1 Chapter

SYSTEMS DEVELOPMENT FUNDAMENTALS

CHAPTER OUTLINE

After studying this chapter, students will be able to understand the:

- The Systems Development Environment
- The Origins of Software
- Managing the Information Systems Project

THE SYSTEMS DEVELOPMENT ENVIRONMENT

SYSTEM

A system is a set of components that interact to accomplish some purpose. For examples, College system, Economic system, Language system, a Business and its parts - Marketing, Sales, Research, Shipping, Accounting, Government and so on.

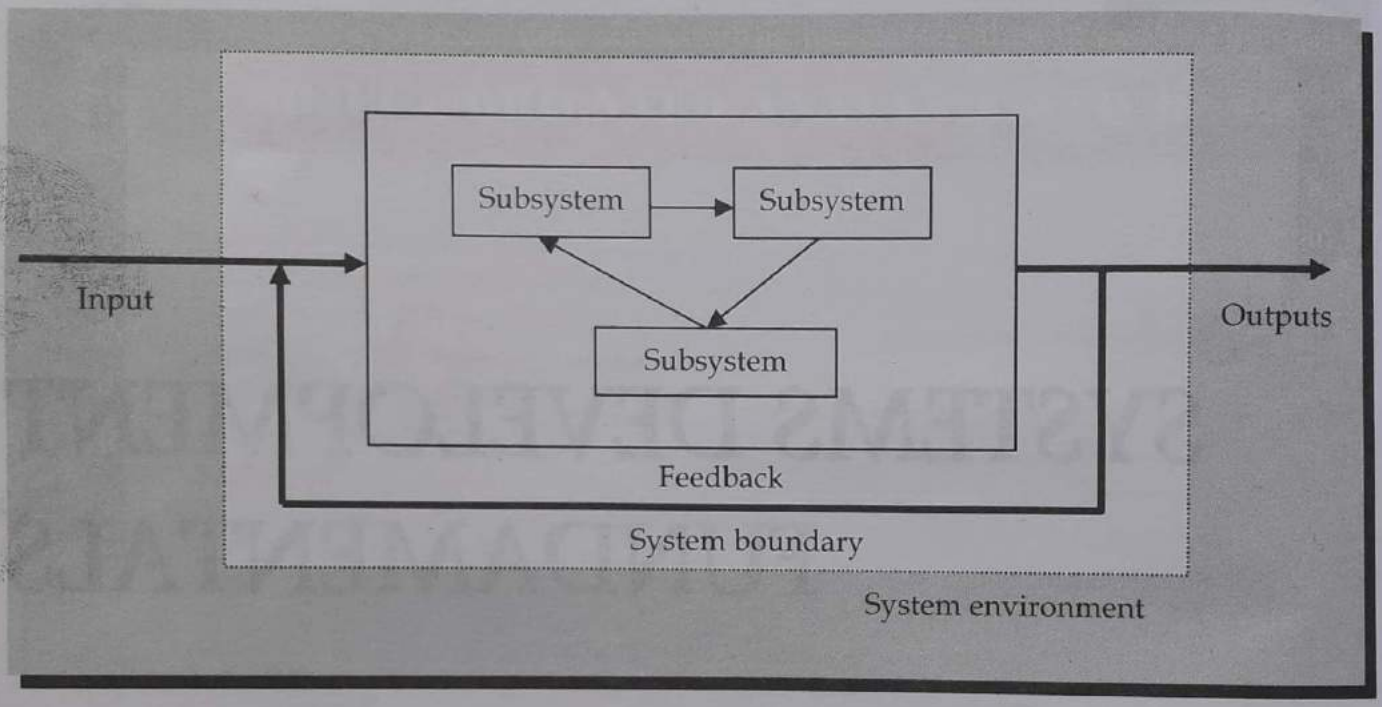


Figure 1.1: Basic System Model

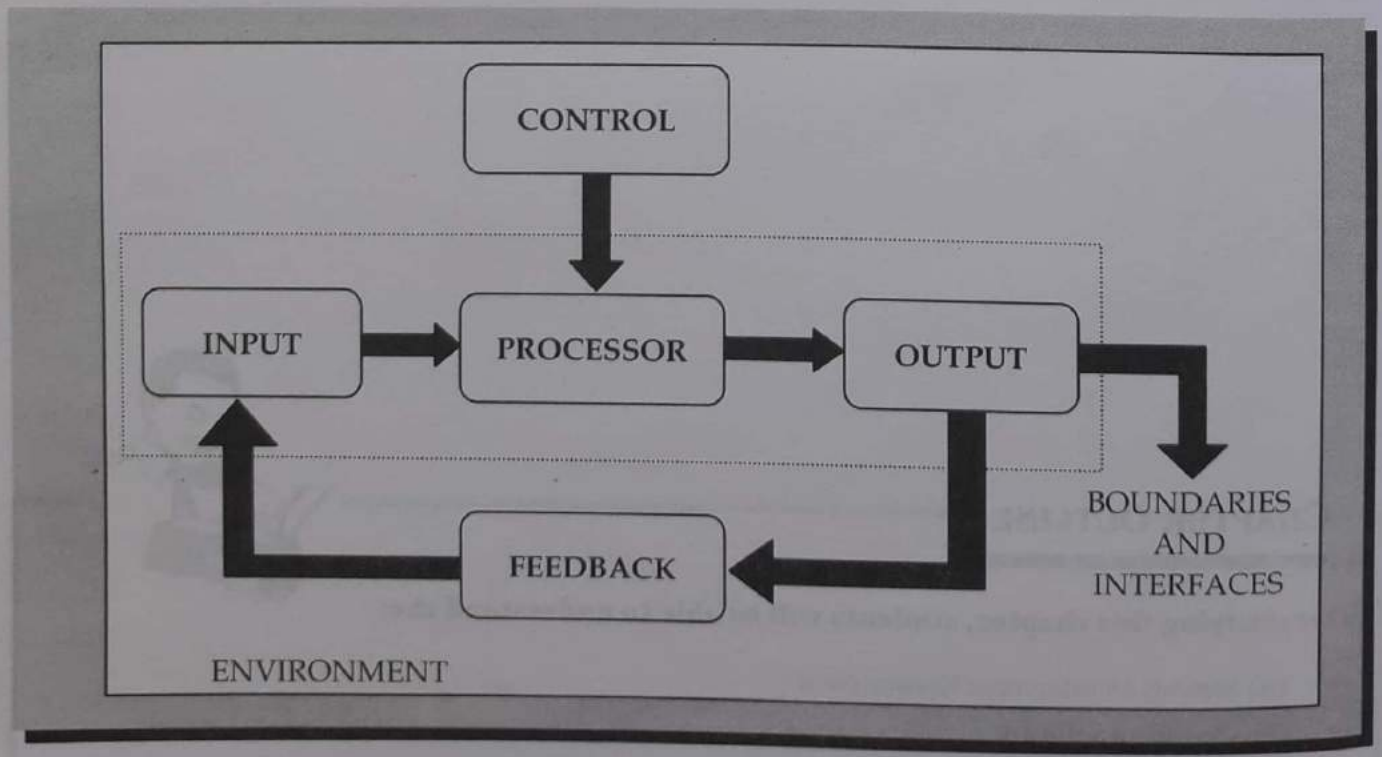


Figure 1.2: Elements of system

Elements of the system

The figure 1.2 shows the elements of a system and elements are:

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

The Characteristics of a system are:

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

INFORMATION SYSTEM

It is interrelated components working together to collect, process, store, and disseminate information to support decision making, coordination control analysis and visualization in an organization.

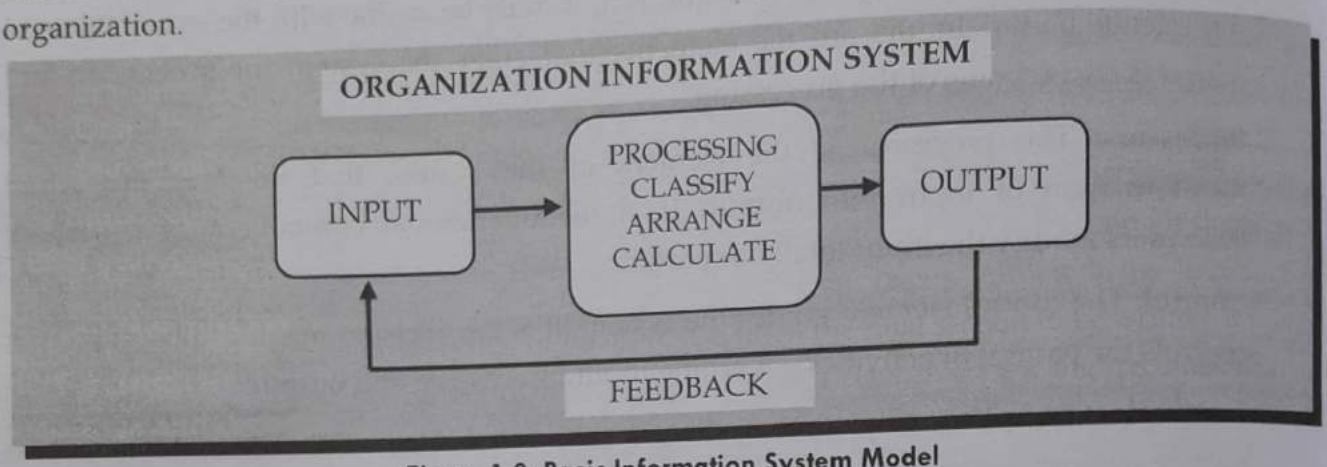


Figure 1.3: Basic Information System Model

The product of systems analysis and design is an information system. Many information systems now use computer systems for manipulating information and are sometimes called computer-based information systems. 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. **Information** is the data shaped into a form that is meaningful and useful to human beings. **Data**, on the other hand, are the collection of raw facts representing events occurring in organizations.

The three activities to produce information in an information system are input, processing, and output. **Input** captures or collects row data from within the organization or from its external environment for processing. **Processing** converts, manipulates, and analyze these raw data into the meaningful information. **Output** transfers this information to the people who will use it or to the activities for which it will be used. Information systems also require **feedback**, which is output that is returned to the appropriate members of the organization to help them evaluate or correct input.

The two types of information systems are formal and informal. **Formal information systems** are based on accepted and fixed definitions of data and procedures for collecting, storing, processing, disseminating, and using these data with predefined rules. **Informal information systems,** in contrast, relay on unstated rules. Formal information systems can be manual as well as computer based. Manual information systems use paper-and-pencil technology. In contrast, **computer-based information systems (CBIS)** relay on computer hardware and software for processing and disseminating information.

Typical Components of Information Systems

While information systems may differ in how they are used within an organization, they typically contain the following components:

- **Hardware:** Computer-based information systems use computer hardware, such as processors, monitors, keyboard and printers.
- **Software:** These are the programs used to organize, process, and analyze data.
- **Databases:** Information systems work with data, organized into tables and files.
- **Network:** Different elements need to be connected to each other, especially if many different people in an organization use the same information system.
- **Procedures:** These describe how specific data are processed and analyzed in order to get the answers for which the information system is designed.

The first four components (Hardware, Software, Database, and Network) are part of the general information technology (IT) of an organization. Procedures, the fifth component, are very specific to the information needed to answer a specific question.

Types of Information System

Information systems are developed for different purposes, depending on the needs of human users and the business. Transaction processing systems (TPS) function at the operational level of the organization; office automation systems (OAS) and knowledge work systems (KWS) support work at the knowledge level. Higher-level systems include management information systems (MIS) and decision support systems (DSS). Expert systems apply the expertise of decision makers to solve specific, structured problems. On the strategic level of management we find executive support systems (ESS). Group decision support systems (GDSS) and the more generally described computer-supported collaborative work systems (CSCWS) aid group-level decision making of a semi-structured or unstructured variety.

The variety of information systems that analysts may develop is shown in the figure 1.4. Notice that the figure presents these systems from the bottom up, indicating that the operational, or lowest, level of the organization is supported by TPS, and the strategic, or highest, level of semi-structured and unstructured decisions is supported by ESS, GDSS, and CSCWS at the top. This text uses the terms management information systems, information systems (IS), computerized information systems, and computerized business information systems interchangeably to denote computerized information systems that support the broadest range of user interactions with technologies and business activities through the information they produce in organizational contexts.

TRANSACTION PROCESSING SYSTEMS

Transaction processing systems (TPS) are computerized information systems that were developed to process large amounts of data for routine business transactions such as payroll and inventory. A TPS eliminates the tedium of necessary operational transactions and reduces the time once required to perform them manually, although people must still input data to computerized systems.

Transaction processing systems are boundary-spanning systems that permit the organization to interact with external environments. Because managers look to the data generated by the TPS for up-to-the-minute information about what is happening in their companies, it is essential to the day-to-day operations of business that these systems function smoothly and without interruption.

