interested parties, and
- What kinds of information will be shared with vendors and external contractors involved with the project.

Determining project standards and procedures:

- During this activity, you will specify how various deliverables are produced and tested by you and your project team.
- For example, the team must decide which tools to use, how the standard SDLC might be modified, which SDLC methods will be used, documentation styles (e.g., type fonts and margins for user manuals), how team members will report the status of their assigned activities, and terminology.
- Setting project standards and procedures for work acceptance is a way to ensure the development of a high-quality system.
- Also, it is much easier to train new team members when clear standards are in place. Organizational standards for project management and conduct make the determination of individual project standards easier and the interchange or sharing of personnel among different projects feasible.

Identifying and assessing risk:

- The goal of this activity is to identify sources of project risk and estimate the consequences of those risks.
- Risks might arise from the use of new technology, prospective users' resistance to change, availability of critical resources, competitive reactions or changes in regulatory actions due to the construction of a system, or team member inexperience with technology or the business area.
- You should continually try to identify and assess project risk.

Creating a preliminary budget:

- During this phase, you need to create a preliminary budget that outlines the planned expenses and revenues associated with your project.
- The project justification will demonstrate that the benefits are worth these costs.

Developing a Project Scope Statement:

- An important activity that occurs near the end of the project planning phase is the development of the Project Scope Statement.
- Developed primarily for the customer, this document outlines work that will be done and clearly describes what the project will deliver.
- The Project Scope Statement is useful to make sure that you, the customer, and other project team members have a clear understanding of the intended project size, duration, and outcomes.

Setting a Baseline Project Plan:

- Once all of the prior project planning activities have been completed, you will be able to develop a Baseline Project Plan.
- This baseline plan provides an estimate of the project's tasks and resource requirements and is used to guide the next project phase—execution.
- As new information is acquired during project execution, the baseline plan will continue to be updated.

c. Executing the Project:

Project execution puts the Baseline Project Plan into action. Within the context of the SDLC, project execution occurs primarily during the analysis, design, and implementation phases.

Executing the Baseline Project Plan:

- This means that you initiate the execution of project activities, acquire and assign resources, orient and train new team members, keep the project on schedule, and ensure the quality of project deliverables.
- This is a formidable task, but a task made much easier through the use of sound project management techniques.

Monitoring project progress against the Baseline Project Plan:

- While you execute the Baseline Project Plan, you should monitor your progress.
- If the project gets ahead of (or behind) schedule, you may have to adjust resources, activities, and budgets. Monitoring project activities can result in modifications to the current plan.
- Measuring the time and effort expended on each activity will help you improve the accuracy of estimations for future projects.
- Monitoring progress means that the team leader must evaluate and appraise (to examine someone or something in order to judge their qualities, success or needs)each team member, occasionally change work assignments or request changes in personnel, and provide feedback to the employee's supervisor.

Managing changes to the Baseline Project Plan:

You will encounter pressure to make changes to the baseline plan.

Numerous events may initiate a change to the Baseline Project Plan, including the following possibilities:

- A slipped completion date for an activity
- A bungled (to do something wrong, in a careless or stupid way) activity that must be redone
- The identification of a new activity that becomes evident later in the project
- An unforeseen change in personnel due to sickness, resignation, or termination

Maintaining the project workbook:

- As in all project phases, maintaining complete records of all project events is necessary.
- The workbook provides the documentation new team members require to assimilate (to take in, fit into, or become similar) project tasks quickly.
- It explains why design decisions were made and is a primary source of information for producing all project reports.

Communicating the project status:

- The project manager is responsible for keeping all stakeholders— system developers, managers, and customers—abreast (describes two or more people who are next to each other and moving in the same direction) of the project status.

d. Closing Down the project

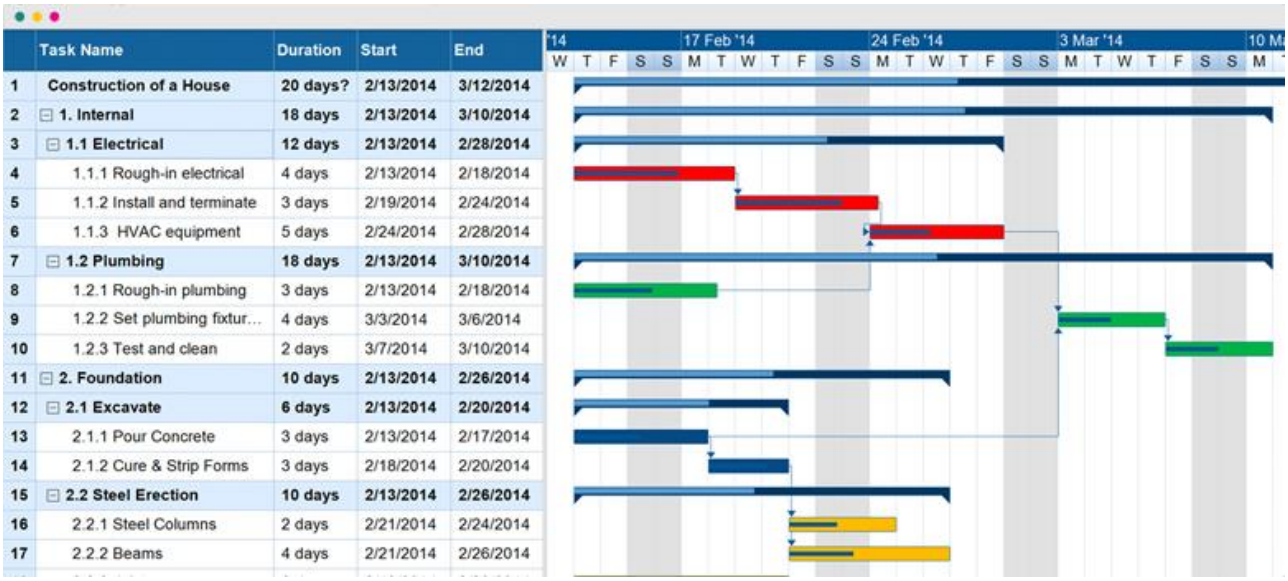
- The focus of project closedown is to bring the project to an end. Projects can conclude with a natural or unnatural termination.
- A natural termination occurs when the requirements of the project have been met—the project has been completed and is a success.
- An unnatural termination occurs when the project is stopped before completion.
- Several events can cause an unnatural termination of a project.
- For example, it may be learned that the assumption used to guide the project proved to be false, that the performance of the systems or development group was somehow inadequate (lack), or that the requirements are no longer relevant or valid in the customer's business environment.

- The most likely reasons for the unnatural termination of a project relate to running out of time or money, or both.

2. Representing and Scheduling Project Plans

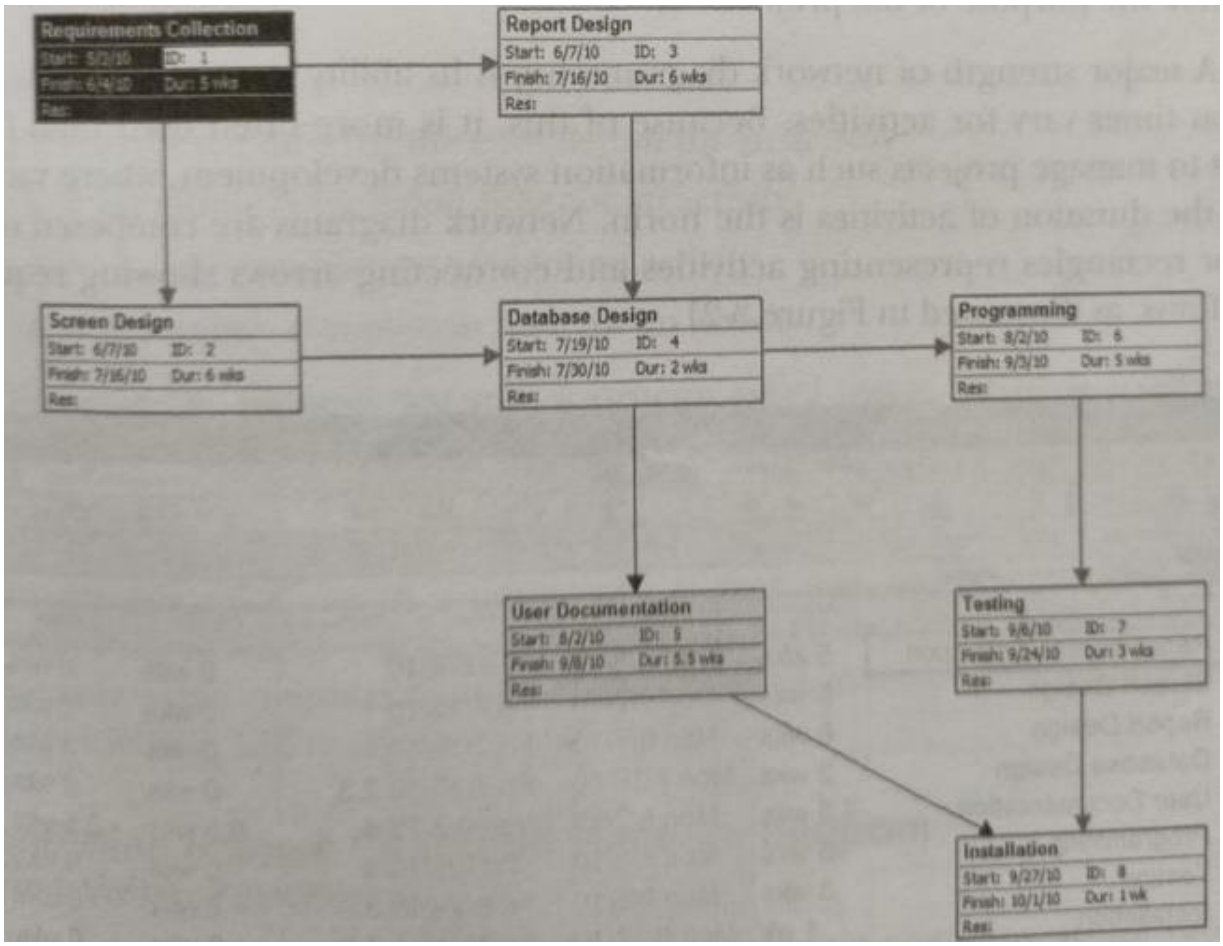
- A project manager has a wide variety of techniques available for depicting (to represent or show something in a picture or story) and documenting project plans.
- These planning documents can take the form of graphical or textual reports, although graphical reports have become most popular for depicting project plans.
- The most commonly used methods are Gantt charts and network diagrams. Because Gantt charts do not (typically) show how tasks must be ordered (precedence) but simply show when a task should begin and when it should end, they are often more useful for depicting (representing) relatively simple projects or subparts of a larger project, showing the activities of a single worker, or monitoring the progress of activities compared to scheduled completion dates.
- Graphical diagrams that depict project plans:

A Gantt char



	Task Name	Duration	Start	End
1	Construction of a House	20 days?	2/13/2014	3/12/2014
2	1. Internal	18 days	2/13/2014	3/10/2014
3	1.1 Electrical	12 days	2/13/2014	2/28/2014
4	1.1.1 Rough-in electrical	4 days	2/13/2014	2/18/2014
5	1.1.2 Install and terminate	3 days	2/19/2014	2/24/2014
6	1.1.3 HVAC equipment	5 days	2/24/2014	2/28/2014
7	1.2 Plumbing	18 days	2/13/2014	3/10/2014
8	1.2.1 Rough-in plumbing	3 days	2/13/2014	2/18/2014
9	1.2.2 Set plumbing fixtur...	4 days	3/3/2014	3/6/2014
10	1.2.3 Test and clean	2 days	3/7/2014	3/10/2014
11	2. Foundation	10 days	2/13/2014	2/26/2014
12	2.1 Excavate	6 days	2/13/2014	2/20/2014
13	2.1.1 Pour Concrete	3 days	2/13/2014	2/17/2014
14	2.1.2 Cure & Strip Forms	3 days	2/18/2014	2/20/2014
15	2.2 Steel Erection	10 days	2/13/2014	2/26/2014
16	2.2.1 Steel Columns	2 days	2/21/2014	2/24/2014
17	2.2.2 Beams	4 days	2/21/2014	2/26/2014

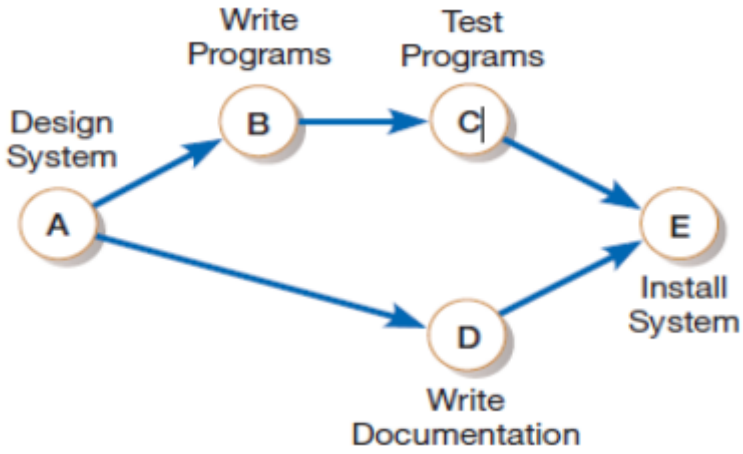
A network diagram



- Project scheduling and management require that time, costs, and resources be controlled.
- Resources are any person, group of people, piece of equipment, or material used in accomplishing an activity.
- Network diagramming is a critical path scheduling technique used for controlling resources.
- A critical path refers to a sequence of task activities whose order and durations directly affect the completion date of a project.
- A network diagram is one of the most widely used and best-known scheduling methods.

You would use a network diagram when tasks:

- are well defined and have a clear beginning and end point,
- can be worked on independently of other tasks,
- are ordered, and
- serve the purpose of the project



Calculating Expected time durations using PERT:(Program Evaluation Review Technique):

One of the most difficult and most error-prone activities when constructing a project schedule is the determination of the time duration for each task within a work breakdown structure.

It is particularly problematic to make these estimates when there is a high degree of complexity and uncertainty about a task.

PERT is a technique that uses optimistic, pessimistic, and realistic time estimates to calculate the expected time for a particular task.

This technique can help you to obtain a better time estimate when there is some uncertainty as to how much time a task will require to be completed.

The optimistic (o) and pessimistic (p) times reflect the minimum and maximum possible periods of time for an activity to be completed.

The realistic (r) time, or most likely time, reflects the project manager's "best guess" of the amount of time the activity actually will require for completion.

Once each of these estimates is made for an activity, an expected time (ET) can be calculated.

Because the expected completion time should be closest to the realistic (r) time, it is typically weighted four times more than the optimistic (o) and pessimistic (p) times.

Once you add these values together, it must be divided by six to determine the ET. This equation is shown in the following formula:

$$ET = (o+4r+p)/6$$

Where

- ET = expected time for the completion for an activity
- o= optimistic completion time for an activity
- r = realistic completion time for an activity
- p = pessimistic completion time for an activity

3. Using project management software

- A wide variety of automated project management tools is available to help you manage a development project.
- New versions of these tools are continuously being developed and released by software vendors.
- Most of the available tools have a set of common features that include the ability to define and order tasks, assign resources to tasks, and easily modify tasks and resources.
- Project management tools are available to run on IBM-compatible personal computers, the Macintosh, and larger mainframe and workstation-based systems.
- These systems vary in the number of task activities supported, the complexity of relationships, system processing and storage requirements, and, of course, cost.
- For example, numerous shareware project management programs (e.g., OpenProj, Bugzilla, and eGroupWare) can be downloaded from the web

Functions:

- **Plan Projects** – Easily prepare projects while taking previous experience into account.
- **Keeping Track of Completion, Time and Cost of the Project** – Alert the right people when things go off course.
- **Manage Time and Schedules** – Manage time on work items and take people's work schedules into consideration.
- **The Allocation of Resources** – Ensuring that the right people work on the right things at the right time.
- **Estimated Project Budget, Including Personnel Costs** – Ensuring real-time monitoring of not only the time but also the budget as well.
- **Collaboration and Communication** – Post comments and concerns, communicate with external stakeholders while keeping a complete historical record of all actions.
- **Files & Documentation** – Easily document requirements and specifications directly or via a file.
- **Simple to Use** – The software enables users to complete tasks, and it's simple to use.

Chapter 2: Planning

A. System Development Projects: Identification and Selection

1. Introduction:

- The scope of information systems today is the whole enterprise. Managers, knowledge workers, and all other organizational members expect to easily access and retrieve information, regardless of its location.
- Non-integrated systems used in the past (islands of information) are being replaced with cooperative, integrated enterprise systems that can easily support information sharing.
- The clear direction for information systems development is building bridges between these islands.
- Obtaining integrated enterprise-wide computing presents significant challenges for both corporate and information system management.

2. Identifying and Selecting Systems Development Projects.

Committee to identify and assess all possible systems development projects that an organization unit could undertake. This committee may be consists of:

- Senior manager.
- Business group.
- IS manager.

After identification, they should know: buy a ready-made (on-shelf), or build from scratch. (Depending in many factors-- always, it depends.)

Project identification and selection consists of three primary activities:

- Identifying potential development projects
- Classifying and ranking IS development projects
- Selecting IS development project

Identifying potential development projects:

Organizations vary as to how they identify projects.

This process can be performed by a key member of top management, either the CEO of a small- or medium sized organization or a senior executive in a larger organization:

- a steering committee, composed of a cross section of managers with an interest in systems
- user departments, in which either the head of the requesting unit or a committee from the requesting department decides which projects to submit (often you, as a systems analyst, will help users prepare such requests);
- or the development group or a senior IS manager

All methods of identification have been found to have strengths and weaknesses. Research has found, for example, that projects identified by top management more often have a strategic organizational focus. Alternatively, projects identified by steering committees more often reflect the diversity of the committee

and therefore have a cross-functional focus. Projects identified by individual departments or business units most often have a narrow, tactical focus. Finally, a dominant characteristic of projects identified by the development group is the ease with which existing hardware and systems will integrate with the proposed project.

Other factors, such as project cost, duration, complexity, and risk, are also influenced by the source of a given project.

TABLE 4-1 Characteristics of Alternative Methods for Making Information Systems Identification and Selection Decisions

Selection Method	Characteristics
Top Management	Greater strategic focus
	Largest project size
	Longest project duration
	Enterprise-wide consideration
Steering Committee	Cross-functional focus
	Greater organizational change
	Formal cost–benefit analysis
	Larger and riskier projects
Functional Area	Narrow, nonstrategic focus
	Faster development
	Fewer users, management layers, and business functions involved
Development Group	Integration with existing systems focus
	Fewer development delays
	Less concern with cost–benefit analysis

Classifying and ranking IS development projects:

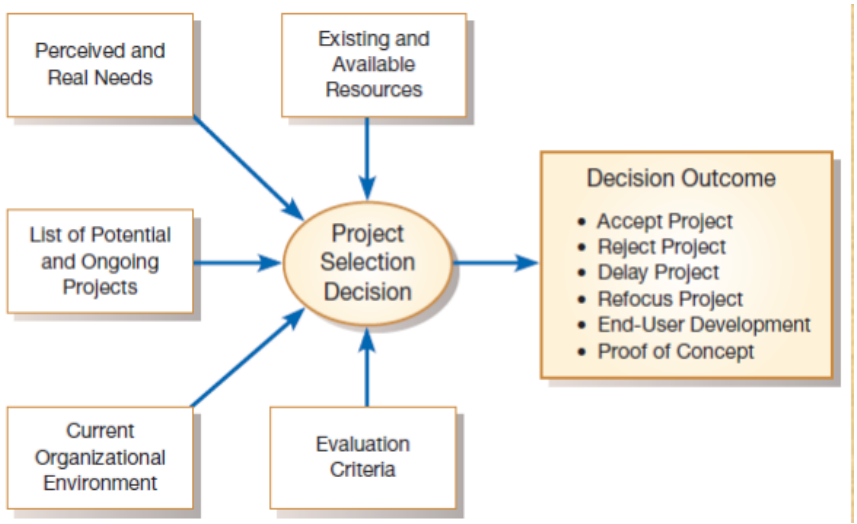
- The second major activity in the project identification and selection process focuses on assessing (to judge or decide the amount, value, quality or importance of something) the relative merit of potential projects.
- As with the project identification process, classifying and ranking projects can be performed by top managers, a steering committee, business units, or the IS development group.
- Additionally, the criteria used when assigning the relative merit of a given project can vary.

Selecting IS development projects:

- The final activity in the project identification and selection process is the actual selection of projects for further development.
- Project selection is a process of considering both short- and long-term projects and selecting those most likely to achieve business objectives.

- Additionally, as business conditions change over time, the relative importance of any single project may substantially change.
- Thus, the identification and selection of projects is a very important and ongoing activity.

FIGURE 4-3
Project selection decisions must consider numerous factors and can have numerous outcomes



- Figure shows that a selection decision requires that the perceived needs of the organization, existing systems and ongoing projects, resource availability, evaluation criteria, current business conditions, and the perspectives of the decision makers will all play a role in project selection decisions.
- Numerous outcomes can occur from this decision process. Of course, projects can be accepted or rejected. Acceptance of a project usually means that funding to conduct the next phase of the SDLC has been approved.
- Rejection means that the project will no longer be considered for development.
- However, projects may also be conditionally accepted; they may be accepted pending the approval or availability of needed resources or the demonstration that a particularly difficult aspect of the system can be developed.
- Projects may also be returned to the original requesters, who are told to develop or purchase the requested system.
- Finally, the requesters of a project may be asked to modify and resubmit their request after making suggested changes or clarifications.

3. Corporate and Information Systems Planning

- Organizations have not traditionally used a systematic planning process when determining how to allocate IS resources. Instead, projects have often resulted from attempts to solve isolated organizational problems. In effect, organizations have asked the question: “What procedure (application program) is required to solve this particular problem as it exists today?”
- The difficulty with this approach is that the required organizational procedures are likely to change over time as the environment changes.
- For example, a company may decide to change its method of billing customers or a university may change its procedure for registering students. When such changes occur, it is usually necessary to again modify existing information systems.

- In contrast, planning-based approaches essentially ask the question: "What information (or data) requirements will satisfy the decisionmaking needs or business processes of the enterprise today and well into the future?"
- A major advantage of this approach is that an organization's informational needs are less likely to change (or will change more slowly) than its business processes. For example, unless an organization fundamentally changes its business, its underlying data structures may remain reasonably stable for more than 10 years. However, the procedures used to access and process the data may change many times during that period. Thus, the challenge of most organizations is to design comprehensive information models containing data that are relatively independent from the languages and programs used to access, create, and update them.

The need for improved information systems project identification and selection is seen when we consider factors such as the following:

a. Corporate Strategic Planning:

A prerequisite for making effective project selection decisions is to gain a clear idea of where an organization is, its vision of where it wants to be in the future, and how to make the transition to its desired future state. Figure represents this as a three-step process.

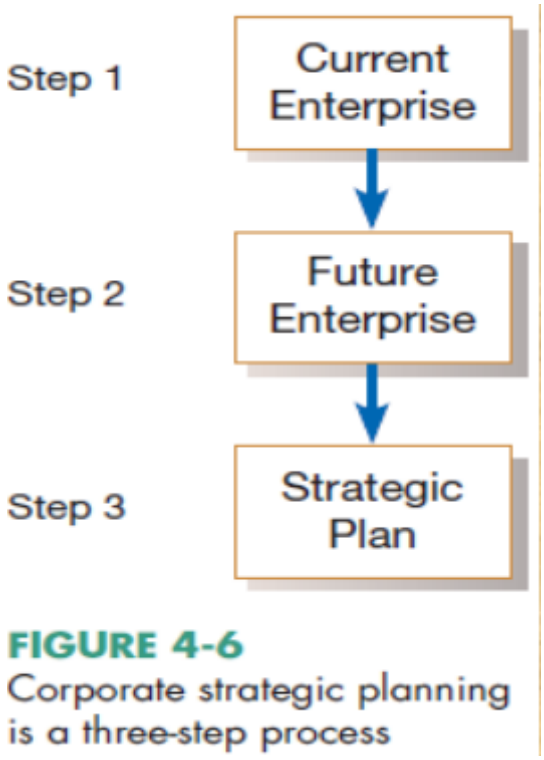


FIGURE 4-6
Corporate strategic planning is a three-step process

- The first step focuses on gaining an understanding of the current enterprise. In other words, if you don't know where you are, it is impossible to tell where you are going.
- Next, top management must determine where it wants the enterprise to be in the future.
- Finally, after gaining an understanding of the current and future enterprise, a strategic plan can be developed to guide this transition.

The process of developing and refining models of the current and future enterprise as well as a transition strategy is often referred to as corporate strategic planning. During corporate strategic planning, executives typically develop a mission statement, statements of future corporate objectives, and strategies designed to help the organization reach its objectives.

All successful organizations have a mission. The mission statement of a company typically states in very simple terms what business the company is in.

For example:

After reviewing PVF's mission statement, it becomes clear that it is in the business of constructing and selling high-quality wood furniture to the general public, businesses, and institutions such as universities and hospitals.

After defining its mission, an organization can then define its **objectives:**

- Objective statements refer to "broad and timeless" goals for the organization. These goals can be expressed as a series of statements that are either qualitative or quantitative but that typically do not contain details likely to change substantially over time.

Objectives are often referred to as critical success factors. Here, we will simply use the term objectives. The objectives for PVF are shown in Figure 4-8, with most relating to some aspect of the organizational mission.

For example, the second objective relates to how PVF views its relationships with customers. This goal would suggest that PVF might want to invest in a web-based order tracking system that would contribute to high-quality customer service. Once a company has defined its mission and objectives, a **competitive strategy can be formulated**.

Pine Valley Furniture
Statement of Objectives

1. PVF will strive to increase market share and profitability (prime objective).
2. PVF will be considered a market leader in customer service.
3. PVF will be innovative in the use of technology to help bring new products to market faster than our competition.
4. PVF will employ the fewest number of the highest-quality people necessary to accomplish our prime objective.
5. PVF will create an environment that values diversity in gender, race, values, and culture among employees, suppliers, and customers.

- **Strive** (to try very hard to do something or to make something happen, especially for a long time or against difficulties)

A competitive strategy is the method by which an organization attempts to achieve **its mission and objectives.**

Michael Porter (1980) defined three generic strategies

- low-cost producer,
- product differentiation, and
- product focus or niche—for reaching corporate objectives

TABLE 4-3 Generic Competitive Strategies

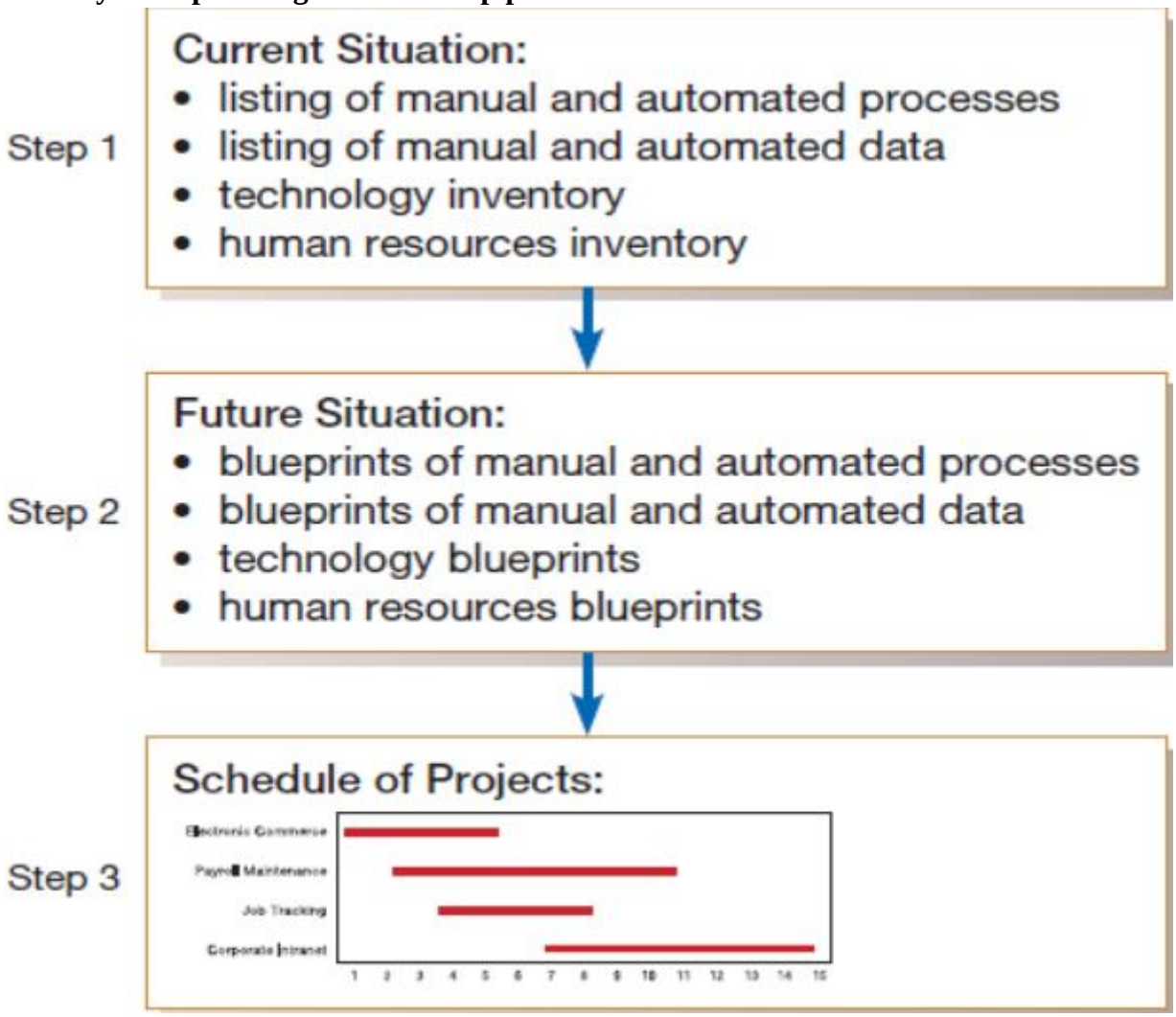
Strategy	Description
Low-Cost Producer	This strategy reflects competing in an industry on the basis of product or service cost to the consumer. For example, in the automobile industry, the South Korean–produced Hyundai is a product line that competes on the basis of low cost.
Product Differentiation	This competitive strategy reflects capitalizing on a key product criterion requested by the market (for example, high quality, style, performance, roominess). In the automobile industry, many manufacturers are trying to differentiate their products on the basis of quality (e.g., "At Ford, quality is job one.").
Product Focus or Niche	This strategy is similar to both the low-cost and differentiation strategies but with a much narrower market focus. For example, a niche market in the automobile industry is the convertible sports car market. Within this market, some manufacturers may employ a low-cost strategy and others may employ a differentiation strategy based on performance or style.

niche: a job or position which is very suitable for someone, especially one that they like or an area or position which is exactly suitable for a small group of the same type.

b. Information Systems Planning

- The second planning process that can play a significant role in the quality of project identification and selection decisions is called information systems planning (ISP).
- ISP is an orderly means of assessing the information needs of an organization and defining the information systems, databases, and technologies that will best satisfy those needs.
- This means that during ISP you (or, more likely, senior IS managers responsible for the IS plan) must model current and future organization informational needs and develop strategies and project plans to migrate the current information systems and technologies to their desired future state. ISP is a **top-down process**.

Information systems planning is a three-step process:

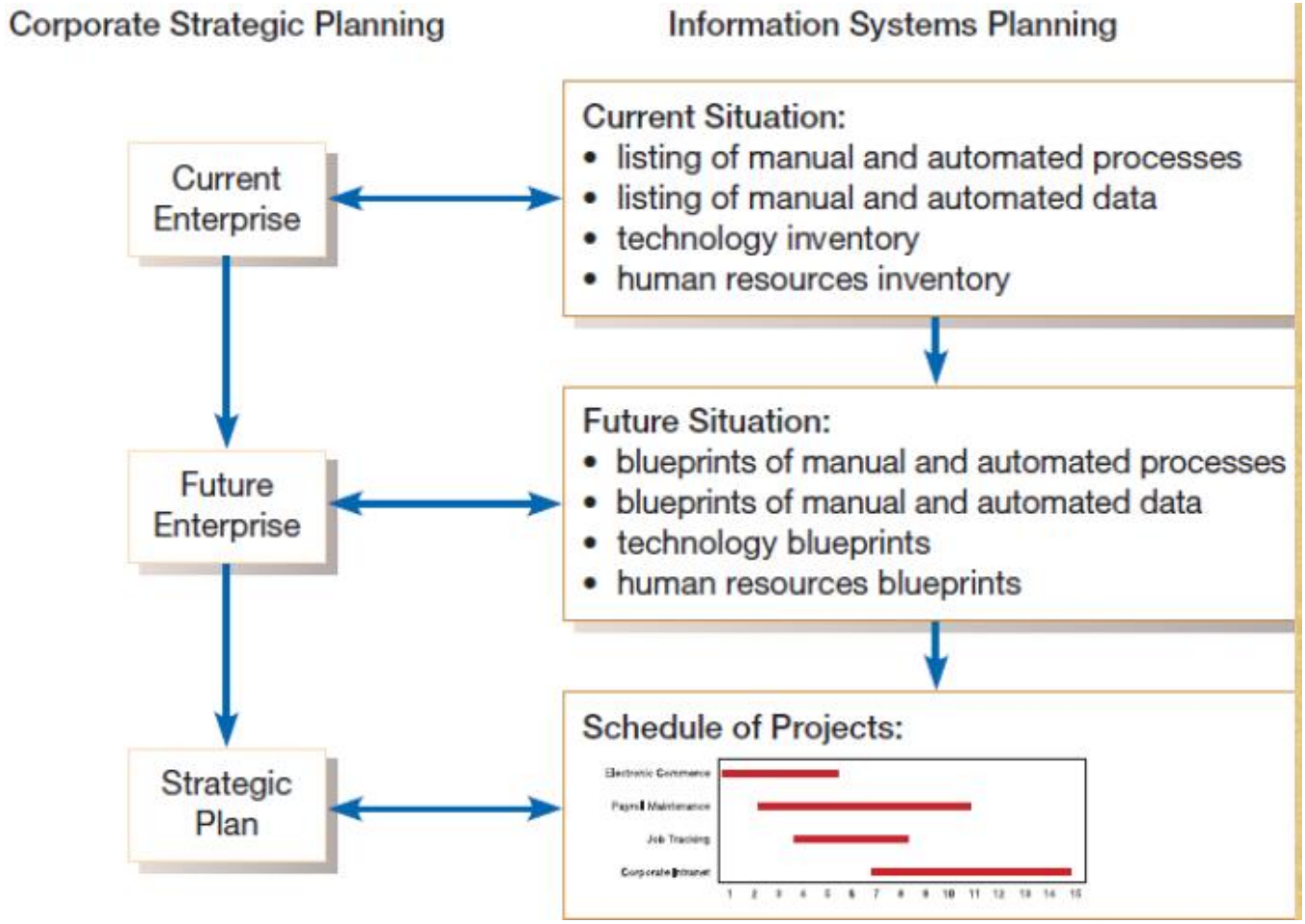


The **three key activities** of this modeling process are represented in Figure:

- Like corporate strategic planning, ISP is a three-step process in which the first step is to assess current IS-related assets—human resources, data, processes, and technologies.
- Next, target blueprints of these resources are developed. These blueprints reflect the desired future state of resources needed by the organization to reach its objectives as defined during strategic planning.

- Finally, a series of scheduled projects is defined to help move the organization from its current to its future desired state.

Parallel activities of corporate strategic planning and information system planning:



i. Describe the current situation:

The most widely used approach for describing the current organizational situation is generically referred to as top-down planning.

Top-down planning attempts to gain a broad understanding of the informational needs of the entire organization. The approach begins by conducting an extensive analysis of the organization's mission, objectives, and strategy and determining the information requirements needed to meet each objective.

Advantages of to-down planning:

- Broader perspective
- Improved integration
- Improved management support
- Better understanding

In contrast to the top-down planning approach, a bottom-up planning approach requires the identification of business problems and opportunities that are used to define projects.

Using the bottom-up approach for creating IS plans can be faster and less costly than using the top-down approach; it also has the advantage of identifying pressing organizational problems.

ii. Describing the target situation, trends, and constraints:

After describing the current situation, the next step in the ISP (information system planning) process is to define the target situation that reflects the desired future state of the organization.