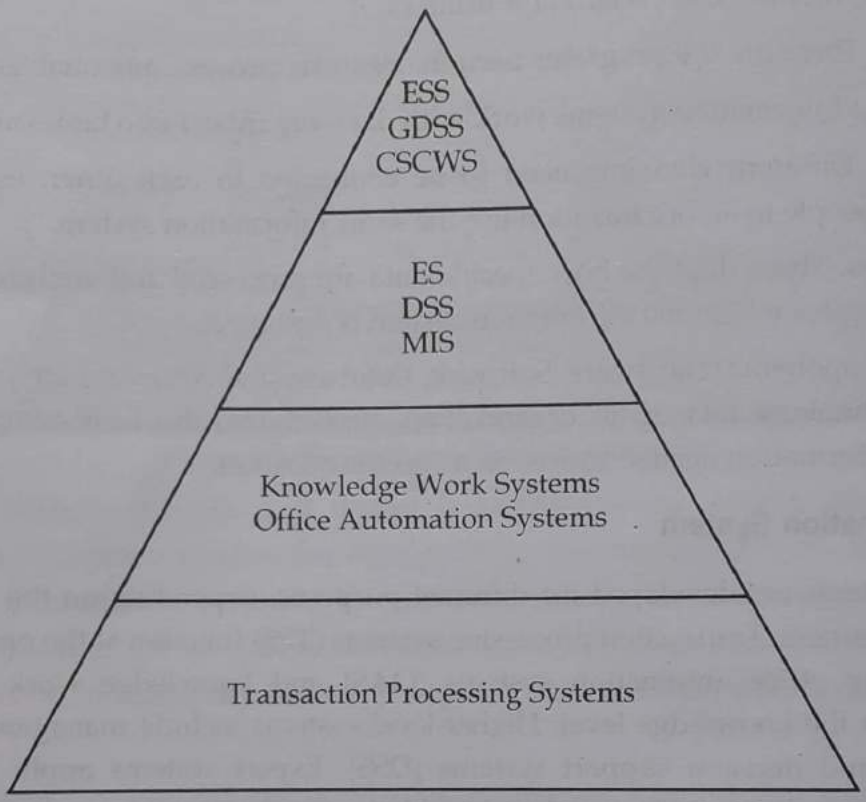


Figure 1.4: A systems analyst may be involved with any or all of these systems

Office Automation Systems and Knowledge Work Systems

At the knowledge level of the organization are two classes of systems. Office automation systems (OAS) support data workers, who do not usually create new knowledge but rather analyze information to transform data or manipulate it in some way before sharing it with, or formally disseminating it throughout, the organization and, sometimes, beyond. Familiar aspects of OAS include word processing, spreadsheets, desktop publishing, electronic scheduling, and communication through voice mail, email (electronic mail), and teleconferencing.

Knowledge work systems (KWS) support professional workers such as scientists, engineers, and doctors by aiding them in their efforts to create new knowledge (often in teams) and by allowing them to contribute it to their organization or to society at large.

Management Information Systems

Management information systems (MIS) do not replace transaction processing systems; rather, all MIS include transaction processing. MIS are computerized information systems that work because of the purposeful interaction between people and computers. By requiring people, software, and hardware to function in concert, management information systems support users in accomplishing a broader spectrum of organizational tasks than transaction processing systems, including decision analysis and decision making.

To access information, users of the management information system share a common database. The database stores both data and models that help the user interact with, interpret, and apply that data. Management information systems output information that is used in decision making. A management information system can also help integrate some of the computerized information functions of a business.

Decision Support Systems

A higher-level class of computerized information systems is decision support systems (DSS). DSS are similar to the traditional management information system because they both depend on a database as a source of data. A decision support system departs from the traditional management information system because it emphasizes the support of decision making in all its phases, although the actual decision is still the exclusive province of the decision maker. Decision support systems are more closely tailored to the person or group using them than is a traditional management information system. Sometimes they are discussed as systems that focus on business intelligence.

Artificial Intelligence and Expert Systems

Artificial intelligence (AI) can be considered the overarching field for expert systems. The general thrust of AI has been to develop machines that behave intelligently. Two avenues of AI research are (1) understanding natural language and (2) analyzing the ability to reason through a problem to its logical conclusion. Expert systems use the approaches of AI reasoning to solve the problems put to them by business (and other) users.

Expert systems are a very special class of information system that has been made practicable for use by business as a result of widespread availability of hardware and software such as personal computers (PCs) and expert system shells. An expert system (also called a knowledge-based system) effectively captures and uses the knowledge of a human expert or experts for solving a particular problem experienced in an organization. Notice that unlike DSS, which leave the ultimate judgment to the decision maker, an expert system selects the best solution to a problem or a specific class of problems.

The basic components of an expert system are the knowledge base, an inference engine connecting the user with the system by processing queries via languages such as structured query language (SQL), and the user interface. People called knowledge engineers capture the expertise of experts, build a computer system that includes this expert knowledge, and then implement it.

Group Decision Support Systems and Computer-Supported Collaborative Work Systems

Organizations are becoming increasingly reliant on groups or teams to make decisions together. When groups make semi-structured or unstructured decisions, a group decision support system may afford a solution. Group decision support systems (GDSS), which are used in special rooms equipped in a number of different configurations, permit group members to interact with electronic support-often in the form of specialized software-and a special group facilitator.

Group decision support systems are intended to bring a group together to solve a problem with the help of various supports such as polling, questionnaires, brainstorming, and scenario creation. GDSS software can be designed to minimize typical negative group behaviors such as lack of participation due to fear of reprisal for expressing an unpopular or contested viewpoint, domination by vocal group members, and "group think" decision making. Sometimes GDSS are discussed under the more general term computer-supported collaborative work systems (CSCWS), which might include software support called groupware for team collaboration via networked computers. Group decision support systems can also be used in a virtual setting.

Executive Support Systems

When executives turn to the computer, they are often looking for ways to help them make decisions on the strategic level. Executive support systems (ESS) help executives organize their interactions with the external environment by providing graphics and communications technologies in accessible places such as boardrooms or personal corporate offices. Although ESS rely on the information generated by TPS and MIS, executive support systems help their users address unstructured decision problems, which are not application specific, by creating an environment that helps them think about strategic problems in an informed way. ESS extends and supports the capabilities of executives, permitting them to make sense of their environments.

Overview of Systems Analysis and Design

Systems Analysis and Design is the complex, challenging, and simulating organizational process that a team of business and systems professionals uses to develop and maintain computer based information systems. **System Analysis** - Process of gathering and interpreting facts, diagnosing problems, and using the facts to improve the system. **Systems Design** - Process of planning a new system to replace or complement the old. Analysis specifies what the system should do and design states how to achieve the objective.

An important result of system analysis and design is application software i.e. software designed to support organizational functions or processes such as inventory management, payroll, or mark-sheet analysis. In addition to application software, the total information system includes the hardware and systems software on which the application software runs, documentation and training materials, the specific job roles associated with the overall system, controls and the people who use the software along with their work methods.

In systems analysis and design, we use various methodologies, techniques and tools. **Methodologies** define the set of steps that will guide your work and influence the quality of your final product: the Information System. Most methodologies incorporate several development techniques. **Techniques** are particular process that will help to ensure that the work is well thought-out, complete, and comprehensible to other on the project team. Techniques also provide support for a wide range of tasks like conducting interviews, planning and managing the activities in a system development project, diagramming the system's logic, and designing the reports that the system will generate. **Tools** are typically computer programs that make it easy to use and benefit from techniques and to faithfully follow the guidelines of

the overall development methodology. To be effective, both techniques and tools must be consistent with an organizations system development methodology. These make easy for system developers to conduct the steps in methodology. Techniques and tools must make it easy for systems developers to conduct the steps called for in the methodology.

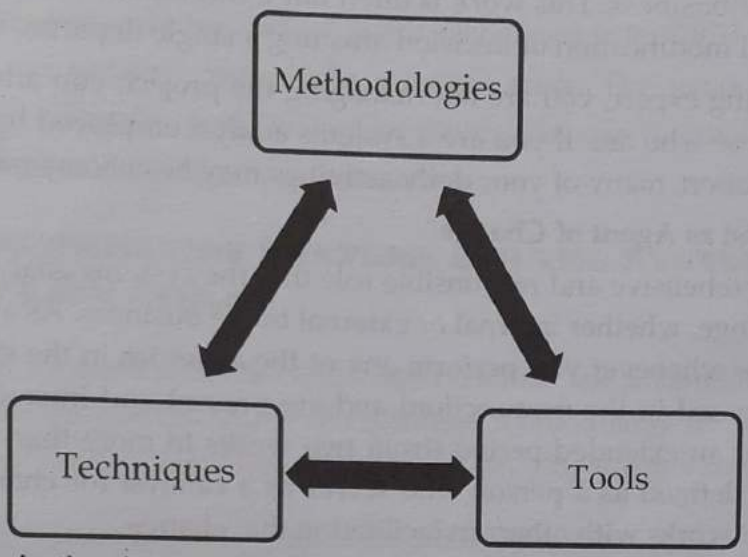


Figure 1.5: An organizational approach to systems analysis and design is driven by methodologies, techniques, and tools.

Although many people in organizations are responsible for systems analysis and design, in most organizations **the systems analyst** has the primary responsibility. When you begin your career in systems development, you will most likely begin as a systems analyst or as a programmer with some systems analysis responsibilities. The primary role of a systems analyst is to study the problems and needs of an organization in order to determine how people, methods, and information technology can best be combined to bring about improvements in the organization. A systems analyst helps system users and other business managers define their requirements for new or enhanced information services. As such, a systems analyst is an agent of change and innovation.

Roles of the Systems Analyst

The analyst plays many roles, sometimes balancing several at the same time. The three primary roles of the systems analyst are consultant, supporting expert, and agent of change.

1. **Systems Analyst as Consultant**

   The systems analyst frequently acts as a systems consultant to humans and their businesses and, thus, may be hired specifically to address information systems issues within a business. Such hiring can be an advantage because outside consultants can bring with them a fresh perspective that other people in an organization do not possess. It also means that outside analysts are at a disadvantage because an outsider can never know the true organizational culture. As an outside consultant, you will rely heavily on the systematic methods discussed throughout this text to analyze and design appropriate information systems for users working in a particular business. In addition, you will rely on information systems users to help you understand the organizational culture from others' viewpoints.

2. **Systems Analyst as Supporting Expert**

Another role that you may be required to play is that of supporting expert within a business for which you are regularly employed in some systems capacity. In this role the analyst draws on professional expertise concerning computer hardware and software and their uses in the business. This work is often not a full-blown systems project, but rather it entails a small modification or decision affecting a single department.

As the supporting expert, you are not managing the project; you are merely serving as a resource for those who are. If you are a systems analyst employed by a manufacturing or service organization, many of your daily activities may be encompassed by this role.

3. **Systems Analyst as Agent of Change**

The most comprehensive and responsible role that the systems analyst takes on is that of an agent of change, whether internal or external to the business. As an analyst, you are an agent of change whenever you perform any of the activities in the systems development life cycle (discussed in the next section) and are present and interacting with users and the business for an extended period (from two weeks to more than a year). An agent of change can be defined as a person who serves as a catalyst for change, develops a plan for change, and works with others in facilitating that change.

Your presence in the business changes it. As a systems analyst, you must recognize this fact and use it as a starting point for your analysis. Hence, you must interact with users and management (if they are not one and the same) from the very beginning of your project. Without their help you cannot understand what they need to support their work in the organization, and real change cannot take place.

If change (that is, improvements to the business that can be realized through information systems) seems warranted after analysis, the next step is to develop a plan for change along with the people who must enact the change. Once a consensus is reached on the change that is to be made, you must constantly interact with those who are changing.

As a systems analyst acting as an agent of change, you advocate a particular avenue of change involving the use of information systems. You also teach users the process of change, because changes in the information system do not occur independently; rather, they cause changes in the rest of the organization as well.

A Modern Approach to Systems Analysis and Design

The growth of computer-based information systems analysis and design methodologies started during the year 1950 to 1960. The researchers argued that the software crisis was due to the lack of discipline of programmers and Some believed that if formal engineering methodologies would be applied to software development, then production of software would become as predictable an industry as other branches of engineering and they advocated proving all programs correct using models such as the Capability Maturity Model.

In 1986, No Silver Bullet article was published by Fred Brooks which described that no individual technology or practice would ever make a 10fold improvement in productivity of software within 10 years. So they realized the need for developing the software in a structured manner. However, it could also be said that there are, in fact, a range of silver bullets today, including spreadsheet calculators, lightweight methodologies, in-site search engines,

customized browsers, integrated design-test coding-editors, database report generators and each issue in software is related with only a small portion of the entire problem which makes the systems analysis and design approaches too complex for finding complete solution to all problems.

The new technologies and practices which were developed after 1970 -1990 were primarily focused on solving the software issues like software crisis. The major elements used were software tools, formal methods, well defined processes that use the methodologies like OOP, CASE tools and Structured Programming approaches.

DEVELOPING INFORMATION SYSTEMS AND THE SYSTEM DEVELOPMENT LIFE CYCLE

When developing information systems, most organizations use a standard of steps called the systems development lifecycle (SDLC) at the common methodology for systems development. SDLC includes phases such as planning, analysis, design, implementation, and maintenance as shown in figure 1.6.

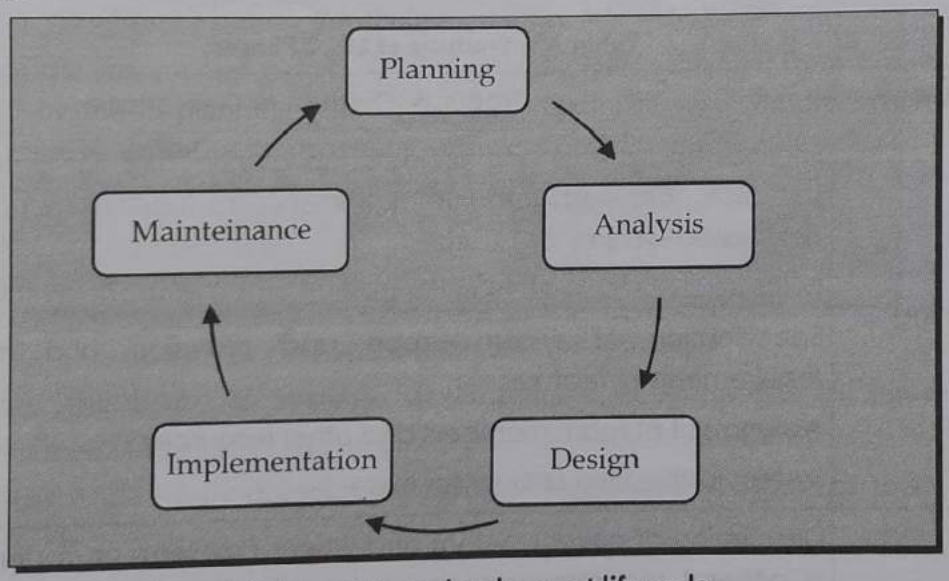


Figure 1.6: The systems development life cycle

The first phase is called **planning**. In this phase, an organization's total information system needs are identified, analyzed, prioritized, and arranged. At the heart of systems development analysis and design are the second and third phases of SDLC. The **analysis** phase usually requires a careful study of the current system, which continues two sub phases: requirements determination and analysis study. **Requirements determination** process usually involves a careful study of the current manual and computerized systems that may be replaced or improved within the project. **Analysis study** process usually involves analysts to study the structural requirements according to the components interrelationships and eliminate redundancies. In the **design** phase, the description of the recommended solution is converted into logical and then physical system specification. Analysts design all aspects of the system, provide physical specifics on the system from input and output screens to reports, databases, and computer processes. **Logical design** is the part of the design process that is independent of

any specific hardware or software platform. Theoretically, the system could be implemented on any hardware and systems software. **Physical design** is the part of the design phase in which the logical specifications of the system form logical design are transformed into technology-specific details from which all programming and system construction can be accomplished. The fourth phase is called **implementation**. In this phase, the information system is coded, tested, installed, and supported in the organization. During coding, programmers write the programs that make up the information system. During testing, programmers and analysts test individual programs and the entire system in order to find and correct errors. During installation, the new system becomes a part of the daily activities of the organization. Implementation activities also include initial user support such as the finalization of documentation, training programs, and ongoing user assistance. The final phase of SDLC is called **maintenance**. In this phase, information system is systematically repaired and improved. When a system is operating in an organization, users sometimes find problems with how it works and often think of better ways to perform its functions. Also the organization's needs with respect to the system change over time.

Table 1.1: Products of SDLC Phases

Phase	Products, Outputs, or Deliverables
Planning	Priorities for systems and projects; an architecture for data, networks, and selection hardware, and information systems management are the result of associated systems Detailed steps, or work plan, for project Specification of system scope and planning and high-level system requirements or features Assignment of team members and other resources System justification or business case
Analysis	Description of current system and where problems or opportunities exist, with a general recommendation on how to fix, enhance, or replace current system Explanation of alternative systems and justification for chosen alternative
Design	Functional, detailed specifications of all system elements (data, processes, inputs, and outputs) Technical, detailed specifications of all system elements (programs, files, network, system software, etc.) Acquisition plan for new technology
Implementation	Code, documentation, training procedures, and support capabilities
Maintenance	New versions or releases of software with associated updates to documentation, training, and support

THE HEART OF THE SYSTEM DEVELOPMENT PROCESS

In many ways, though, the SDLC is fiction. Although almost all systems development projects adhere to some type of life cycle, the exact location of activities and the specific sequencing of steps can vary greatly from one project to the next. Current practice combines the activities traditionally thought of as belonging to analysis, design, and implementation into a single process. Instead of systems requirements being produced in analysis, systems specifications being created in design, and coding and testing being done at the beginning of implementation, current practice combines all of these activities into a single analysis-design-code-test process as shown in figure 1.7.

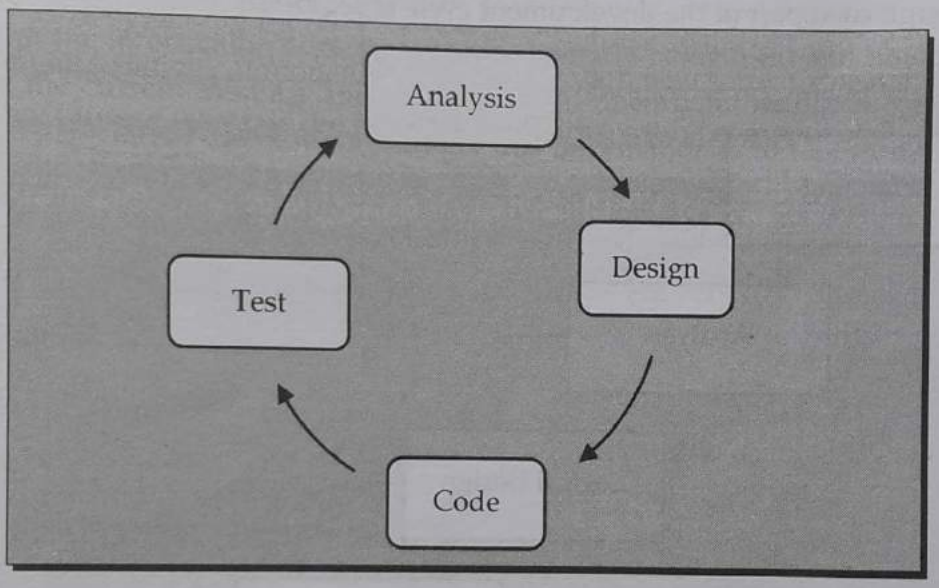


Figure 1.7: The analysis-design-code-test loop

These activities are the heart of systems development, as suggested in figure 1.8.This combination of activities is typical of current practices in Agile Methodologies. A well-known instance of one of the Agile Methodologies is eXtreme Programming, although there are other variations.

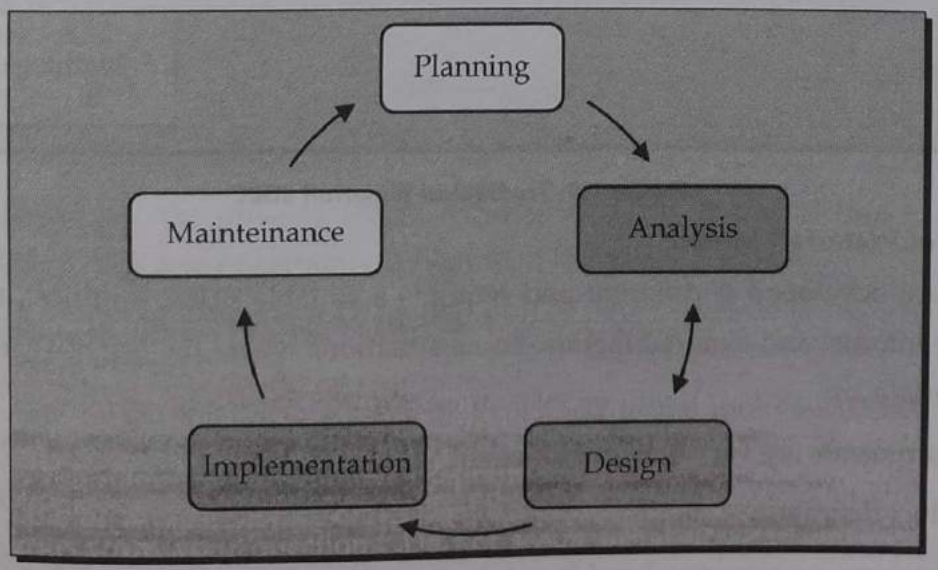


Figure 1.8: The heart of system development

TRADITIONAL WATERFALL SDLC

Waterfall model (sometimes called **classic life cycle** or **the linear sequential model**) is the oldest and the most widely used paradigm for information systems development. This structured approach looks at the system from a top-down view. It is a formalized step by step approach to the SDLC which consists of phases or activities. The activities of one phase must be completed before moving to the next phase. At the completion of each activity or phase, a milestone has been reached and a document is produced to be approved by the stakeholders before moving to the next activity or phase; painstaking amounts of documentation and signoffs through each part of the development cycle is required.

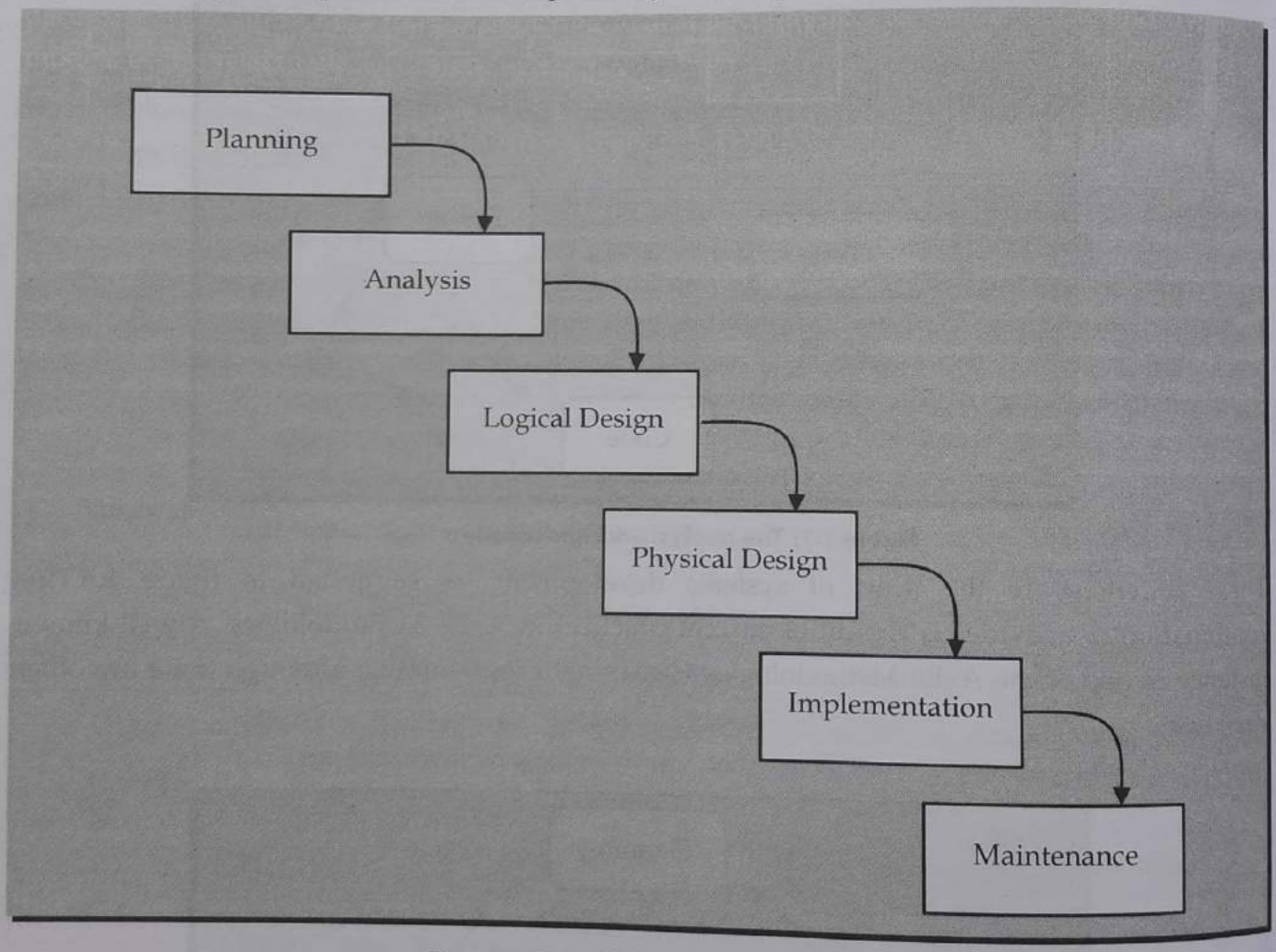


Figure 1.9: Traditional Waterfall SDLC

Application of Waterfall Model

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed
- Product definition is stable
- Technology is understood and is not dynamic

- There are no ambiguous requirements
- Ample resources with required expertise are available to support the product
- The project is short

Advantages of waterfall model

The advantage of waterfall model is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one. Each phase of development proceeds in strict order. Following are lists of the advantages of waterfall model.

- Simple and easy to understand and use.
- Easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Disadvantages of waterfall model

The disadvantage of waterfall model is that it does not allow for much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage. The disadvantages of the waterfall model are listed below:

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model of complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- No working software is produced until late in the life cycle.
- Adjusting scope during the life cycle can end a project
- Integration is done as a "big-bang" at the very end, which doesn't allow identifying and technological or business bottleneck or challenges early.

CASE Tools

Computer-aided systems engineering (CASE), also called **computer-aided software engineering**, is a technique that uses powerful software, called **CASE tool**, to help systems analysts develop and maintain information systems. CASE tools provide an overall framework for systems development and support a wide variety of design methodologies, including structured analysis and object-oriented analysis.

Because CASE tools make it easier to build an information system, they boost IT productivity and improve the quality of the finished product. After developing a model, many CASE tools can generate program code, which speeds the implementation process. Figure 1.10 shows the sample for Visual Case Tool Software.