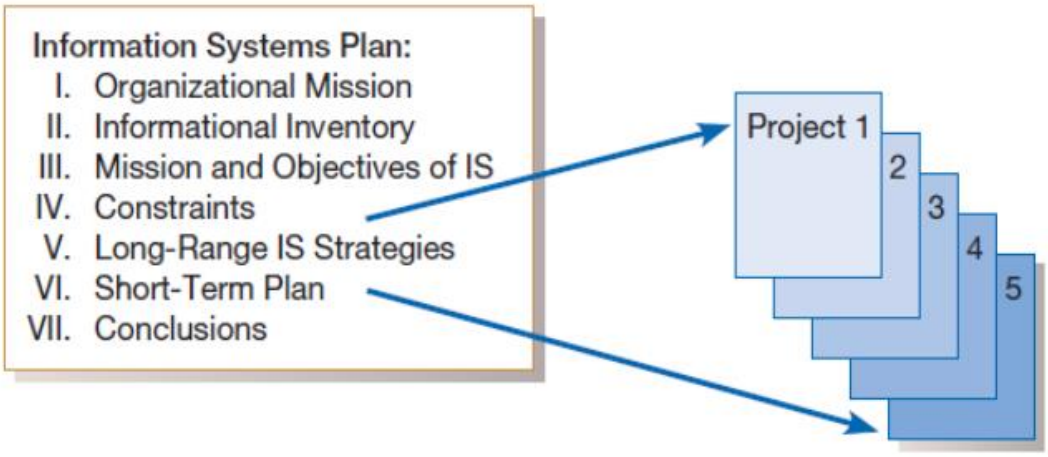
This means that the target situation consists of the desired state of the locations, units, functions, processes, data, and IS.

- **For example**, if a desired future state of the organization is to have several new branch offices or a new product line that requires several new employee positions, functions, processes, and data, then most lists and matrices will need to be updated to reflect this vision. The target situation must be developed in light of technology and business trends, in addition to organizational constraints (something which controls what you do by keeping you within particular limits).

iii. Developing a transition strategy and plans:

- Once the creation of the current and target situations is complete, a detailed transition strategy and plan are developed by the IS planning team.
- This plan should be very comprehensive, reflecting broad, longrange issues in addition to providing sufficient detail to guide all levels of management concerning what needs to be done, how, when, and by whom in the organization.
- The IS plan is typically a very comprehensive document that looks at both short- and long-term organizational development needs. The short- and long-term developmental needs identified in the plan are typically expressed as a series of projects (see Figure).
- Projects from the long-term plan tend to build a foundation for later projects (such as transforming databases from old technology into newer technology).
- Projects from the short-term plan consist of specific steps to fill the gap between current and desired systems or respond to dynamic business conditions.
- The top-down (or plan-driven) projects join a set of bottom-up or needs driven projects submitted as system service requests from managers to form the short-term systems development plan.

Systems development projects flow from the information systems plan:



B. System Development Projects: Initiation and Planning

1. Introduction:

- The first, project identification and selection, focuses on the activities during which the need for a new or enhanced system is recognized. Thus, project identification and selection is often thought of as a "pre project" step in the life cycle.
- The next step is to conduct a more detailed assessment during project initiating and planning.
- This assessment does not focus on how the proposed system will operate but rather on understanding the scope of a proposed project and its feasibility of completion given the available resources.
- In the next section, the project initiation and planning process is briefly reviewed.
- Numerous techniques for assessing project feasibility are then described.

2. Initiating and Planning Systems Development Projects

A key consideration when conducting project initiation and planning (PIP) is deciding when PIP ends and when analysis, the next phase of the SDLC, begins.

This is a concern because many activities performed during PIP could also be completed during analysis.

Pressman (2014) speaks of three important questions that must be considered when making this decision on the division between PIP and analysis:

1 **How much effort should be expended on the project initiation and planning process?**
2 **Who is responsible for performing the project initiation and planning process?**
3 **Why is project initiation and planning such a challenging activity?**

- Finding an answer to the **first question**, how much effort should be expended on the PIP process, is often difficult. Practical experience has found, however, that the time and effort spent on initiation and planning activities easily pay for themselves later in the project. Proper and insightful project planning, including determining project scope as well as identifying project activities, can easily

reduce time in later project phases. A rule of thumb is that between 10 and 20 percent of the entire development effort should be expended on the PIP study. Thus, you should not be reluctant to spend considerable time in PIP in order to fully understand the motivation for the requested system.

- For the **second question**, who is responsible for performing PIP, most organizations assign an experienced systems analyst, or a team of analysts for large projects, to perform PIP. The analyst will work with the proposed customers (managers and users) of the system and other technical development staff in preparing the final plan. Experienced analysts working with customers who fully understand their information services needs should be able to perform PIP without the detailed analysis typical of the analysis phase of the life cycle.
- Less-experienced analysts with customers who only vaguely (not enough) understand their needs will likely expend more effort during PIP in order to be certain that the project scope and work plan are feasible.
- As to the **third question**, PIP is viewed as a challenging activity because the objective of the PIP study is to transform a vague system request document into a tangible project description. This is an open-ended process. Getting all parties to agree on the direction of a project may be difficult for cross-department projects where different parties have different business objectives. Thus, more complex organizational settings for projects will result in more time required for analysis of the current and proposed systems during PIP.

3. The Process of Initiating and Planning IS Development Projects

Project Initiation Process

- Project initiation focuses on activities designed to assist in organizing a team to conduct project planning. During initiation, one or more analysts are assigned to work with a customer—that is, a member of the business group that requested or will be affected by the project—to establish work standards and communication procedures.
- Depending upon the size, scope, and complexity of the project, some project initiation activities may be unnecessary or may be very involved. Also, many organizations have established procedures for assisting with common initiation activities.
- For E.g.

TABLE 5-1 Elements of Project Initiation

- Establishing the Project Initiation Team
- Establishing a Relationship with the Customer
- Establishing the Project Initiation Plan
- Establishing Management Procedures
- Establishing the Project Management Environment and Project Workbook
- Developing the Project Charter

Project Planning Process:

- Project planning is the process of defining clear, discrete activities and the work needed to complete each activity within a single project.
- The objective of the project planning process is the development of a Baseline Project Plan (BPP) and the Project Scope Statement (PSS).
- The **BPP** becomes the foundation for the remainder of the development project. The **PSS** produced by the team clearly outlines the objectives and constraints of the project for the customer.
- As with the project initiation process, the size, scope, and complexity of a project will dictate the comprehensiveness of the project planning process and resulting documents.
- Further, numerous assumptions about resource availability and potential problems will have to be made. Analysis of these assumptions and system costs and benefits forms a business case.
- The range of activities performed during project planning is shown in figure:

TABLE 5-2 Elements of Project Planning

- Describing the Project Scope, Alternatives, and Feasibility
- Dividing the Project into Manageable Tasks
- Estimating Resources and Creating a Resource Plan
- Developing a Preliminary Schedule
- Developing a Communication Plan
- Determining Project Standards and Procedures
- Identifying and Assessing Risk
- Creating a Preliminary Budget
- Developing the Project Scope Statement
- Setting a Baseline Project Plan

4. Assessing Project Feasibility

Unfortunately, most projects must be developed within tight budgetary and time constraints.

This means that assessing project feasibility is a required activity for all information systems projects and is a potentially large undertaking.

It requires that you, as a systems analyst, evaluate a wide range of factors. Typically, the relative importance of these factors will vary from project to project.

Most feasibility factors are represented by the following categories:

- **Economic**
- **Technical**
- **Operational**
- **Scheduling**
- **Legal and contractual**
- **Political**

A. Assessing Economic Feasibility:

The purpose of assessing economic feasibility is:

- To identify the financial benefits and costs associated with the development project.
- Often referred to as cost–benefit analysis.

During project initiation and planning, it will be impossible for you to precisely define all benefits and costs related to a particular project.

Yet it is important that you spend adequate time identifying and quantifying these items or it will be impossible for you to conduct an adequate economic analysis and make meaningful comparisons between rival projects.

- Here we will describe typical benefits and costs resulting from the development of an information system and provide several useful worksheets for recording costs and benefits.
- Additionally, several common techniques for making cost–benefit calculations are presented.
- These worksheets and techniques are used after each SDLC phase as the project is reviewed in order to decide whether to continue, redirect, or kill a project.

How to analysis Economic feasibility?

a. Determining Project Benefits:

- An information system can provide many benefits to an organization.
- **For example**, a new or renovated information system can automate monotonous jobs and reduce errors; provide innovative services to customers and suppliers; and improve organizational efficiency, speed, flexibility, and morale.
- In general, the benefits can be viewed as being both **tangible** and **intangible**.

Tangible:

- Tangible benefits refer to items that can be measured in dollars and with certainty.
- **Examples of tangible** benefits might include reduced personnel expenses, lower transaction costs, or higher profit margins.
- It is important to note that not all tangible benefits can be easily quantified (to measure something in amount). **For example**, a tangible benefit that allows a company to perform a task in 50 percent of the time may be difficult to quantify in terms of hard dollar savings.

Most tangible benefits will fit within the following categories:

- Cost reduction and avoidance
- Error reduction
- Increased flexibility
- Increased speed of activity
- Improvement of management planning and control
- Opening new markets and increasing sales opportunities

Intangible:

Intangible benefits of the system could not be quantified.

Intangible benefits refer to items that cannot be easily measured in dollars or with certainty.

Intangible benefits may have direct organizational benefits, such as the improvement of employee morale, or they may have broader societal implications, such as the reduction of waste creation or resource consumption.

Actual benefits will vary from system to system. After determining project benefits, project costs must be identified.

Below Table lists numerous intangible benefits often associated with the development of an information system.

TABLE 5-3 Intangible Benefits from the Development of an Information System

• Competitive necessity	• More confidence in decision quality
• More timely information	• Improved processing efficiency
• Improved organizational planning	• Improved asset utilization
• Increased organizational flexibility	• Improved resource control
• Promotion of organizational learning and understanding	• Increased accuracy in clerical operations
• Availability of new, better, or more information	• Improved work process that can improve employee morale or customer satisfaction
• Ability to investigate more alternatives	• Positive impacts on society
• Faster decision making	• Improved social responsibility
	• Better usage of resources ("greener")

(Source: Based on Parker and Benson, 1988; Brynjolfsson and Yang, 1997; Keen, 2003; Cresswell, 2004.)

b. Determining Project Costs:

Similar to benefits, an information system can have both **tangible and intangible** costs.

- **Tangible** costs refer to items that you can easily measure in dollars and with certainty. From an IS development perspective, tangible costs include items such as hardware costs, labor costs, and operational costs including employee training and building renovations.
- Alternatively, **intangible** costs are items that you cannot easily measure in terms of dollars or with certainty. Intangible costs can include loss of customer goodwill, employee morale, or operational inefficiency.
- One goal of a cost–benefit analysis is to accurately determine the total cost of ownership (TCO) for an investment. TCO is focused on understanding not only the total cost of acquisition but also all costs associated with ongoing use and maintenance of a system.
- Consequently, besides tangible and intangible costs, you can distinguish IS-related development costs as either one-time or recurring.
- **One-time costs** refer to those associated with project initiation and development and the start-up of the system. These costs typically encompass activities such as systems development, new hardware

and software purchases, user training, site preparation, and data or system conversion. When conducting an economic cost–benefit analysis, a worksheet should be created for capturing these expenses. For very large projects, one-time costs may be staged over one or more years. In these cases, a separate onetime cost worksheet should be created for each year. This separation will make it easier to perform present value calculations.

- Recurring costs refer to those costs resulting from the ongoing evolution and use of the system. Examples of these costs typically include the following:
  - Application software maintenance
  - Incremental data storage expenses
  - Incremental communications
  - New software and hardware leases
  - Supplies and other expenses (e.g., paper, forms, data center personnel)

- Both **one-time and recurring costs** can consist of items that are fixed or variable in nature. Fixed costs are costs that are billed or incurred at a regular interval and usually at a fixed rate (a facility lease payment). Variable costs are items that vary in relation to usage (long-distance phone charges).

c. The Time Value of Money:

- Most techniques used to determine economic feasibility encompass the concept of the time value of money **(TVM)**, which reflects the notion that money available today is worth more than the same amount tomorrow. As previously discussed, the development of an information system has both one-time and recurring costs. Furthermore, benefits from systems development will likely occur sometime in the future.
- Because many projects may be competing for the same investment dollars and may have different useful life expectancies, all costs and benefits must be viewed in relation to their present value when comparing investment options.

ONE-TIME COSTS WORKSHEET
Customer Tracking System Project

	Year 0
A. Development costs	$20,000
B. New hardware	15,000
C. New (purchased) software, if any	
1. Packaged applications software	5,000
2. Other _____	0
D. User training	2,500
E. Site preparation	0
F. Other _____	0
TOTAL one-time costs	$42,500

RECURRING COSTS WORKSHEET Customer Tracking System Project	Year 1 through 5
A. Application software maintenance	$25,000
B. Incremental data storage required: 20 GB $50 (estimated cost/GB = $50)	1000
C. Incremental communications (lines, messages, . . .)	2000
D. New software or hardware leases	0
E. Supplies	500
F. Other _____	0
TOTAL recurring costs	$28,500

Example:

Most of us would gladly accept $4500 today rather than three payments of $1500, because a dollar today (or $4500 for that matter) is worth more than a dollar tomorrow or next year, given that money can be invested. The rate at which money can be borrowed or invested is referred to as the cost of capital, and is called the discount rate for TVM calculations. Let's suppose that the seller could put the money received for the sale of the car in the bank and receive a 10 percent return on her investment. A simple formula can be used when figuring out the present value of the three $1500 payments:

$$PV_n = Y \times \frac{1}{(1 + i)^n}$$

where PV^n is the present value of Y dollars n years from now when i is the discount rate.

From our example, the present value of the three payments of $1500 can be calculated as

$$PV_1 = 1500 \times \frac{1}{(1 + .10)^1} = 1500 \times .9091 = 1363.65$$

$$PV_2 = 1500 \times \frac{1}{(1 + .10)^2} = 1500 \times .8264 = 1239.60$$

$$PV_3 = 1500 \times \frac{1}{(1 + .10)^3} = 1500 \times .7513 = 1126.95$$

where PV_1, PV_2, and PV_3 reflect the present value of each $1500 payment in years 1, 2, and 3, respectively.

To calculate the *net present value (NPV)* of the three $1500 payments, simply add the present values calculated previously (NPV = PV_1 = PV_2 = PV_3 = 1363.65 = 1239.60 = 1126.95 = $3730.20). In other words, the seller could accept a lump-sum payment of $3730.20 as equivalent to the three payments of $1500, given a discount rate of 10 percent.

Given that we now know the relationship between time and money, the next step in performing the economic analysis is to create a summary worksheet reflecting the present values of all benefits and costs as well as all pertinent analyses. Due to the fast pace of the business world, PVF's System Priority Board feels that the useful life

Example 2:

Imagine a project that costs $1000 and will provide three cash flow of $500, $300 and $800 over the next 3 years. Assume there is no salvage value at the end of project and the required rate of return is 8%. Find the NPV of project?

$$NPV = \frac{-\$1000}{(1+0.08)^0} + \frac{\$500}{(1+0.08)^1} + \frac{\$300}{(1+0.08)^2} + \frac{\$800}{(1+0.08)^3}$$
$$= \$355.23$$

d. FORMS OF COCOMO (Constructive Cost MOdel) MODELS:

Basic model:

The basic COCOMO model takes the following form:

E=ab (KLOC)bb persons-months

D=cb (E)db months

Where,

E - Stands for the effort applied in terms of person months

D - Development time in chronological months

KLOC - Kilo lines of code of the project.

From E & D we can compute the no. of people required to accomplish the project as: N=E/D

Example:

The coefficients of A_a, b_b, c_c, d_d for the three Modes are:

	A_a	b_b	c_c	d_d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Example: Consider your team is developing a software project using semi-detached mode with 30,000 lines of code . We will obtain estimation for this project as follows:

(1)Effort estimation

$E= a (KLOC)^b$ person-months

$E=3.0(30)^{1.12}$ = 135.36 (**136**)

(2) Duration estimation

$D=c (E)^d$ months

$D=2.5 (136)^{0.35}$ = 13.95 (**14**)

(3)Person estimation

N=E/D persons

N =136/14 = 9.7 (**10**)

Intermediate model:

Intermediate COCOMO takes the form.

--$E = ai\ (KLOC)^{bi} \times EAF$

Where

E - Effort applied in terms of person-months

KLOC - Kilo lines of code for the project

EAF - It is the effort adjustment factor

--The duration and person estimate is same as in basic COCOMO model i.e.; $D = c_b\ (E)^{db}$ months i.e.; use values of c_b and d_b coefficients.

--N=E/D persons

THE VALUES OF AI AND BI FOR VARIOUS CLASS

Software Projects	A_i	B_i
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Example: Consider your team is developing a project having 30,000 lines of code which in an embedded software with critical area hence reliability is high .

The estimation can be

$E = a\ (KLOC)^{bi} \times EAF$

As reliability is high

EAF=1.15 (product attribute)

$a_i = 2.8$

$b_i = 1.20$ for embedded software

$E = 2.8\ (30)^{1.20} \times 1.15$

$= 191$ person month

$D = c_b\ (E)^{db}$

$= 2.5(191)^{0.32}$

$= 13$ months approximately

$N = E/D$

$= 191/13$

$N = 15$ persons approx.

Typical values for EAF range from 0.9 to 1.4

e. Break-even analysis:

The objective of the break-even analysis is to discover at what point (if ever) benefits equal costs (i.e., when breakeven occurs). To conduct this analysis, the NPV of the yearly cash flows are determined.

Here, the yearly cash flows are calculated by subtracting both the one-time cost and the present values of the recurring costs from the present value of the yearly benefits.

The overall NPV of the cash flow reflects the total cash flows for all preceding years.

Examination of line 30 of the worksheet shows that breakeven occurs between years 2 and 3.

Because year 3 is the first in which the overall NPV cash flow figure is non-negative, the identification of what point during the year breakeven occurs can be derived.

$$\text{Break-Even Ratio} = \frac{\text{Yearly NPV Cash Flow} - \text{Overall NPV Cash Flow}}{\text{Yearly NPV Cash Flow}}$$

B. Assessing Technical Feasibility:

The purpose of assessing technical feasibility is to gain an understanding of the organization's ability to construct the proposed system.

This analysis should include an assessment of the development group's understanding of the possible target hardware, software, and operating environments to be used, as well as system size, complexity, and the group's experience with similar systems.

It is important to note that all projects have risk and that risk is not necessarily something to avoid.

Yet it is also true that, because organizations typically expect a greater return on their investment for riskier projects, understanding the sources and types of technical risks proves to be a valuable tool when you assess a project.

Also, risks need to be managed in order to be minimized; you should, therefore, identify potential risks as early as possible in a project. The potential consequences of not assessing and managing risks can include the following:

- Failure to attain expected benefits from the project
- Inaccurate project cost estimates
- Inaccurate project duration estimates
- Failure to achieve adequate system performance levels
- Failure to adequately integrate the new system with existing hardware, software, or organizational procedures.

The amount of technical risk associated with a given project is contingent (or depends) on four primary factors: project size, project structure, the development group's experience with the application and technology area, and the user group's experience with systems development projects and the application area.

Large projects are riskier than small projects.

A system in which the requirements are easily obtained and highly structured will be less risky than one in which requirements are messy, ill-structured, ill defined, or subject to the judgment of an individual

The development of a system employing commonly used or standard technology will be less risky than one employing novel (a long printed story about imaginary characters and events) or nonstandard technology.

A project is less risky when the user group is familiar with the systems development process and application area than if the user group is unfamiliar with them.

C. Assessing operational feasibility:

- Its purpose is to gain an understanding of the degree to which the proposed system will likely solve the business problems or take advantage of the opportunities outlined in the System Service Request or project identification study.
- For a project motivated from information systems planning, operational feasibility includes justifying the project on the basis of being consistent with or necessary for accomplishing the information systems plan.
- Your assessment of operational feasibility should include an analysis of how the proposed system will affect organizational structures and procedures.
- Systems that have substantial and widespread impact on an organization's structure or procedures are typically riskier projects to undertake.
- Thus, it is important for you to have a clear understanding of how an information system will fit into the current day-to-day operations of the organization.

D. Assessing Schedule Feasibility:

Another feasibility concern relates to project duration is assessing schedule feasibility.

The purpose of assessing schedule feasibility is for you, as a systems analyst, to gain an understanding of all potential time frames and completion date schedules can be met and that meeting these dates will be sufficient for dealing with the needs of the organization.

Further, detailed activities may only be feasible if resources are available when called for in the schedule. **For example**:

- The schedule should not call for system testing during rushed business periods or for key project meetings during annual vacation or holiday periods.
- The schedule of activities produced during project initiation and planning will be very precise and detailed for the analysis phase.
- The estimated activities and associated times for activities after the analysis phase are typically not as detailed (e.g., it will take two weeks to program the payroll report module) as the life-cycle-phase level (e.g., it will take six weeks for physical design, four months for programming, and so on).

This means that assessing schedule feasibility during project initiation and planning is more of a "rough-cut" analysis of whether the system can be completed within the constraints of the business opportunity or the desires of the users.

For example:

Factors such as project team size, availability of key personnel, subcontracting or outsourcing activities, and changes in development environments may all be considered as having a possible impact on the eventual schedule. As with all forms of feasibility, schedule feasibility will be reassessed after each phase when you can specify with greater certainty the details of each step for the next phase.

E. Assessing Legal and Contractual Feasibility:

- In this area, you need to gain an understanding of any potential legal ramifications (the possible results of an action) due to the construction of the system.
- Possible considerations might include copyright or nondisclosure infringements (an action that breaks a rule, law, etc.), labor laws, antitrust legislation (which might limit the creation of systems to share data with other organizations), foreign trade regulations (e.g., some countries limit access to employee data by foreign corporations), and financial reporting standards, as well as current or pending contractual obligations.
- Contractual obligations may involve ownership of software used in joint ventures, license agreements for use of hardware or software, nondisclosure agreements with partners, or elements of a labor agreement (e.g., a union agreement may preclude certain compensation or work-monitoring capabilities a user may want in a system).

F. Assessing Political Feasibility:

- A final feasibility concern focuses on assessing political feasibility in which you attempt to gain an understanding of how key stakeholders within the organization view the proposed system.
- Because information system may affect the distribution of information within the organization, and thus the distribution of power, the construction of an information system can have political ramifications.
- Those stakeholders not supporting the project may take steps to block, disrupt, or change the intended focus of the project.

5. Baseline Project Plan and the Project Scope Statement

The major outcomes and deliverables from the project initiation and planning phase are the Baseline Project Plan and the Project Scope Statement (PSS).

Baseline Project Plan (BPP):

- The **Baseline Project Plan (BPP)** contains all information collected and analyzed during project initiation and planning.
- The plan reflects the best estimate of the project's scope, benefits, costs, risks, and resource requirements given the current understanding of the project.
- The BPP specifies detailed project activities for the next life cycle phase— analysis—and less detail for subsequent project phases (because these depend on the results of the analysis phase).
- Similarly, benefits, costs, risks, and resource requirements will become more specific and quantifiable as the project progresses.
- The BPP is used by the project selection committee to decide whether the project should be accepted, redirected, or canceled.
- If selected, the BPP becomes the foundation document for all subsequent SDLC activities; however, it is also expected to evolve (to develop gradually) as the project evolves. That is, as new information is learned during subsequent SDLC phases, the baseline plan will be updated.

The Project Scope Statement (PSS:

- **The Project Scope Statement (PSS**) is a short document prepared for the customer that describes what the project will deliver and outlines all work required to complete the project.
- The PSS ensures that both you and your customer gain a common understanding of the project. It is also a very useful communication tool.
- The PSS is a very easy document to create because it typically consists of a high-level summary of the BPP information.
- Depending upon your relationship with your customer, the role of the PSS may vary.
- At one extreme, the PSS can be used as the basis of a formal contractual agreement outlining firm deadlines, costs, and specifications.
- At the other extreme, the PSS can simply be used as a communication vehicle to outline the current best estimates of what the project will deliver, when it will be completed, and the resources it may consume.
- A contract programming or consulting firm, for example, may establish a very formal relationship with a customer and use a PSS that is extensive and formal.
- Alternatively, an internal development group may develop a PSS that is only one to two pages in length and is intended to inform customers rather than to set contractual obligations and deadlines.

Chapter 3: Analysis

A. System Requirements

1. Introduction:

Systems analysis is the part of the systems development life cycle in which you determine how the current information system functions and assess what users would like to see in a new system.

Analysis has two sub phases: requirements determination and requirements structuring.

Techniques used in requirements determination have evolved over time to become more structured and increasingly rely on computer support.

We will first study the more traditional requirements determination methods, including interviewing, observing users in their work environment, and collecting procedures and other written documents.

We will then discuss more current methods for collecting system requirements.

The first of these methods is Joint Application Design (JAD).

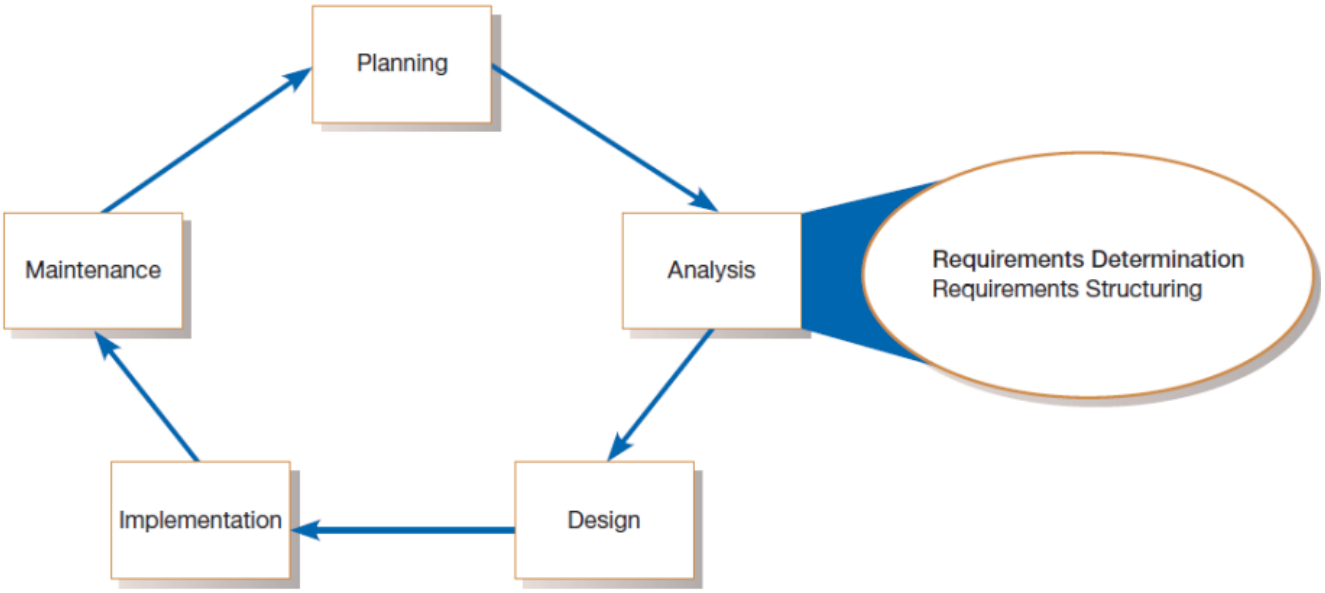
Next, we will read about how analysts rely more and more on information systems to help them perform analysis.

As you will see, CASE tools, discussed in Chapter 1, are useful in requirements determination, and prototyping has become a key tool for some requirements determination efforts.

Finally, you will learn how requirements analysis continues to be an important part of systems analysis and design, whether the approach involves business process redesign, new Agile techniques (such as constant user involvement or usage-centered design), or focuses on developing Internet applications.

2. Performing Requirements Determination

- There are two sub phases to systems analysis: requirements determination and requirements structuring.
- We will address these as separate steps, but you should consider the steps as parallel and iterative.
- **For example**, as we determine some aspects of the current and desired system(s), we begin to structure these requirements or build prototypes to show users how a system might behave. Inconsistencies and deficiencies discovered through structuring and prototyping lead us to explore further the operation of current system(s) and the future needs of the organization. Eventually, our ideas and discoveries converge in a thorough and accurate depiction (the way that something is represented or shown) of current operations and requirements for the new system.



- Once management has granted permission to pursue (follow or start) development of a new system (this was done at the end of the project identification and selection phase of the SDLC) and a project is initiated and planned, you begin determining what the new system should do.
- During requirements determination, you and other analysts gather information on what the system should do from as many sources as possible: from users of the current system; from observing users; and from reports, forms, and procedures.
- All of the system requirements are carefully documented and prepared for structuring.
- In many ways, gathering system requirements is like conducting any investigation.

These characteristics include the following:

a. Impertinence:

You should question everything. You need to ask questions such as: Are all transactions processed the same way? Could anyone be charged something other than the standard price? Might we someday want to allow and encourage employees to work for more than one department?

b. Impartiality:

Your role is to find the best solution to a business problem or opportunity. It is not, for example, to find a way to justify the purchase of new hardware or to insist on incorporating what users think they want into the new system requirements. You must consider issues raised by all parties and try to find the best organizational solution.

c. Relax constraints:

Assume that anything is possible and eliminate the infeasible. For example, do not accept this statement: "We've always done it that way, so we have to continue the practice." Traditions are different from rules and policies. Traditions probably started for a good reason but, as the organization and its environment change, they may turn into habits rather than sensible procedures.

d. Attention to details:

Every fact must fit with every other fact. One element out of place means that even the best system will fail at some time. For example, an imprecise (not accurate or exact) definition of who a customer is may mean that you purge (remove) customer data when a customer has no active orders, yet these past customers may be vital contacts for future sales.

e. Reframing:

Analysis is, in part, a creative process. You must challenge yourself to look at the organization in new ways. You must consider how each user views his or her requirements. You must be careful not to jump to the following conclusion: "I worked on a system like that once—this new system must work the same way as the one I built before."

f. Deliverables and outcomes:

The primary deliverables from requirements determination are the various forms of information gathered during the determination process: transcripts of interviews; notes from observation and analysis of documents; sets of forms, reports, job descriptions, and other documents; and computer-generated output such as system prototypes. In short, anything that the analysis team collects as part of determining system requirements is included in the deliverables resulting from this sub phase of the systems development life cycle. Examples of some specific information that might be gathered during requirements determination.

We could group all this information in three groups:

- Information collected from conversations with or observations of users (interview transcripts, notes from observations etc.)
- Existing written information (business mission or strategy, forms, reports, training manuals, flow charts and documentation of existing system etc.)
- Computer-based information (results from JRP sessions, CASE repository contents)

g. Analysis Paralysis:

Too much analysis is not productive, and the term analysis paralysis has been coined (invented) to describe a systems development project that has become bogged down (to be/become so involved in something difficult or complicated that you cannot do anything else) in an abundance of analysis work. Because of the dangers of excessive analysis, today's systems analysts focus more on the system to be developed than on the current system. The techniques you will learn about later in this chapter, JAD and prototyping, were developed to keep the analysis effort at a minimum yet still keep it effective. Newer techniques have also been developed to keep requirements determination fast and flexible, including continual user involvement, usage centered design, and the Planning Game from eXtreme Programming.

3. Traditional methods for determining requirements

TABLE 6-2 Traditional Methods of Collecting System Requirements

- Individually interview people informed about the operation and issues of the current system and future systems needs
- Interview groups of people with diverse needs to find synergies and contrasts among system requirements
- Observe workers at selected times to see how data are handled and what information people need to do their jobs
- Study business documents to discover reported issues, policies, rules, and directions as well as concrete examples of the use of data and information in the organization

- One of the best ways to get this information is to talk to the people who are directly or indirectly involved in the different parts of the organizations affected by the possible system changes: users, managers, funders, and so on.
- Another way to find out about the current system is to gather copies of documentation relevant to current systems and business processes.

a. Interviewing and listening:

Interviewing is one of the primary ways analysts gather information about an information systems project. Early in a project, an analyst may spend a large amount of time interviewing people about their work, the information they use to do it, and the types of information processing that might supplement their work. Other stakeholders are interviewed to understand organizational direction, policies, expectations managers have on the units they supervise, and other non-routine aspects of organizational operations.

During interviewing you will gather facts, opinions, and speculation(Guess, when you guess possible answers to a question without having enough information to be certain) and observe body language, emotions, and other signs of what people want and how they assess current systems.

There are many ways to effectively interview someone, and no one method is necessarily better than another.

TABLE 6-3 Guidelines for Effective Interviewing

Plan the Interview
- Prepare interviewee: appointment, priming questions
- Prepare checklist, agenda, and questions

Listen carefully and take notes (record if permitted)

Review notes within 48 hours of interview

Be neutral

Seek diverse views

Open-ended (no pre-specified answer)

- Open-ended questions are usually used to probe for information for which you cannot anticipate all possible responses or for which you do not know the precise question to ask.
- The person being interviewed is encouraged to talk about whatever interests him or her within the general bounds of the question.
- An example is, "What would you say is the best thing about the information system you currently use to do your job?" or "List the three most frequently used menu options." You must react quickly to answers and determine whether or not any follow-up questions are needed for clarification or elaboration.
- A major disadvantage of openended questions is the length of time it can take for the questions to be answered. In addition, open-ended questions can be difficult to summarize.

Closed-ended

- Closed-ended questions provide a range of answers from which the interviewee may choose.
- Closed-ended questions work well when the major answers to questions are well known.
- Another plus is that interviews based on closed-ended questions do not necessarily require a large time commitment—more topics can be covered.
- You can see body language and hear voice tone, which can aid in interpreting the interviewee's responses. Closed-ended questions can also be an easy way to begin an interview and to determine which line of open-ended questions to pursue.
- Closed-ended questions, like objective questions on an examination, can follow several forms, including the following choices: True or false or MCQ.

Interview Guidelines:

- **First,** with either open- or closed-ended questions, do not phrase a question in a way that implies a right or wrong answer. The respondent must feel that he or she can put his or her true opinion and perspective and that his or her idea will be considered equally with those of others.
- **The second** guideline to remember about interviews is to listen very carefully to what is being said. Take careful notes or, if possible, record the interview (be sure to ask permission first!).

- **Third,** once the interview is over, go back to your office and type up your notes within 48 hours. If you recorded the interview, use the recording to verify the material in your notes. After 48 hours, your memory of the interview will fade quickly.
- **Fourth,** be careful during the interview not to set expectations about the new or replacement system unless you are sure these features will be part of the delivered system.
- **Fifth,** seek a variety of perspectives from the interviews. Find out what potential users of the system, users of other systems that might be affected by changes, managers and superiors, information systems staff who have experience with the current system, and others think the current problems and opportunities are and what new information services might better serve the organization.

b. Interviewing groups:

- In a group interview, several key people are interviewed at once. To make sure all of the important information is collected, you may conduct the interview with one or more analysts. In the case of multiple interviewers, one analyst may ask questions while another takes notes, or different analysts might concentrate on different kinds of information.
- For example, one analyst may listen for data requirements while another notes the timing and triggering of key events. The number of interviewees involved in the process may range from two to however many you believe can be comfortably accommodated.
- A group interview has a few advantages. One, it is a much more effective use of your time than a series of interviews with individuals (although the time commitment of the interviewees may be more of a concern). Two, interviewing several people together allows them to hear the opinions of other key people and gives them the opportunity to agree or disagree with their peers.
- The primary disadvantage of a group interview is the difficulty in scheduling it.

c. Directly observing users:

Employees who know they are being observed may be nervous and make more mistakes than normal, may be careful to follow exact procedures they do not typically follow, and may work faster or slower than normal. Moreover, because observation typically cannot be continuous, you receive only a snapshot image of the person or task you observe, which may not include important events or activities.

Because observation is very time consuming, you will not only observe for a limited time, but also a limited number of people and a limited number of sites. Again, observation yields only a small segment of data from a possibly vast variety of data sources. Exactly which people or sites to observe is a difficult selection problem. You want to pick both typical and atypical people and sites, and observe during normal and abnormal conditions and times to receive the richest possible data from observation.

d. Analyzing Procedures and Other Documents

By examining existing system and organizational documentation, system analyst can find out details about current system and the organization these systems supports.

In documents analyst can find information such as problem with existing systems, opportunities to meet new needs if only certain information or information processing were available, organizational direction that can influence information system requirements, and the reason why current systems are designed as they are, etc.

What we get from documents:

- Problems with existing system
- Opportunity to meet new need
- Organize direction
- Reasons for current system design
- Rules for processing data

4. Contemporary Methods for determining system Requirements

Even though we called interviews, observation, and document analysis traditional methods for determining a system's requirements, all of these methods are still very much used by analysts to collect important information. Today, however, there are additional techniques to collect information about the current system, the organizational area requesting the new system, and what the new system should be like.

In this section, you will learn about several contemporary information-gathering techniques for analysis: **JAD, CASE tools to support JAD, and prototyping.**

As we said earlier, these techniques can support effective information collection and structuring while reducing the amount of time required for analysis.

TABLE 6-5 Contemporary Methods for Collecting System Requirements

- Bringing session users, sponsors, analysts, and others together in a JAD session to discuss and review system requirements
- Using *CASE tools* during a JAD to analyze current systems to discover requirements that will meet changing business conditions
- Iteratively developing system *prototypes* that refine the understanding of system requirements in concrete terms by showing working versions of system features

a. Joint Application Design (JAD):

A structured process in which users, managers and analysts work together for several days in a series of intensive meetings to specify or review system requirements.

Joint Application Design (JAD) started in the late 1970s at IBM, and since then the practice of JAD has spread throughout many companies and industries.

The main idea behind JAD is to bring together the **key users, managers, and systems analysts** involved in the analysis of a current system.

The primary purpose of using JAD in the analysis phase is to collect systems requirements simultaneously from the key people involved with the system.

The result is an intense (extreme and forceful or (of a feeling) very strong) and structured, but highly effective, process. As with a group interview, having all the key people together in one place at one time allows analysts to see where there are areas of agreement and where there are conflicts.

Meeting with all of these important people for over a week of intense sessions allows you the opportunity to resolve conflicts, or at least to understand why a conflict may not be simple to resolve.

The idea behind such a practice is to keep participants away from as many distractions as possible so that they can concentrate on systems analysis.

A JAD may last anywhere from four hours to an entire week and may consist of several sessions.

A JAD employs thousands of dollars of corporate resources, the most expensive of which is the time of the people involved.

The **typical participants in a JAD** are listed below:

JAD session leader:

The JAD session leader organizes and runs the JAD. This person has been trained in group management and facilitation as well as in systems analysis. The JAD leader sets the agenda and sees that it is met; he or she remains neutral on issues and does not contribute ideas or opinions, but rather concentrates on keeping the group on the agenda, resolving conflicts and disagreements, and soliciting (manage) all ideas.

Users:

The key users of the system under consideration are vital participants in a JAD.

They are the only ones who have a clear understanding of what it means to use the system on a daily basis.

Managers:

Managers of the work groups who use the system in question provide insight into new organizational directions, motivations for and organizational impacts of systems, and support for requirements determined during the JAD.

Sponsor:

As a major undertaking due to its expense, a JAD must be sponsored by someone at a relatively high level in the company. If the sponsor attends any sessions, it is usually only at the very beginning or the end.

Systems analysts:

Members of the systems analysis team attend the JAD, although their actual participation may be limited. Analysts are there to learn from users and managers, not to run or dominate the process.

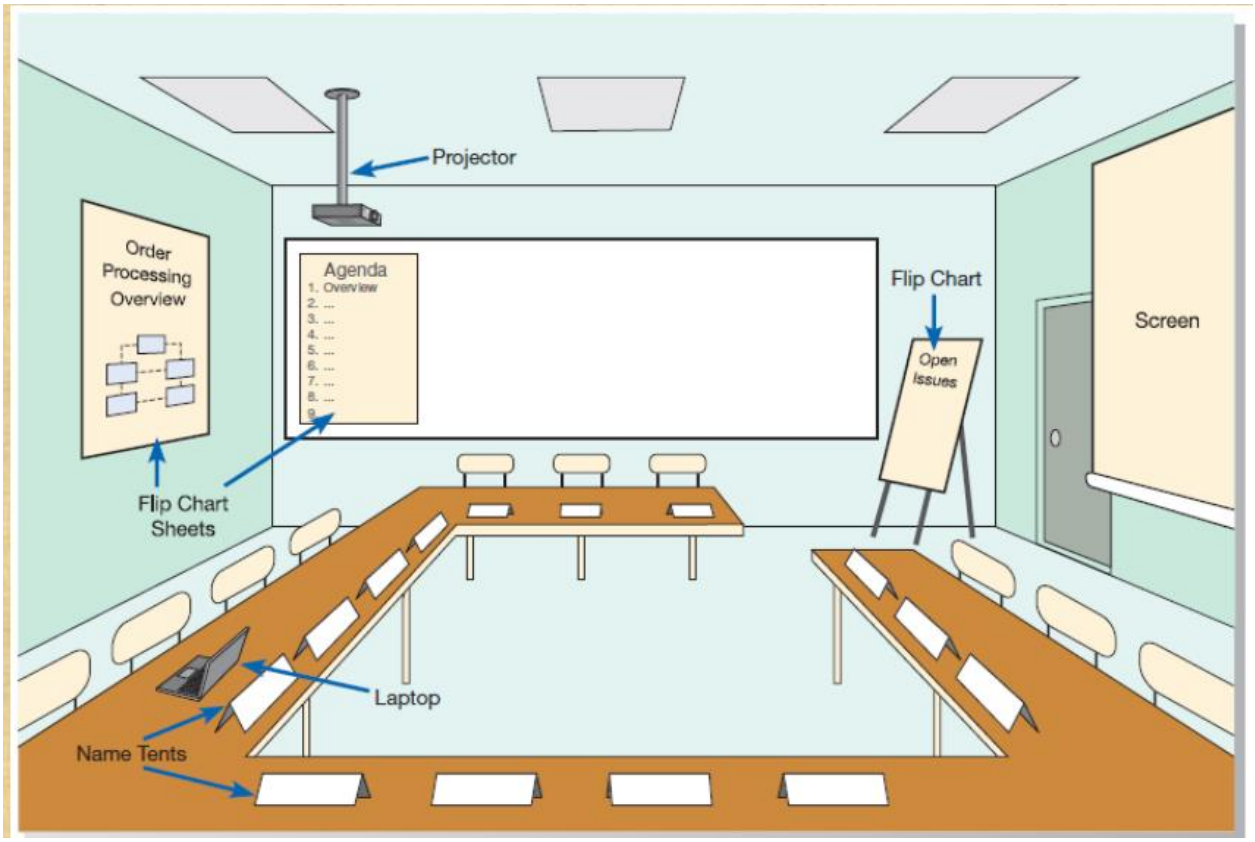
Scribe:

The scribe takes notes during the JAD sessions. This is usually done on a laptop. Notes may be taken using a word processor, or notes and diagrams may be entered directly into a CASE tool.

IS staff:

Besides systems analysts, other information systems (IS) staff, such as programmers, database analysts, IS planners, and data center personnel, may attend to learn from the discussion and possibly contribute their ideas on the technical feasibility of proposed ideas or the technical limitations of current systems.

JAD Meeting Room design:



When a JAD is completed, the end result is a set of documents that detail the workings of the current system related to the study of a replacement system. Depending on the exact purpose of the JAD, analysts may also walk away from the JAD with some detailed information on what is desired of the replacement system.

CASE Tools During JAD: For requirements determination and structuring, the most useful CASE tools are used for diagramming and form and report generation.

Some observers advocate using CASE tools during JADs (Lucas, 1993). Running a CASE tool during a JAD enables analysts to enter system models directly into a CASE tool, providing consistency and reliability in the joint model-building process.

The CASE tool captures system requirements in a more flexible and useful way than can a scribe or an analysis team making notes. Further, the CASE tool can be used to project menu, display, and report designs, so users can directly observe old and new designs and evaluate their usefulness for the analysis team.

Advantage:

- JAD allows you to resolve difficulties more simply and produce better, error-free software.
- The joint collaboration between the company and the clients lowers all risks.
- JAD reduces costs and time needed for project development.
- Well-defined requirements improve system quality.
- Due to the close communication, progress is faster.
- JAD encourages the team to push each other to work faster and deliver on time.

Disadvantage:

- Different opinions within the team make it difficult to align goals and maintain focus
- Depending on the size of the project, JAD may require a significant time commitment.

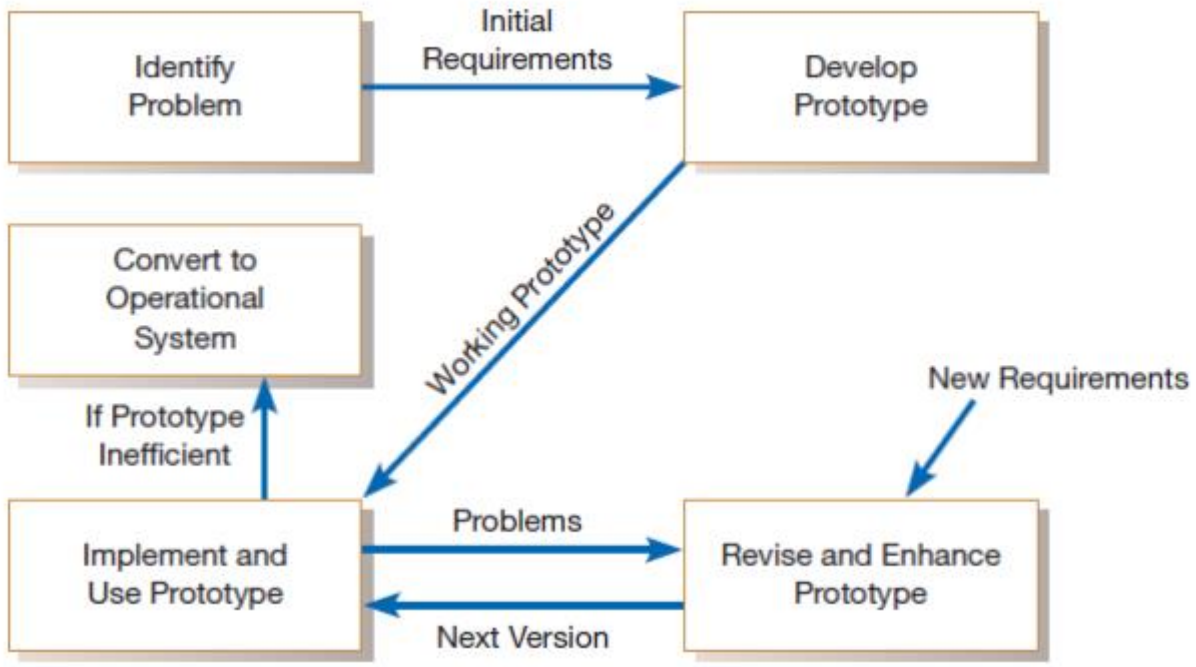
b. Prototyping:

An iterative process of systems development in which requirements are converted to a working system that is continually revised through close collaboration between an analyst and users.

Prototyping will enable you to quickly convert basic requirements into a working, though limited, version of the desired information system.

The prototype will then be viewed and tested by the user. Typically, seeing verbal descriptions of requirements converted into a physical system will prompt the user to modify existing requirements and generate new ones.

For example, in the initial interviews, a user might have said that he wanted all relevant utility billing information (e.g., the client's name and address, the service record, and payment history) on a single computer display form. Once the same user sees how crowded and confusing such a design would be in the prototype, he might change his mind and instead ask to have the information organized on several screens, but with easy transitions from one screen to another. He might also be reminded of some important requirements (data, calculations, etc.) that had not surfaced during the initial interviews.



The prototyping methodology

Evolutionary prototyping:

- As the prototype changes through each iteration more and more of the design specifications for the system are captured in the prototype. The prototype can then serve as the basis for the production system in a process called **evolutionary prototyping.**

- In evolutionary prototyping, you begin by modeling parts of the target system and, if the prototyping process is successful, you evolve the rest of the system from those parts.
- A life-cycle model of evolutionary prototyping illustrates the iterative nature of the process and the tendency to refine the prototype until it is ready to release.
- Systems must be designed to support scalability, multiuser support, and multiplatform support. Few of these design specifications will be coded into a prototype. Further, as much as 90 percent of a system's functioning is devoted to handling exceptional cases.
- Prototypes are designed to handle only the typical cases, so exception handling must be added to the prototype as it is converted to the production system. Clearly, the prototype captures only part of the system requirements.

Throwaway prototyping:

- Alternatively, the prototype can serve only as a model, which is then used as a reference for the construction of the actual system. In this process, called **throwaway prototyping**, the prototype is discarded after it has been used.
- Unlike evolutionary prototyping, throwaway prototyping does not preserve the prototype that has been developed. With throwaway prototyping, there is never any intention to convert the prototype into a working system.
- Instead, the prototype is developed quickly to demonstrate some aspect of a system design that is unclear or to help users decide among different features or interface characteristics.
- Once the uncertainty the prototype was created to address has been reduced, the prototype can be discarded, and the principles learned from its creation and testing can then become part of the requirements determination.

5. Radical Methods for Determining System Requirements

The overall process by which current methods are replaced with radically new methods is generally referred to as **business process reengineering (BPR).**

To better understand **BPR**, consider the following analogy. Suppose you are a successful European golfer who has tuned your game to fit the style of golf courses and weather in Europe. You have learned how to control the flight of the ball in heavy winds, roll the ball on wide open greens, putt on large and undulating greens, and aim at a target without the aid of the landscaping common on North American courses. When you come to the United States to make your fortune on the US tour, you discover that incrementally improving you're putting, driving accuracy, and sand shots will help, but the new competitive environment is simply not suited to your style of the game. You need to reengineer your whole approach, learning how to aim at targets, spin and stop a ball on the green, and manage the distractions of crowds and the press. If you are good enough, you may survive, but without reengineering, you will never be a winner. Organizations realize that creatively using information technologies can yield significant improvements in most business processes.

Identifying Processes To Reengineer

A first step in any BPR effort relates to understanding what processes to change. To do this, you must first understand which processes represent the key business processes for the organization. Key business processes are the structured set of measurable activities designed to produce a specific output for a

particular customer or market. The important aspect of this definition is that key processes are focused on some type of organizational outcome, such as the creation of a product or the delivery of a service. Key business processes are also customer focused. In other words, key business processes would include all activities used to design, build, deliver, support, and service a particular product for a particular customer.

Disruptive technologies:

Once key business processes and activities have been identified, information technologies must be applied to radically improve business processes. To do this, Hammer and Champy (1993) suggest that organizations think "inductively" about information technology. Induction is the process of reasoning from the specific to the general, which means that managers must learn the power of new technologies and think of innovative (using new methods or ideas) ways to alter the way work is done. This is contrary to deductive thinking, where problems are first identified and solutions are then formulated.

6. Requirements Management Tools

Organizations and developers have always been looking for more effective and creative ways to create and maintain requirements documents. One method that has been developed is computer-based requirements management tools. These tools make it easier for analyst to keep requirements, current documents and to define links between different parts of the overall specifications package. Requirements management tools are typically designed to work with many of the standards now available for requirements specification such as the Unified Modeling Language 2.0(Structural thing, behavioral thing, grouping thing and notational thing) (help to specify, visualize and document models), System modeling language, Business process modeling notation etc.

Requirements management tools is best to traditional, planning based systems development approaches. They do not work well with **agile methodologies** which tend to employ different approaches to requirements gathering.

Requirements Determination Using Agile Methodologies:

Three techniques are presented in this section.

- The first is **continual user involvement** in the development process, a technique that works especially well with small and dedicated development teams.
- The second approach is a **JAD-like process called Agile Usage-Centered Design.**
- The third approach is the **Planning Game, which was developed as part of eXtreme Programming.**

Continual User Involvement

In Chapter 1, we read about the criticisms of the traditional waterfall SDLC. One of those criticisms was that the waterfall SDLC allowed users to be involved in the development process only in the early stages of analysis. Once requirements had been gathered from them, the users were not involved again in the process until the system was being installed and they were asked to sign off on it. Typically, by the time the users saw the system again, it was nothing like what they had imagined. The system most likely did not adequately address user needs. One approach to the problem of limited user involvement is to involve the users continually, throughout the entire analysis and design process. Such an approach works best when

development can follow the analysis–design–code–test cycle favored by the Agile Methodologies (Figure 6-9), because the user can provide information on requirements and then watch and evaluate as those requirements are designed, coded, and tested. This iterative process can continue through several cycles, until most of the major functionality of the system has been developed.

Extensive involvement users in the analysis and design process are a key part of many Agile approaches, but it was also a key part of Rapid Application Development (see Chapter 1).

Agile Usage-Centered Design

Continual user involvement in systems development is an excellent way to ensure that requirements are captured accurately and immediately implemented in system design. However, such constant interaction works best when the development team is small, as was the case in the Boeing example. Also, it is not always possible to have continual access to users for the duration of a development project. Thus, agile developers have come up with other means for effectively involving users in the requirements determination process. One such method is called Agile Usage-Centered Design, originally developed by Larry Constantine (2002) and adapted for Agile Methodologies by Jeff Patton (2002).

The Planning Game From eXtreme Programming

You read about eXtreme Programming in Chapter 1, and you know that it is an approach to software development put together by Kent Beck (Beck and Andres, 2004). You also know that it is distinguished by its short cycles, its incremental planning approach, its focus on automated tests written by programmers and customers to monitor the process of development, and its reliance on an evolutionary approach to development that lasts throughout the lifetime of the system. One of the key emphases of eXtreme Programming is its use of two-person programming teams and having a customer on-site during the development process.

The relevant parts of eXtreme Programming that relate to requirements determination are:

(1) How planning, analysis, design, and construction are all fused together into a single phase of activity

(2) Its unique way of capturing and presenting system requirements and design specifications.

All phases of the life cycle converge into a series of activities based on the basic processes of coding, testing, listening, and designing.

B. System Process Requirements:

1. Introduction:

- In this topic, our focus will be on one tool that is used to coherently represent the information gathered as part of requirements determination—data flow diagrams.
- Data flow diagrams enable you to model how data flow through an information system, the relationships among the data flows, and how data come to be stored at specific locations.
- Data flow diagrams also show the processes that change or transform data. Because data flow diagrams concentrate on the movement of data between processes, these diagrams are called process models.

- Decision tables allow you to represent the conditional logic that is part of some data flow diagram processes.

2. Process Modeling:

Process modeling involves graphically representing the functions, or processes, that capture, manipulate, store, and distribute data between a system and its environment and between components within a system. A common form of a process model is a data flow diagram (DFD).

DFDs, the traditional process modeling technique of structured analysis and design and one of the techniques most frequently used today for process modeling.

Modeling a system's Process for structured Analysis:

Analysis phase of the systems development life cycle has two sub phases: **requirements determination and requirements structuring.**

The analysis team enters the requirements structuring phase with an abundance of information gathered during the requirements determination phase.

During requirements structuring, you and the other team members must organize the information into a meaningful representation of the information system that currently exists and of the requirements desired in a replacement system.

Deliverables and outcomes:

TABLE 7-1 Deliverables for Process Modeling

1. Context DFD
2. DFDs of the system (adequately decomposed)
3. Thorough descriptions of each DFD component

In structured analysis, the primary deliverables from process modeling are a set of coherent, interrelated DFDs.

Table 7-1 provides a more detailed list of the deliverables that result when DFDs are used to study and document a system's processes.

First, a context diagram shows the scope of the system, indicating which elements are inside and which are outside the system.

Second, DFDs of the system specify which processes move and transform data, accepting inputs and producing outputs. These diagrams are developed with sufficient detail to understand the current system and to eventually determine how to convert the current system into its replacement.

Finally, entries for all of the objects included in all of the diagrams are included in the project dictionary or CASE repository.

3. Data Flow Diagramming Mechanics:

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself.

DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart.

Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools.

Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

DFD consists four symbols:

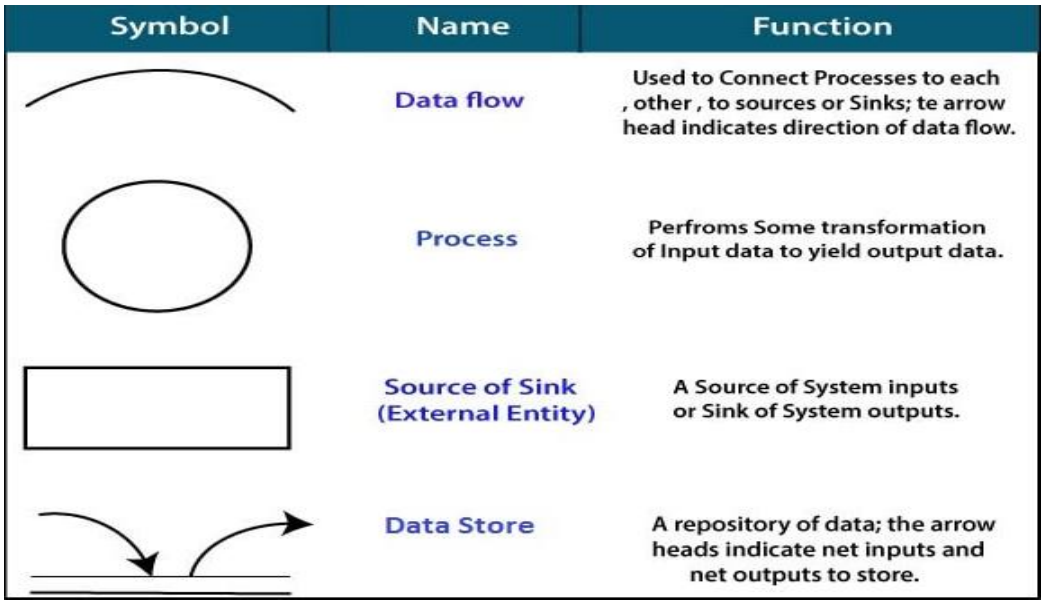
- data flows,
- data stores,
- processes, and
- Sources/sinks (or external entities).

Data flow is the path for data to move from one part of the system to another. It may be a single data element or set of data element. The symbol of data flow is the arrow which shows the flow direction.

A **data store** is data at rest. A data store may represent one of many different physical locations for data; for example, a file folder, one or more computer-based file(s), or a notebook. A data store might contain data about customers, students, customer orders, or supplier invoices.

A **process** is the work or actions performed on data so that they are transformed, stored, or distributed. When modeling the data processing of a system, it does not matter whether a process is performed manually or by a computer.

Finally, a **source/sink** is the origin and/or destination of the data. Sources/sinks are sometimes referred to as external entities because they are outside the system. Once processed, data or information leave the system and go to some other place.



Advantages of DFD

- It helps us to understand the functioning and the limits of a system.
- It is a graphical representation which is very easy to understand as it helps visualize contents.
- Data Flow Diagram represent detailed and well explained diagram of system components.
- It is used as the part of system documentation file.
- Data Flow Diagrams can be understood by both technical and nontechnical person because they are very easy to understand.

Disadvantages of DFD

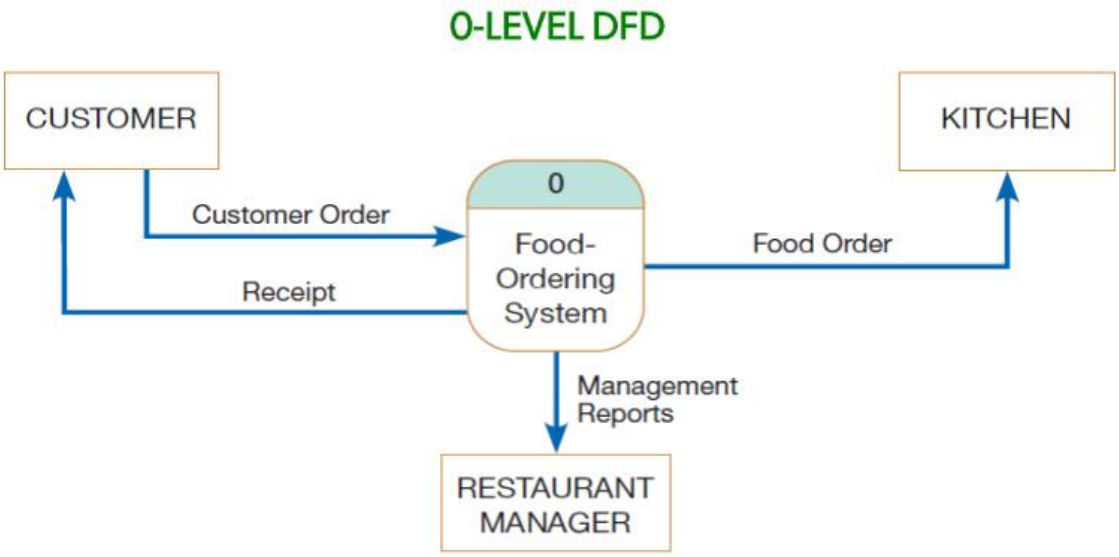
- At times DFD can confuse the programmers regarding the system.
- Data Flow Diagram takes long time to be generated, and many times due to this reasons analysts are denied permission to work on it.

Levels in Data Flow Diagrams (DFD):

Three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

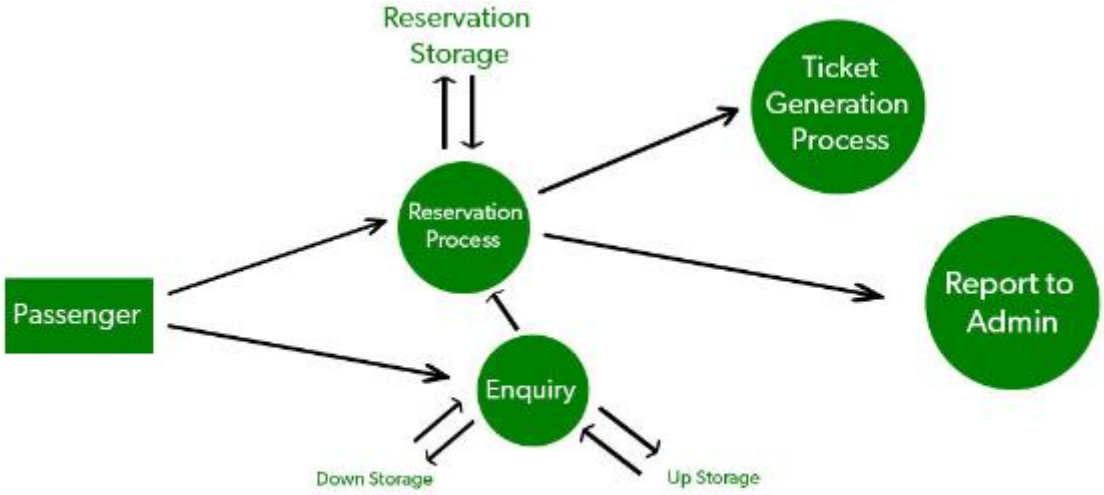
a. 0-level DFD:

DFD Level 0 is also called a **Context Diagram**. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.



b. 1-level DFD:

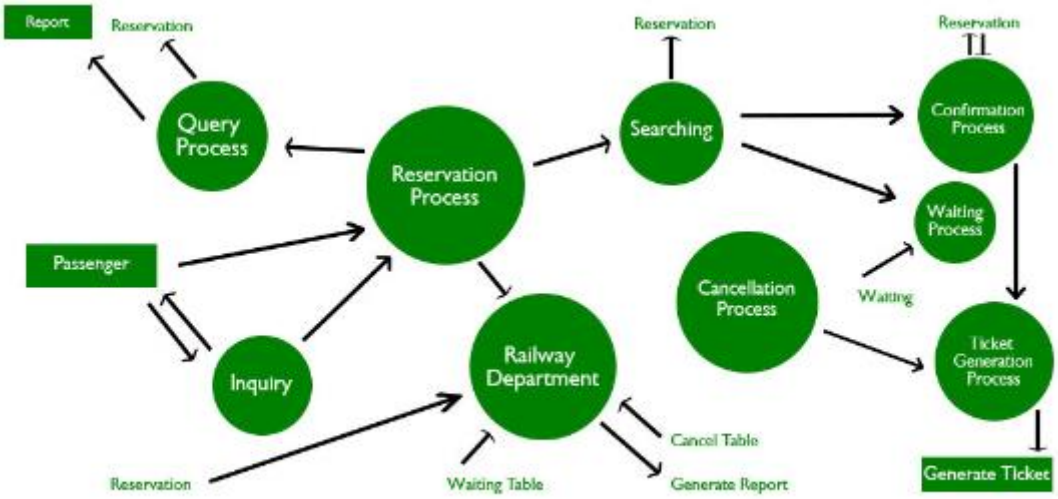
DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes.



1-LEVEL DFD

c. 2-level DFD:

DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system's functioning.



2-LEVEL DFD

DFD rules and tips:

- Each process should have at least one input and an output.
- Each data store should have at least one data flow in and one data flow out.
- Data stored in a system must go through a process.
- All processes in a DFD go to another process or a data store.

Process:

- No process can have only outputs (a miracle)
- No process can have only inputs (black hole)
- No process can have partial inputs but complete output (grey hole)
- A process has a verb phrase label

Data Store:

- Data cannot be moved directly from one store to another
- Data cannot move directly from an outside source to a data store
- Data cannot move directly from a data store to a data sink
- Data store has a noun phrase label

Source/Sink:

- Data cannot move directly from a source to a sink
- A source/sink has a noun phrase label

Data Flow:

- A data flow has only one direction of flow between symbols
- A fork means that exactly the same data goes from a common location to two or more processes, data stores or sources/sinks

Decomposition of DFDs

The act of going from a single system to four component processes is called (functional) decomposition. Functional decomposition is an iterative process of breaking the description or perspective of a system down into finer and finer detail.

This process creates a set of hierarchically related charts in which one process on a given chart is explained in greater detail on another chart.

- Creates a set of charts in which one process on a given chart is explained in greater detail on another chart.
- Continues until no sub-process can logically be broken down any further.
- Lowest level is called a primitive DFD

Level-N Diagrams

A DFD that is the result of n nested decompositions of a series of sub processes from a process on a level-0 diagram.

Balancing DFDs:

Conservation Principle:

- Conserve inputs and outputs to a process at the next level of decomposition.

Balancing:

- Conservation of inputs and outputs to a data flow diagram process when that process is decomposed to a lower level.

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.

When you decompose a DFD from one level to the next, there is a conservation principle at work. You must conserve inputs and outputs to a process at the next level of decomposition. In other words, Process 1.0, which appears in a level-0 diagram, must have the same inputs and outputs when decomposed into a level-1 diagram. This conservation of inputs and outputs is called balancing.

4. Using Data flow Diagramming in the Analysis Process:

Learning the mechanics of drawing DFDs is important because DFDs have proven to be essential tools for the structured analysis process. Beyond the issue of drawing mechanically correct DFDs, there are other issues related to process modeling with which an analyst must be concerned. Such issues, including whether the DFDs are complete and consistent across all levels, which covers guidelines for drawing DFDs. Another issue to consider is how you can use DFDs as a useful tool for analysis.

Guidelines for drawing DFDs:

a. Completeness:

The concept of DFD completeness refers to whether you have included in your DFDs all of the components necessary for the system you are modeling. If your DFD contains data flows that do not lead anywhere or data stores, processes, or external entities that are not connected to anything else, your DFD is not complete.

b. Consistency:

The concept of DFD consistency refers to whether or not the depiction of the system shown at one level of a nested set of DFDs is compatible with the depictions of the system shown at other levels. A gross violation of consistency would be a level-1 diagram with no level-0 diagram. Another example of inconsistency would be a data flow that appears on a higher-level DFD but not on lower levels (also a violation of balancing).

c. Timing:

You may have noticed in some of the DFD examples we have presented that DFDs do not do a very good job of representing time. On a given DFD, there is no indication of whether a data flow occurs constantly in real time, once per week, or once per year. There is also no indication of when a system would run.

d. Iterative Development:

The first DFD you draw will rarely capture perfectly the system you are modeling. You should count on drawing the same diagram over and over again, in an iterative fashion. With each attempt, you will come closer to a good approximation of the system or aspect of the system you are modeling. One rule of thumb is that it should take you about three revisions for each DFD you draw

e. Primitive DFDs:

One of the more difficult decisions you need to make when drawing DFDs is when to stop decomposing processes. One rule is to stop drawing when you have reached the lowest logical level; however, it is not always easy to know what the lowest logical level is.

Rules for stopping decomposition

- 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, on-line 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

5. Modeling logic with decision Tables

- Data flow diagrams do not show the logic inside the processes
- Logic modeling involves representing internal structure and functionality of processes depicted on a DFD
- Logic modeling can also be used to show when processes on a DFD occur

The table has three parts:

The **condition stubs** contain the various conditions that apply to the situation the table is modeling.

- There are two condition stubs for employee type and hours worked. Employee type has two values: "**S**," which stands for salaried, and "**H**," which stands for hourly. Hours worked has three values: **less than 40, exactly 40, and more than 40**.

The **action stubs** contain all the possible courses of action that result from combining values of the condition stubs.

- There are four possible courses of action in this table: **Pay Base Salary, Calculate Hourly Wage, Calculate Overtime, and Produce Absence Report.**

Rules specify which actions are to be followed for a given set of conditions

Complete decision table for payroll system

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type	S	H	S	H	S	H
	Hours worked	<40	<40	40	40	>40	>40
Action Stubs	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce absence report		X				

Decision Table (Extended)

Steps:

- Identify
  - Conditions

    Criteria: What and How many?
  - Actions
  - Rules

    Figure out total number of rules
  - Draw a 2D grid

Decision Table Compression

Steps:

- Start from extended table
- Identify indifferent rules

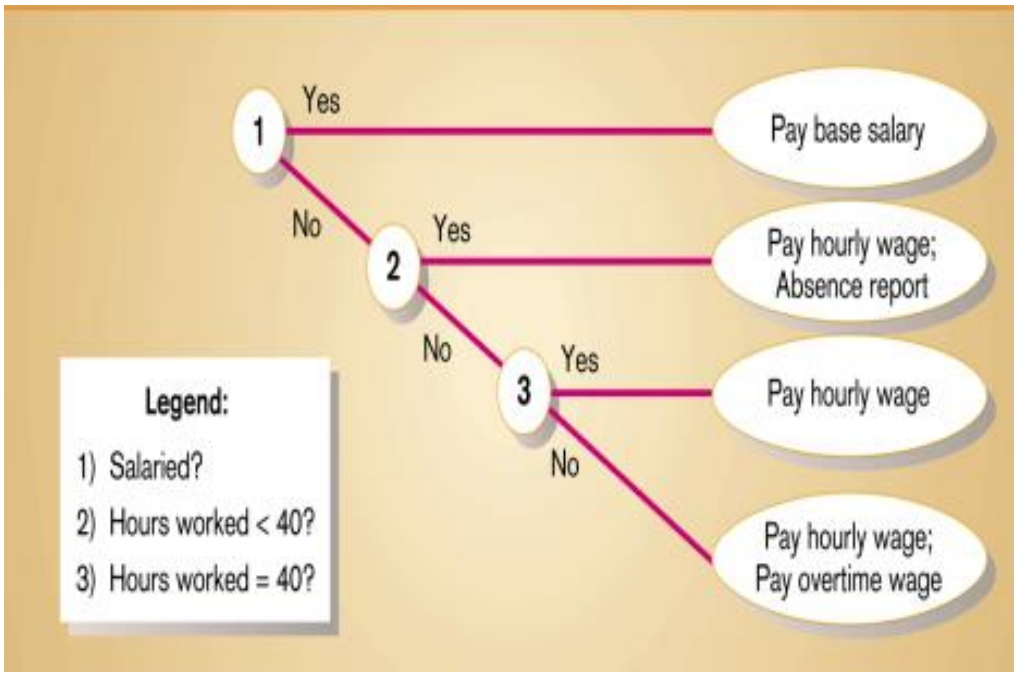
  Does not change the action if one or more conditions are changed
- Replace that condition by a dash "–"
- Merge two identical rules
- It is better to do this one condition a time

Reduced (Compression) decision table for payroll system

Conditions/ Courses of Action	Rules			
	1	2	3	4
Employee type	S	H	H	H
Hours worked	–	<40	40	>40
Pay base salary	X			
Calculate hourly wage		X	X	X
Calculate overtime				X
Produce absence report		X		

6. Modeling Logic with Decision Trees

- A graphical representation of a decision situation
- Decision situation points are connected together by arcs and terminate in ovals
- Two main **components**:
  - Decision points represented by nodes
  - Actions represented by ovals
- Read from **left to right**
- Each node corresponds to a numbered choice a legend
- All possible actions are listed on the **far right**



C. System Data Requirements:

1. Introduction:

- In previous chapter, we learned how to model and analyze data. We learned how to show data stores, or data at rest, in a data flow diagram (DFD). DFDs, use cases, and various processing logic techniques show how, where, and when data are used or changed in an information system, but these techniques do not show the definition, structure, and relationships within the data.
- **Data modeling** develops these missing, and crucial, descriptive pieces of a system. The most common format used for data modeling is entity-relationship (E-R) diagramming.
- **Data models** that use E-R and class diagram notations explain the characteristics and structure of data independent of how the data may be stored in computer memory. A data model is usually developed iteratively, either from scratch or from a purchased data model for the industry or business area to be supported.

2. Conceptual Data Modeling

A conceptual data model is a representation of organizational data.

The purpose of a conceptual data model is to show as many rules about the meaning and interrelationships among data as are possible.

Conceptual data modeling is typically done in parallel with other requirements analysis and structuring steps during systems analysis.

On larger systems development teams, a subset of the project team concentrates on data modeling while other team members focus attention on process or logic modeling.

Analysts develop (or use from prior systems development) a conceptual data model for the current system and then build or refine a purchased conceptual data model that supports the scope and requirements for the proposed or enhanced system.

The work of all team members is coordinated and shared through the project dictionary or repository.