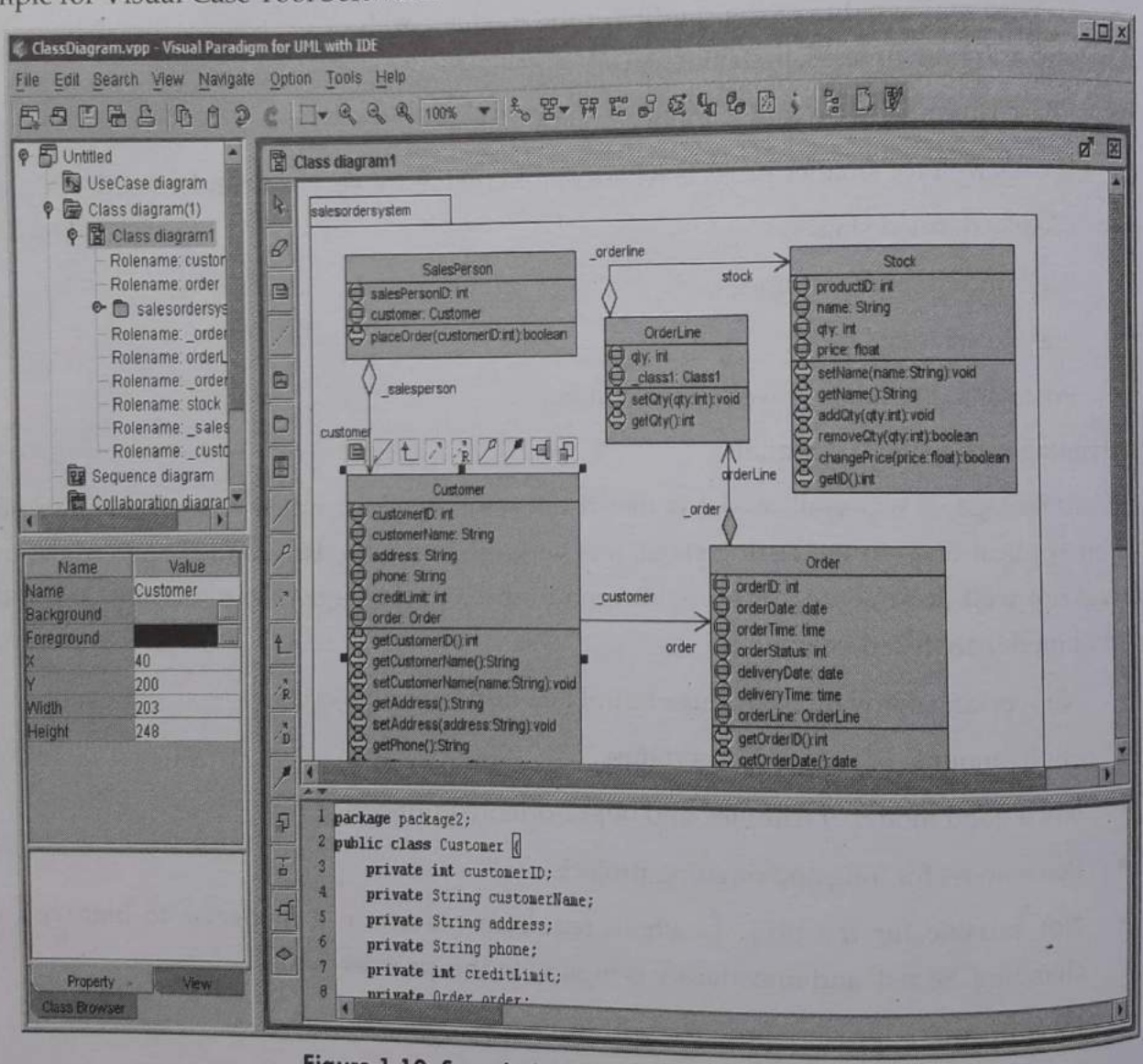


Figure 1.10: Sample for visual case tool software

Types of CASE Tools

- **Diagram Tools:** It helps in diagrammatic and graphical representations of the data and system processes. It represents system elements, control flow and data flow among different software components and system structure in a pictorial form. For example, Flow Chart Maker tool for making state-of-the-art flowcharts.

- **Computer Display and Report Generators:** It helps in understanding the data requirements and the relationships involved.
- **Analysis Tools:** It focuses on inconsistent, incorrect specifications involved in the diagram and data flow. It helps in collecting requirements; automatically check for any irregularity, imprecision in the diagrams, data redundancies or erroneous omissions.
- **Central Repository:** It provides the single point of storage for data diagrams, reports and documents related to project management.
- **Documentation Generators:** It helps in generating user and technical documentation as per standards. It creates documents for technical users and end users. For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.
- **Code Generators:** It aids in the auto generation of code, including definitions, with the help of the designs, documents and diagrams.

Advantages of the CASE Tool

- As special emphasis is placed on redesign as well as testing, the servicing cost of a product over its expected lifetime is considerably reduced.
- The overall quality of the product is improved as an organized approach is undertaken during the process of development.
- Chances to meet real-world requirements are more likely and easier with a computer-aided software engineering approach.
- CASE indirectly provides an organization with a competitive advantage by helping ensure the development of high-quality products.

Disadvantages of CASE tools

- Purchasing of CASE tools is Not an Easy Task. The cost of CASE tools is very high. For this reason small software development firms do not invest in CASE tools.
- Learning Curve. In general, programmer productivity may fall in the initial phase of implementation as users need time to learn this technology.
- Tool Mix. It is important to make proper selection of CASE tools to get maximum benefits from the tools, as the wrong selection may lead to the wrong results.

Today's CASE tools provide two distinct ways to develop system models - forward engineering and reverse engineering. Forward engineering requires the system analyst to draw system models, either from scratch or from templates. The resulting models are subsequently transformed into program code. Reverse engineering, on the other hand, allows a CASE tool to read existing program code and transform that code into a representative system model that can be edited and refined by the systems analyst. CASE tools that allow for bi-directional, forward and reverse engineering are said to provide for "round-trip engineering".

OTHER APPROACHES

In the efforts to improve the systems analysis and design processes, different approaches have been developed. The traditional waterfall approach focuses on compartmentalizing project into several phases. The agile approach focuses on self-adaptive processes with an emphasis on individual talents. The object-oriented approach focuses on combining data and processes into objects and shares the iterative development approach of the agile method. These approaches all have different pros and cons in a way that they could be used to fit and optimize different kinds of projects. Some approaches are explained below:

PROTOTYPING APPROACH

The Prototyping Model is a methodology that is treated as a model for software development where a prototype - which is a premature approximated sample of the final product, is constructed and then tested. After that rework is done on that unfinished product as per requirement in anticipation of building a suitable prototype that is, at last, attain after the entire software is developed and then it is delivered to the customer. It is a useful model for those whose project requirement is not fully known or there is a constant update required based on customer satisfaction. It can be considered as a trial-and-error method which takes place involving the developers as well as the users. The model has two types. These are:

1. **Rapid Throwaway Prototyping**: In this method, developers can explore the ideas as well as get proper customer feedback. Here, the prototype need not be a final one, and so it can be further iterated to develop new versions of the final product.
2. **Evolutionary Prototyping:** Here your developed prototype will primarily be incremented for refining on the foundation of customer opinion until the final one gets accepted. It provides an improved way which can save time and effort.

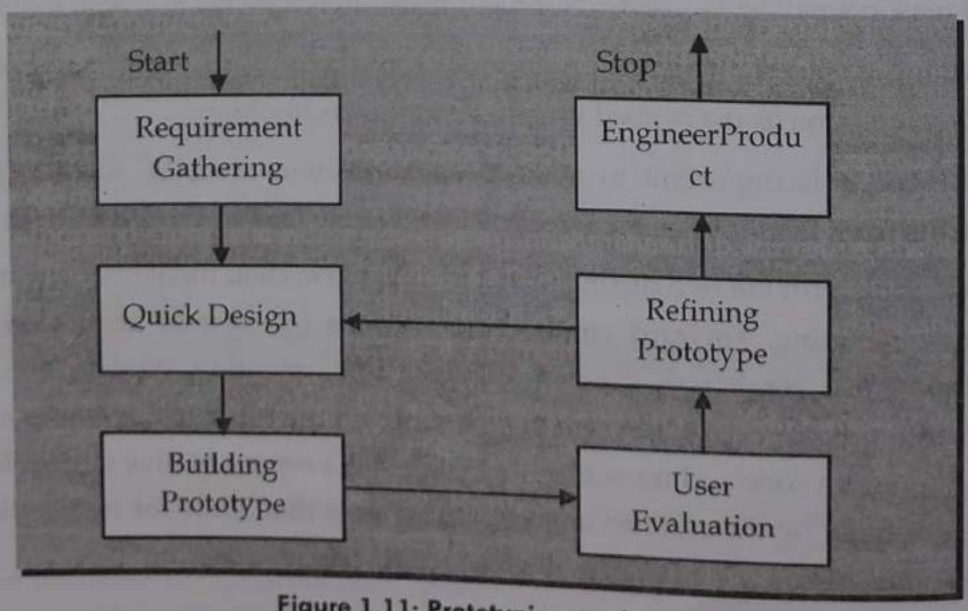


Figure 1.11: Prototyping model

Phases of Prototyping Model

1. **Requirements Gathering:** A prototyping model begins with requirements analysis and the requirements of the system are defined in detail. The user is interviewed in order to know the requirements of the system.
2. **Quick Design:** When requirements are known, a preliminary design or quick design for the system is created. It is not a detailed design and includes only the important aspects of the system, which gives an idea of the system to the user. A quick design helps in developing the prototype.
3. **Build Prototype:** Information gathered from quick design is modified to form the first prototype, which represents the working model of the required system.
4. **User Evaluation:** Next, the proposed system is presented to the user for thorough evaluation of the prototype to recognize its strengths and weaknesses such as what is to be added or removed. Comments and suggestions are collected from the users and provided to the developer.
5. **Refining Prototype:** Once the user evaluates the prototype and if he is not satisfied, the current prototype is refined according to the requirements. That is, a new prototype is developed with the additional information provided by the user. The new prototype is evaluated just like the previous prototype. This process continues until all the requirements specified by the user are met. Once the user is satisfied with the developed prototype, a final system is developed on the basis of the final prototype.
6. **Engineer Product:** Once the requirements are completely met, the user accepts the final prototype. The final system is evaluated thoroughly followed by the routine maintenance on regular basis for preventing large-scale failures and minimizing downtime.

Advantages of Prototyping Model

- The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.
- New requirements can be easily accommodated as there is scope for refinement.
- Missing functionalities can be easily figured out.
- Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
- The developed prototype can be reused by the developer for more complicated projects in the future.
- Flexibility in design.

Disadvantages of Prototyping Model

- There are no parallel deliverables
- It is a time consuming if customer asks for changes in prototype
- This methodology may increase the system complexity as scope of the system may expand beyond original plans.
- The invested effort in the preparation of prototypes may be too much if not properly monitored.
- Customer may get confused in the prototypes and real systems.

SPIRAL APPROACH

The spiral development model is a risk-driven process model generator that is used to guide multi-stakeholder concurrent engineering of software intensive systems. It has two main distinguishing features. One is a cyclic approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk. The other is a set of anchor point milestones for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions.

In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using spiral model. The Radius of the spiral at any point represents the expenses (cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

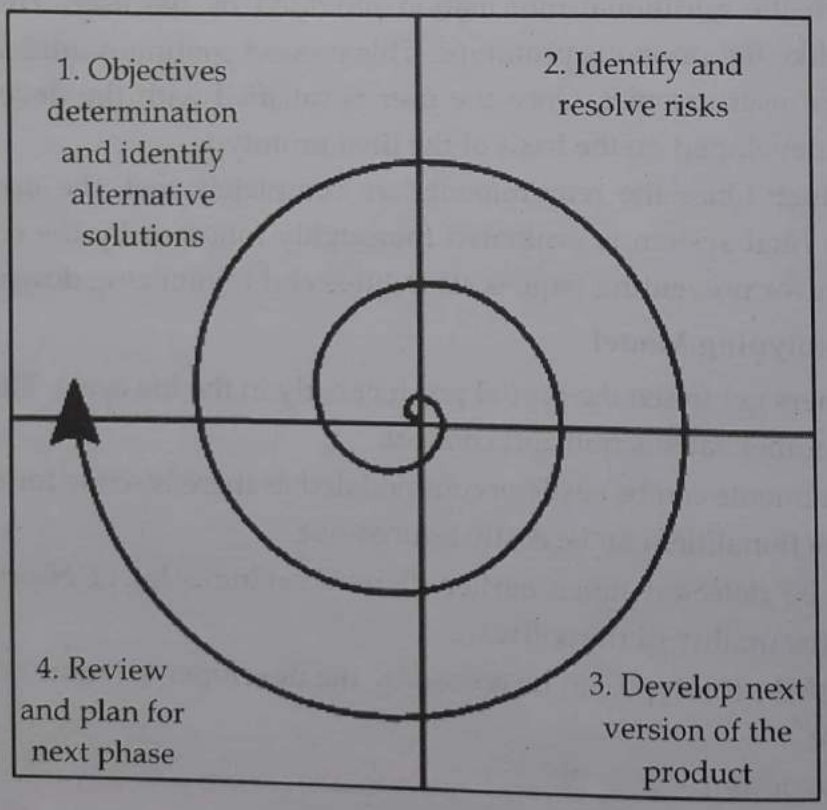


Figure 1.12: Different phases of spiral model

Each phase of Spiral Model is divided into four quadrants as shown in figure 1.12. The functions of these four quadrants are discussed below:

1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

2. **Identify and resolve Risks:** During the second quadrant all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution is identified and the risks are resolved using the best possible strategy. At the end of this quadrant, Prototype is built for the best possible solution.
3. **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
4. **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

Advantages of Spiral Model

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the software life cycle.

Disadvantages of Spiral Model

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects.

RAPID APPLICATION DEVELOPMENT APPROACH

Rapid application development (RAD) is an object-oriented approach to systems development that includes a method of development as well as software tools. It makes sense to discuss RAD and prototyping in the same chapter, because they are conceptually very close. Both have as their goal the shortening of time typically needed in a traditional SDLC between the design and implementation of the information system. Ultimately, both RAD and prototyping are trying to meet rapidly changing business requirements more closely. Once you have learned the concepts of prototyping, it is much easier to grasp the essentials of RAD, which can be thought of as a specific implementation of prototyping.

Some developers are looking at RAD as a helpful approach in new ecommerce, Web-based environments in which so-called first-mover status of a business might be important. In other words, to deliver an application to the Web before their competitors, businesses may want their development team to experiment with RAD.