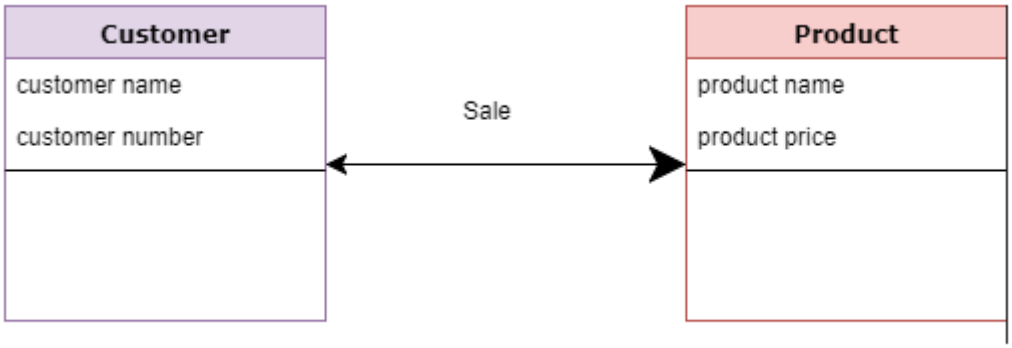
This repository is often maintained by a common Computer- Aided Software Engineering (CASE) or data modeling software tool.

The 3 basic terms of Conceptual Data Model are:

- **Entity:** A real-world thing
- **Attribute:** Characteristics or properties of an entity
- **Relationship:** Dependency or association between two entities

Data model example:

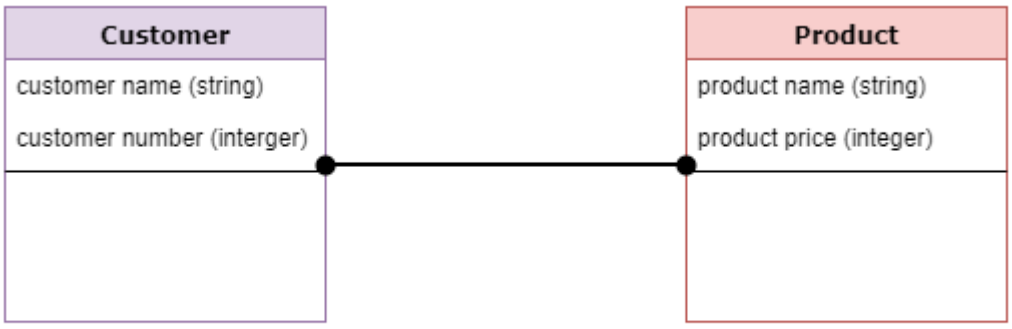
- Customer and Product are two entities. Customer number and name are attributes of the Customer entity
- Product name and price are attributes of product entity
- Sale is the relationship between the customer and product



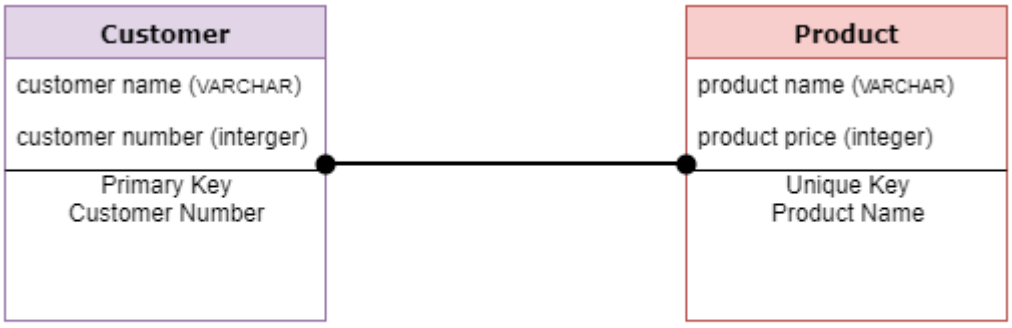
Conceptual Data Model

The Logical Data Model is used to define the structure of data elements and to set relationships between them. The logical data model adds further information to the conceptual data model elements. The advantage of using a Logical data model is to provide a foundation to form the base for the Physical model. However, the modeling structure remains generic.

A Physical Data Model describes a database-specific implementation of the data model. It offers database abstraction and helps generate the schema. This is because of the richness of meta-data offered by a Physical Data Model. The physical data model also helps in visualizing database structure by replicating database column keys, constraints, indexes, triggers, and other RDBMS features.



Logical Data Model



Physical Data Model

The Conceptual Data Modeling process

- The process of conceptual data modeling begins with developing a conceptual data model for the system being replaced, if a system already exists.
- This is essential for planning the conversion of the current files or database into the database of the new system.
- Further, this is a good, but not a perfect, starting point for your understanding of the data requirements of the new system. Then, a new conceptual data model is built (or a standard one is purchased) that includes all of the data requirements for the new system.

Deliverables and outcomes:

Most organizations today do conceptual data modeling using **E-R (entity-relationship) modeling**, which uses a special notation to represent as much meaning about data as possible. Because of the rapidly increasing interest in **object-oriented methods, class diagrams using unified modeling language (UML-- modeling language that is most often used for software engineering but has extended its use to business processes and other project workflows, e.g.: Lucid chart, draw.io) drawing tools** such as IBM's Rational products or Microsoft Visio are also popular.

- The primary deliverable from the conceptual data modeling step within the analysis phase is an E-R diagram.
- The other deliverable from conceptual data modeling is a full set of entries about data objects that will be stored in the project dictionary, repository, or data modeling software. The repository is the mechanism to link data, process and logic models of an information system.

3. Gathering Information for Conceptual Data Modeling:

Requirements determination methods must include questions and investigations that take a data, not only a process and logic, focus. For example, during interviews with potential system users—during Joint Application Design (JAD) sessions or through requirements interviews—you must ask specific questions in order to gain the perspective on data that you need to develop or tailor a purchased data model.

You typically do data modeling from a combination of perspectives.

- The first perspective is generally called the top-down approach. This perspective derives the business rules for a data model from an intimate understanding of the nature of the business, rather than from any specific information requirements in computer displays, reports, or business forms.
- You can also gather the information you need for data modeling by reviewing specific business documents—computer displays, reports, and business forms— handled within the system.
- This process of gaining an understanding of data is often called a bottom-up approach. These items will appear as data flows on DFDs and will show the data processed by the system and, hence, probably the data that must be maintained in the system's database. Consider, for example, Figure 8-4, which shows a customer order form used at Pine Valley Furniture (PVF).

ORDER NO
ORDER DATE
PROMISED DATE
PRODUCT NO
DESCRIPTION
QUANTITY ORDERED
UNIT PRICE

CUSTOMER NO
NAME
ADDRESS
CITY-STATE-ZIP

PVF CUSTOMER ORDER

ORDER NO: 61384 CUSTOMER NO: 1273

NAME: Contemporary Designs
ADDRESS: 123 Oak St.
CITY-STATE-ZIP: Austin, TX 28384

ORDER DATE: 11/04/2014 PROMISED DATE: 11/21/2017

PRODUCT NO	DESCRIPTION	QUANTITY ORDERED	UNIT PRICE
M128	Bookcase	4	200.00
B381	Cabinet	2	150.00
R210	Table	1	500.00

FIGURE 8-4
Sample customer form

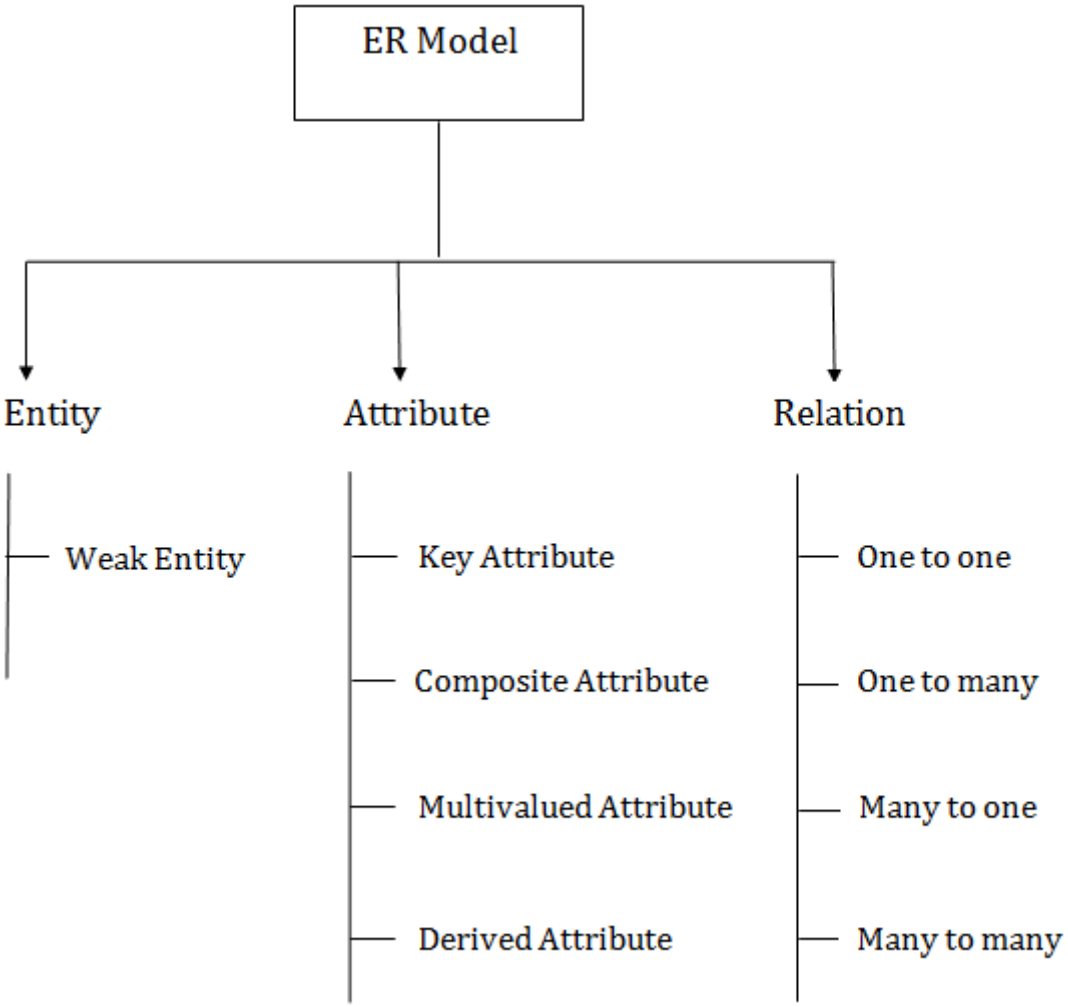
1. *What are the subjects/objects of the business?* What types of people, places, things, materials, events, etc. are used or interact in this business, about which data must be maintained? How many instances of each object might exist?—**data entities and their descriptions**
2. *What unique characteristic (or characteristics) distinguishes each object from other objects of the same type?* Might this distinguishing feature change over time or is it permanent? Might this characteristic of an object be missing even though we know the object exists?—**primary key**
3. *What characteristics describe each object?* On what basis are objects referenced, selected, qualified, sorted, and categorized? What must we know about each object in order to run the business?—**attributes and secondary keys**
4. *How do you use these data?* That is, are you the source of the data for the organization, do you refer to the data, do you modify it, and do you destroy it? Who is not permitted to use these data? Who is responsible for establishing legitimate values for these data?—**security controls and understanding who really knows the meaning of data**
5. *Over what period of time are you interested in these data?* Do you need historical trends, current "snapshot" values, and/or estimates or projections? If a characteristic of an object changes over time, must you know the obsolete values?—**cardinality and time dimensions of data**
6. *Are all instances of each object the same?* That is, are there special kinds of each object that are described or handled differently by the organization? Are some objects summaries or combinations of more detailed objects?—**supertypes, subtypes, and aggregations**
7. *What events occur that imply associations among various objects?* What natural activities or transactions of the business involve handling data about several objects of the same or a different type?—**relationships and their cardinality and degree**
8. *Is each activity or event always handled the same way or are there special circumstances?* Can an event occur with only some of the associated objects, or must all objects be involved? Can the associations between objects change over time (for example, employees change departments)? Are values for data characteristics limited in any way?—**integrity rules, minimum and maximum cardinality, time dimensions of data**

TABLE 8-1 Requirements Determination Questions for Data Modeling

4. Introduction to E-R Modeling

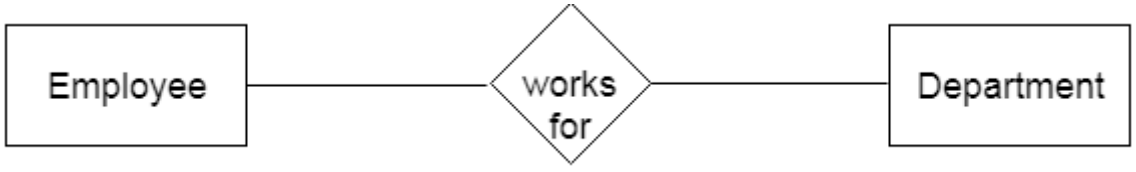
- An entity-relationship data model (E-R model) is a detailed, logical representation of the data for an organization or for a business area. The E-R model is expressed in terms of entities in the business environment, the relationships or associations among those entities, and the attributes or properties of both the entities and their relationships.
- An E-R model is normally expressed as an entity-relationship diagram (E-R diagram), which is a graphical representation of an E-R model.
- The basic E-R modeling notation uses three main constructs: data entities, relationships and their associated attributes.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

Component/Structural/Constraints of ER Diagram



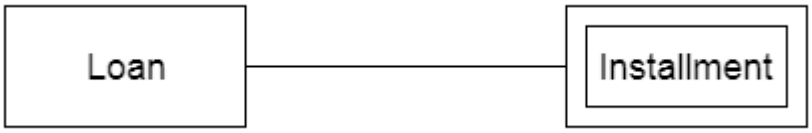
a. Entity:

- An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
- Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



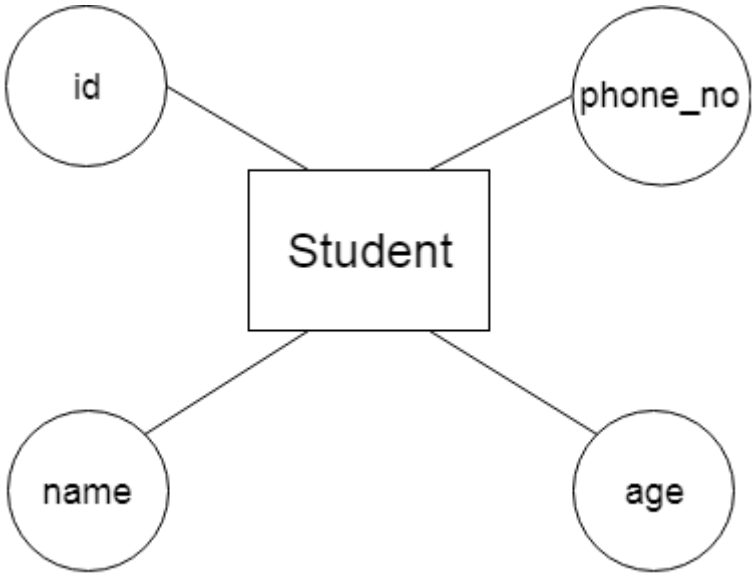
i. Weak Entity

An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



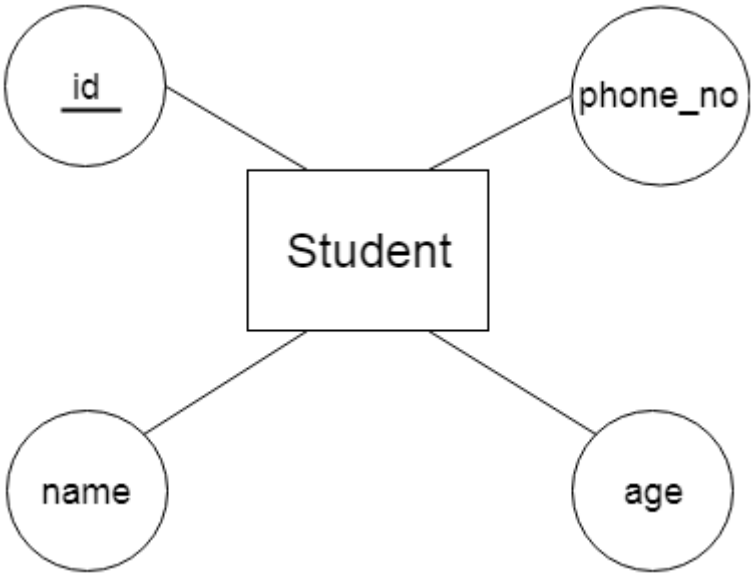
b. Attribute

The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.
For example, id, age, contact number, name, etc. can be attributes of a student.



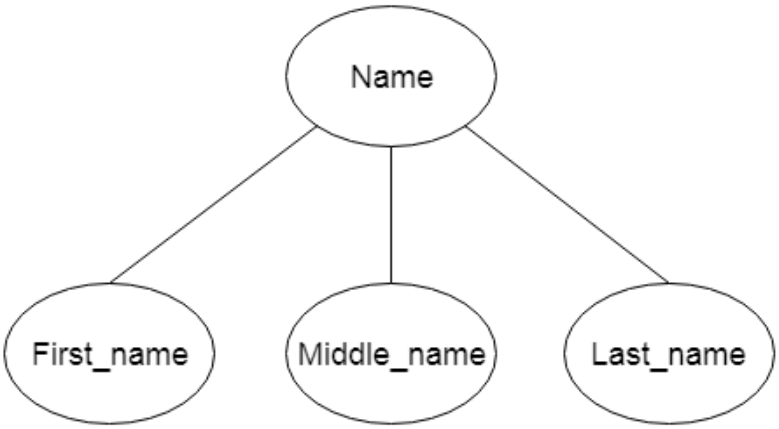
i. Key Attribute:

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



ii. Composite Attribute

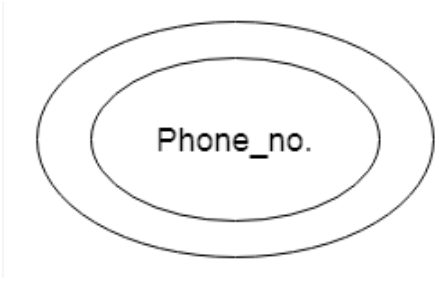
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



iii. Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

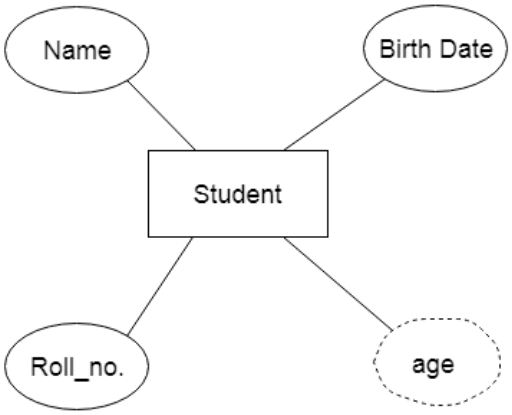
For example, a student can have more than one phone number.



iv. Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, a person's age changes over time and can be derived from another attribute like Date of birth.



c. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

i. One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, a female can marry to one male, and a male can marry to one female.

ii. One-to-many relationship

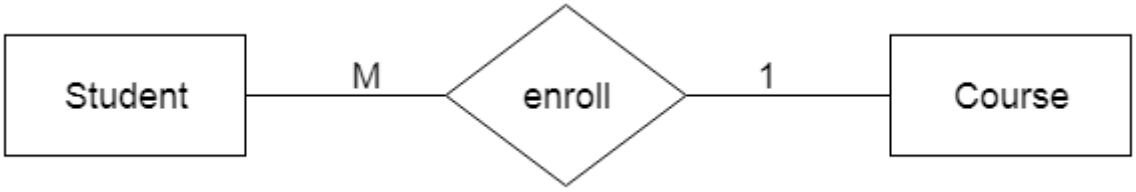
When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.

iii. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.



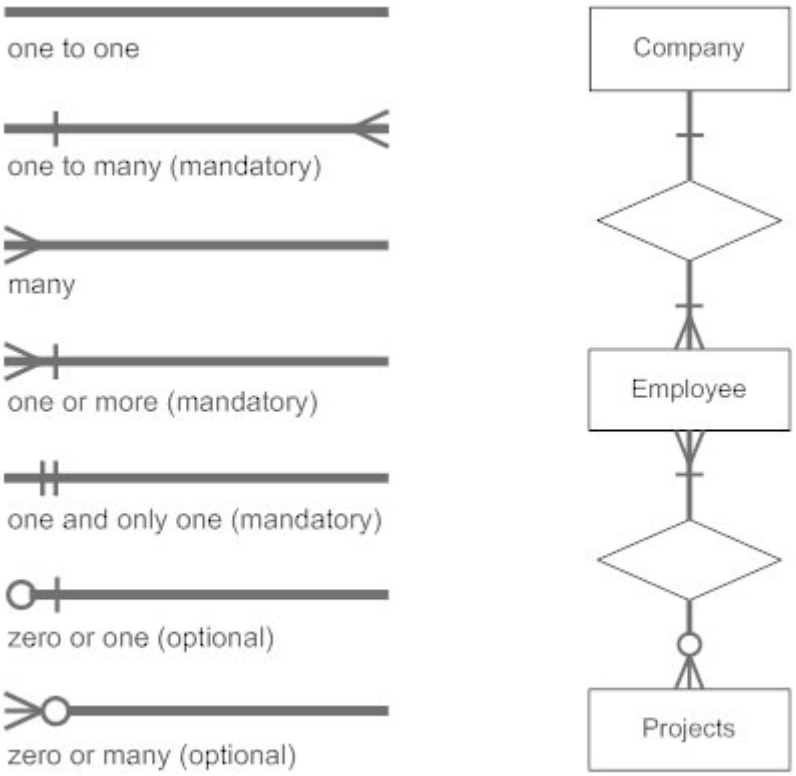
iv. Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.

Notation of ER diagram

Database can be represented using the notations. In ER diagram, many notations are used to express the cardinality. These notations are as follows:



5. Conceptual Data Modeling and the E-R Model

Degree of a Relationship

Number of entity sets that participate in a relationship set is called degree of the relationship set. On the basis of degree, relationships can be divided as below:

- **Unary Relationship**
- **Binary Relationship**
- **N-ary Relationship**

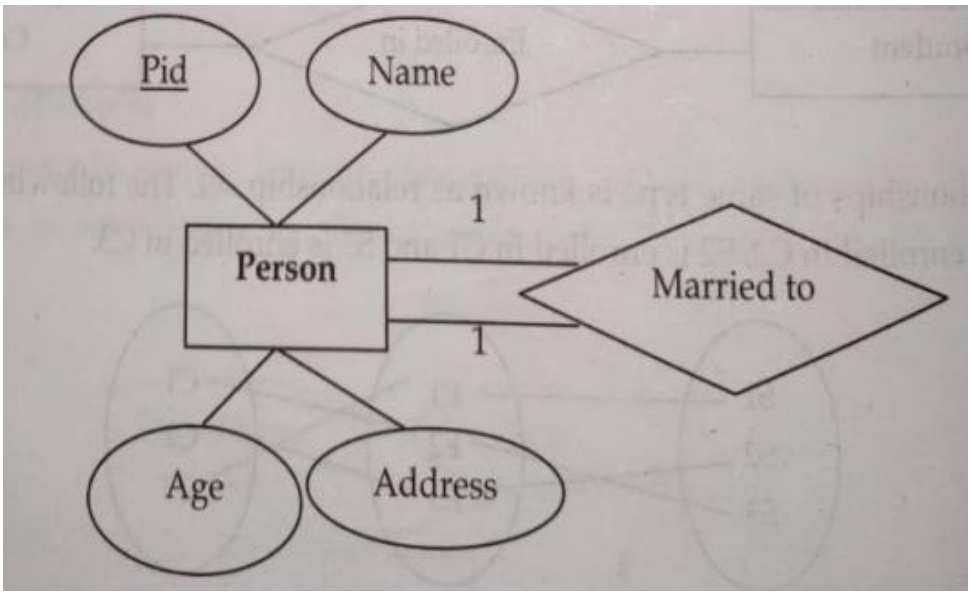
I. Unary Relationship

If only one entity set participates in a relation, the relationship is called as unary relationship. Here same entity set participates in relationship twice with different roles. Role names are specified above the link joining entity set and relationship set. This type of relationship set is sometimes called a recursive relationship set. There are three types of unary relationships:

- **1:1 unary relationship**
- **1: M unary relationship**
- **M: N unary relationship**

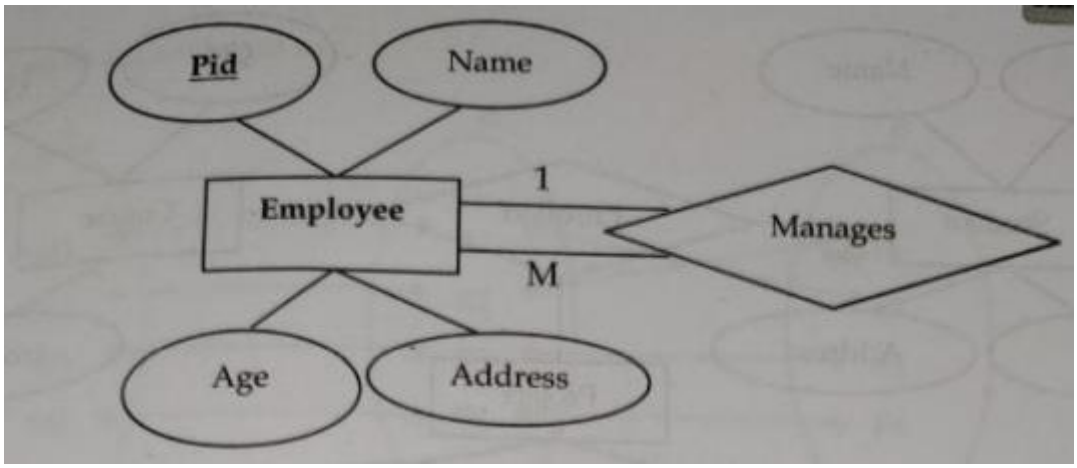
a. One to one (1:1) unary relationship

In the example below, one person is married to only one person.



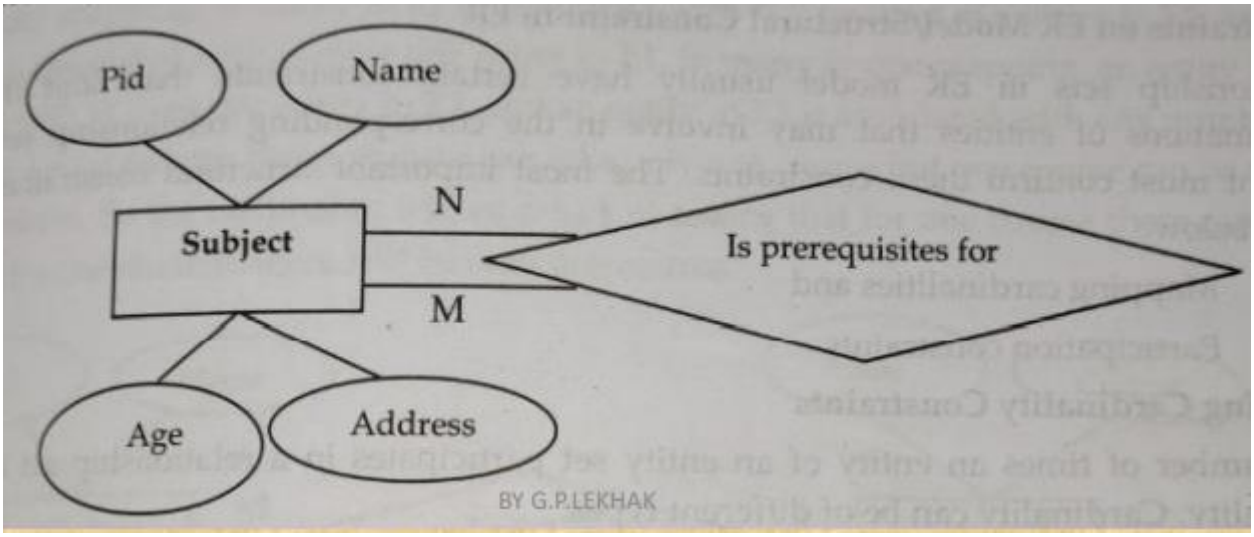
b. One to many (1: M) unary relationship

An employee may manage many employees but an employee is managed by only one employee. This types of relationship with employee relationship set itself is called 1: M unary relationship as shown in below;



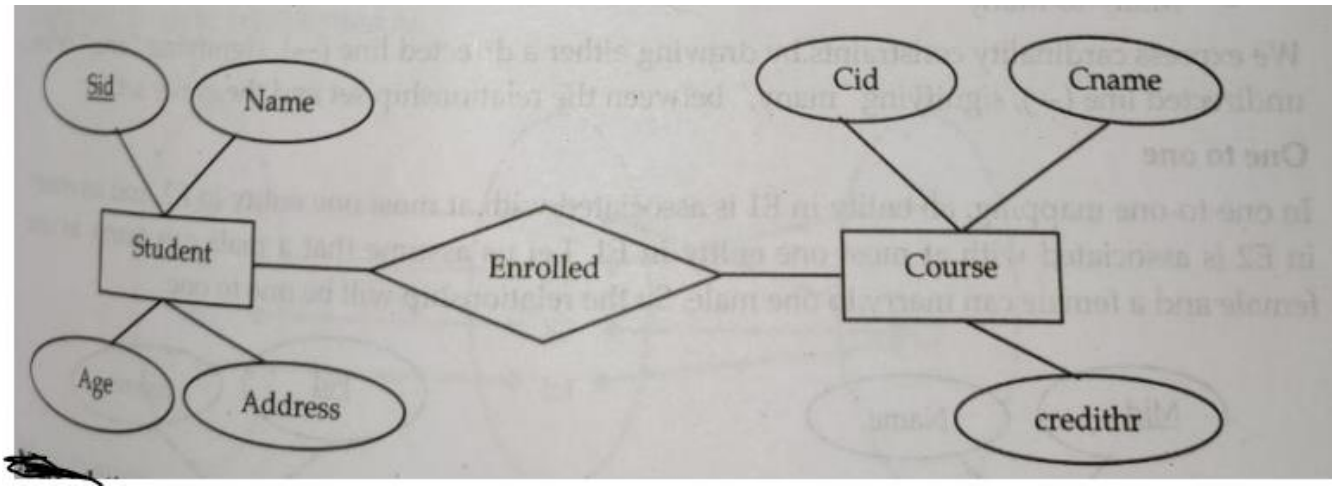
c. Many to many (M: N) unary relationship

A subject may have many other subjects as prerequisites and each subject may be a prerequisite to many other subjects.



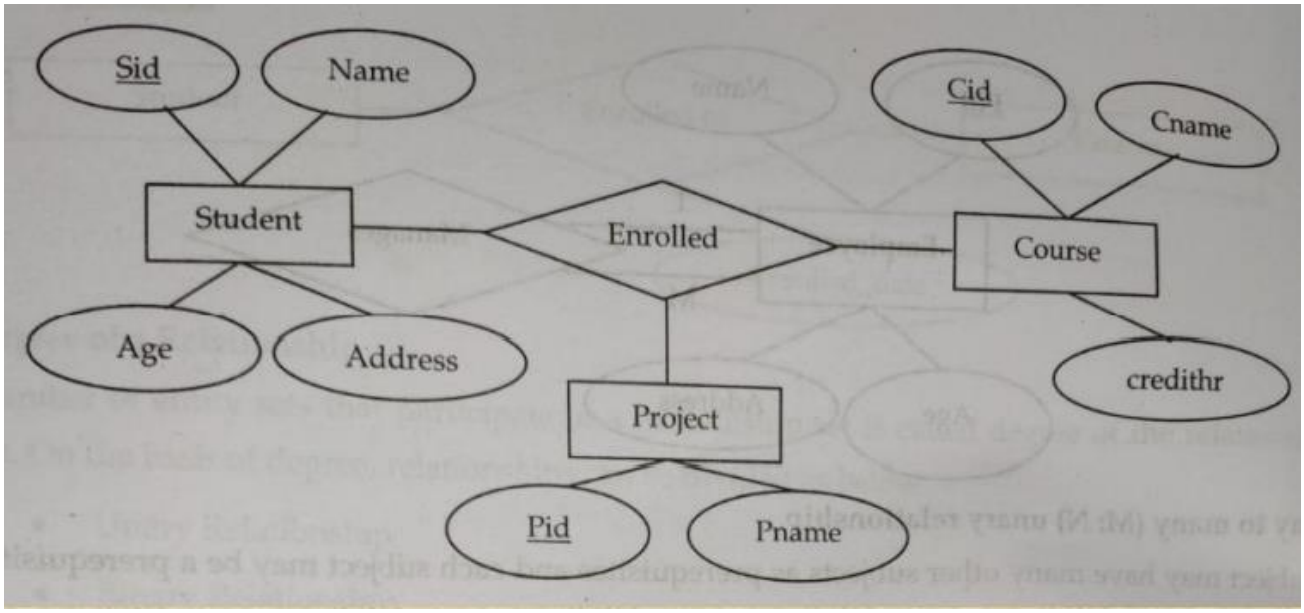
II. Binary relationship

When there are two entities set participating in a relation, the relationship is called as binary relationship. For example, Student is enrolled in Course. This is the most common type of relationship in database systems.



III. N-ary relationship

When there are n entities set participating in a relation, the relationship is called as n-ary relationship. If n=1 then it is called unary relationship, if n=2 then it is called binary relationship. Generally in N-ary relationship there are more than two entities participating with a single relationship i.e. n>2.



6. Representing Super types And Subtypes

- Often two or more entity types seem very similar (maybe they have almost the same name), but there are a few differences. That is, these entity types share common properties but also have one or more distinct attributes or relationships. To address this situation, the E-R model has been extended to include super type/subtype relationships.
- A **subtype** is a sub grouping of the entities in an entity type that is meaningful to the organization. **For example,** STUDENT is an entity type in a university. Two subtypes of STUDENT are GRADUATE STUDENT and UNDERGRADUATE STUDENT.

- A **super type** is a generic entity type that has a relationship with one or more subtypes.

Business Rules:
Conceptual data modeling is a step-by-step process for documenting information requirements, and it is concerned with both the structure of data and with rules about the integrity of those data. Business rules are specifications that preserve the integrity of the logical data model. Four basic types of business rules are as follows:

1. Entity integrity: Each instance of an entity type must have a unique identifier that is not null.
2. Referential integrity constraints: Rules concerning the relationships between entity types.
3. Domains: Constraints on valid values for attributes.
4. Triggering operations: Other business rules that protect the validity of attribute values.

Domains: A domain is the set of all data types and ranges of values that attributes may assume.

Triggering operations: A triggering operation (also called a trigger) is an assertion or rule that governs the validity of data manipulation operations such as insert, update, and delete. The scope of triggering operations may be limited to attributes within one entity or it may extend to attributes in two or more entities.

7. Role of Packaged Conceptual Data Models in Database:

There are two principal types of packaged data models: **universal data models** applicable to nearly any business or organization and **industry-specific data models**.

a. Universal Data Models:

Numerous core subject areas are common to many (or even most) organizations, such as customers, products, accounts, documents, and projects. Although they differ in detail, the underlying data structures are often quite similar for these subjects. Further, there are core business functions such as purchasing, accounting, receiving, and project management that follow common patterns. Universal data models are templates for one or more of these subject areas and/or functions. All of the expected components of data models are generally included: **entities, relationships, attributes, primary and foreign keys, and even sample data.**

- **Identify the entities.** The process of data modeling begins with the identification of the things, events or concepts that are represented in the data set that is to be modeled. Each entity should be cohesive and logically discrete from all others.
- **Identify key properties of each entity.** Each entity type can be differentiated from all others because it has one or more unique properties, called attributes. For instance, an entity called "customer" might possess such attributes as a first name, last name, telephone number and salutation, while an entity called "address" might include a street name and number, a city, state, country and zip code.
- Identify relationships among entities. The earliest draft of a data model will specify the nature of the relationships each entity has with the others. In the above example, each customer "lives at" an address. If that model were expanded to include an entity called "orders," each order would be

shipped to and billed to an address as well. These relationships are usually documented via unified modeling language (UML).

- **Map attributes to entities completely.** This will ensure the model reflects how the business will use the data. Several formal data modeling patterns are in widespread use. Object-oriented developers often apply analysis patterns or design patterns, while stakeholders from other business domains may turn to other patterns.
- **Assign keys as needed, and decide on a degree of normalization that balances the need to reduce redundancy with performance requirements.** Normalization is a technique for organizing data models (and the databases they represent) in which numerical identifiers, called keys, are assigned to groups of data to represent relationships between them without repeating the data. For instance, if customers are each assigned a key, that key can be linked to both their address and their order history without having to repeat this information in the table of customer names. Normalization tends to reduce the amount of storage space a database will require, but it can at cost to query performance.
- **Finalize and validate the data model**. Data modeling is an iterative process that should be repeated and refined as business needs change.

b. Industry-Specific Data Models:

Industry-specific data models are generic data models that are designed to be used by organizations within specific industries. Data models are available for nearly every major industry group, including health care, telecommunications, discrete manufacturing, process manufacturing, banking, insurance, and higher education. These models are based on the premise that data model patterns for organizations are very similar within a particular industry ("a bank is a bank"). However, the data models for one industry (such as banking) are quite different from those for another (such as hospitals).

Chapter 4: Design

A. Designing Databases

1. Introduction

File and database design occurs in two steps.

Develop a logical database model, which describes data using notation that corresponds to a data organization used by a database management system.

- Relational database model.

Prescribe the technical specifications for computer files and databases in which to store the data.

- Physical database design provides specifications.

Logical and physical database design in parallel with other system design steps.

Database design occurs in three stages:

a. Conceptual Design
- Entities and their relationship are identified by analyzing input and output of the system

b. Logical Design
- Attributes are identified
- Primary keys are defines
- ER diagram is used for logical representation

c. Physical Design
- Convert ER diagram to database tables
- Foreign keys are defined
- Normalization

2. Database Design

Features	Conceptual Design	Logical Design	Physical Design
Entity Names	√	√	
Entity Relationships	√	√	
Attributes		√	√
Primary Keys		√	√
Foreign Keys			√
Table Names			√
Normalization			√

Generic steps of database design are:

- Discover and organize entities and their attributes
- Define primary keys
- Define the relationships between different entities
- Create ER diagram
- Convert ER diagram to tables using sample data using sample data
- Normalize the tables

Process of Database Design

Physical Design

Based upon results of logical database design

Key decisions

- Choosing the storage format (called data type) for each attribute from the logical database model; the format is chosen to minimize storage space and to maximize data quality. Data type involves choosing length, coding scheme, number of decimal places, minimum and maximum values, and potentially many other parameters for each attribute.
- Grouping attributes from the logical database model into physical records (in general, this is called selecting a stored record, or data, structure).
- Arranging related records in secondary memory (hard disks and magnetic tapes) so that individual records and groups of records can be stored, retrieved, and updated rapidly (called file organization). You should also consider protecting data and recovering data after errors are found.
- Selecting media and structures for storing data to make access more efficient. The choice of media affects the utility of different file organizations. The primary structure used today to make access to data more rapid is key indexes on unique and no unique keys.

3. Relational Database Model

Data represented as a set of related tables or relations.

Relation is two-dimensional table of data. Each relation consists of a set of named columns and an arbitrary number of unnamed rows.

Relations have several properties that distinguish them from non-relational tables:

- Entries in cells are simple. An entry at the intersection of each row and column has a single value.
- Entries in a given column are from the same set of values.
- Each row is unique. Uniqueness is guaranteed because the relation has a non-empty primary key value.
- The sequence of columns can be interchanged without changing the meaning or use of the relation.
- The rows may be interchanged or stored in any sequences.

Well-structured relation

A relation that contains a minimum amount of redundancy and that allows users to insert, modify, and delete the rows without error or inconsistencies; also known as a table.

4. Normalization

We have presented an intuitive discussion of well-structured relations; however, we need rules and a process for designing them.

Normalization is a process for converting complex data structures into simple, stable data structures.

We have presented an intuitive discussion of well-structured relations; however, we need rules and a process for designing them. Normalization is a process for con verting complex data structures into simple, stable data structures

Rules of Normalization

Normalization is based on well-accepted principles and rules. There are many normalization rules, more than can be covered (for a more complete coverage).

a. First Normal Form (1NF)

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First normal form disallows the multi-valued attribute, composite attribute, and their combinations.

Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

b. Second Normal Form (2NF)

- In the 2NF, relational must be in 1NF.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key

Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.

To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology
47	English
83	Math

c. Third Normal Form (3NF)

- A relation will be in 3NF if it is in 2NF and not contain any transitive partial dependency.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds at least one of the following conditions for every non-trivial function dependency X → Y.

1. X is a super key.
2. Y is a prime attribute, i.e., each element of Y is part of some candidate key.

Example:

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Super key in the table above:

1. {EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on

Candidate key: {EMP_ID}

Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID. The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key (EMP_ID). It violates the rule of third normal form.

That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

d. Boyce Codd normal form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is super key.

Example: Let's assume there is a company where employees work in more than one department.

EMPLOYEE table:

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

In the above table Functional dependencies are as follows:

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate key: {EMP-ID, EMP-DEPT}

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.

To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

Functional dependencies:

1. EMP_ID → EMP_COUNTRY
2. EMP_DEPT → {DEPT_TYPE, EMP_DEPT_NO}

Candidate keys:

For the first table: EMP_ID
For the second table: EMP_DEPT
For the third table: {EMP_ID, EMP_DEPT}

Now, this is in BCNF because left side part of both the functional dependencies is a key.

e. Fourth normal form (4NF)

- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
- For a dependency A → B, if for a single value of A, multiple values of B exists, then the relation will be a multi-valued dependency.

Example

STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.

In the STUDENT relation, a student with STU_ID, **21** contains two courses, **Computer** and **Math** and two hobbies, **Dancing** and **Singing**. So there is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data.

So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

f. Fifth normal form (5NF)

- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.
- 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy.
- 5NF is also known as Project-join normal form (PJ/NF).

Example

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

5. Transforming E-R Diagrams into Relations

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 nonkey attributes associative entity 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 foreign key attributes).
Supertype/subtype	Create a relation for the superclass, which contains the primary relationship key and all nonkey attributes in common with all subclasses, plus create a separate relation for each subclass with the same primary key (with the same or local name) but with only the nonkey attributes related to that subclass.

Figure 5-8: Mapping a regular entity

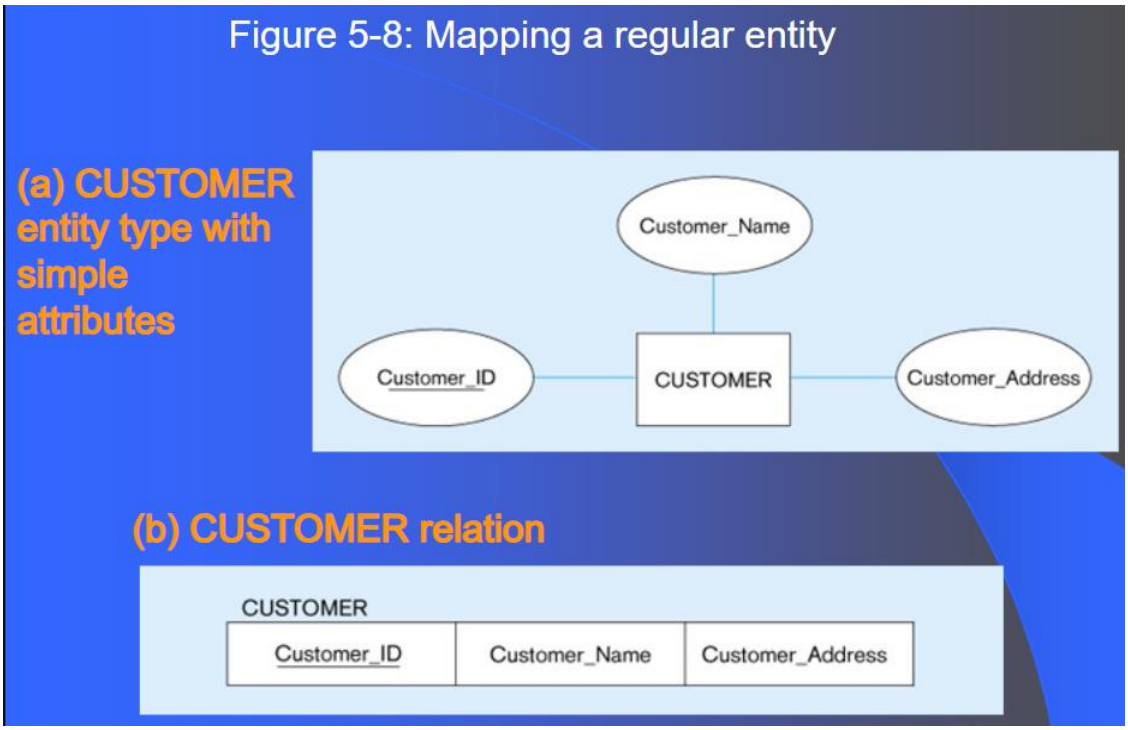


Figure 5-9: Mapping a composite attribute

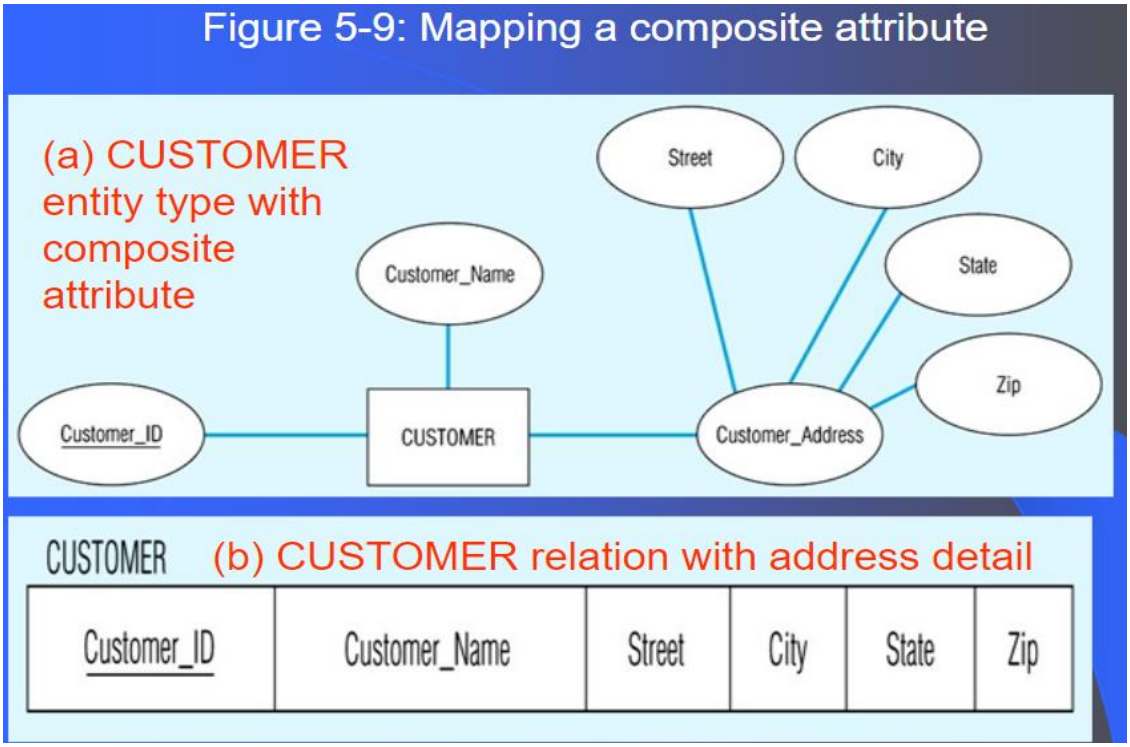


Figure 5-10: Mapping a multivalued attribute

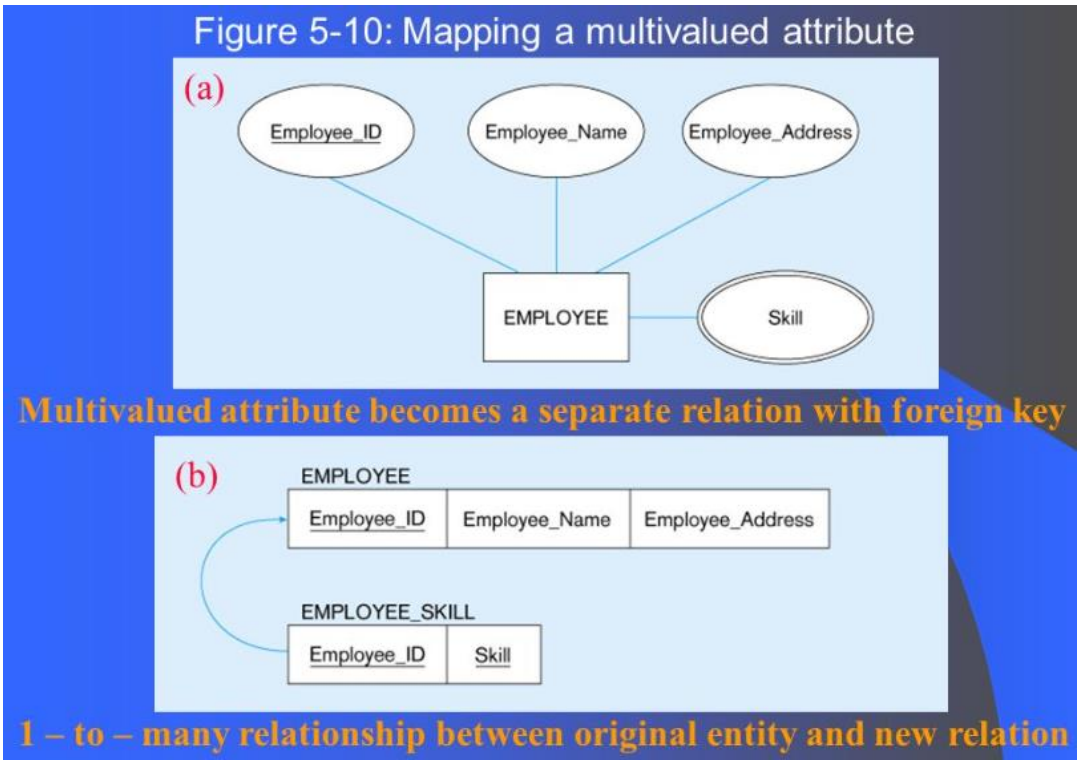


Figure 5-11: Example of mapping a weak entity

(a) Weak entity DEPENDENT

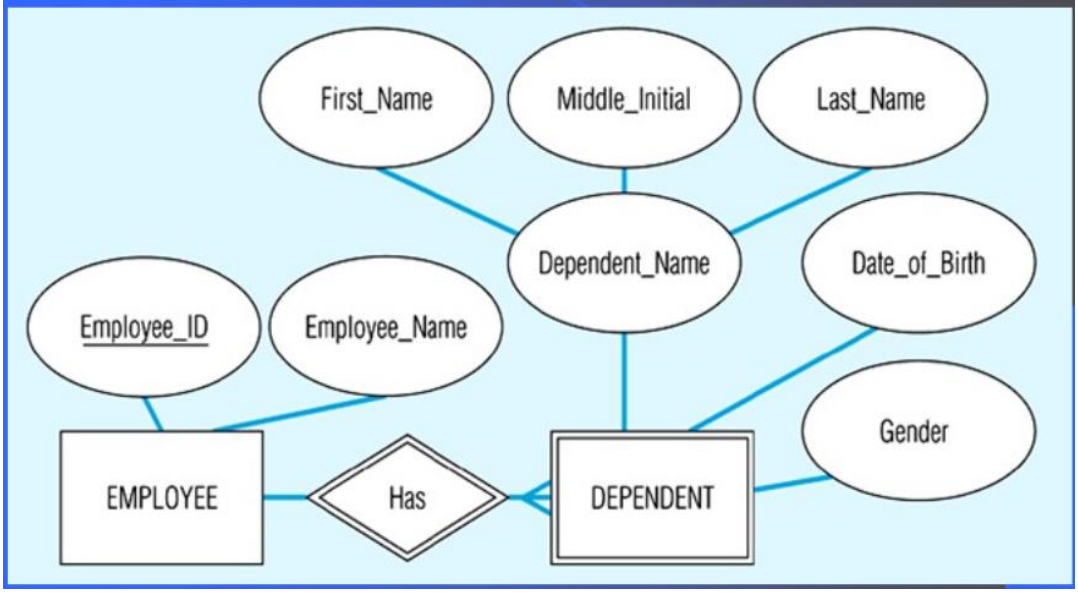


Figure 5-11(b) Relations resulting from weak entity

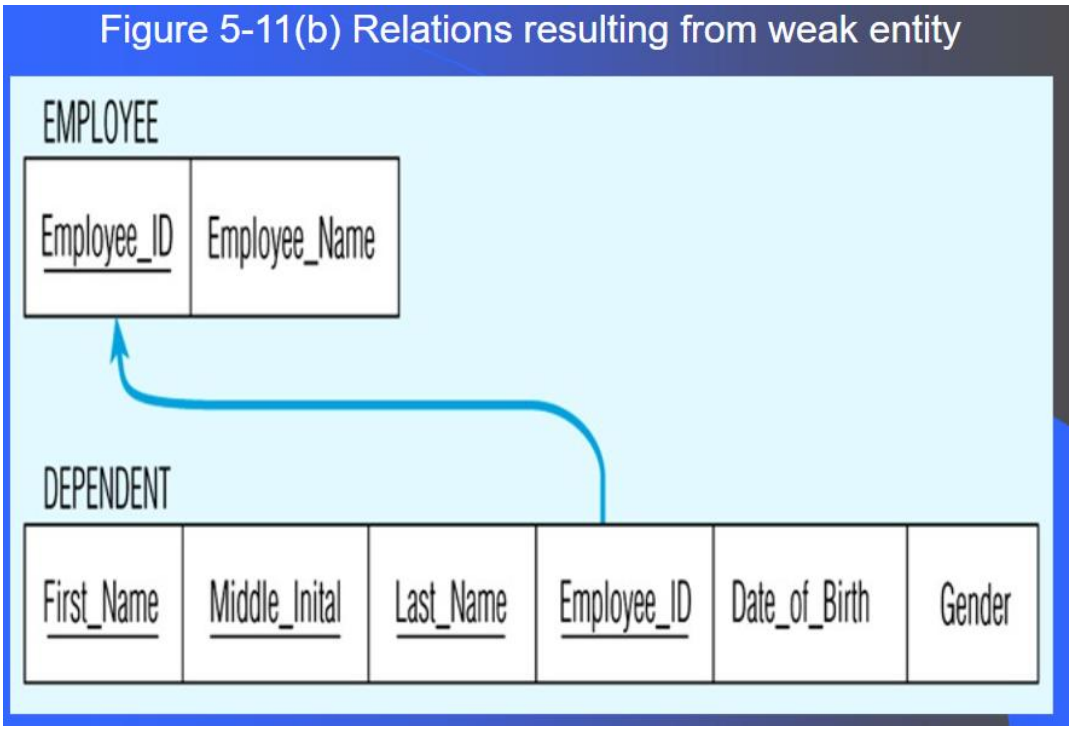


Figure 5-12: Example of mapping a 1:M relationship

(a) Relationship between customers and orders

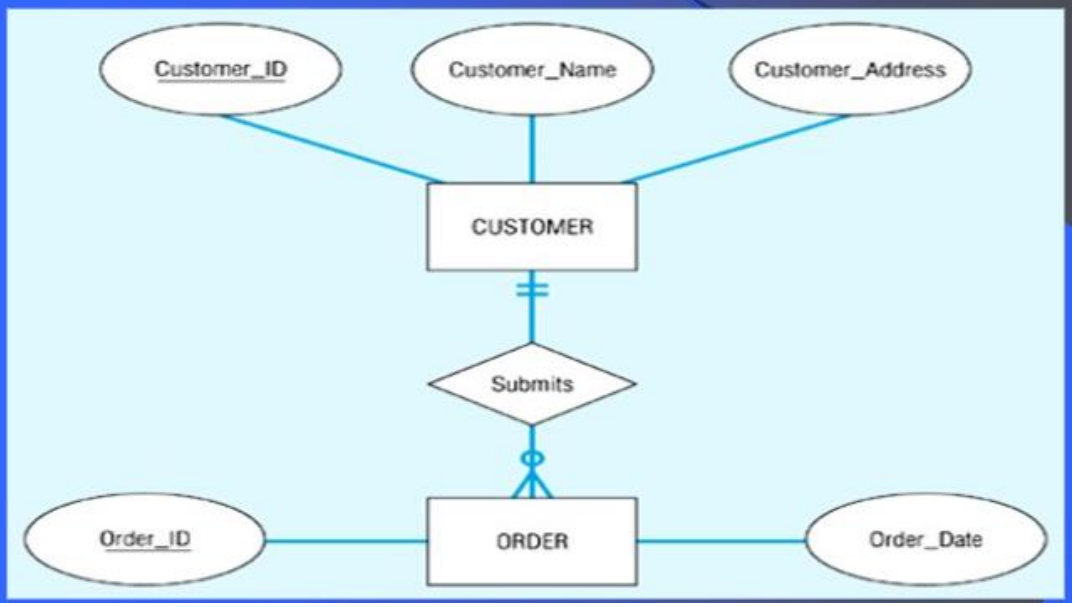


Figure 5-12(b) Mapping the relationship

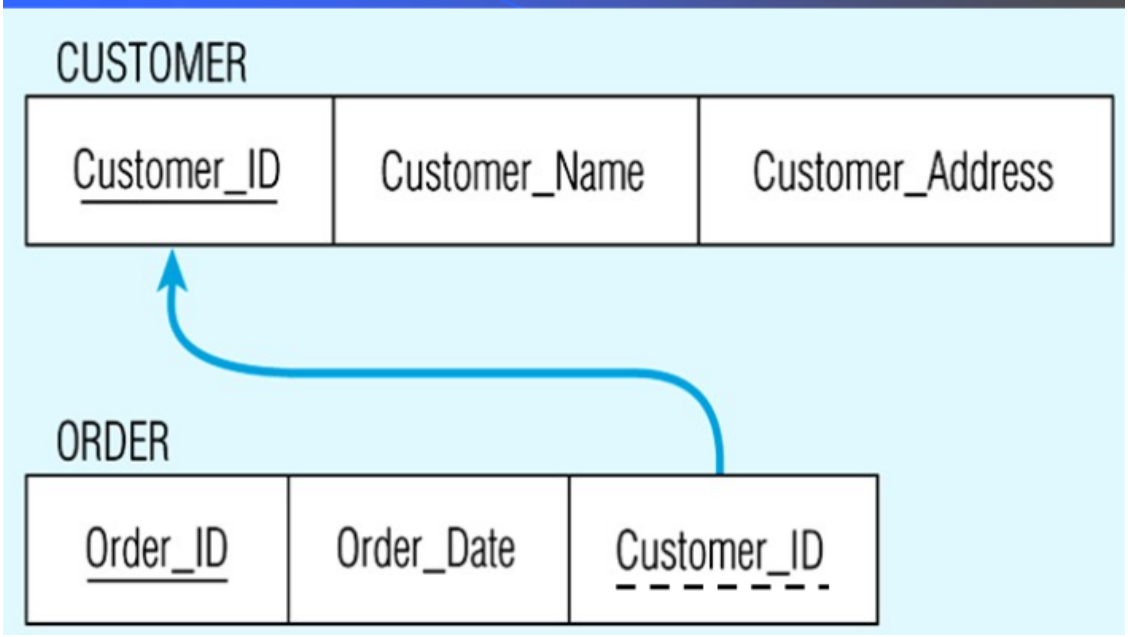


Figure 5-14: Mapping a binary 1:1 relationship

(a) Binary 1:1 relationship

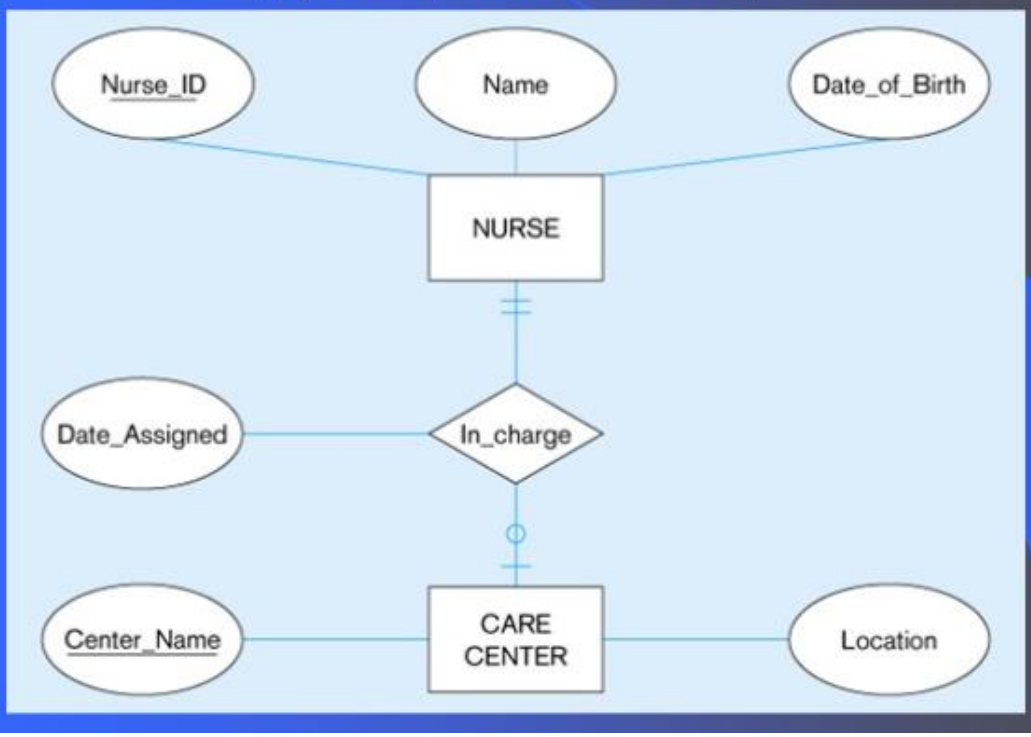


Figure 5-14(b) Resulting relations

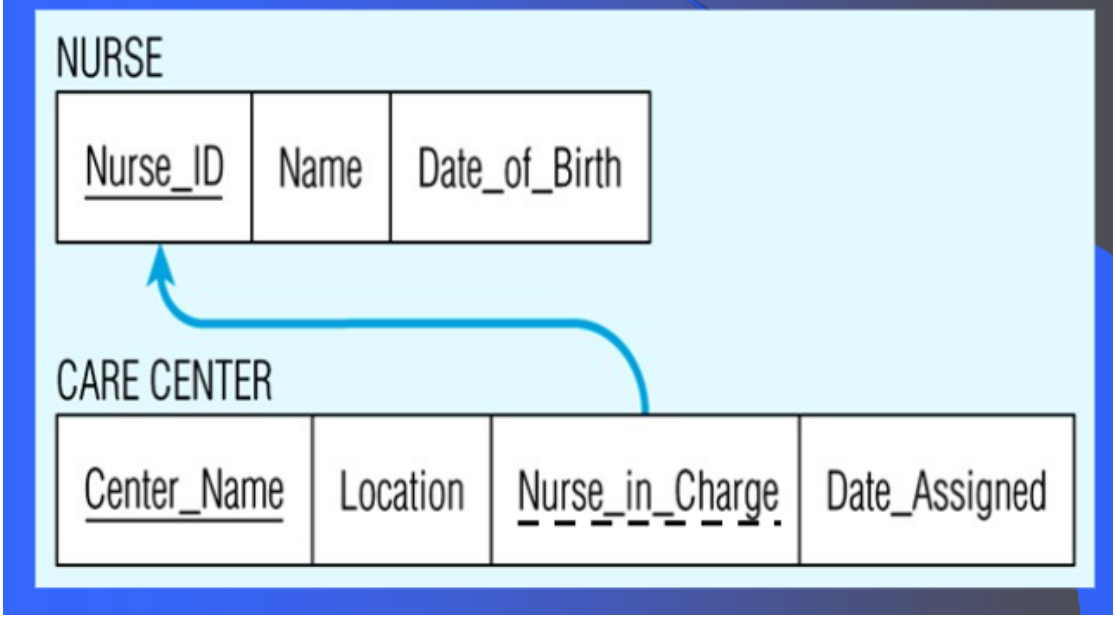


Figure 5-13: Example of mapping an M:N relationship

(a) ER diagram (M:N)

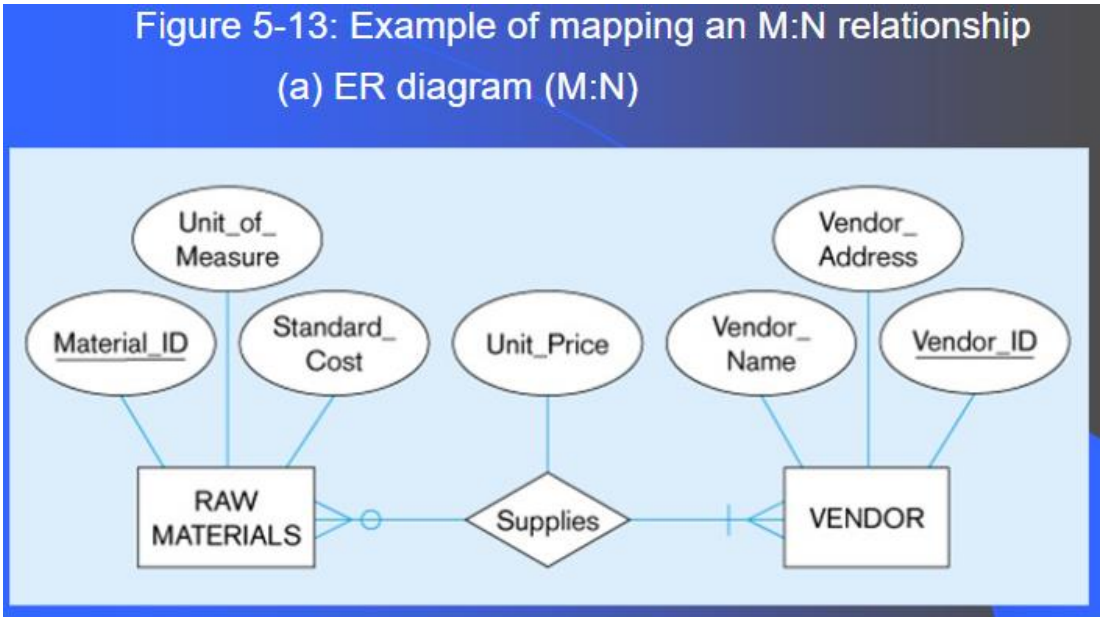


Figure 5-13(b) Three resulting relations

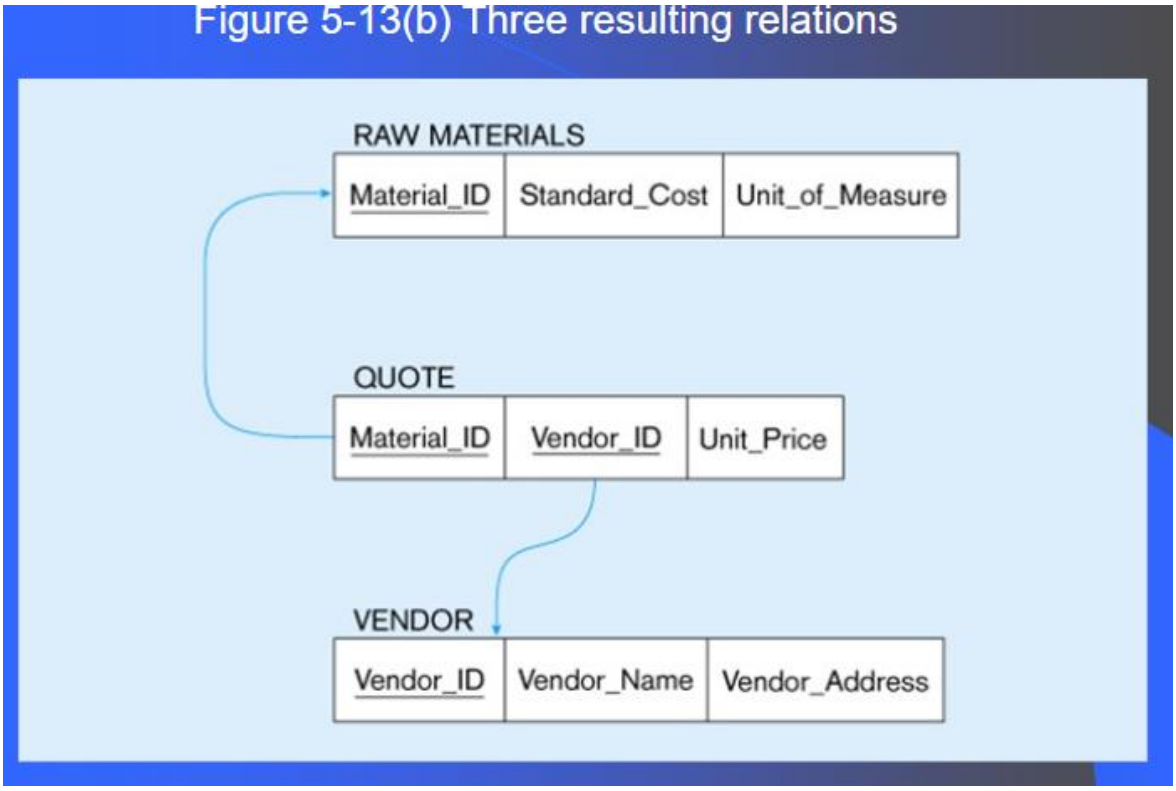


Figure 5-15: Mapping an associative entity

(a) Associative entity

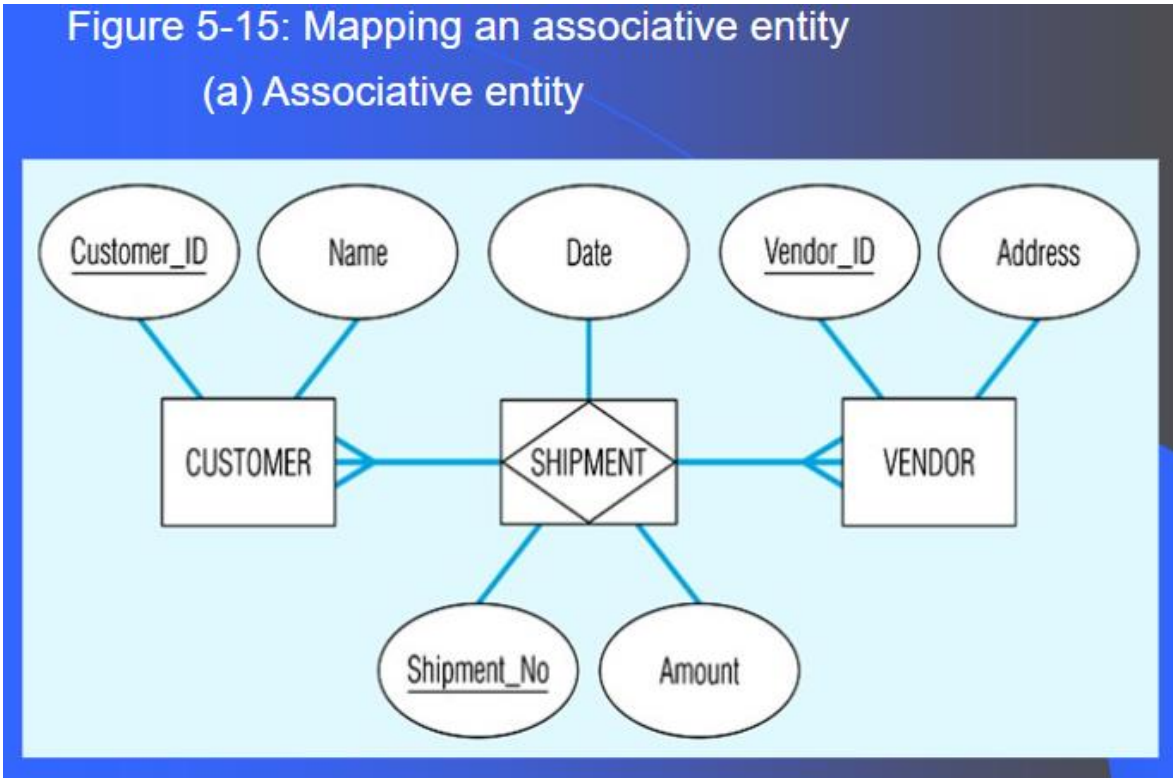


Figure 5-15(b) Three resulting relations

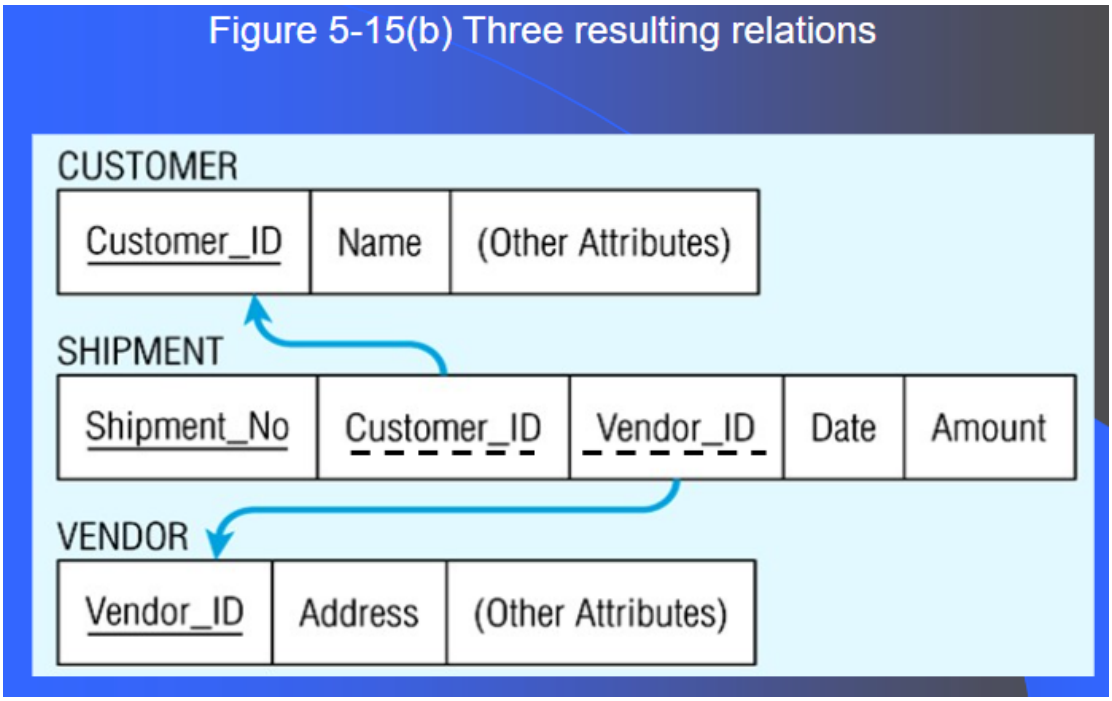


Figure 5-17: Mapping a unary 1:N relationship

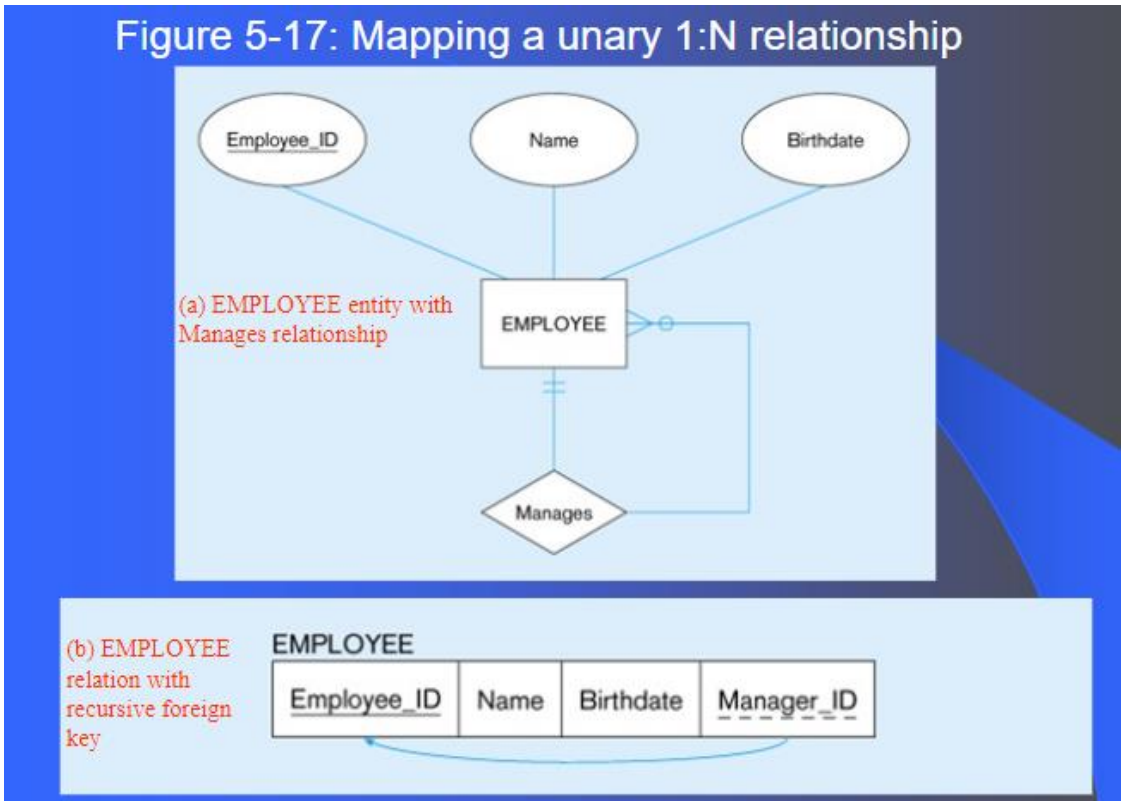
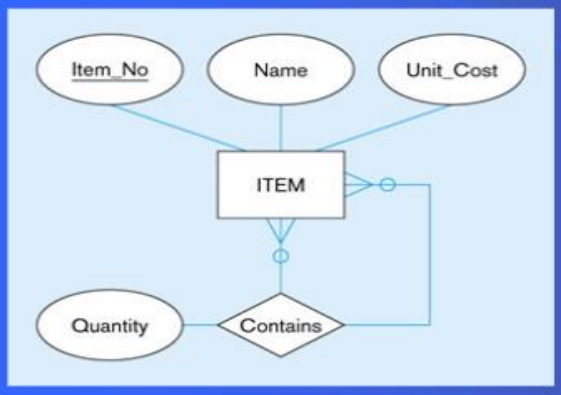