Phases of RAD Model

There are three broad phases to RAD that engage both users and analysts in assessment, design, and implementation. These three phases are shown in figure 1.13. Notice that RAD involves users in each part of the development effort, with intense participation in the business part of the design.

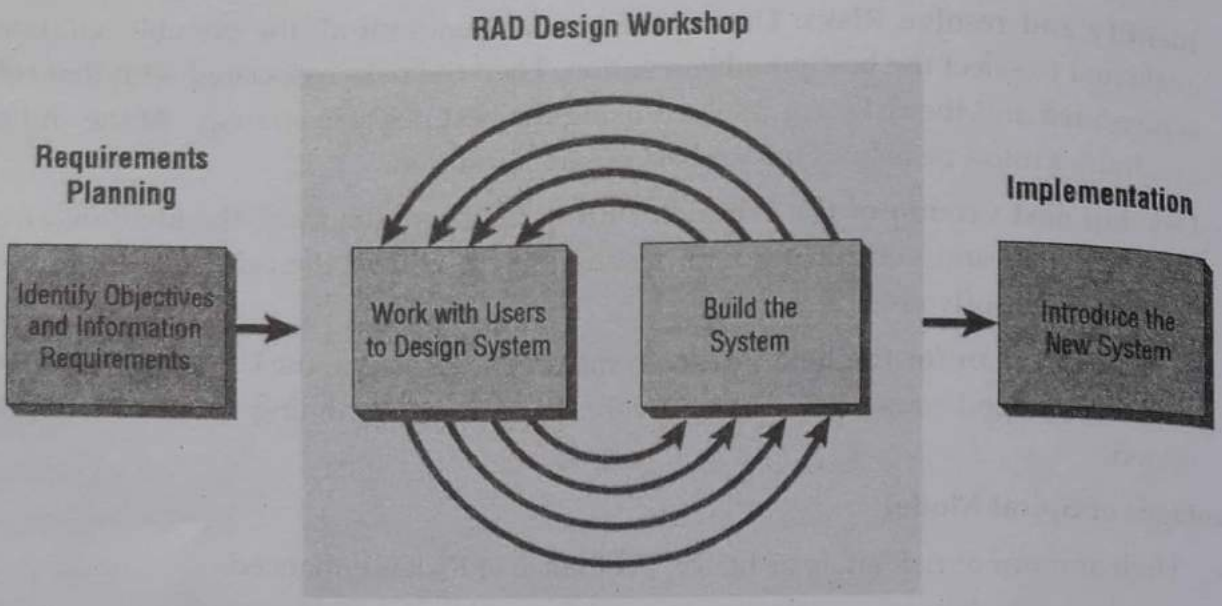


Figure 1.13: The RAD design workshop is the heart of the interactive development process

Requirements Planning Phase

In the requirements planning phase, users and analysts meet to identify objectives of the application or system and to identify information requirements arising from those objectives. This phase requires intense involvement from both groups; it is not just signing off on a proposal or document. In addition, it may involve users from different levels of the organization. In the requirements planning phase, when information requirements are still being addressed, you may be working with the Chief information officer (CIO) (if it is a large organization) as well as with strategic planners, especially if you are working with an ecommerce application that is meant to further the strategic aims of the organization. The orientation in this phase is toward solving business problems. Although information technology and systems may even drive some of the solutions proposed, the focus will always remain on reaching business goals.

RAD Design Workshop

The RAD design workshop phase is a design-and-refine phase that can best be characterized as a workshop. When you imagine a workshop, you know that participation is intense, not passive, and that it is typically hands on. Usually participants are seated at round tables or in a U-shaped configuration of chairs with attached desks where each person can see the other and where there is space to work on a notebook computer. If you are fortunate enough to have a group decision support systems (GDSS) room available at the company or through a local university, use it to conduct at least part of your RAD design workshop.

During the RAD design workshop, users respond to actual working prototypes and analysts refine designed modules based on user responses. The workshop format is very exciting and stimulating, and if experienced users and analysts are present, there is no question that this creative endeavor can propel development forward at an accelerated rate.

Implementation Phase

Analysts work with users intensely during the workshop to design the business or nontechnical aspects of the system. As soon as these aspects are agreed on and the systems are built and

refined, the new systems or part of systems are tested and then introduced to the organization. Because RAD can be used to create new ecommerce applications for which there is no old system, there is often no need to (and no real way to) run the old and new systems in parallel before implementation.

By this time, the RAD design workshop will have generated excitement, user ownership, and acceptance of the new application. Typically, change brought about in this manner is far less wrenching than when a system is delivered with little or no user participation.

Advantages of RAD Model

- Changing requirements can be accommodated.
- Progress can be measured.
- Iteration time can be sort with use of powerful RAD tools.
- Productivity with fewer people in short time.
- Reduce development time.
- Increases reusability of components.

Disadvantages of RAD Model

- 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 modeling skills.
- Inapplicable to cheaper projects as cost.

INTRODUCTION TO AGILE DEVELOPMENT APPROACH

The Agile software development model was mainly intended for helping developers build a project which can adapt to transforming requests quickly. So, the most important endeavor for developing the Agile model is to make easy and rapid project achievement. For attaining this task, developers need to preserve the agility during development. Agility can be achieved by correcting the progression to the project by eliminating activities which may not be crucial for that specific project.

Developmental Process for an Agile Project

There are activities and behaviors that shape the way development team members and customers act during the development of an agile project. Two words that characterize a project done with an agile approach are interactive and incremental. By examining the figure 1.14 we can see that there are five distinct stages: exploration, planning, iterations to the first release, productionizing, and maintenance. Notice that the three red arrows that loop back into the "Iterations" box symbolize incremental changes created through repeated testing and feedback that eventually lead to a stable but evolving system. Also note that there are multiple looping arrows that feed back into the productionizing phase. These symbolize that the pace of

iterations is increased after a product is released. The red arrow is shown leaving the maintenance stage and returning to the planning stage, so that there is a continuous feedback loop involving customers and the development team as they agree to alter the evolving system.

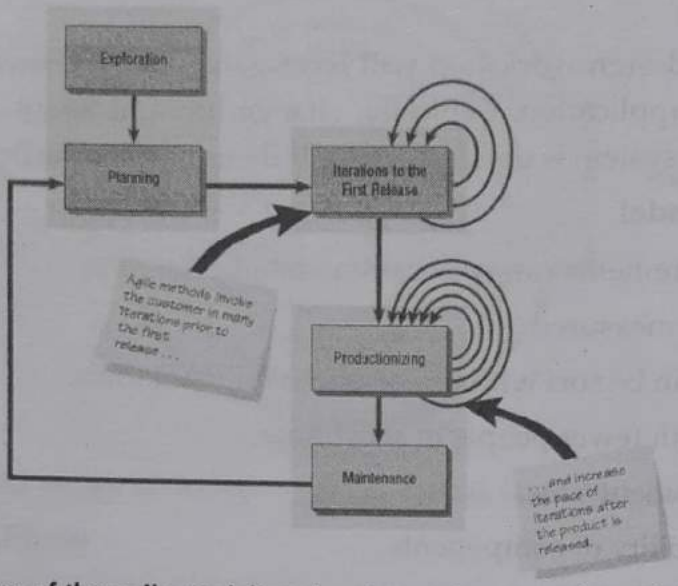


Figure 1.14: The five stages of the agile modeling development process show that frequent iterations are essential to successful system development

Exploration

During exploration, you will explore your environment, asserting your conviction that the problem can and should be approached with agile development, assemble the team, and assess team member skills. This stage will take anywhere from a few weeks (if you already know your team members and technology) to a few months (if everything is new). You also will be actively examining potential technologies needed to build the new system. During this stage you should practice estimating the time needed for a variety of tasks. In exploration, customers also are experimenting with writing user stories. The point is to get the customer to refine a story enough so that you can competently estimate the amount of time it will take to build the solution into the system you are planning. This stage is all about adopting a playful and curious attitude toward the work environment, its problems, technologies, and people.

Planning

The next stage of the agile development process is called planning. In contrast to the first stage, planning may only take a few days to accomplish. In this stage you and your customers agree on a date anywhere from two months to half a year from the current date to deliver solutions to their most pressing business problems (you will be addressing the smallest, most valuable set of stories). If your exploration activities were sufficient, this stage should be very short.

The entire agile planning process has been characterized using the idea of a planning game as devised by Beck. The planning game spells out rules that can help formulate the agile development team's relationship with their business customers. Although the rules form an idea of how you want each party to act during development, they are not meant as a replacement for a relationship. They are a basis for building and maintaining a relationship.

So, we use the metaphor of a game. To that end we talk in terms of the goal of the game, the strategy to pursue, the pieces to move, and the players involved. The goal of the game is to maximize the value of the system produced by the agile team. In order to figure the value, you

have to deduct costs of development, and the time, expense, and uncertainty taken on so that the development project could go forward.

The strategy pursued by the agile development team is always one of limiting uncertainty (downplaying risk). To do that they design the simplest solution possible, put the system into production as soon as possible, get feedback from the business customer about what's working, and adapt their design from there.

Story cards become the pieces in the planning game that briefly describe the task, provide notes, and provide an area for task tracking. There are two main players in the planning game: the development team and the business customer. Deciding which business group in particular will be the business customer is not always easy, because the agile process is an unusually demanding role for the customer to play. Customers decide what the development team should tackle first. Their decisions will set priorities and check functionalities throughout the process.

Iterations to the First Release

The third stage in the agile development process is composed of iterations to the first release. Typically these are iterations (cycles of testing, feedback, and change) of about three weeks in duration. You will be pushing yourself to sketch out the entire architecture of the system, even though it is just in outline or skeletal form. One goal is to run customer-written functional tests at the end of each iteration. During the iterations stage you should also question whether the schedule needs to be altered or whether you are tackling too many stories. Make small rituals out of each successful iteration, involving customers as well as developers. Always celebrate your progress, even if it is small, because this is part of the culture of motivating everyone to work extremely hard on the project.

Productionizing

Several activities occur during this phase. In this phase the feedback cycle speeds up so that rather than receiving feedback for an iteration every three weeks, software revisions are being turned around in one week. You may institute daily briefings so everyone knows what everyone else is doing. The product is released in this phase, but may be improved by adding other features. Getting a system into production is an exciting event. Make time to celebrate with your teammates and mark the occasion. One of the watchwords of the agile approach, with which we heartily agree, is that it is supposed to be fun to develop systems!

Maintenance

Once the system has been released, it needs to be kept running smoothly. New features may be added, riskier customer suggestions may be considered, and team members may be rotated on or off the team. The attitude you take at this point in the developmental process is more conservative than at any other time. You are now in a "keeper of the flame" mode rather than the playful one you experienced during exploration.

Advantages of Agile Development Model

- Working through Pair programming produce well written compact programs which has fewer errors as compared to programmers working alone.
- It reduces total development time of the whole project.
- Customer representative get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.

Disadvantages of Agile Development Model

- Due to lack of formal documents, it creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.
- Due to absence of proper documentation, when the project completes and the developers are assigned to another project, maintenance of the developed project can become a problem.

EXTREME PROGRAMMING

Extreme programming (XP) is one of the most important software development frameworks of Agile models. It is used to improve software quality and responsive 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.

Good practices needs to practiced extreme programming: Some of the good practices that have been recognized in the extreme programming model and suggested to maximize their use are given below:

- **Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their works between them every hour.
- **Testing:** Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach test cases are written even before any code is written.
- **Incremental development:** Incremental development is very good because customer feedback is gained and based on this development team come up with new increments every few days after each iteration.
- **Simplicity:** Simplicity makes it easier to develop good quality code as well as to test and debug it.
- **Design:** Good quality design is important to develop good quality software. So, everybody should design daily.
- **Integration testing:** It helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.

Basic principles of Extreme programming: XP is based on the frequent iteration through which the developers implement User Stories. User stories are simple and informal statements of the customer about the functionalities needed. A User story is a conventional description by the user about a feature of the required system. It does not mention finer details such as the different scenarios that can occur. On the basis of User stories, the project team proposes Metaphors. Metaphors are a common vision of how the system would work. The development team may decide to build a Spike for some feature. A Spike is a very simple program that is constructed to explore the suitability of a solution being proposed. It can be considered similar to a prototype. Some of the basic activities that are followed during software development by using XP model are given below:

- **Coding:** The concept of coding which is used in XP model is slightly different from traditional coding. Here, coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system and choosing among several alternative solutions.
- **Testing:** XP model gives high importance on testing and considers it be the primary factor to develop fault-free software.
- **Listening:** The developers need to carefully listen to the customers if they have to develop good quality software. Sometimes programmers may not have the depth knowledge of the system to be developed. So, it is desirable for the programmers to understand properly the functionality of the system and they have to listen to the customers.
- **Designing:** Without a proper design, a system implementation becomes too complex and very difficult to understand the solution, thus it makes maintenance expensive. A good design results elimination of complex dependencies within a system. So, effective use of suitable design is emphasized.
- **Feedback:** One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs. Frequent contact with the customer makes the development effective.
- **Simplicity:** The main principle of the XP model is to develop a simple system that will work efficiently in present time, rather than trying to build something that would take time and it may never be used. It focuses on some specific features that are immediately needed, rather than engaging time and effort on speculations of future requirements.

Applications of Extreme Programming (XP): Some of the projects that are suitable to develop using XP model are given below:

- **Small projects:** XP model is very useful in small projects consisting of small teams as face to face meeting is easier to achieve.
- **Projects involving new technology or Research projects:** This type of projects faces changing of requirements rapidly and technical problems. So XP model is used to complete this type of projects.

Advantages of Extreme Programming

Extreme Programming solves the following problems often faced in the software development projects –

- **Slipped schedules:** and achievable development cycles ensure timely deliveries.
- **Cancelled projects:** Focus on continuous customer involvement ensures transparency with the customer and immediate resolution of any issues.
- **Costs incurred in changes:** Extensive and ongoing testing makes sure the changes do not break the existing functionality. A running working system always ensures sufficient time for accommodating changes such that the current operations are not affected.
- **Production and post-delivery defects: Emphasis is on:** the unit tests to detect and fix the defects early.

- **Misunderstanding the business and/or domain:** Making the customer a part of the team ensures constant communication and clarifications.
- **Business changes:** Changes are considered to be inevitable and are accommodated at any point of time.
- **Staff turnover:** Intensive team collaboration ensures enthusiasm and good will. Cohesion of multi-disciplines fosters the team spirit.

Disadvantages of Extreme Programming

Here are the disadvantages of Extreme Programming:

- **Difficulty:** This is technically a tough software practice so convincing developers and programmers to adopt it won't be easy. It requires customer devotion as well as lots and lots of team discipline. Its life cycle has very many different changes which mean that those who manage this type of software projects are bound to be faced with numerous difficulties.
- **XP Relies on Very Many Factors:** This is basically a minimalist process. Its lack of vigor is made up by the number of practices involved. The project has a high risk of failure if something goes wrong. This happens to be a very big disadvantage when it comes to this type of software development. The fact of the matter is that extreme programming is a highly risky endeavor.
- **Code Centric:** This type of programming methodology is code-centric rather than design-centric. This can prove to be very tiring when larger software projects are involved.

SERVICE-ORIENTED ARCHITECTURE

Modular development has led to a concept called **service-oriented architecture (SOA),** but one that is very different from the modules in the structure chart. Instead of being hierarchical like the top-down approach found in structure charts, the SOA approach is to make individual SOA services that are unassociated or only loosely coupled to one another.

Each service executes one action. One service may return the number of days in this month; another may tell us if this is a leap year; a third service may reserve five nights in a hotel room from the end of February to the beginning of March. Although the third service needs to know the values obtained from the first and second services, they are independent of one another. Each service can be used in other applications within the organization or even in other organizations.

We can say that service-oriented architecture is simply a group of services that can be called upon to provide specific functions. Rather than including calls to other services, a service can use certain defined protocols so that it can communicate with other services. **Figure 1.15** illustrates how services are called upon throughout the system. Services can be general in nature and can be outsourced or even be available on the Web. Other services are more specialized and oriented toward the business itself. These enterprise-based services provide business rules and can also differentiate one business from another. Services can be called upon at a time and can be called on repeatedly in many application modules.

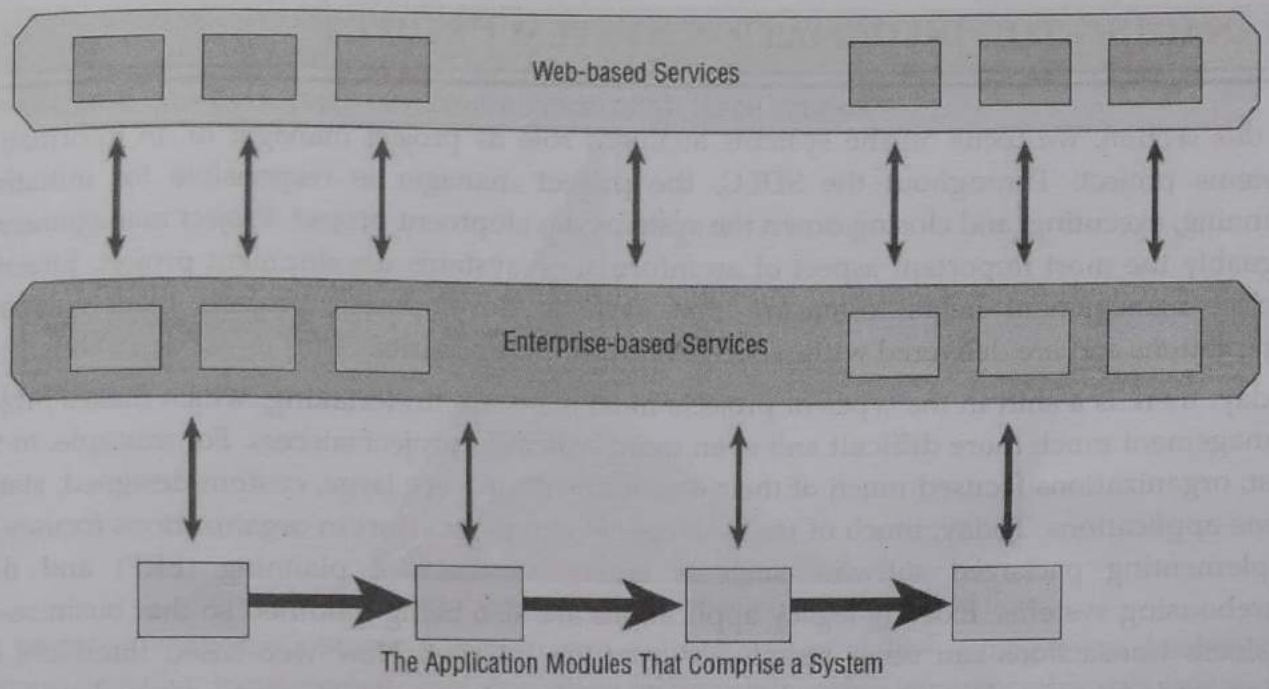


Figure 1.15: Modules in SOA are independent and can be ubiquitous

The burden of connecting services in a useful fashion, a process called orchestration, is placed upon the systems designer. This can even be accomplished by selecting services from a menu of services and monitoring them by setting up an SOA dashboard.

In order to set up an SOA, the services must be:

1. Modular,
2. Reusable,
3. Work together with other modules (interoperability),
4. Able to be categorized and identified,
5. Able to be monitored, and
6. Comply with industry-specific standards.

While the advantages of reusability and interoperability are obvious, SOA is not without its challenges. First industry standards must be agreed upon. Next, a library must be maintained so that developers can find the services they need. Finally, security and privacy can be issues when using software developed by someone else.

Object-Oriented Analysis and Design

In this section overview of Object-oriented analysis and design (OOAD) will be discussed (see unit 6: Introduction to Object-Oriented Development for detail). **OOAD** is often called the third approach to systems development, after the process-oriented and data-oriented approaches. 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. The goal of OOAD is to make systems elements more reusable, thus improving system quality and the productivity of systems analysis and design. Another key idea behind object orientation is **inheritance**. Objects are organized into **object classes**, which are groups of objects sharing structural and behavioral characteristics.

MANAGING THE INFORMATION SYSTEM PROJECT

In this section, 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.

Today, there is a shift in the types of projects most firms are undertaking, which makes project management much more difficult and even more critical to project success. For example, in the past, organizations focused much of their development on very large, custom-designed, stand-alone applications. Today, much of the systems development effort in organizations focuses on implementing packaged software such as enterprise resource planning (ERP) and data warehousing systems. Existing legacy applications are also being modified so that business-to-business transactions can occur seamlessly over the Internet. New web-based interfaces are being added to existing legacy systems so that a broader range of users, often distributed globally, can access corporate information and systems. Additionally, software developed by global outsourcing partners that must be integrated into an organization's existing portfolio of applications is now common practice. Working with vendors to supply applications, with customers or suppliers to integrate systems, or with a broader and more diverse user community requires that project managers be highly skilled. Consequently, it is important that you gain an understanding of the project management process; this will become a critical skill for your future success.

Project management is an important aspect of the development of information systems and a critical skill for a systems analyst. The focus of project management is to ensure that systems development projects meet customer expectations and are delivered within budget and time constraints. A **project** is a planned undertaking of a series of related activities to reach an objective that has a beginning and an end.

Shaping a project

A successful project must be completed on time, within budget, and deliver a quality product that satisfies users and meets requirements. Project management techniques can be used throughout the SDLC. System developers can initiate a formal project as early as the preliminary investigation stage, or later on, as analysis, design, and implementation activities occur. Systems development projects tend to be dynamic and challenging. There is always a balance between constraints, and interactive elements such as project cost, scope, and time.

Project Triangle

For each project, it must be decided what is most important, because the work cannot be **good** and **fast** and **cheap**. When it comes to project management, things are not quite so simple. Decisions do not need to be all-or-nothing, but recognize that any change in one leg of the triangle will affect the other two legs. Figure 1.16 represents a common view of a **project triangle**, where the three legs are cost, scope, and time. The challenge is to find the optimal balance among these factors. Most successful project managers rely on personal experience,

communication ability, and resourcefulness. For example, if an extremely time-critical project starts to slip, the project manager might have to trim some features, seek approval for a budget increase, add new personnel, or a combination of all three actions.

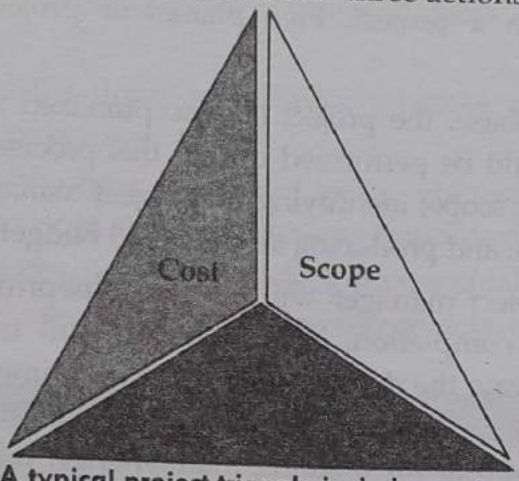


Figure 1.16: A typical project triangle includes cost, scope, and time.

The **project manager** is a systems analyst with a diverse set of skills—management, leadership, technical, conflict management, and customer relationship—who is responsible for initiating, planning, executing, and closing down a project. As a project manager, your environment is one of continual change and problem solving. In some organizations, the project manager is a very experienced systems analyst, whereas in others, both junior and senior analysts are expected to take on this role, managing parts of a project or actively supporting a more senior colleague who assumes the project manager role. Understanding the project management process is a critical skill for your future success. Creating and implementing successful projects requires managing the resources, activities, and tasks needed to complete the information systems project.

Table 1.2: Common activities and skills of a project manager

Activity	Description	Skill
Leadership	Influencing the activities of others toward the attainment of a common goal through the use of intelligence, personality, and abilities	Communication; liaison between management, users, and developers; assigning activities; monitoring progress
Management	Getting projects completed through the effective utilization of resources	Defining and sequencing activities; communicating expectations; assigning resources to activities; monitoring outcomes
Customer relations	Working closely with customers to ensure that project deliverables meet expectations	Interpreting system requests and specifications; site preparation and user training; contact point for customers
Technical problem solving	Designing and sequencing activities to attain project goals	Interpreting system requests and specifications; defining activities and their sequence; making trade-offs between alternative solutions; designing solutions to problems
Conflict management	Managing conflict within a project team to assure that conflict is not too high or too low	Problem solving; smoothing out personality differences; compromising; goal setting
Team management	Managing the project team for effective team performance	Communication within and between teams; peer evaluations; conflict resolution; team building; self-management
Risk and change management	Identifying, assessing, and managing the risks and day-to-day changes that occur during a project	Environmental scanning; risk and opportunity identification and assessment; forecasting; resource redeployment

Phases of Project Management

Project management is a controlled process of initiating, planning, executing, monitoring, controlling, and closing down a project. Five phases of project management process are explained below:

1. **Initiating:** During this phase, the project is conceptualized and feasibility is determined. Some activities that should be performed during this process include defining the project goal; defining the project scope; identifying the project manager and the key stakeholders; identifying potential risks; and producing an estimated budget and timeline.
2. **Planning:** Next, the project manager will create a blueprint to guide the entire project from ideation through completion. This blueprint will map out the project's scope; resources required to create the deliverables; estimated time and financial commitments; communication strategy to ensure stakeholders are kept up to date and involved; execution plan; and proposal for ongoing maintenance. If the project has not yet been approved, this blueprint will serve as a critical part of the pitch.
3. **Executing:** During this phase, the project manager will conduct the procurement required for the project as well as staff the team. Further, execution of the project objectives requires effective management of the team members on the ground. PMs are responsible for delegating and overseeing the work on the project while maintaining good relationships with all team members and keeping the entire project on time and on budget. The PM must therefore be highly organized and an exceptional leader. That's because they'll need to address team concerns and any issues that arise along the way, requiring frequent and open communication with all team members and stakeholders.
4. **Monitoring and control:** During this process group, project managers will closely measure the progress of the project to ensure it is developing properly. Documentation such as data collection and verbal and written status reports may be used. Monitoring and controlling is closely related to project planning. While planning determines what is to be done, monitoring and controlling establish how well it has been done. Monitoring will detect any necessary corrective action or change in the project to keep the project on track.
5. **Closing:** The closing process group occurs once the project deliverables have been produced and the stakeholders validate and approve them. During this phase, the project manager will close contracts with suppliers, external vendors, consultants, and other third-party providers. All documentation will be archived and a final project report will be produced. Further, the final parts of the project plan - the plan for troubleshooting and maintenance.

REPRESENTING AND SCHEDULING PROJECT PLANS

A project manager has a wide variety of techniques available for depicting 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 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 relatively simple projects or subparts of a larger project, the activities of a single worker, or for monitoring the progress of

activities compared to scheduled completion dates as shown in figure 1.17. Recall that a Network diagram shows the ordering of activities by connecting a task to its predecessor and successor tasks (see figure 1.18). Sometimes a Network diagram is preferable; other times a Gantt chart more easily shows certain aspects of a project. Here are the key differences between these two representations:

- A Gantt chart shows the duration of tasks, whereas a Network diagram shows the sequence dependencies between tasks.
- A Gantt chart shows the time overlap of tasks, whereas a Network diagram does not show time overlap but does show which tasks could be done in parallel.
- Some forms of Gantt charts can show slack time available within an earliest start and latest finish date. A Network diagram shows these data within activity rectangles.