Figure 5-18: Mapping a unary M:N relationship



(a) Bill-of-materials relationships (M:N)

(b) ITEM and COMPONENT relations

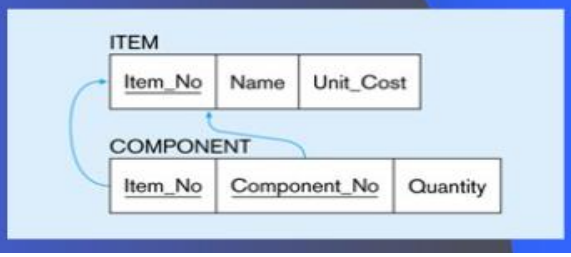


Figure 5-19: Mapping a ternary relationship

(a) Ternary relationship with associative entity

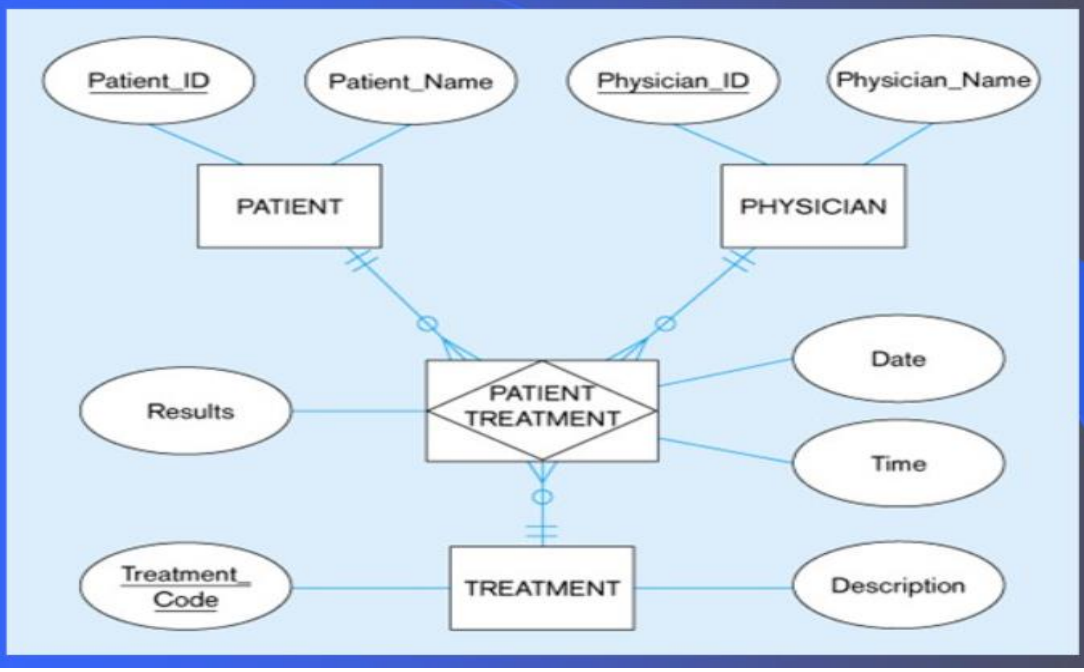


Figure 5-19(b) Mapping the ternary relationship

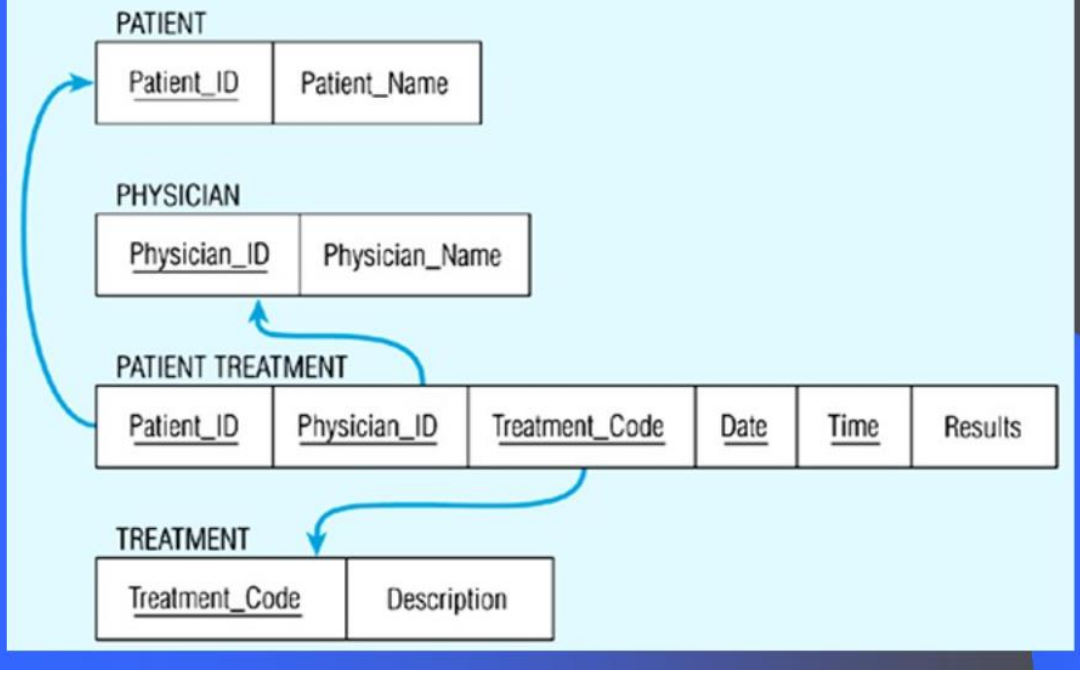
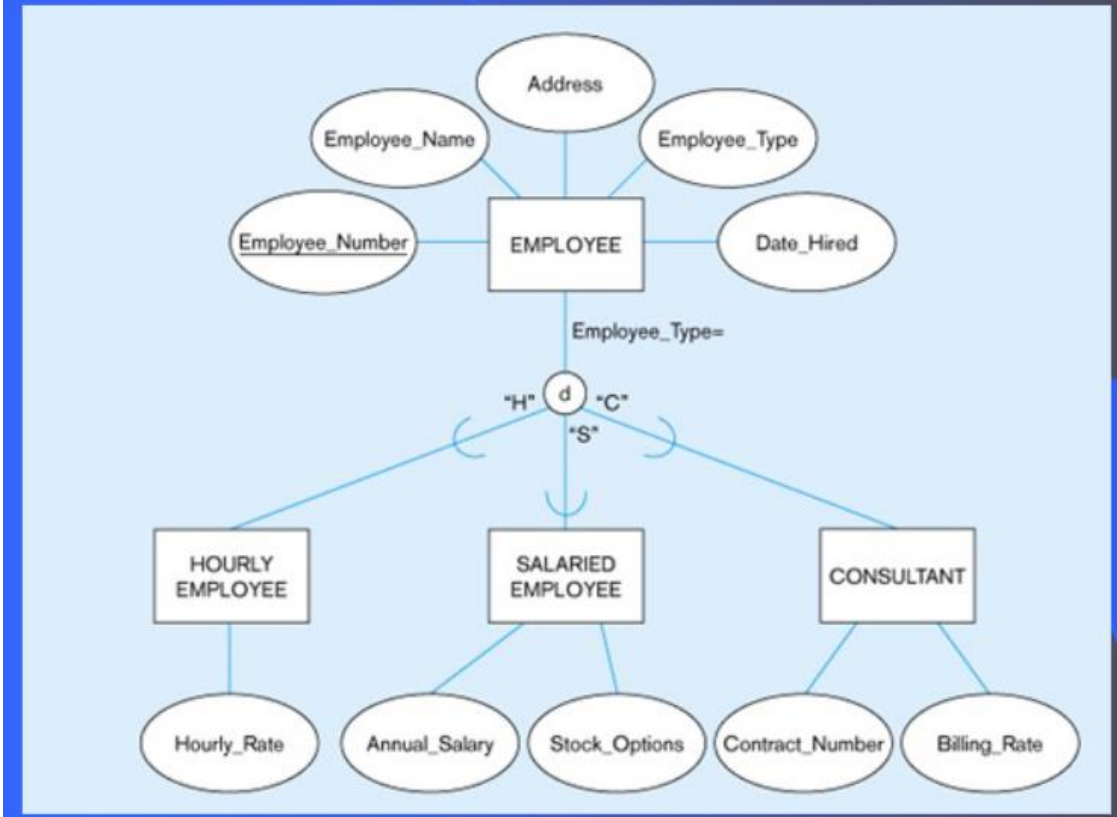
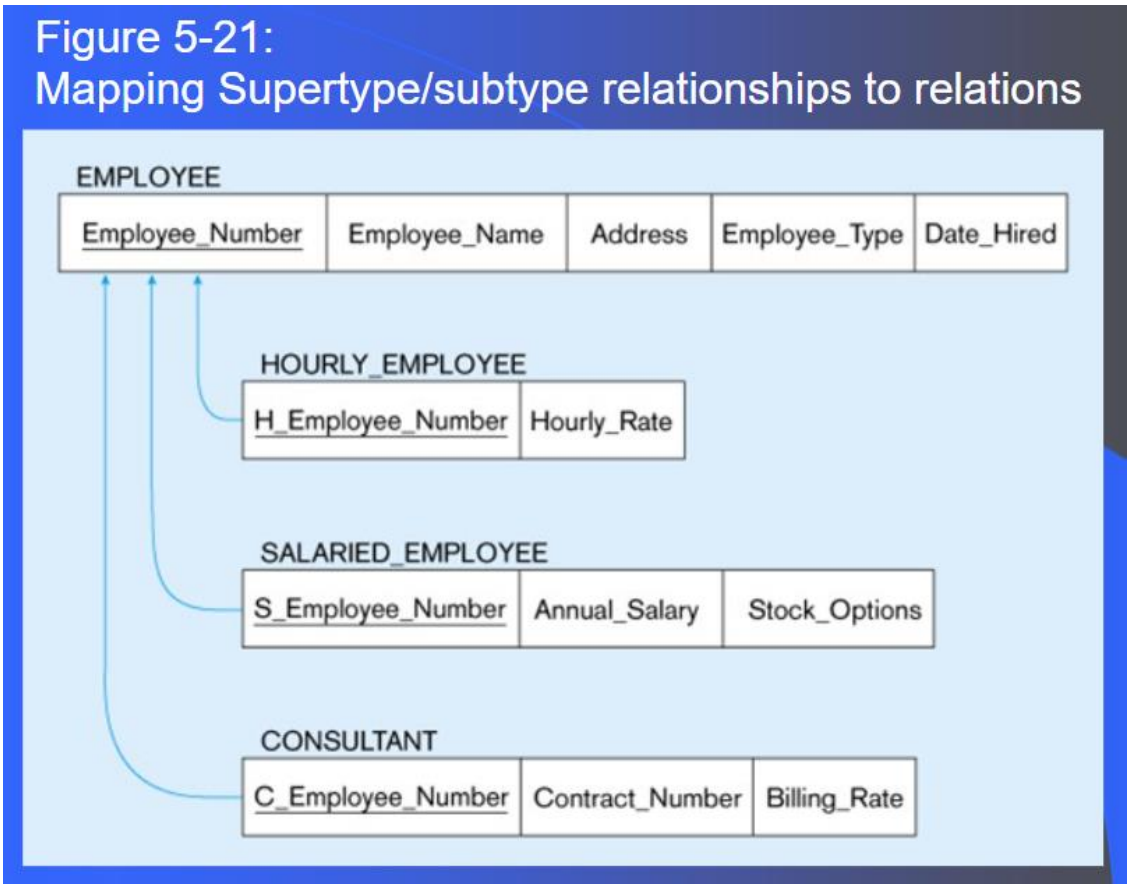


Figure 5-20: Supertype/subtype relationships





6. Physical File and Database Design

Designing physical files and databases requires certain information that should have been collected and produced during prior SDLC phases.

This information includes the following:

Normalized relations, including volume estimates

- Definitions of each attribute
- Descriptions of where and when data are used: entered, retrieved, deleted, and updated (including frequencies)
- Expectations or requirements for response time and data integrity
- Descriptions of the technologies used for implementing the files and database so that the range of required decisions and choices for each is known

Normalized relations are, of course, the result of logical database design. Statistics on the number of rows in each table as well as the other information listed above may have been collected during requirements determination in systems analysis. If not, these items need to be discovered to proceed with database design.

We take a bottom-up approach to reviewing physical file and database design. Thus, we begin the physical design phase by addressing the design of physical fields for each attribute in a logical data model.

7. Designing Fields

A field is the smallest unit of application data recognized by system software, such as a programming language or database management system. An attribute from a logical database model may be represented by several fields.

For example, a student name attribute in a normalized student relation might be represented as three fields: last name, first name, and middle initial. In general, you will represent each attribute from each normalized relation as one or more fields.

The basic decisions you must make in specifying each field concern the type of data (or storage type) used to represent the field and data integrity controls for the field.

Choosing Data types:

A data type is a coding scheme recognized by system software for representing organizational data. The bit pattern of the coding scheme is usually immaterial to you, but the space to store data and the speed required to access data are of consequence in the physical file and database design. The specific file or database management software you use with your system will dictate which choices are available to you.

Selecting a data type balances four objectives that will vary in degree of impor tance depending on the application:

- Minimize storage space
- Represent all possible values of the field
- Improve data integrity for the field
- Support all data manipulations desired on the field

Calculated Fields: It is common for an attribute to be mathematically related to other data. A field that can be derived from other database fields is called a calculated field (or a computed field or a derived field). If you specify a field as calculated, you would then usually be prompted to enter the formula for the calculation; the formula can involve other fields from the same record and possibly fields from records in related files. The database technology will either store the calculated value or compute it when requested.

Coding and Compression Techniques: Some attributes have very few values from a large range of possible values. **For example,** suppose that each product from PVF has a finish attribute, with possible values of Birch, Walnut, Oak, and so forth. To store this attribute as text might require 12, 15, or even 20 bytes to represent the longest finish value. Suppose that even a liberal estimate is that PVF will never have more than 25 finishes. Thus, a single alphabetic or alphanumeric character would be more than sufficient. We not only reduce storage space but also increase integrity (by restricting input to only a few values. Codes also have disadvantages. If used in system inputs and outputs, they can be more difficult for users to remember, and programs must be written to decode fields if codes will not be displayed.

8. Designing Physical tables:

A relational database is a set of related tables (tables are related by foreign keys referencing primary keys). In logical database design, you grouped into a relation those attributes that concern some unifying, normalized business concept, such as a customer, product, or employee.

Physical table is a named set of rows and columns that specifies the fields in each row of the table. The design of a physical table has two goals: efficient use of secondary storage and data processing speed.

The efficient use of secondary storage (disk space) relates to how data are loaded on disks. Disks are physically divided into units (called pages) that can be read or written in one machine operation. Space is used efficiently when the physical length of a table row divides close to evenly into the length of the storage unit. For many information systems, this even division is very difficult to achieve because it depends on factors, such as operating system parameters, outside the control of each database.

A second and often more important consideration when selecting a physical table design is efficient data processing. Data are most efficiently processed when they are stored close to one another in secondary memory, thus minimizing the number of input/output (I/O) operations that must be performed. Typically, the data in one physical table (all the rows and fields in those rows) are stored close together on disk.

B. Designing Forms and Reports

1. Introduction

Form: A business document that contains some organizational related 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.

Report: A business document that contains only predefined data. A passive document for reading or viewing data. Typically contains data from many databases records or transactions.

Common Types of Business Reports

Report Name	Description
Scheduled Reports	Reports produced at predefined intervals—daily, weekly, or monthly—to support the routine informational needs of an organization.
Key-Indicator Reports	Reports that provide a summary of critical information on a recurring basis.
Exception Reports	Reports that highlight data that are out of the normal operating range.
Drill-Down Reports	Reports that provide details behind the summary values on a key-indicator or exception report.
Ad-hoc Reports	Unplanned information requests in which information is gathered to support a nonroutine decision.

2. Designing Forms and Reports

- ✓ User-focused activity
- ✓ Follows a prototyping approach

Requirements determination:

- Who will use the form or report?
- What is the purpose of the form or report?
- When is the report needed or used?
- Where does the form or report need to be delivered and used?
- How many people need to use or view the form or report?

Prototyping

- Initial prototype is designed from requirements.
- Users review prototype design and either accept the design or request changes.
- If changes are requested, the construction-evaluation-refinement cycle is repeated until the design is accepted.

A coding sheet is an "old" tool for designing forms and reports, usually associated with text-based forms and reports for mainframe applications.

Visual Basic and other development tools provide computer-aided GUI form and report generation.

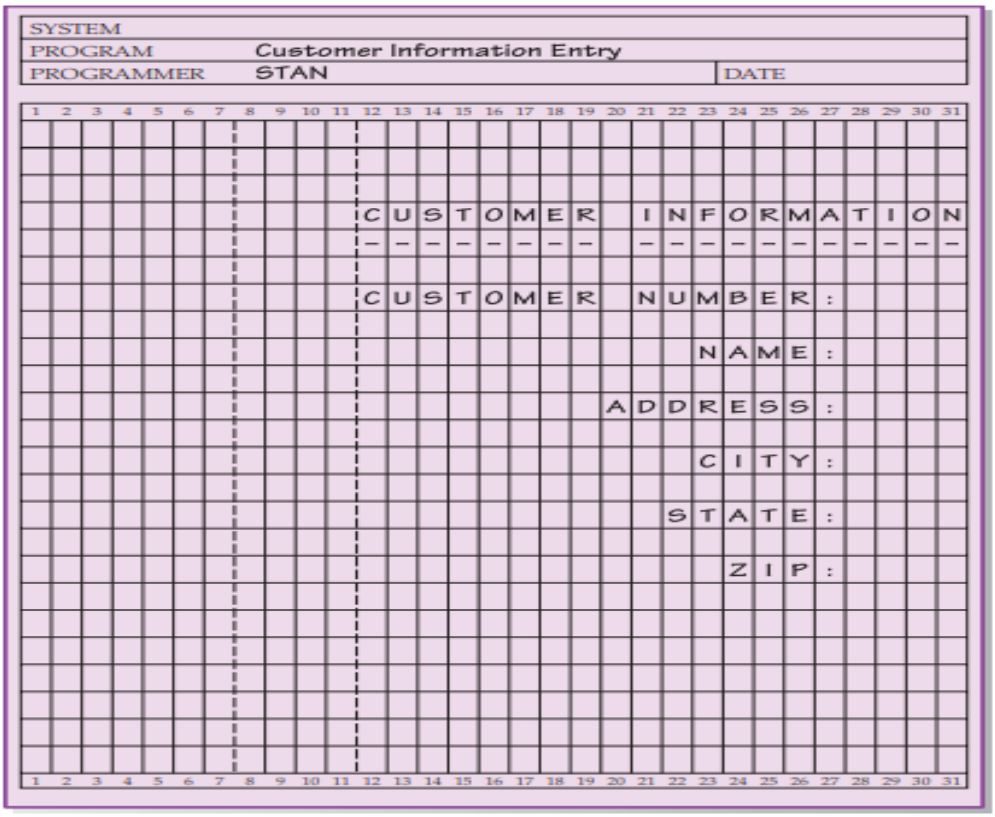
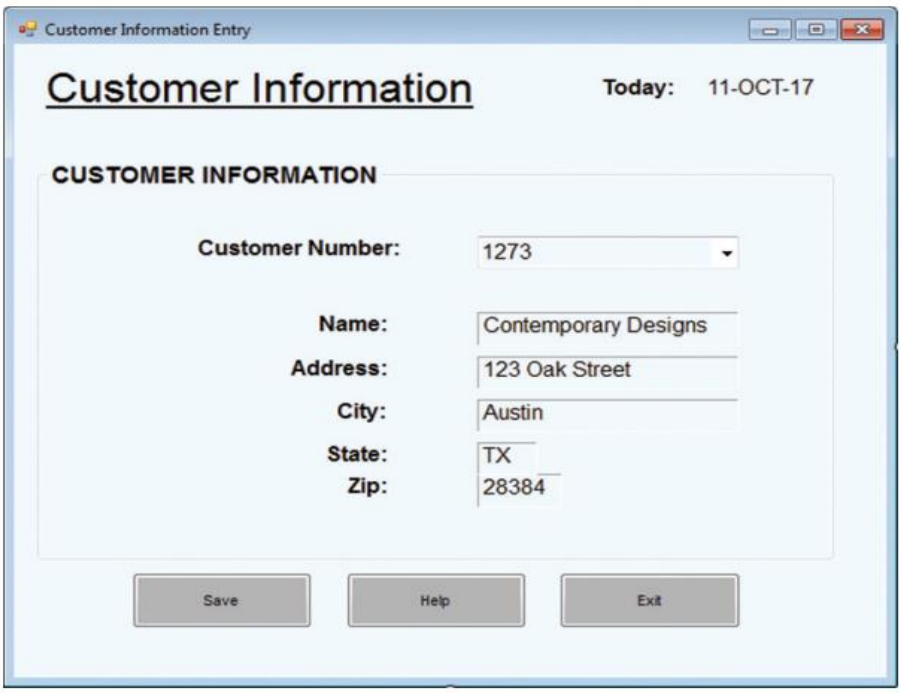


FIGURE 10-2
The layout of a data input form using a coding sheet

FIGURE 10-3B
A data input screen designed in Microsoft's Visual Basic.NET
(*Source:* Microsoft Corporation.)



Deliverables and outcomes
Design specifications have three sections:

- **Narrative overview**: characterizes users, tasks, system, and environmental factors
- **Sample design:** image of the form (from coding sheet or form building development tool)
- **Testing and usability assessment:** measuring test/usability results (consistency, sufficiency, accuracy, etc.)

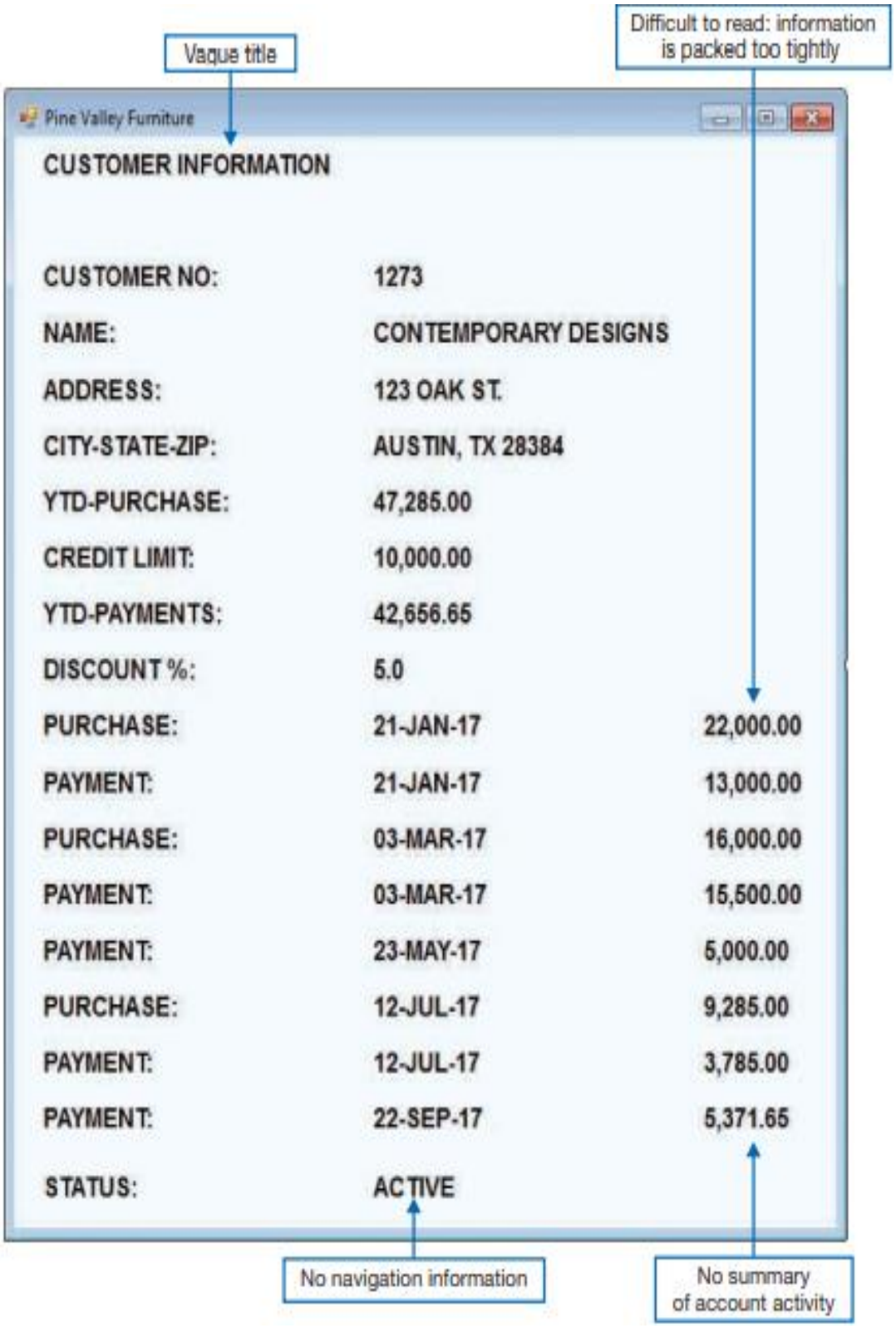
3. Formatting Forms and Reports

Guidelines:

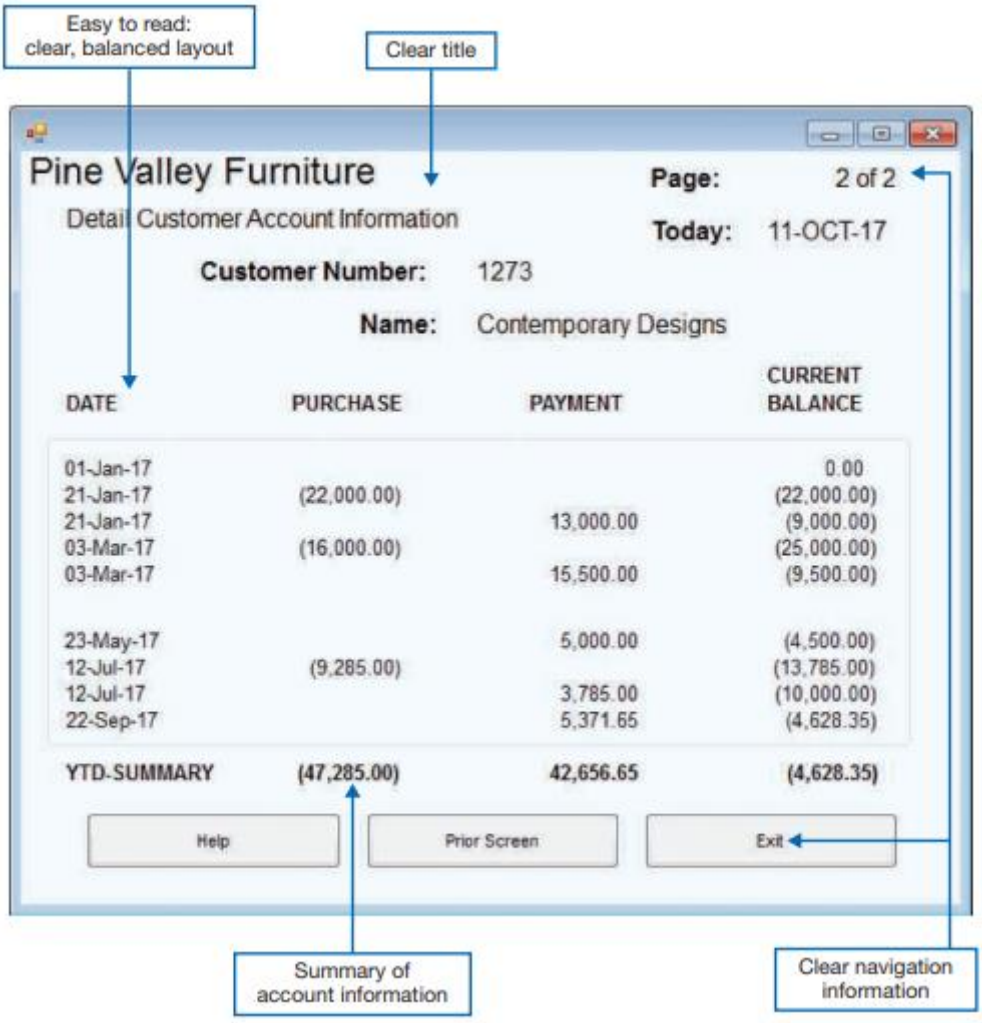
- **Meaningful titles:** clear, specific, version information, current date
- **Meaningful information:** include only necessary information
- **Balanced layout:** adequate spacing, margins, and clear labels
- **Easy navigation system:** show how to move forward and backward, and where you are currently

General Formatting Guidelines

Poorly Designed form



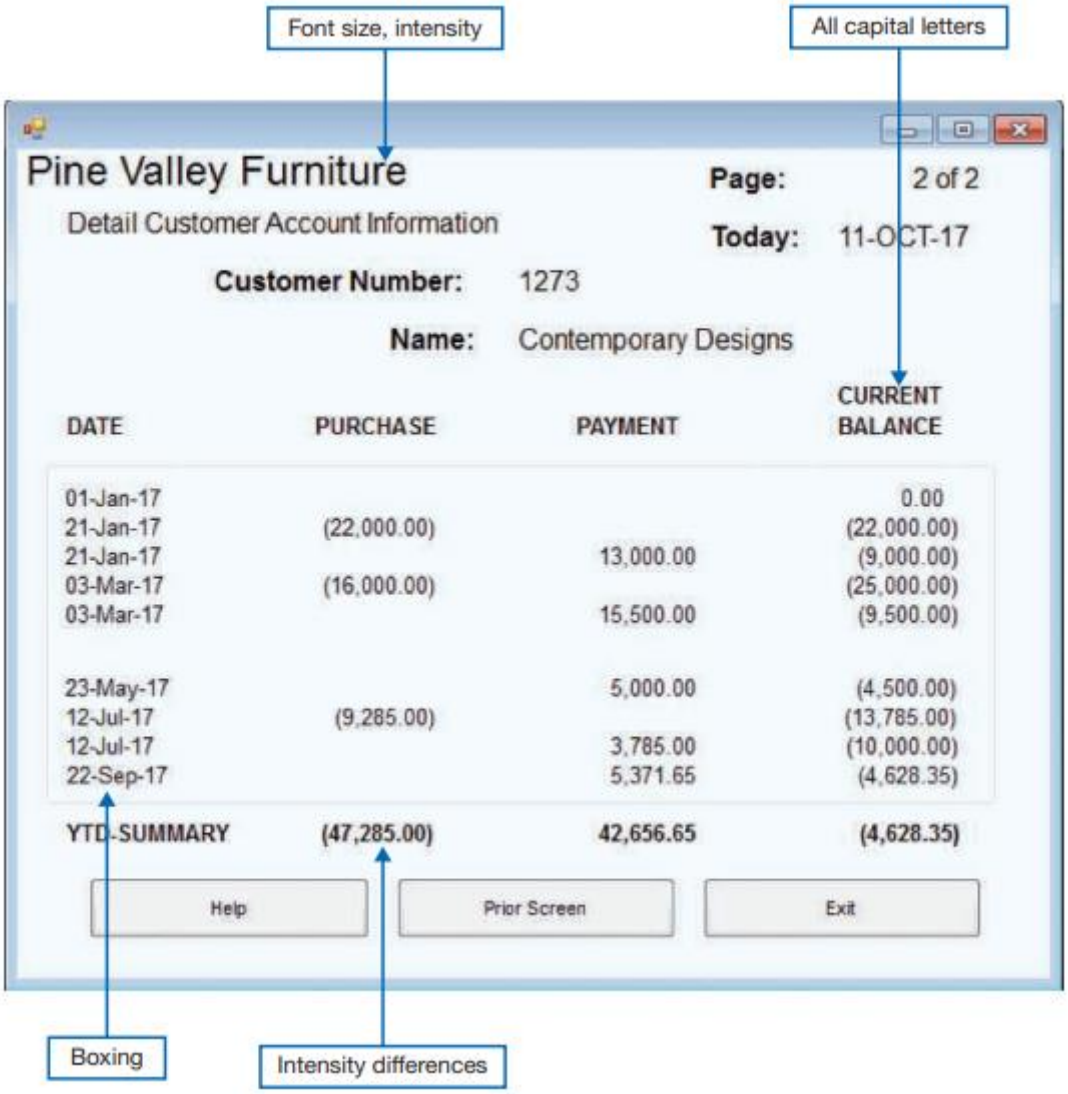
Improved design of Form



Highlighting Information

There are several situations when highlighting can be a valuable technique for conveying special information:

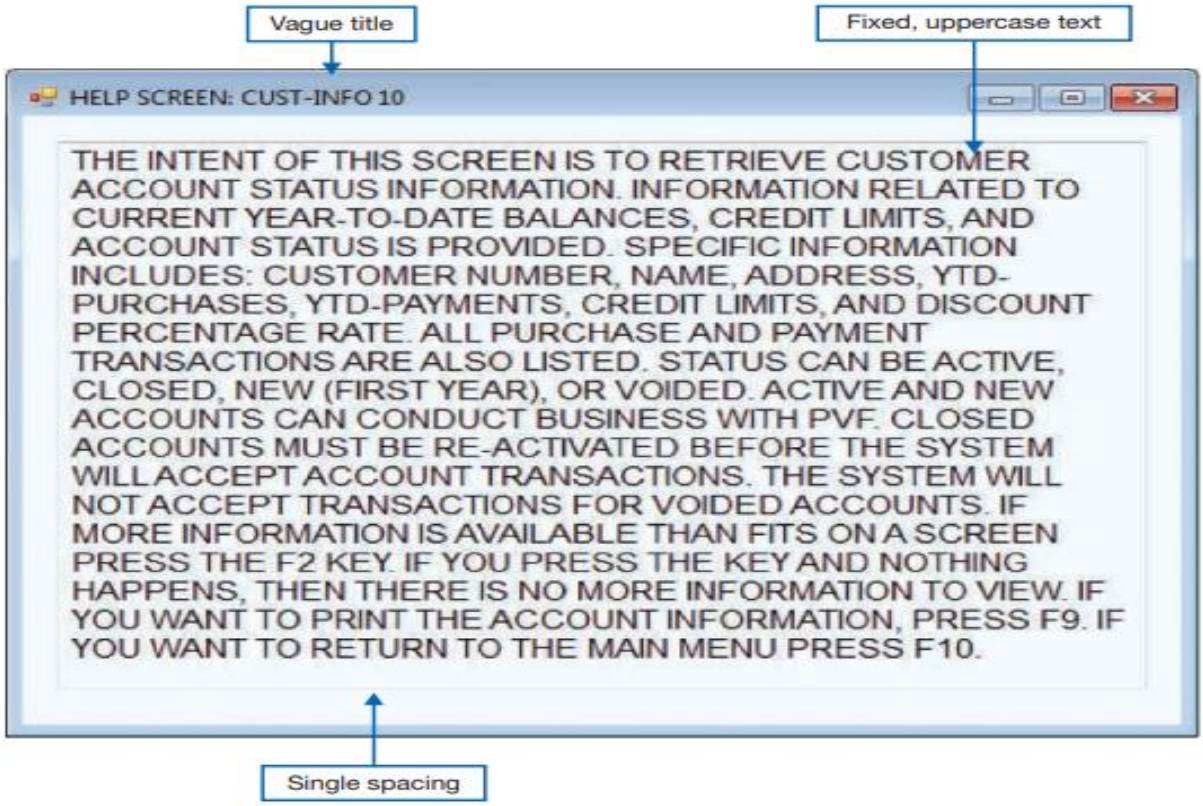
- Notifying users of errors in data entry or processing
- Providing warnings to users regarding possible problems such as unusual data values or an unavailable device
- Drawing attention to keywords, commands, high-priority messages, and data that have changed or gone outside normal operating ranges



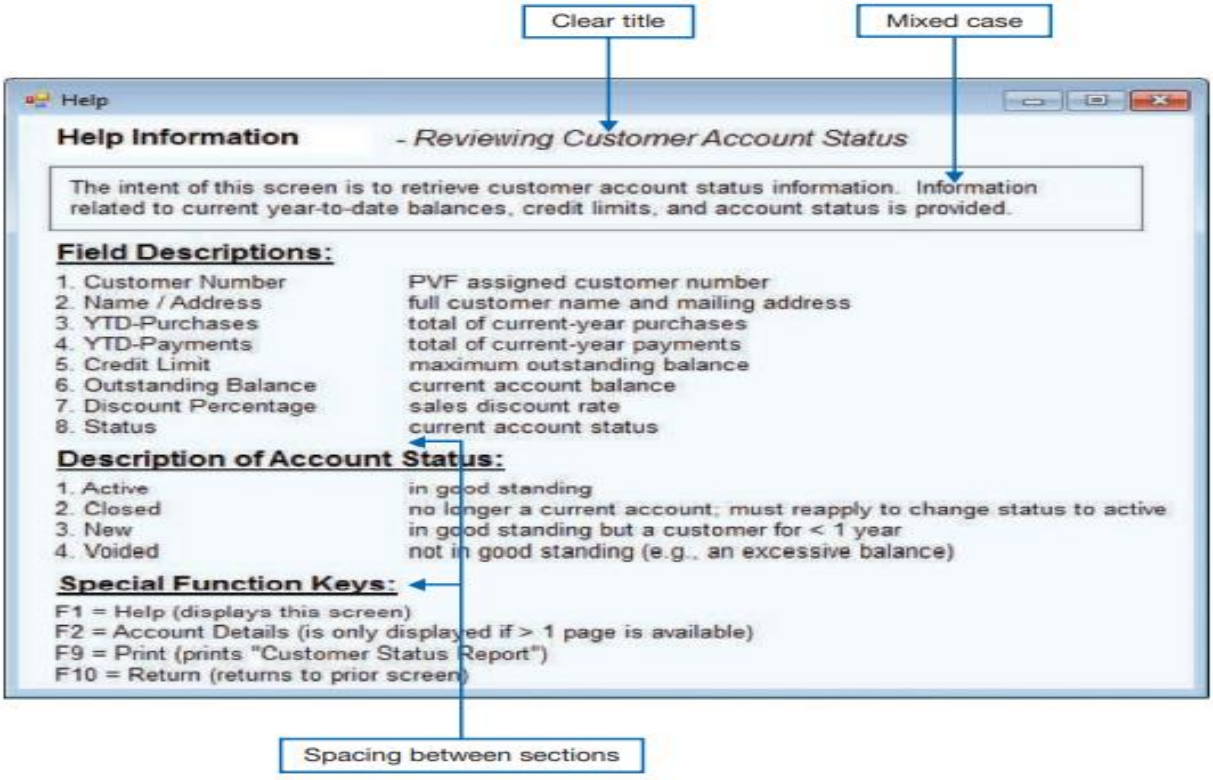
Guidelines for Displaying Text

Case	Display text in mixed uppercase and lowercase and use conventional punctuation.
Spacing	Use double spacing if space permits. If not, place a blank line between paragraphs.
Justification	Left-justify text and leave a ragged-right margin.
Hyphenation	Do not hyphenate words between lines.
Abbreviations	Use abbreviations and acronyms only when they are widely understood by users and are significantly shorter than the full text.

Poor Design for displaying Text



Improved design for displaying Text



Color vs. No Color

Benefits - Color:

- Soothes or strikes the eye.
- Accents an uninteresting display.
- Facilitates subtle discriminations in complex displays.
- Emphasizes the logical organization of information.
- Draws attention to warnings.
- Evokes more emotional reactions.

Problems from Using Color

- Color pairings may wash out or cause problems for some users.
- Resolution may degrade with different displays.
- Color fidelity may degrade on different displays.
- Printing or conversion to other media may not easily translate.

Designing Tables and Lists

Labels

- All columns and rows should have meaningful labels.
- Labels should be separated from other information by using highlighting.
- Redisplay labels when the data extend beyond a single screen or page.

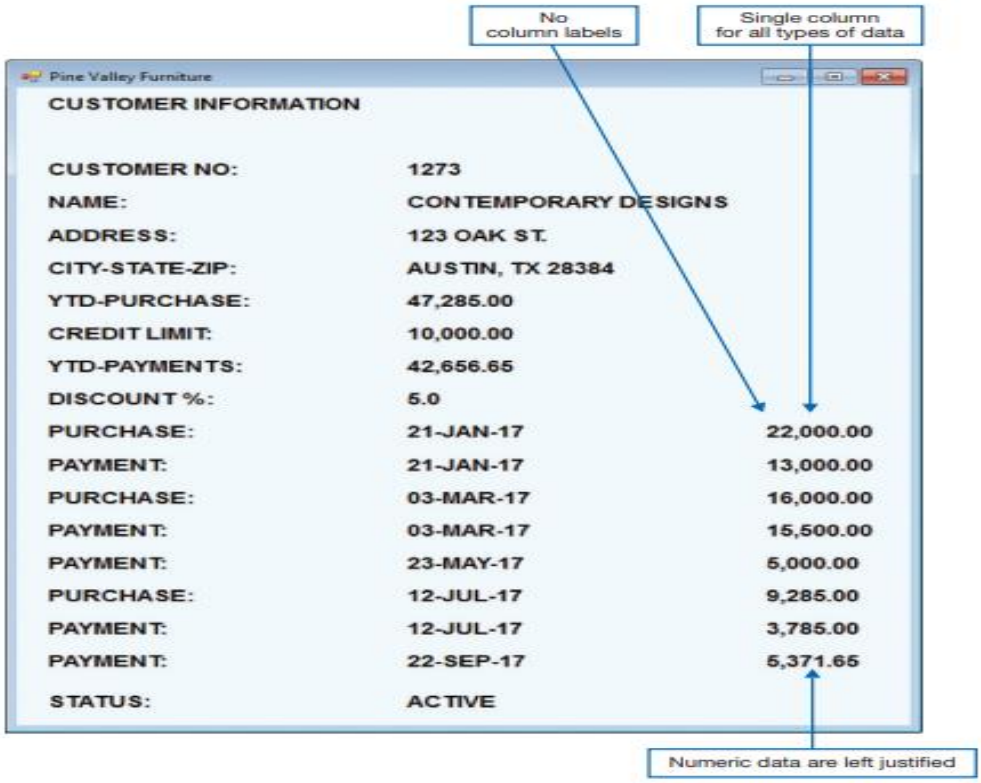
Formatting columns, rows and text:

- Sort in a meaningful order.
- Place a blank line between every five rows in long columns.
- Similar information displayed in multiple columns should be sorted vertically.
- Columns should have at least two spaces between them.
- Allow white space on printed reports for user to write notes.
- Use a single typeface, except for emphasis.
- Use same family of typefaces within and across displays and reports.
- Avoid overly fancy fonts.

Formatting numeric, textual and alphanumeric data:

- Right justify numeric data and align columns by decimal points or other delimiter.
- Left justify textual data. Use short line length, usually 30 to 40 characters per line.
- Break long sequences of alphanumeric data into small groups of three to four characters each.

Poorly Design form



Improved Design form

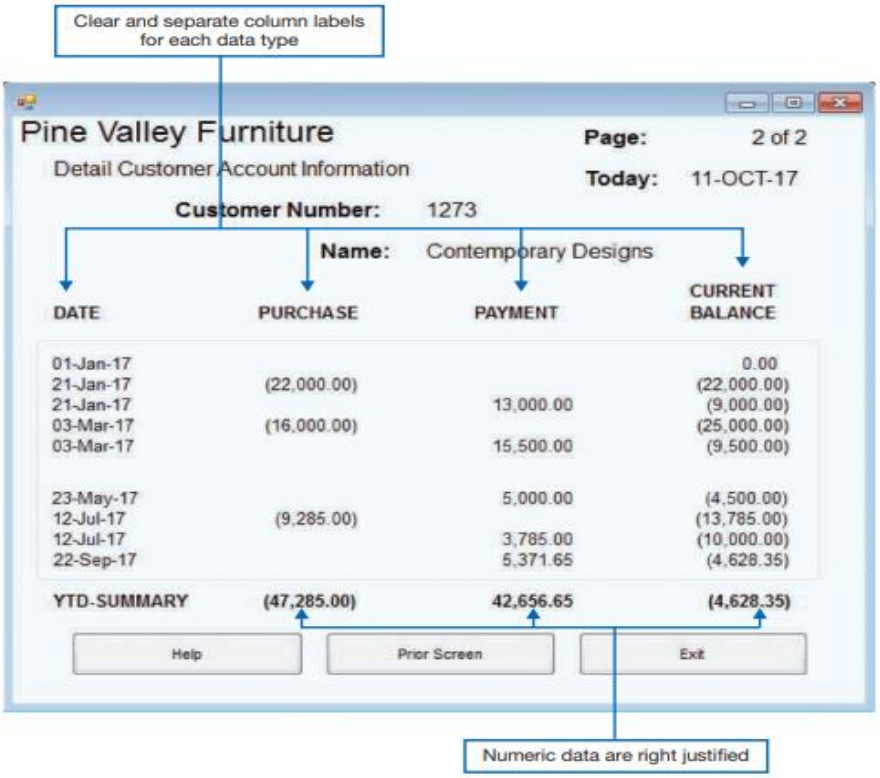
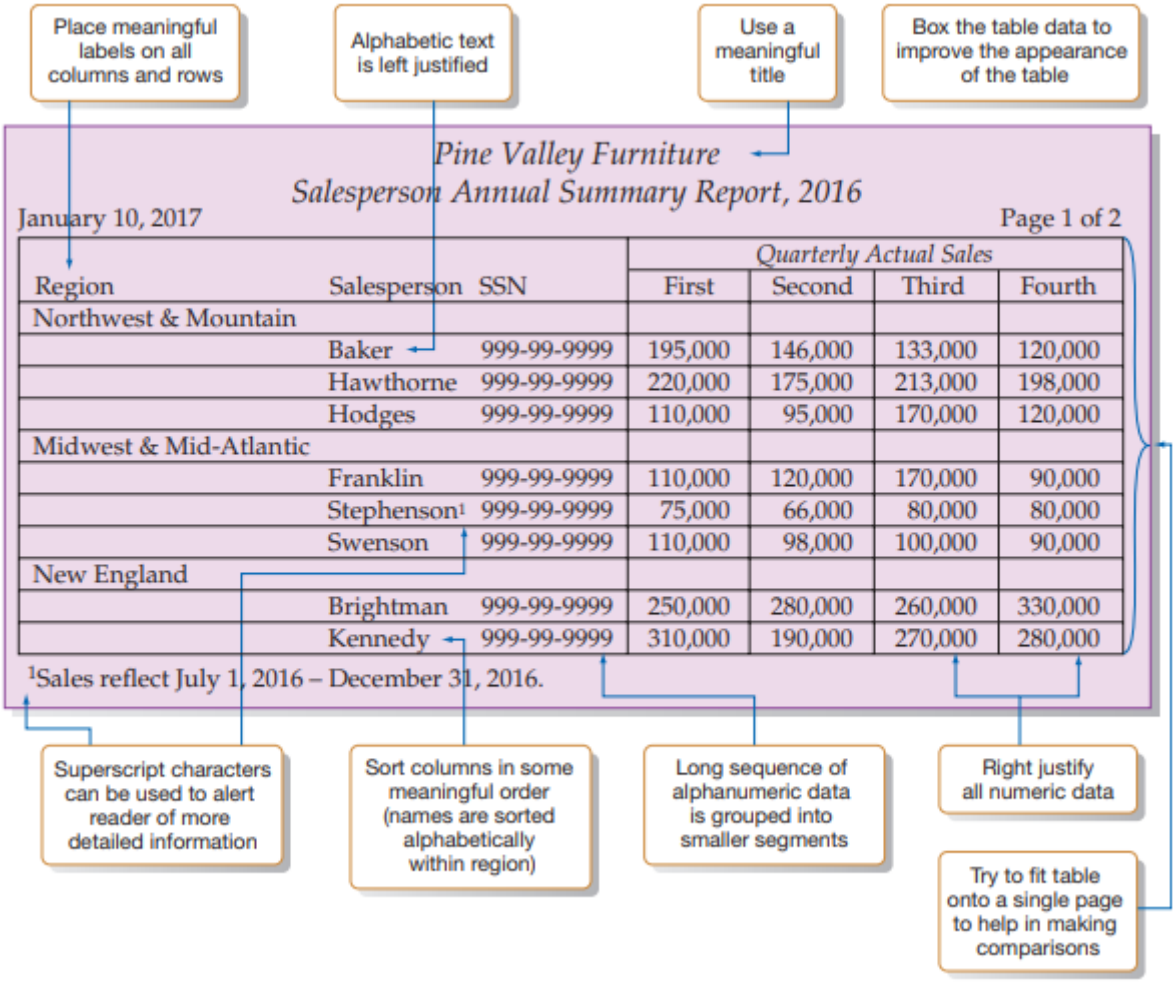


Table Vs Graph

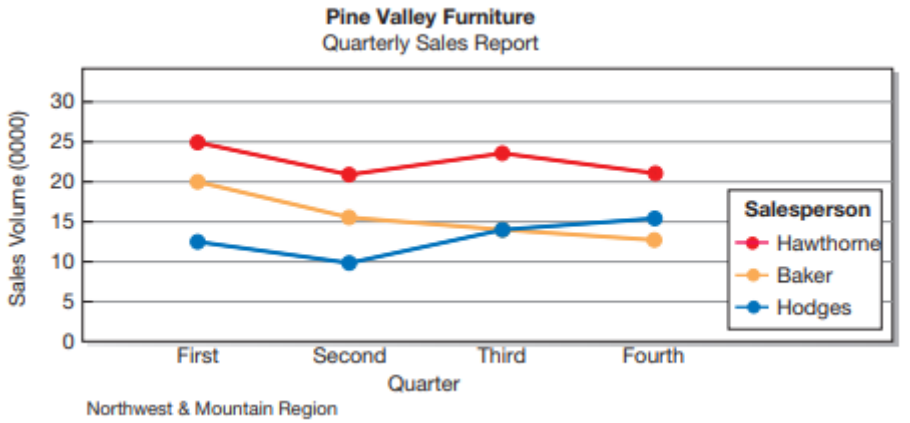
Use tables for reading individual data values
Use graphs for:

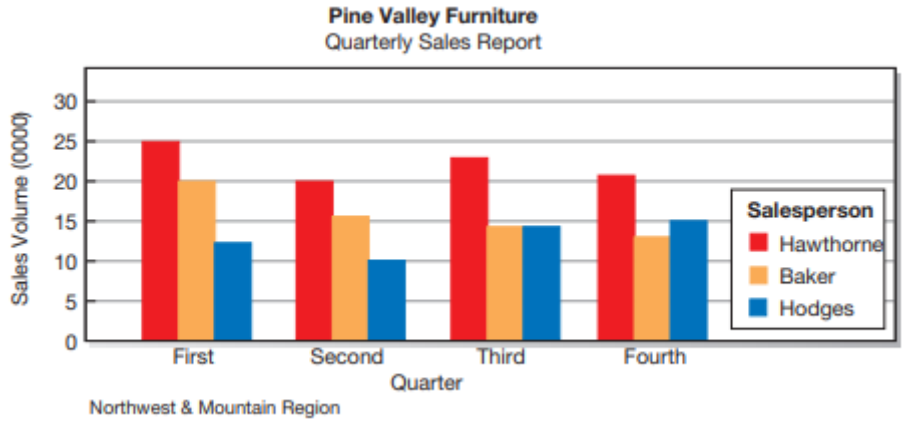
- Providing quick summary
- Displaying trends over time
- Comparing points and patterns of variables
- Forecasting activity
- Simple reporting of vast quantities of information

Tables



Graphs





4. Assessing Usability

Overall evaluation of how a system performs in supporting a particular user for a particular task.
Objective for designing forms, reports and all human-computer interactions is usability.
There are three **characteristics:**

- **Speed** - Can you complete a task efficiently?
- **Accuracy** - Does the output provide what you expect?
- **Satisfaction** - Do you like using the output?

Guidelines for Maximizing Usability

Usability Factor	Guidelines for Achievement of Usability
Consistency	Consistent use of terminology, abbreviations, formatting, titles, and navigation within and across outputs. Consistent response time each time a function is performed.
Organization	Formatting should be designed with an understanding of the task being performed and the intended user. Text and data should be aligned and sorted for efficient navigation and entry. Entry of data should be avoided where possible (e.g., computing rather than entering totals).
Clarity	Outputs should be self-explanatory and not require users to remember information from prior outputs in order to complete tasks. Labels should be extensively used, and all scales and units of measure should be clearly indicated.
Format	Information format should be consistent between entry and display. Format should distinguish each piece of data and highlight, not bury, important data. Special symbols, such as decimal places, dollar signs, and ± signs, should be used as appropriate.
Flexibility	Information should be viewed and retrieved in a manner most convenient to the user. For example, users should be given options for the sequence in which to enter or view data and for use of shortcut keystrokes, and the system should remember where the user stopped during the last use of the system.

Characteristics for Consideration When Designing Forms and Reports

Characteristic	Consideration for Form and Report Design
User	Issues related to experience, skills, motivation, education, and personality should be considered.
Task	Tasks differ in amount of information that must be obtained from or provided to the user. Task demands such as time pressure, cost of errors, and work duration (fatigue) will influence usability.
System	The platform on which the system is constructed will influence interaction styles and devices.
Environment	Social issues such as the users' status and role should be considered in addition to environmental concerns such as lighting, sound, task interruptions, temperature, and humidity. The creation of usable forms and reports may necessitate changes in the users' physical work facilities.

Common errors when designing the layout of Web Pages

Error	Recommendation
Nonstandard Use of GUI Widgets	Make sure that when using standard design items, they behave in accordance with major interface design standards. For example, the rules for radio buttons state that they are used to select one item among a set of items, that is, not confirmed until "OK'ed" by a user. In many websites selecting radio buttons is used as both *selection* and *action*.
Anything That Looks Like Advertising	Because research on web traffic has shown that many users have learned to stop paying attention to web advertisements, make sure that you avoid designing any legitimate information in a manner that resembles advertising (e.g., banners, animations, pop-ups).
Bleeding-Edge Technology	Make sure that users don't need the latest browsers or plug-ins to view your site.
Scrolling Test and Looping Animations	Avoid scrolling text and animations because they are both hard to read and users often equate such content with advertising.
Nonstandard Link Colors	Avoid using nonstandard colors to show links and for showing links that users have already used; nonstandard colors will confuse the user and reduce ease of use.
Outdated Information	Make sure your site is continuously updated so that users "feel" that the site is regularly maintained and updated. Outdated content is a sure way to lose credibility.
Slow Download Times	Avoid using large images, lots of images, unnecessary animations, or other time-consuming content that will slow the downloading time of a page.
Fixed-Formatted Text	Avoid fixed-formatted text that requires users to scroll horizontally to view content or links
Displaying Long Lists as Long Pages	Avoid requiring users to scroll down a page to view information, especially navigational controls. Manage information by showing only *N* items at a time, using multiple pages, or by using a scrolling container within the window.

Good Web Design Practices

Lightweight Graphics: small images to quick image download

Forms and Data Integrity

Template-based HTML:

- Templates to display and process common attributes of higher-level, more abstract items
- Creates an interface that is very easy to maintain

C. Designing Interfaces and Dialogues

1. Introduction

Designing Interfaces and Dialogues

- Interface design focuses on how information is provided to and from users
- Dialog design focuses on sequencing of interface display
- The design of interface and dialogues is the process of defining the manner in which human and computers
- Exchange information
- Similar to designing f&r, the process of designing i&d is user focused activity (prototyping methodology of iterative collecting information, constructing, assessing usability, and making refinements)
- To design usable i&d you must answer the same who, what, when, where and how
- Design specification had 3 section, narrative overview, sample design and testing & usability assessment
- Focus on how information is provided to and captured from users
- Dialogues are analogous to a conversation between two people
- A good human-computer interface provides a unifying structure for finding, viewing and invoking the different components of a system

The Process of Designing Interfaces and Dialogues

- User-focused activity
- Parallels form and report design process
- Employs prototyping methodology:
    - Collect information
    - Construct prototype
    - Assess usability
    - Make refinements

Design Specification:

- Narrative overview
- Sample design
- Testing and usability assessment

Design Specification

1. Narrative Overview
   a. Interface/Dialogue Name
   b. User Characteristics
   c. Task Characteristics
   d. System Characteristics
   e. Environmental Characteristics

2. Interface/Dialogue Designs
   a. Form/Report Designs
   b. Dialogue Sequence Diagram(s) and Narrative Description

3. Testing and Usability Assessment
   a. Testing Objectives
   b. Testing Procedures
   c. Testing Results
      i) Time to Learn
      ii) Speed of Performance
      iii) Rate of Errors
      iv) Retention over Time
      v) User Satisfaction and Other Perceptions

2. Interaction Methods and Devices

Methods of Interacting

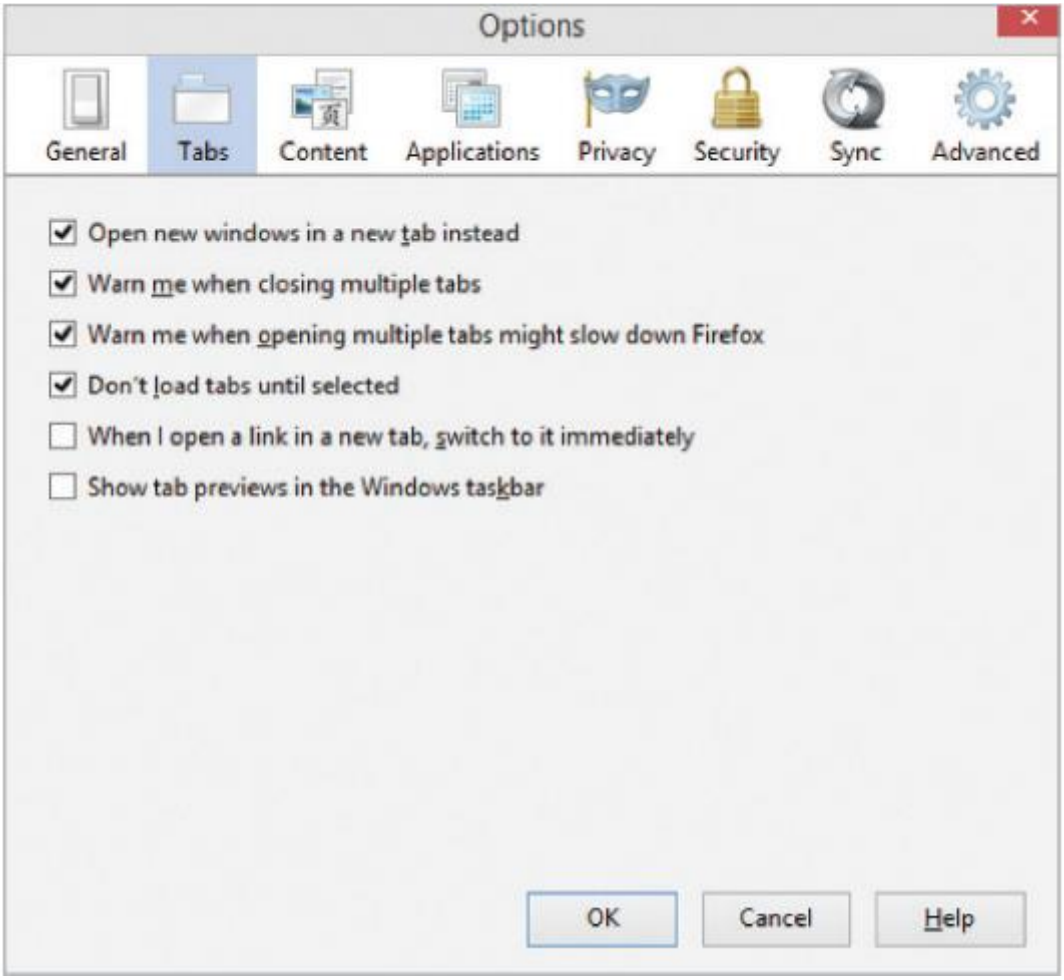
Interface: The method by which a user interacts with the information system

a. Command Line Interaction

- Users enter explicit statements into a system to invoke operations
- Example from MS DOS:
  - COPY C:PAPER.DOC A:PAPER.DOC
  - This copies a file from the C: drive to the A: drive
- Includes keyboard shortcuts and function keys
- Experienced users and for rapid interaction with a system
- User interface standards

b. Menu Interaction

- A list of system options is provided and specific command is invoked by user selection of a menu option
- Two common menu types:
    - Pop-up: menu placed near current cursor position and list of commands or possible values
    - Drop-down: access point to menu placed at top line of display, menu drops down when access point clicked

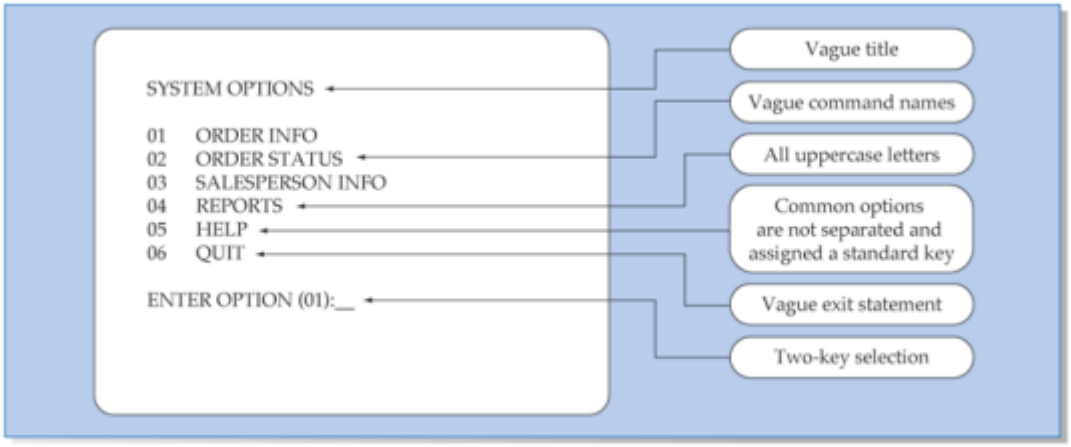


Guidelines for Menu Design

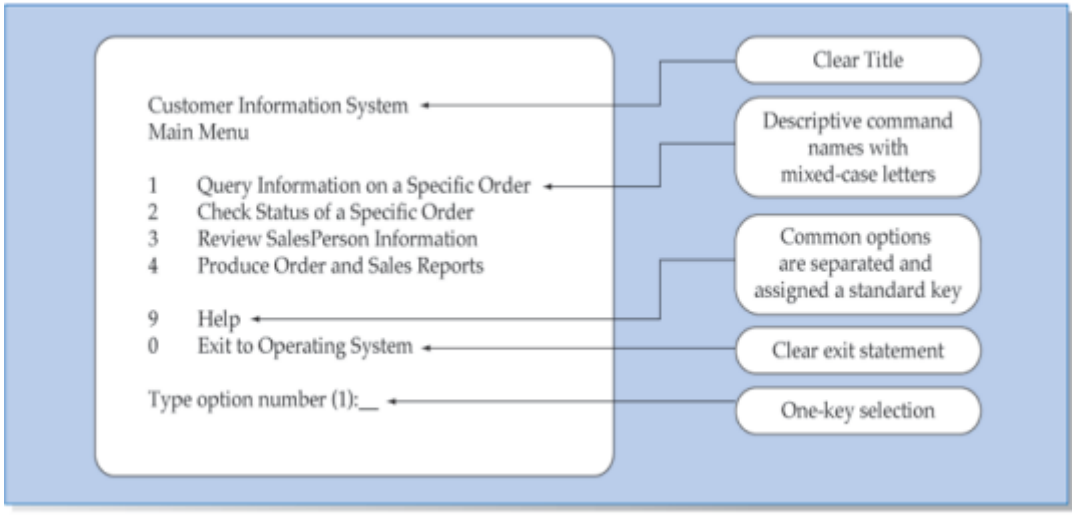
- Wording: meaningful titles, clear command verbs, mixed upper/lower case
    - Quit → prior menu or exit program?
- Organization: consistent organizing principle
    - Related options grouped together
    - Same option should have the same wording
- Length: all choices fit within screen length
    - Use submenus to break up exceedingly long menus
    -

- Selection: consistent, clear and easy selection methods
    - How to select and the consequences of each option - will another menu appear?
- Highlighting: only for selected options (check mark) or unavailable options (dimmed text)
- Use menu building tools

Poor Menu

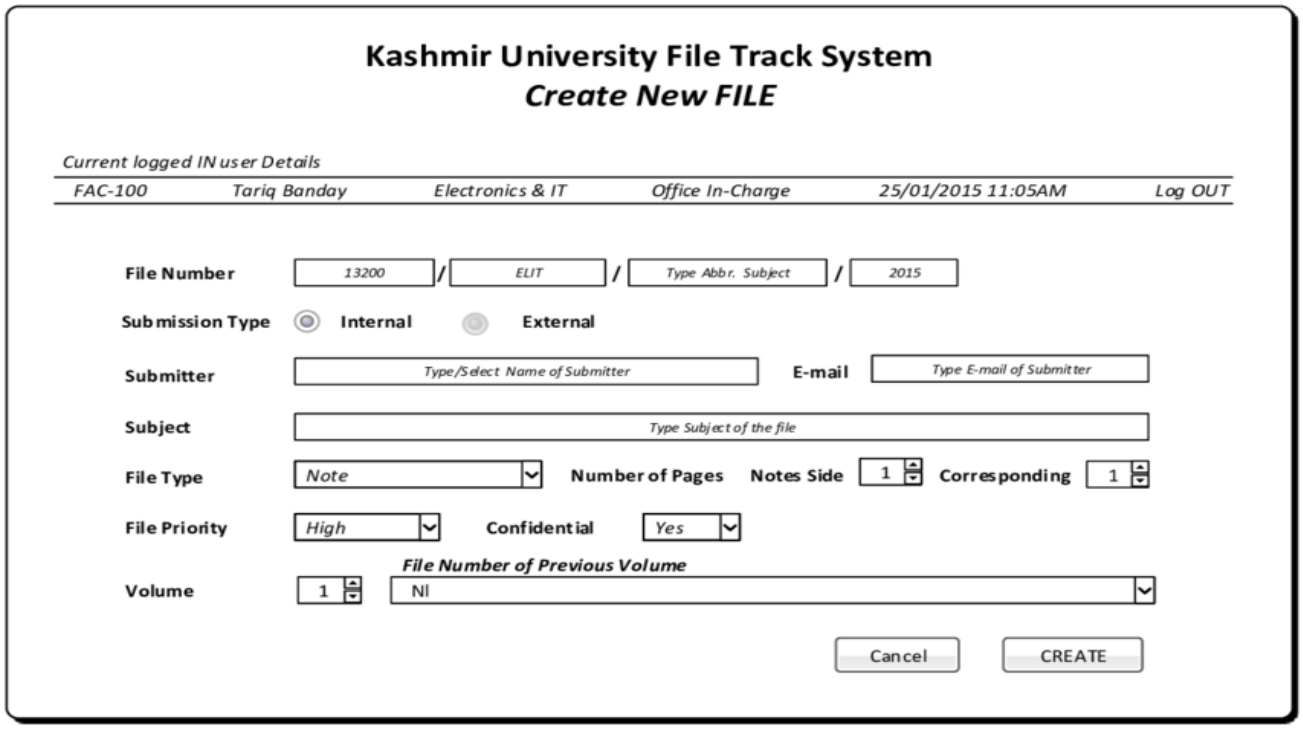


Good Menu



c. Form Interaction

- Allows users to fill in the blanks when working with a system
- Measures of an effective design:
    - Self-explanatory title and field headings
    - Fields organized into logical groupings
    - Distinctive boundaries
    - Default values
    - Displays appropriate field lengths
    - Minimizes the need to scroll windows



d. Object Interaction

- Symbols are used to represent commands or functions.
- Icons:
    - Graphic symbols that look like the processing option they are meant to represent
    - Use little screen space
    - Can be easily understood by users

e. Natural Language Interaction

- Inputs to and outputs from system are in a conventional speaking language like English
- Based on research in artificial intelligence
- Current implementations are tedious and difficult to work with, not as viable as other interaction methods

Hardware options for System Interaction

Common Devices for Interacting with an Information System

Device	Description and Primary Characteristics or Usage
Keyboard	Users push an array of small buttons that represent symbols that are then translated into words and commands. Keyboards are widely understood and provide considerable flexibility for interaction.
Mouse	A small plastic box that users push across a flat surface and whose movements are translated into cursor movement on a computer display. Buttons on the mouse tell the system when an item is selected. A mouse works well on flat desks but may not be practical in dirty or busy environments, such as a shop floor or check-out area in a retail store. Newer pen-based mice provide the user with more of the feel of a writing implement.
Joystick	A small vertical lever mounted on a base that steers the cursor on a computer display. Provides similar functionality to a mouse.
Trackball	A sphere mounted on a fixed base that steers the cursor on a computer display. A suitable replacement for a mouse when work space for a mouse is not available.
Touch Screen	Selections are made by touching a computer display. This works well in dirty environments or for users with limited dexterity or expertise.
Light Pen	Selections are made by pressing a pen-like device against the screen. A light pen works well when the user needs to have a more direct interaction with the contents of the screen.
Graphics Tablet	Moving a pen-like device across a flat tablet steers the cursor on a computer display. Selections are made by pressing a button or by pressing the pen against the tablet. This device works well for drawing and graphical applications.
Voice	Spoken words are captured and translated by the computer into text and commands. This is most appropriate for users with physical challenges or when hands need to be free to do other tasks while interacting with the application.