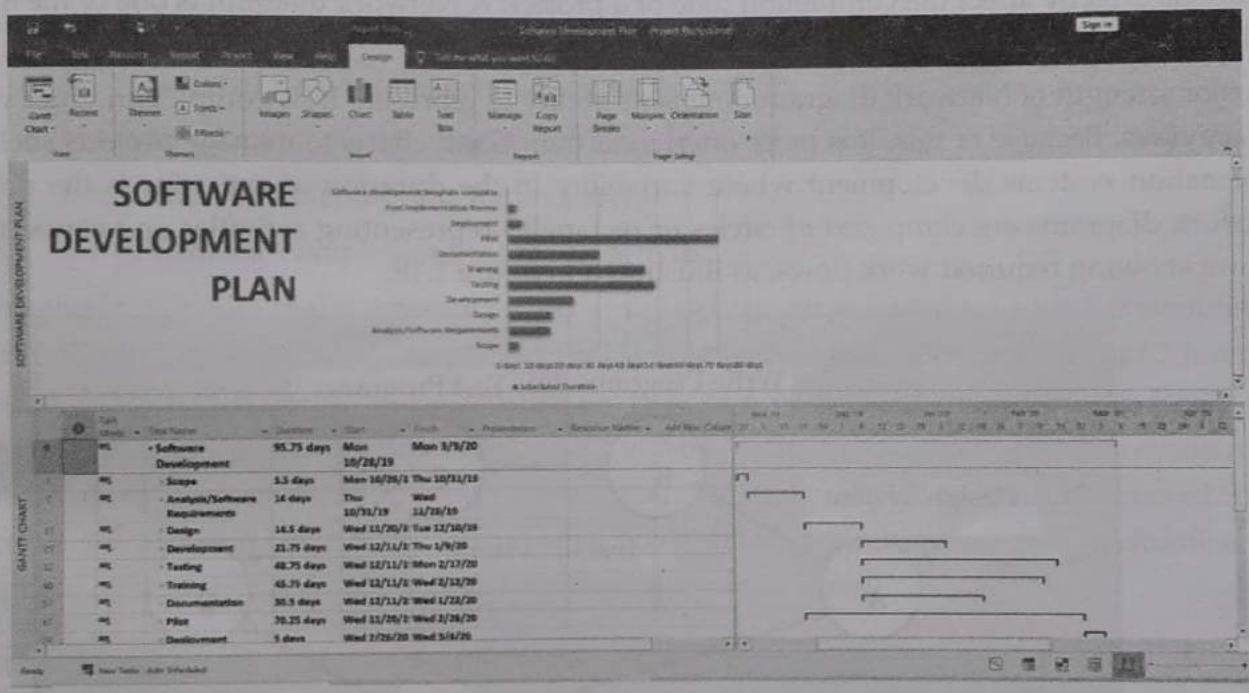


Figure 1.17: A Gantt chart

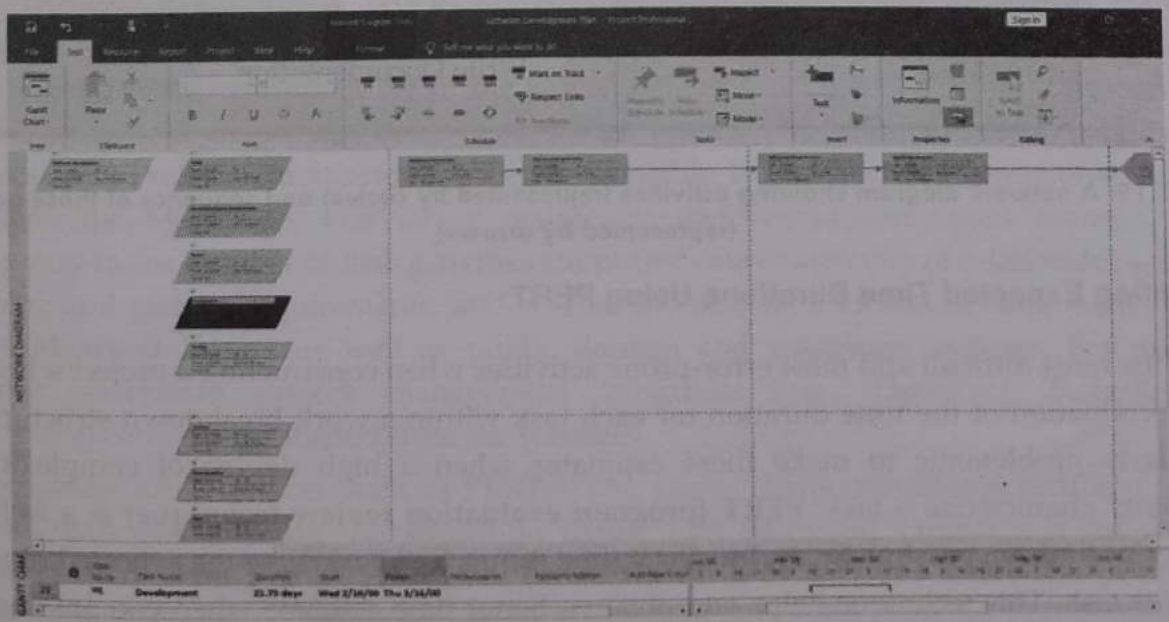


Figure 1.18: A network diagram

Project managers also use textual reports that depict resource utilization by task, complexity of the project, and cost distributions to control activities. Most project managers use computer-based systems to help develop their graphical and textual reports. A project manager will periodically review the status of all ongoing project task activities to assess whether the activities will be completed early, on time, or late.

Representing Project Plans

Project scheduling and management requires 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.

A major strength of Network diagramming is its ability to represent how completion times vary for activities. Because of this, it is more often used than Gantt charts to manage projects such as information systems development where variability in the duration of activities is the norm. Network diagrams are composed of circles or rectangles representing activities and connecting arrows showing required work flows, as illustrated in figure 1.19.

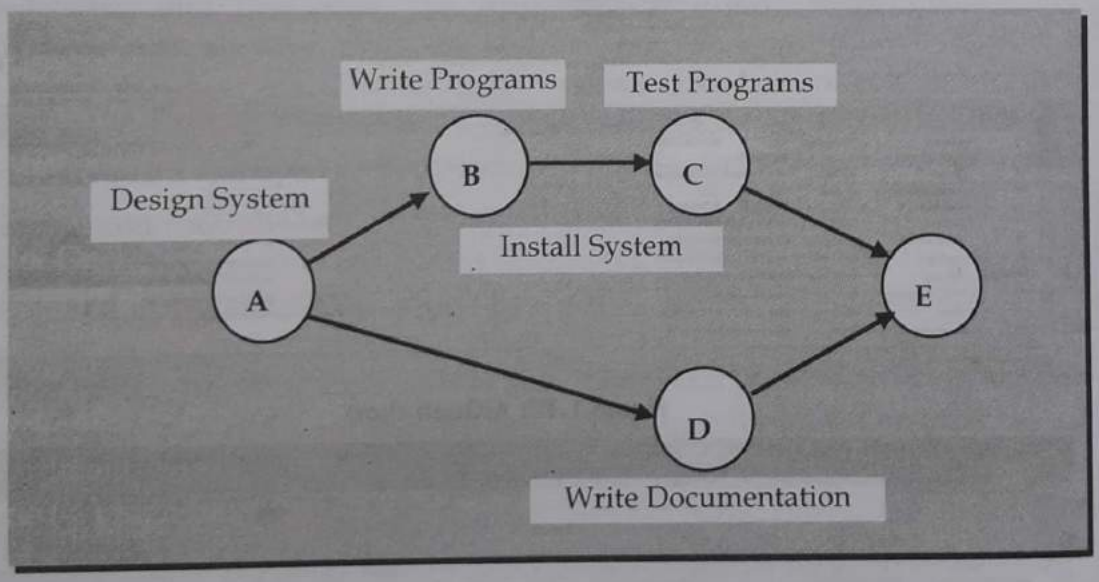


Figure 1.19: A network diagram showing activities (represented by circles) and sequence of those activities (represented by arrows)

Calculating Expected Time Durations Using PERT

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 a high degree of complexity and uncertainty characterize a task. **PERT (program evaluation review technique)** is a technique that uses optimistic, pessimistic, and realistic time estimates to calculate the expected time for a particular task. This technique helps you obtain a better time estimate when you are uncertain 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 time (r), or most likely time, reflects the project manager's "best guess" of the amount of time the activity will require for completion. Once each of these estimates is made for an activity, an expected completion time (ET) can be calculated for that activity. Because the expected completion time should be closer to the realistic time (r), the realistic time 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:

$$ET = \frac{0 + 4r + p}{6}$$

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

For example, suppose that the instructor asked to calculate an expected time for the completion of an upcoming programming assignment. For this assignment, an optimistic time of 2 hours, a pessimistic time of 8 hours, and a most likely time of 6 hours is estimated. Using PERT, the expected time for completing this assignment is 5.67 hours. Commercial project management software such as Microsoft Project assists you in using PERT to make expected time calculations. Additionally, many commercial tools allow project manager to customize the weighing of optimistic, pessimistic, and realistic completion times.

USING PROJECT MANAGEMENT SOFTWARE

A wide variety of automated project management tools is available to help to 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. Yet a lot can be done with systems such as Microsoft Project as well as public domain and shareware systems. For example, numerous shareware project management programs (e.g., OpenProj, Bugzilla, and eGroupWare) can be downloaded from the websites.

Most programs offer features such as PERT/CPM, Gantt charts, resource scheduling, project calendars, and cost tracking. As shown in figure 1.20, Microsoft Project is a full-featured program that holds the dominant share of the market. It is available as a software product for Windows and as an online service as part of Microsoft's Office 365 (Link for online tutorial for Microsoft Project 2013 is available at: https://www.tutorialspoint.com/ms_project/index.htm).

Irrespective of which project management tool used, a step-by-step process is followed to develop a WBS, work with task patterns, and analyze the critical path. The main difference is that the software does most of the work automatically, which enables much more effective management.

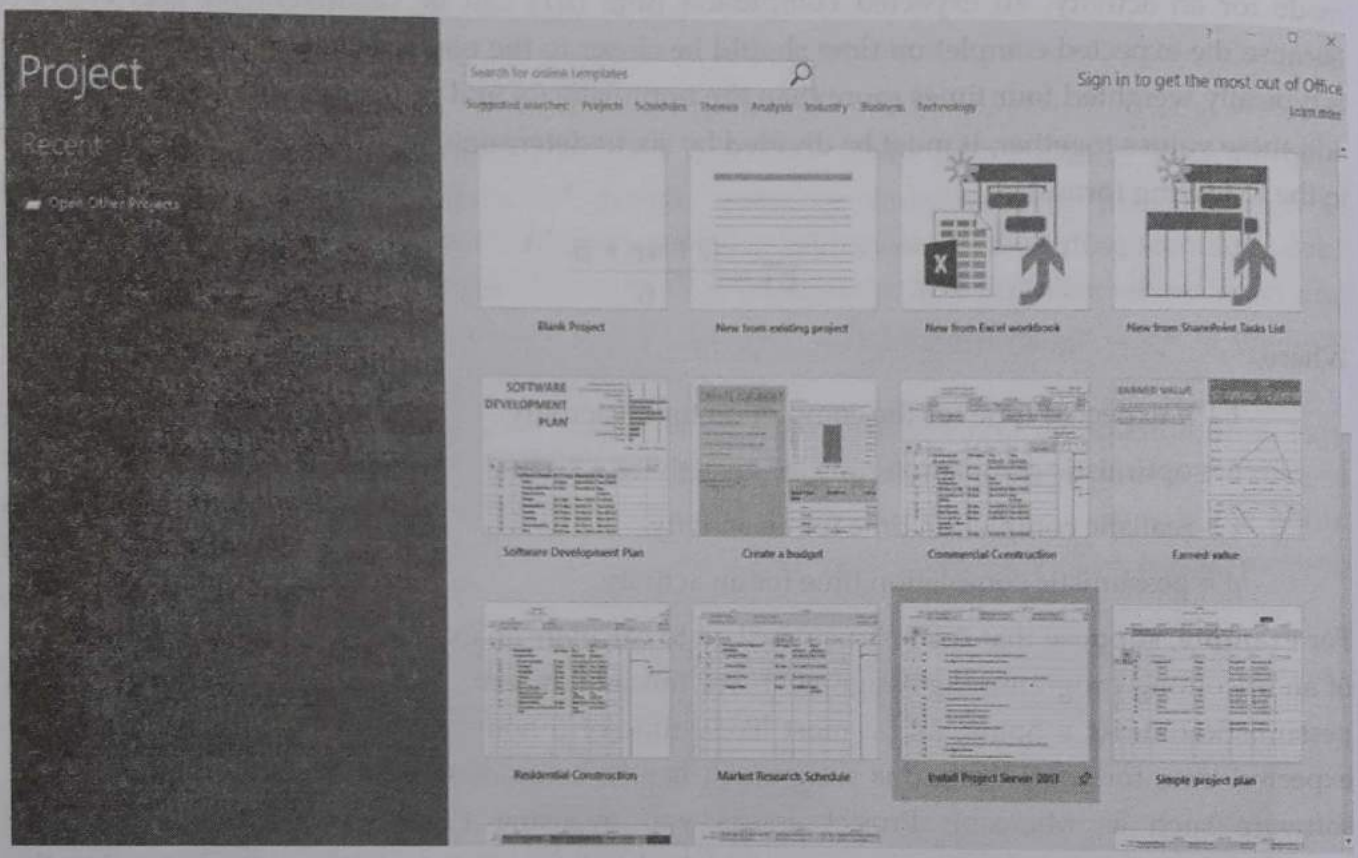


Figure 1.20: Microsoft Project

THE ORIGINS OF SOFTWARE

Even though today's systems analyst has dozens of programming languages and development tools to work with, you could easily argue that systems development is even more difficult now than it was 60 years ago. Then, as well as even more recently, certain issues were decided for you; 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 many of you reading this book will end up working for 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. Instead, the focus will be on where to obtain the many pieces and components that you will combine into the application system you have been asked to create. You and your peers will still write code, mainly to make all the different pieces work together, but more and more of your application software will be written by someone else. Even though you will not write the code, you will still use the basic structure and processes of the systems analysis and design life cycle to build the application systems your organization demands.