Summary of Interaction Device Usability Problems

	Problem						
Device	Visual Blocking	User Fatigue	Movement Scaling	Durability	Adequate Feedback	Speed	Pointing Accuracy
Keyboard	☐	☐	■	☐	■	■	☐
Mouse	☐	☐	■	☐	■	☐	☐
Joystick	☐	☐	■	☐	■	☐	■
Trackball	☐	☐	■	■	■	☐	☐
Touch Screen	■	■	☐	■	☐	☐	■
Light Pen	■	■	☐	☐	☐	☐	■
Graphics Tablet	☐	☐	■	☐	■	☐	☐
Voice	☐	☐	■	☐	■	☐	■

Key:

☐ = little or no usability problems

■ = potentially high usability problems for some applications

- Visual Blocking = extent to which device blocks display when using
- User Fatigue = potential for fatigue over long use
- Movement Scaling = extent to which device movement translates to equivalent screen movement

- Durability = lack of durability or need for maintenance (e.g., cleaning) over extended use
- Adequate Feedback = extent to which device provides adequate feedback for each operation
- Speed = cursor movement speed
- Pointing Accuracy = ability to precisely direct cursor

Summary of General Conclusions from experimental Comparisons of Input Devices in Relation to Specific Task activities

Task	Most Accurate	Shortest Positioning	Most Preferred
Target Selection	trackball, graphics tablet, mouse, joystick	touch screen, light pen, mouse, graphics tablet, trackball	touch screen, light pen
Text Selection	mouse	mouse	—
Data Entry	light pen	light pen	—
Cursor Positioning	—	light pen	—
Text Correction	light pen, cursor keys	light pen	light pen
Menu Selection	touch screen	—	keyboard, touch screen

Key:

- Target Selection = moving the cursor to select a figure or item
- Text Selection = moving the cursor to select a block of text
- Data Entry = entering information of any type into a system
- Cursor Positioning = moving the cursor to a specific position
- Text Correction = moving the cursor to a location to make a text correction
- Menu Selection = activating a menu item

— = no clear conclusion from the research

3. Designing Interfaces and Dialogues in Graphical Environments

Designing Interfaces

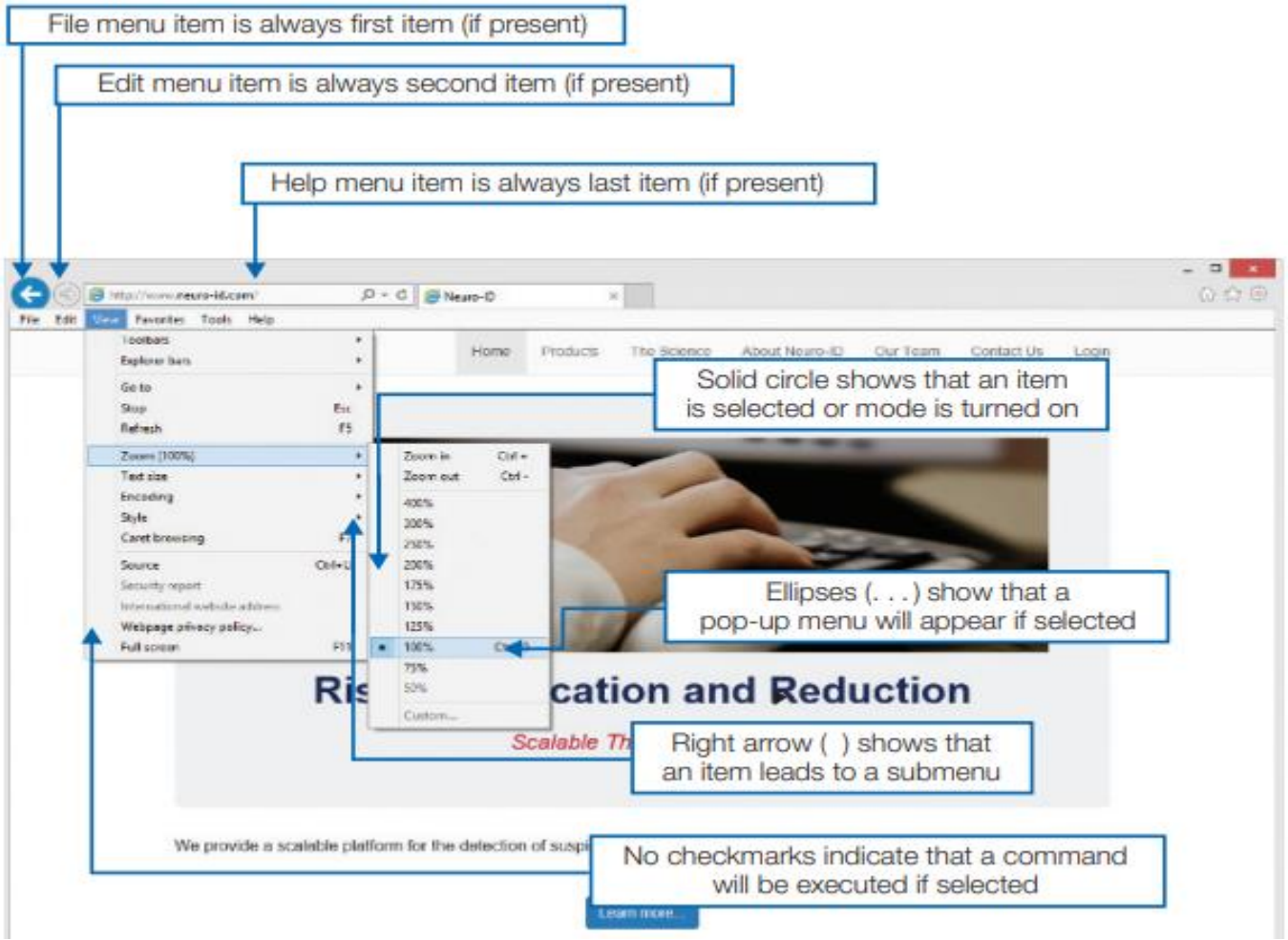
In most discussions of GUI programming, two rules repeatedly emerge as composing the first step to becoming an effective GUI designer:

1 Become an expert user of the GUI environment.
2 Understand the available resources and how they can be used.

Common Properties of Windows and Forms in a GUI environment That Can be active or Inactive

Property	Explanation
Modality	Requires users to resolve the request for information before proceeding (e.g., need to cancel or save before closing a window)
Resizable	Allows users to resize a window or form (e.g., to make room to see other windows that are also on the screen)
Movable	Allows users to move a window or form (e.g., to allow another window to be seen)
Maximize	Allows users to expand a window or form to a full-size screen (e.g., to avoid distraction from other active windows or forms)
Minimize	Allows users to shrink a window or form to an icon (e.g., to get the window out of the way while working on other active windows)
System Menu	Allows a window or form to also have a system menu to directly access system-level functions (e.g., to save or copy data)

GUI Design Interface



Designing Dialogues

- Dialogue: Sequence in which information is displayed to and obtained from a user
- Primary design guideline is consistency in sequence of actions, keystrokes and terminology
- Three step process
    1. Design dialogue sequence
    2. Build a prototype
    3. Assess usability

Dialogue Design Issues

- Goal is to establish the sequence of displays that users will encounter when working with system
- Ability of some GUI environments to jump from application to application or screen to screen makes sequencing a challenge
- One approach is to make users always resolve requests for information before proceeding
- Dialogue diagramming helps analysts better manage the complexity of designing graphical interfaces

Common errors when designing the Interface and Dialogues of Websites

Error	Description
Opening New Browser Window	Avoid opening a new browser window when a user clicks on a link unless it is clearly marked that a new window will be opened; users may not see that a new window has been opened, which will complicate navigation, especially moving backward.
Breaking or Slowing Down the Back Button	Make sure users can use the back button to return to prior pages. Avoid opening new browser windows, using an immediate redirect where, when users click the back button, they are pushed forward to an undesired location, or prevent caching such that each click of the back button requires a new trip to the server.
Complex URLs	Avoid overly long and complex URLs because it makes it more difficult for users to understand where they are and can cause problems if users want to e-mail page locations to colleagues.
Orphan Pages	Avoid having pages with no "parent" that can be reached by using a back button; requires users to "hack" the end of the URL to get back to some other prior page.
Scrolling Navigation Pages	Avoid placing navigational links below where a page opens because many users may miss these important options that are below the opening window.
Lack of Navigation Support	Make sure your pages conform to users' expectations by providing commonly used icon links such as a site logo at the top or other major elements. Also place these elements on pages in a consistent manner.
Hidden Links	Make sure you leave a border around images that are links, don't change link colors from normal defaults, and avoid embedding links within long blocks of text.
Links That Don't Provide Enough Information	Avoid not turning off link-marking borders so that links clearly show which links users have clicked and which they have not. Make sure users know which links are internal anchor points versus external links, and indicate if a link brings up a separate browser window from those that do not. Finally, make sure link images and text provide enough information to users so that they understand the meaning of the link.
Buttons That Provide No Click Feedback	Avoid using image buttons that don't clearly change when being clicked; use web GUI toolkit buttons, HTML form-submit buttons, or simple textual links.

Chapter 5: Implementation and Maintenance

A. System Implementation

1. Introduction

- Implementation and maintenance are the last two phases of the systems development life cycle.
- The purpose of implementation is to build a properly working system, install it in the organization, replace old systems and work methods, finalize system and user documentation, train users, and prepare support systems to assist users.
- Implementation also involves closedown of the project, including evaluating personnel, reassigning staff, assessing the success of the project, and turning all resources over to those who will support and maintain the system.
- The purpose of maintenance is to fix and enhance the system to respond to problems and changing business conditions. Maintenance includes activities from all systems development phases.
- Maintenance also involves responding to requests to change the system, transforming requests into changes, designing the changes, and implementing them.

2. System Implementation

System implementation is made up of many activities. The six major activities we are concerned with this are coding, testing, installation, documentation, training, and support.

The purpose of these steps is to convert the physical system specifications into working and reliable software and hardware, document the work that has been done, and provide help for current and future users and caretakers of the system.

Coding and testing may have already been completed by this point if Agile Methodologies have been followed. Using a plan-driven methodology, coding and testing are often done by other project team members besides analysts, although analysts may do some programming.

In any case, analysts are responsible for ensuring that all of these various activities are properly planned and executed.

Next, we will briefly discuss these activities in two groups:

- Coding, testing, and installation and
- Documenting the system and training and supporting users.

1 **Coding, testing, and Installation** are the process whereby the physical design specifications created by the analysis team are turned into working computer code by the programming team.

- Depending on the size and complexity of the system, coding can be an involved, intensive activity. Once coding has begun, the testing process can begin and proceed in parallel.
- Installation is the process during which the current system is replaced by the new system. This includes conversion of existing data, software, documentation, and work procedures to those consistent with the new system.

2 The Processes of Documenting the System, Training Users, and Supporting

- The process of documentation proceeds throughout the life cycle, it receives formal attention during the implementation phase because the end of implementation largely marks the end of the analysis team's involvement in systems development.
- As the team is getting ready to move on to new projects, you and the other analysts need to prepare documents that reveal (give information) all of the important information you have accumulated about this system during its development and implementation.
- There are two audiences for this final documentation: (1) the information systems personnel who will maintain the system throughout its productive life, and (2) the people who will use the system as part of their daily lives.
- The analysis team in a large organization can get help in preparing documentation from specialized staff in the information systems department.

3. Software application testing

- Software testing can be stated as the process of verifying and validating that a software or application is bug free, meets the technical requirements as guided by its design and development and meets the user requirements effectively and efficiently with handling all the exceptional and boundary cases.
- Software testing is method of assessing the functionality of a software program.
- The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy and usability.
- It mainly aims at measuring specification, functionality and performance of a software program or application.

Different types of tests

Static testing means that the code being tested is not executed. The results of running the code are not an issue for that particular test. **Dynamic testing**, on the other hand, involves execution of the code. **Automated testing** means the computer conducts the test, whereas **manual testing** means that people complete the test.

A. Inspections:

A testing technique in which participants examine program code for predictable language-specific errors that is participants manually examine code for occurrences of well-known errors. Syntax, grammar, and some other routine errors can be checked by automated inspection software, so manual inspection checks are used for more subtle (small) errors. Each programming language lends itself to certain types of errors that programmers make when coding, and these common errors are well known and documented. Exactly what the code does is not investigated in an inspection. It has been estimated that code inspections detect from 60 to 90 percent of all software defects as well as provide programmers with feedback that enables them to avoid making the same types of errors in future work.

B. Desk checking:

A testing technique in which the program code is sequentially executed manually by the reviewer. It is informal process in which 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.

C. Unit testing:

Sometimes called module testing, is an automated technique whereby each module is tested alone in an attempt to discover any errors that may exist in the module's code. But because modules coexist and work with other modules in programs and the system, they must also be tested together in larger groups.

D. Integration testing:

Combining modules and testing them is called integration testing. Integration testing is gradual. First you test the coordinating module and only one of its subordinate modules. After the first test, you add one or two other subordinate modules from the same level. Once the program has been tested with the coordinating module and all of its immediately subordinate modules, you add modules from the next level and then test the program. You continue this procedure until the entire program has been tested as a unit.

E. System testing:

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.

Current practice calls for a **top-down** approach to writing and testing modules. Under a **top-down** approach, the coordinating module is written first. Then the modules at the next level in the structure chart are written, followed by the modules at the next level, and so on, until all of the modules in the system are done. Each module is tested as it is written. Because top level modules contain many calls to subordinate modules.

System testing can be performed in two ways:

i. Black box testing:

It is defined as a testing technique in which functionality of the application under test is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications. In this testing, we just focus on inputs and output of the software system without bothering about internal knowledge of the software program. In black box test (also called functional test) internal code of the program are tested. It is called black box testing because the test cases are totally hidden for the general users.

ii. White box testing:

White box testing is a testing technique that examines the program structure and derives test data from the program logic/code. It is a software testing methodology that uses a program's source code to design tests and test cases for quality assurance (QA). The code structure is known and understood by the tester in

white box testing. In white box test (also called glass box test) structure of the program is tested. It is called white box testing because the test cases are totally visible to the general users and they can also make test cases.

F. Stub testing:

Stubs are two or three lines of code written by a programmer to stand in for the missing modules. During testing, the coordinating module calls the stub instead of the subordinate module. The stub accepts control and then returns it to the coordinating module.

G. User 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.

4. Installation

The process of moving from the current information system to the new one is called installation. All employees who use a system, whether they were consulted during the development process or not, must give up their reliance on the current system and begin to rely on the new system. Four different approaches to installation have emerged over the years: **direct, parallel, single-location, and phased**. The approach an organization decides to use will depend on the scope and complexity of the change associated with the new system and the organization's risk aversion.

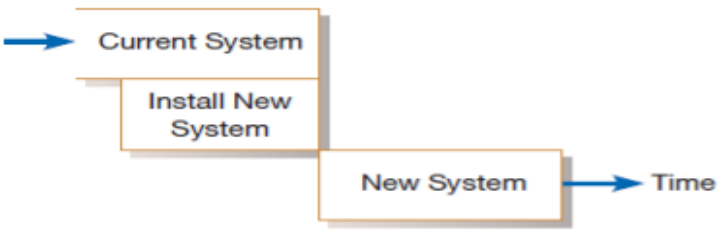
a. Direct Installation:

The old system is turned off and the new system is turned on.

Under direct installation, users are at the mercy of the new system. Any errors resulting from the new system will have a direct impact on the users and how they do their jobs and, in some cases—depending on the centrality of the system to the organization—on how the organization performs its business.

If the new system fails, considerable delay may occur until the old system can again be made operational and business transactions are reentered to make the database up to date.

For these reasons, direct installation can be very risky. Further, direct installation requires a complete installation of the whole system. For a large system, this may mean a long time until the new system can be installed, thus delaying system benefits or even missing the opportunities that motivated the system request. On the other hand, it is the least expensive installation method, and it creates considerable interest in making the installation a success. Sometimes, a direct installation is the only possible strategy if there is no way for the current and new systems to coexist, which they must do in some way in each of the other installation approaches.

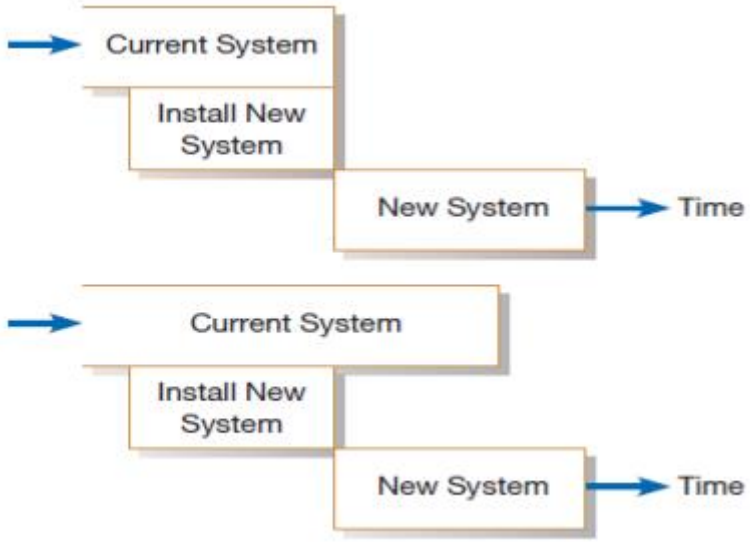


b. Parallel Installation:

It is as riskless as direct installation is risky. Under parallel installation, the old system continues to run alongside the new system until users and management are satisfied that the new system is effectively performing its duties and the old system can be turned off.

All of the work done by the old system is concurrently performed by the new system. Outputs are compared (to the greatest extent possible) to help determine whether the new system is performing as well as the old. Errors discovered in the new system do not cost the organization much, if anything, because errors can be isolated and the business can be supported with the old system.

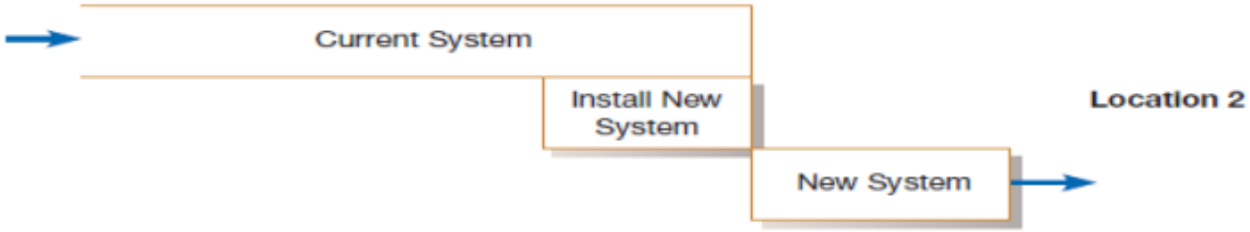
Because all work is essentially done twice, a parallel installation can be very expensive; running two systems implies employing (and paying) two staffs to not only operate both systems, but also to maintain them. A parallel approach can also be confusing to users because they must deal with both systems. As with direct installation, there can be a considerable delay until the new system is completely ready for installation. A parallel approach may not be feasible, especially if the users of the system (such as customers) cannot tolerate redundant effort or if the size of the system (number of users or extent of features) is large.



c. Single-location Installation

Single-location installation, also known as location or pilot installation, is a middle of- the-road approach compared with direct and parallel installation. Rather than convert all of the organization at once, single location installation involves changing from the current to the new system in only one place or in a series of separate sites over time.

The single location may be a branch office, a single factory, or one department, and the actual approach used for installation in that location may be any of the other approaches. The key advantage to single-location installation is that it limits potential damage and potential cost by limiting the effects to a single site. Once management has determined that installation has been successful at one location, the new system may be deployed in the rest of the organization, possibly continuing with installation at one location at a time. Success at the pilot site can be used to convince reluctant personnel at other sites that the system can be worthwhile for them as well. Problems with the system (the actual software as well as documentation, training, and support) can be resolved before deployment to other sites.

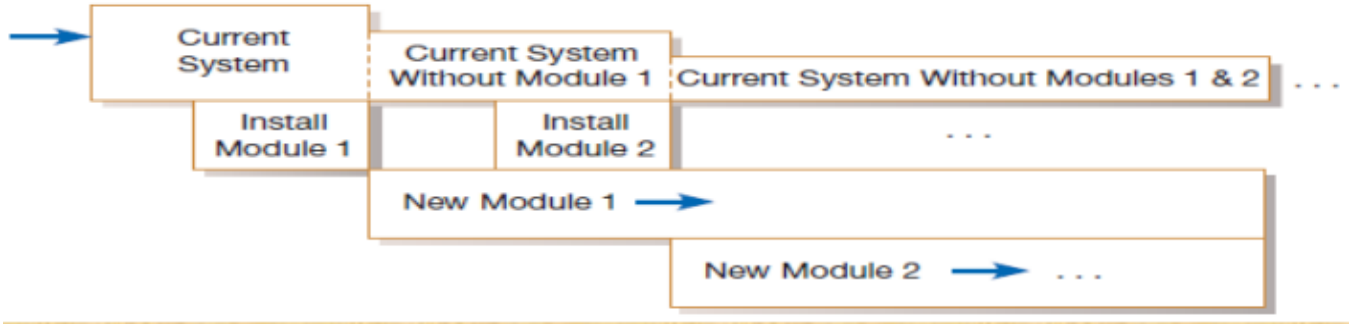


d. Phased Installation

Phased installation, also called staged installation, is an incremental approach. With phased installation, the new system is brought online in functional components; different parts of the old and new systems are used in cooperation until the whole new system is installed.

Phased installation, like single-location installation, is an attempt to limit the organization's exposure to risk, whether in terms of cost or disruption of the business. By converting gradually, the organization's risk is spread out over time and place. Also, a phased installation allows for some benefits from the new system before the whole system is ready.

For example, new data-capture methods can be used before all reporting modules are ready. For a phased installation, the new and replaced systems must be able to coexist and probably share data. Thus, bridge programs connecting old and new databases and programs often must be built. Sometimes, the new and old systems are so incompatible (built using totally different structures) that pieces of the old system cannot be incrementally replaced, so this strategy is not feasible. A phased installation is akin to bringing out a sequence of releases of the system. Thus, a phased approach requires careful version control, repeated conversions at each phase, and a long period of change, which may be frustrating and confusing to users. On the other hand, each phase of change is smaller and more manageable for all involved.



5. Documenting the System

Documentation is the process of collecting, organizing, storing and maintaining a complete record of system and other documents used or prepared during the different phases of the life cycle of system. System cannot be considered to be complete, until it is properly documented.

Proper documentation of system is necessary due to the following reasons:

- It solves the problem of indispensability (not willing) of an individual for an organization. . Even if the person, who has designed or developed the system, leaves the organization, the documented knowledge remains with the organization, which can be used for the continuity of that software.

- It makes system easier to modify and maintain in the future. The key to maintenance is proper and dynamic documentation. It is easier to understand the concept of a system from the documented records.
- It helps in restarting a system development, which was postponed due to some reason. The job need not be started from scratch, and old ideas may still be easily recapitulated from the available documents, which avoids duplication of work, and saves lot of times and effort.

Types of Documentation

a. System Documentation:

System documentation records detailed information about a system's design specification, its internal workings and its functionality. System documentation is intended primarily for maintenance programmers.

It contains the following information:

- A description of the system specifying the scope of the problem, the environment in which it functions, its limitation, its input requirements, and form and types of output required.
- Detailed diagram of system flowchart and program flowchart.
- A source listing of all the full details of any modifications made since its development.
- Specification of all input and output media required for the operation of the system.
- Problem definition and the objective of developing the programs.
- Output and test report of the program.
- Upgrade or maintenance history, if modification of the program is made.

There are two types of system documentation. They are:

i. Internal documentation:

- Internal documentation is part of the program source code or is generated at compile time.

ii. External documentation:

- External documentation includes the outcome of structured diagramming technique such as dataflow an entity-relationship diagrams.

b. User documentation:

User documentation consists of written or other visual information about an application system, how it works and how to use it. User documentation is intended primarily for users.

It contains the following information:

- Set up and operational details of each system.
- Loading and unloading procedures.
- Problems which could arise, their meaning reply and operation action.
- Special checks and security measures.
- Quick reference guides about operating a system in a short, concise format.

6. Training and Supporting Users

The type of training needed will vary by system type and user expertise.

Types of training methods are:

- Resident expert (to fellow users for training).
- Traditional instructor-led classroom training.
- E-learning/ distance learning.
- Blended learning (combination of instructor-led and e-learning).
- Software help components.
- Electronic performance support system: component of a software package or an application in which training and educational information is embedded.
- External sources, such as vendors.

Computing supports for users has been provided in one of a few forums:

i. Automating support:

- Online support forums provides users access to information on new releases, bugs and tips form more effective usage.
- Forums are offered over the internet or over company intranets.

ii. Providing support through a help desk:

- A help desk is an information systems department function and is staffed by IS personnel.
- The help desk is the first place users should call when they need assistance with an information system.
- The help desk staff members either deal with the users questions or refer the users to the most appropriate person.

7. Organizational Issues in Systems Implementation

The best efforts of the systems development team is to design and build a quality system and to manage the change process in the organization, the implementation effort sometimes fails.

Sometimes employees will not use the new system that has been developed for them or, if they do use it, their level of satisfaction with it is very low.

Why do systems implementation efforts fail? This question has been the subject of information systems research for over 60 years.

Why Implementation Sometimes fails?

The conventional wisdom that has emerged over the years is that there are at least two conditions necessary for a successful implementation effort: **management support of the system under development** and **the involvement of users in the development process.**

Conventional wisdom holds that if both of these conditions are met, you should have a successful implementation.

- **Management support and user involvement are important** to implementation success, but they may be overrated compared to other factors that are also important.
- Research has shown that the link between user involvement and implementation success is sometimes weak.
- User involvement can help reduce the risk of failure when the system is complex, but user participation in the development process only makes failure more likely when there are financial and time constraints in the development process.
- Information systems implementation failures are too common, and the implementation process is too complicated, for the conventional wisdom to be completely correct.

Over the years, other studies have found evidence of additional factors that are important to a successful implementation process.

Three such factors are: **commitment to the project, commitment to change, and the extent of project definition and planning**

- **Commitment to the project** involves managing the systems development project so that the problem being solved is well understood and the system being developed to deal with the problem actually solves it.
- **Commitment to change** involves being willing to change behaviors, procedures, and other aspects of the organization.
- **The extent of project definition and planning** is a measure of how well the project was planned. The more extensive the planning effort is, the less likely implementation failure is. Still another important factor related to implementation success is user expectations. The more realistic a user's early expectations about a new system and its capabilities are, the more likely it is that the user will be satisfied with the new system and actually use it.

B. System Maintenance

1. Introduction

In this chapter, we discuss systems maintenance, the largest systems development expenditure for many organizations. In fact, more programmers today work on maintenance activities than work on new development. Your first job after graduation may very well be as a maintenance programmer/analyst.

This disproportionate (too large or too small in comparison to something else) distribution of maintenance programmers is interesting because software does not wear out in a physical manner as do buildings and machines.

There is no single reason why software is maintained; however, most reasons relate to a desire to evolve system functionality in order to overcome internal processing errors or to better support changing business needs.

Thus, maintenance is a fact of life for most systems. This means that maintenance can begin soon after the system is installed.

2. Maintaining Information Systems

- Once an information system is installed, the system is essentially in the maintenance phase of the systems development life cycle (SDLC).
- When a system is in the maintenance phase, some person within the systems development group is responsible for collecting maintenance requests from system users and other interested parties, such as system auditors, data center and network management staff, and data analysts.
- Once collected, each request is analyzed to better understand how it will alter the system and what business benefits and necessities will result from such a change.
- If the change request is approved, a system change is designed and then implemented. As with the initial development of the system, implemented changes are formally reviewed and tested before installation into operational systems.

The Process of Maintaining information Systems

The maintenance phase is the last phase of the SDLC. It is here that the SDLC becomes a cycle, with the last activity leading back to the first. This means that the process of maintaining an information system is the process of returning to the beginning of the SDLC and repeating development steps until the change is implemented.

Four major activities occur within maintenance:

a. **Obtaining maintenance requests**
b. **Transforming requests into changes**
c. **Designing changes**
d. **Implementing changes**

- Obtaining maintenance requests requires that a formal process be established whereby users can submit system change requests.
- Most companies have some sort of document like a System Service Request (SSR) to request new development, to report problems, or to request new features within an existing system.
- When developing the procedures for obtaining maintenance requests, organizations must also specify an individual within the organization to collect these requests and manage their dispersal to maintenance personnel.

3. Conducting Systems Maintenance

A significant within organizations does not go to the development of new systems but **to the maintenance of existing systems.**

We will describe various types of maintenance, factors influencing the complexity and cost of maintenance, and alternatives for managing maintenance.

Types of System Maintenance

a. Corrective Maintenance

Corrective maintenance refers to changes made to repair defects in the design, coding, or implementation of the system. For example, if you had recently purchased a new home, corrective maintenance would involve repairs made to things that had never worked as designed, such as a faulty electrical outlet or a misaligned door. Most corrective maintenance problems surface soon after installation. When corrective maintenance problems surface, they are typically urgent and need to be resolved to curtail possible interruptions in normal business activities.

b. Adaptive Maintenance

Adaptive maintenance involves making changes to an information system to evolve its functionality to changing business needs or to migrate it to a different operating environment. Within a home, adaptive maintenance might be adding storm windows to improve the cooling performance of an air conditioner. Adaptive maintenance is usually less urgent than corrective maintenance because business and technical changes typically occur over some period of time.

c. Perfective Maintenance

Perfective maintenance involves making enhancements to improve processing performance or interface usability or to add desired, but not necessarily required, system features (bells and whistles). In our home example, perfective maintenance would be adding a new room. Many systems professionals feel that perfective maintenance is not really maintenance but rather new development.

d. Preventive Maintenance

Preventive maintenance involves changes made to a system to reduce the chance of future system failure. An example of preventive maintenance might be to increase the number of records that a system can process far beyond what is currently needed or to generalize how a system sends report information to a printer so that the system can easily adapt to changes in printer technology.

A. The Cost of Maintenance

Information systems maintenance costs are a significant expenditure. For some organizations, as much as 60 to 80 percent of their information systems budget is allocated to maintenance activities. These huge maintenance costs are due to the fact that many organizations have accumulated more and older so-called legacy systems that require more and more maintenance. More maintenance means more maintenance work for programmers. For systems developed in-house, on average, 52 percent of a company's programmers are assigned to maintain existing software. In situations where a company has not developed its systems in-house but instead has licensed software, as in the case of ERP systems, maintenance costs remain high. The standard cost of maintenance for most Enterprise Resource Planning (ERP) vendors is 22 percent annually.

B. Managing Maintenance

As maintenance activities consume more and more of the systems development budget, maintenance management has become increasingly important. Today, far more programmers worldwide are working on maintenance than on new development. In other words, maintenance is the largest segment of programming personnel, and this implies the need for careful management.

TABLE 14-2 Advantages and Disadvantages of Different Maintenance Organizational Structures

Type	Advantages	Disadvantages
Separate	Formal transfer of systems between groups improves the system and documentation quality	All things cannot be documented, so the maintenance group may not know critical information about the system
Combined	Maintenance group knows or has access to all assumptions and decisions behind the system's original design	Documentation and testing thoroughness may suffer due to a lack of a formal transfer of responsibility
Functional	Personnel have a vested interest in effectively maintaining the system and have a better understanding of functional requirements	Personnel may have limited job mobility and lack access to adequate human and technical resources

C. Measuring Maintenance Effectiveness

A second management issue is the measurement of maintenance effectiveness. As with the effective management of personnel, the measurement of maintenance activities is fundamental to understanding the quality of the development and maintenance efforts.

To measure effectiveness, we must measure the following 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 metric is referred to as the mean time between failures (MTBF).

As its name implies, the MTBF metric shows the average length of time between the identification of one system failure and the next.

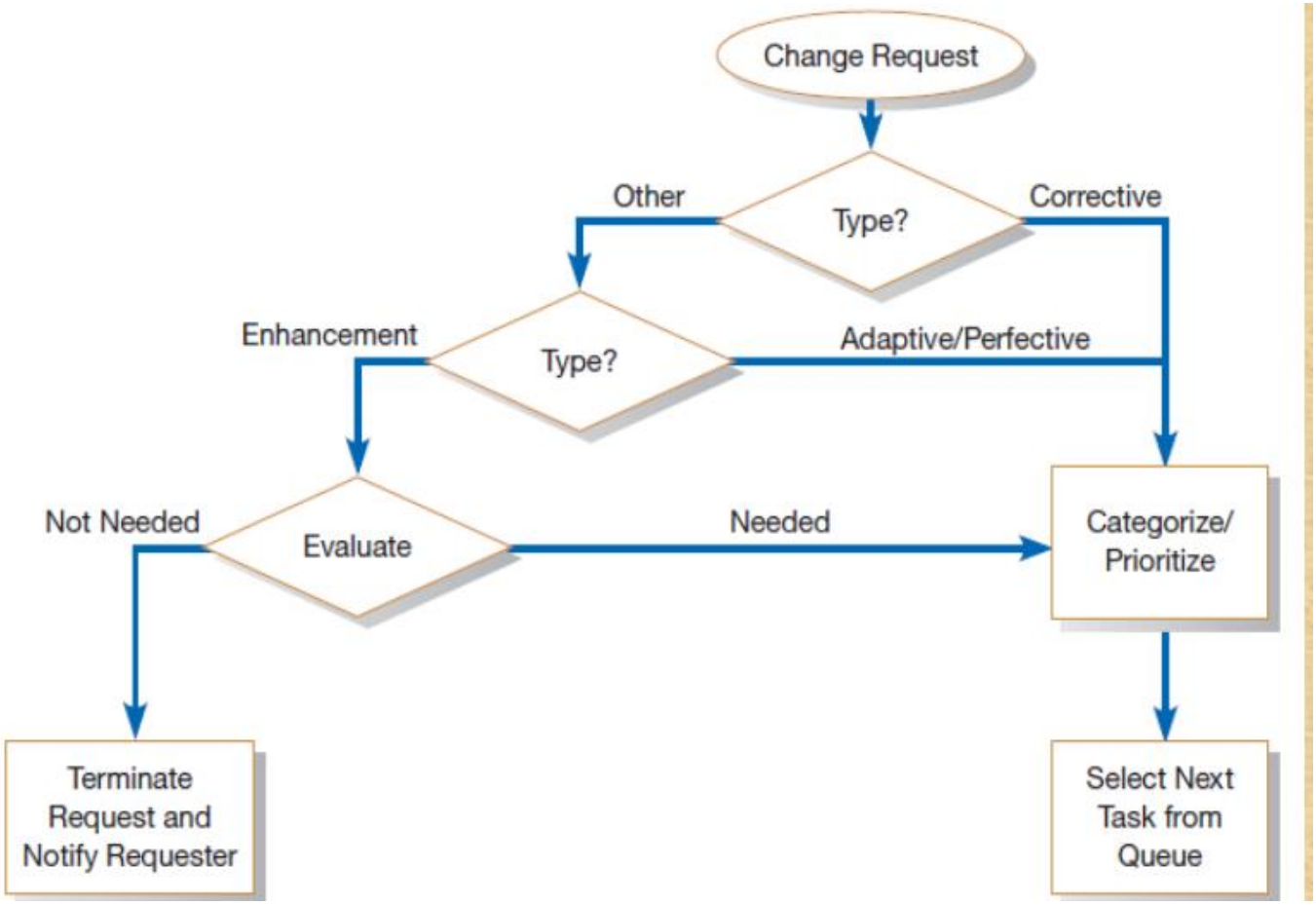
Over time, you should expect the MTBF value to rapidly increase after a few months of use (and corrective maintenance) of the.

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.

D. Controlling Maintenance Requests

Another maintenance activity is managing maintenance requests. There are various types of maintenance requests—some correct minor or severe defects in the systems, whereas others improve or extend system functionality. 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.

If, **for example**, the request is an error—that is, a corrective maintenance request—then the flowchart shows that the request is placed in the queue of tasks waiting to be performed on the system.



E. 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. The baseline modules are the software modules for the most recent version of a system whereby each module has passed the organization's quality assurance process and documentation standards.

A system librarian controls the checking out and checking in of the baseline source code modules.

If maintenance personnel are assigned to make changes to a system, they must first check out a copy of the baseline system modules—no one is allowed to directly modify the baseline modules. Only those modules that have been tested and have gone through a formal check-in process can reside in the library.

Before any code can be checked back in to the librarian, the code must pass the quality control procedures, testing, and documentation standards established by the organization.