SYSTEMS ACQUISITION

Despite of debate about first administrative information system, there is general agreement that in the United States, the first administrative information system was General Electric's (GE) payroll system, which was developed in 1954. Since GE's payroll system was built, in-house development has become a progressively smaller piece of all the 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. Instead, they invest in packaged software, open-source software, and outsourced services. Organizations today have many choices when seeking an information system.

Outsourcing

The practice of turning over responsibility for some or all of an organization's information systems applications and operations to an outside firm is called **outsourcing**. It is important to distinguish between outsourcing and offshoring, which is a form of outsourcing. In recent years, the term "outsourcing" has become associated with organizations that employ overseas firms to perform the bulk of the work required by software development projects (offshoring), generally for cost-reduction reasons. Yet outsourcing (to domestic companies) has been used for many years, long before offshoring was commonplace. Another definition of outsourcing is "The process of retaining resources external to the procuring organization to conduct software development and related activities". The reasons for this definition are as follows:

- It places no restrictions on the size or number of resources external to the procuring organization. The resources could be another company or an individual consultant.
- It places no restrictions on the location of the external resources in relation to the procuring organization; in other words, the resource can be located a long distance from the procuring organization or down the street.
- It places no restrictions on the number of organizations involved. This means that the procuring organization may hire a number of different companies or consultants, working either in a parallel or in a serial fashion. In other words, one contractor develops the system, and another contractor deploys and maintains the system after it is in production.
- It does not preclude the procuring organization from performing some of the work itself and outsourcing only a portion of a project to an external organization.

Notice that nothing in the definition specifically mentions overseas resources or offshoring. The practice of outsourcing to overseas companies for software development tasks is a fairly recent phenomenon. More traditional forms of outsourcing software development projects have been taking place for many years. In particular, the U.S. government has been outsourcing projects to domestic companies for dozens of years. That said, offshoring is increasing in popularity.

Advantages of Outsourcing

- Reduced operating expenses
- Flexibility
- Exposure to new talent

- Focusing on core business processes
- Downsize the business by using outsourcing services
- Managing resources
- Time-saving

Disadvantages of Outsourcing

- Lowered Quality of Service
- Miscommunication
- Risk

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
- 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, and a suitable off-the-shelf system is not available, 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. Note in table 1.3 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.

Table 1.3: Leading software firms and their development specializations

Specialization	Example Firms or Websites
IT Services	Accenture, Computer Science Corporation (CSC), IBM, HP
Packaged Software Providers	Intuit, Microsoft, Oracle, SAP AG, Symantec
Enterprise Software Solutions	Oracle, SAP AG
Cloud Computing	Amazon.com, Google, IBM, Microsoft, Salesforce.com
Open Source	SourceForge.net

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. Also listed in table 1.3, 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**. Microsoft's Word as illustrated in figure 1.21 and Intuit's Quicken, QuickPay, and QuickBooks are popular examples of such software. The packaged software development industry serves many market segments. Software companies develop software to run on many different computer platforms, from microcomputers to large mainframes. The companies range in size from just a few people to thousands of employees.

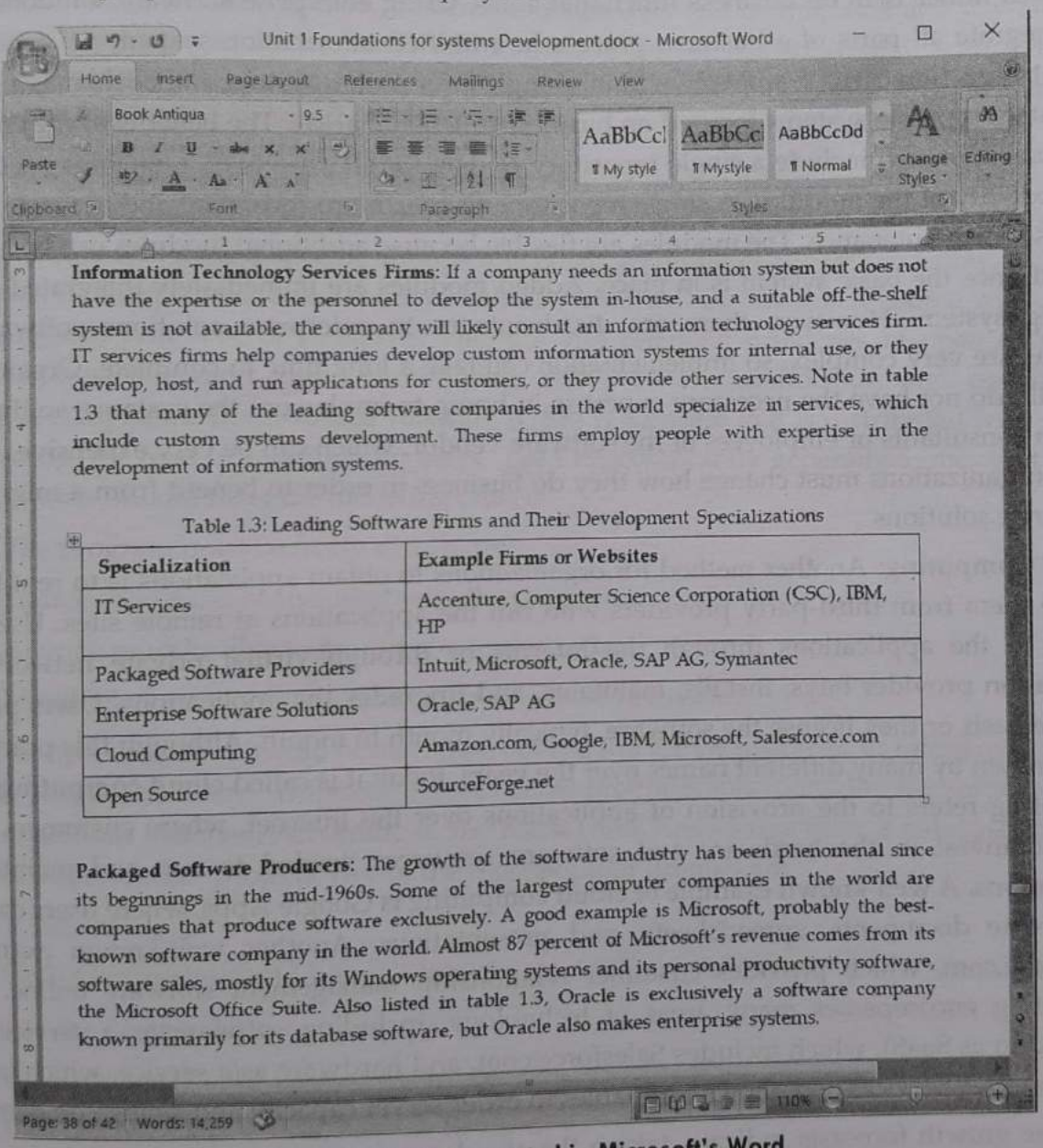


Figure 1.21: A file created in Microsoft's Word

Software companies consult with system users after the initial software design has been completed and an early version of the system has been built. The systems are then tested in actual organizations to determine whether there are any problems or if any improvements can be made. Until testing is completed, the system is not offered for sale to the public. Some off-the-shelf software systems cannot be modified to meet the specific, individual needs of a

particular organization. Such application systems are sometimes called **turnkey systems**. The producer of a turnkey system will make changes to the software only when a substantial number of users ask for a specific change.

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. Using enterprise software solutions, a firm can integrate all parts of a business process in a unified information system. All aspects of a single transaction occur seamlessly within a single information system, rather than as a series of disjointed, separate systems focused on business functional areas. 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 Salesforce.com, which provides customer relationship management software online. Cloud computing encompasses many areas of technology, including software as a service (often referred to as SaaS), which includes Salesforce.com, and hardware as a service, which includes Amazon Web Services and allows companies to order server capacity and storage on demand.

As these growth forecasts indicate, taking the cloud computing route has its advantages. The top three reasons for choosing to go with cloud computing, all of which result in benefits for the company, are (1) freeing internal IT staff, (2) gaining access to applications faster than via internal development, and (3) achieving lower cost access to corporate-quality applications.

IT managers do have some concerns about cloud computing, primary concern is over security. Concerns over security are based on storing company data on machines one does not own and

that others can access. In fact, the top two reasons for not using cloud services are concerns about unauthorized access to proprietary information and unauthorized access to customer information. Another concern is reliability. Some warn that the cloud is actually a network of networks, and as such, it is vulnerable to unexpected risks due to its complexity. Still another concern is compliance with government regulations, such as the Sarbanes-Oxley Act. Experts recommend a three-step process for secure migration to the cloud. First, have the company's security experts involved early in the migration process, so that a vendor who understands the company's security and regulatory requirements can be selected. Second, set realistic security requirements. Make sure the requirements are clearly spelled out as part of the bidding process. Third, do an honest risk assessment. Determine which data will be migrated and pay attention to how it will be managed by the cloud vendor. Once migration has occurred, it is important for companies to continue to monitor their data and systems and actively work with the vendor to maintain security.

Open-Source Software: The term "**open source**" refers to something people can modify and share because its design is publicly accessible. The term originated in the context of software development to designate a specific approach to creating computer programs. Today, however, "open source" designates a broader set of values—what we call "the open source way". Open source projects, products, or initiatives embrace and celebrate principles of open exchange, collaborative participation, rapid prototyping, transparency, meritocracy, and community-oriented development. **Open-source software (OSS)** is software that is distributed with source code that may be read or modified by users. The OSS community generally agrees that open-source software should meet the following criteria:

- The program must be freely distributed
- Source code must be included with the program
- Anyone must be able to modify the source code
- Modified versions of the source code may be redistributed

As well, an open-source software license must not require the exclusion of, or interfere with, the operation of other software. Different licenses allow programmers to modify the software with various conditions attached. According to the Black Duck Knowledge Base, a database of some two million open source projects, five of the most popular licenses are:

- T License
- GNU General Public License (GPL) 2.0
- Apache License 2.0
- GNU General Public License (GPL) 3.0
- BSD License 2.0 (3-clause, New or Revised)

When you change the source code, OSS requires the inclusion of what you altered as well as your methods. The software created after code modifications may or may not be made available for free. Open-source technologies helped establish much of the internet. Furthermore, many of the programs in use every day are based on open-source technologies. Cases in point: Android OS and Apple's OS X are based on the kernel and Unix/BSD open-source technologies, respectively. Other popular open-source software are Mozilla's Firefox web browser,

Thunderbird email client, PHP scripting language, Python programming language, Apache HTTP web server. Open-source software is an alternative to proprietary software. Participating in an OSS project can be a pathway to building a career in software development, allowing programmers to hone their skills by working on the biggest software programs in the world. Facebook, Google, and LinkedIn all release OSS, so developers can share knowledge, innovate solutions, and contribute to stable, functional products.

In-House Development: 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. A study by Banker, Davis, and Slaughter found that using a code generator as the basis for in-house development was related to an increase in maintenance hours, whereas using packaged applications was associated with a decrease in maintenance effort.

Of course, in-house development need not entail development of all of the software that will constitute the total system. Hybrid solutions involving some purchased and some in-house software components are common. The choice between a package and an external supplier will be determined by your needs, not by what the supplier has to sell. Table 1.4 compares the six different software sources discussed in this section.

Table 1.4: Leading software firms and their development specializations

Producers	When to go to this type of organization for software	Internal Staffing Requirements
IT services firms	When task requires custom support and system can't be built internally or system needs to be sourced	Internal staff may be needed, depending on application
Packaged software producers	When supported task is generic	Some IS and user staff to define requirements and evaluate packages
Enterprise-wide solutions vendors	For complete systems that cross functional boundaries	Some internal staff necessary but mostly need consultants
Cloud computing	For instant access to an application; when supported task is generic	Few; frees up staff for other IT work
Open-source software	When supported task is generic but cost is an issue	Some IS and user staff to define requirements and evaluate packages
In-house developers	When resources and staff are available and system must be built from scratch	Internal staff necessary though staff size may vary

Choosing Off-the-Shelf Software

Once you have decided to purchase off-the-shelf software rather than write some or all of the software for your new system, how do you decide what to buy? Several criteria need consideration, and special ones may arise with each potential software purchase. For each standard, an explicit comparison should be made between the software package and the process of developing the same application in-house. The most common criteria, highlighted in are as follows:

- Cost
- Functionality
- Vendor support
- Viability of vendor
- Flexibility
- Documentation
- Response time
- Ease of installation

The relative importance of these standards will vary from project to project and from organization to organization. If you had to choose two criteria that would always be among the most important, those two would probably be vendor support and vendor viability. You don't want to license software from a vendor that has a reputation for poor support. Similarly, you don't want to get involved with a vendor that might not be in business tomorrow. How you rank the importance of the remaining criteria depends primarily on your specific situation.

Cost involves comparing the cost of developing the same system in-house to the cost of purchasing or licensing the software package. Costs for purchasing and developing in-house can be compared based on the economic feasibility measures. **Functionality** refers to the tasks the software can perform and the mandatory, essential, and desired system features. Can the software package perform all, or just some of the tasks your users need? If some, can it perform the necessary core tasks? Note that meeting user requirements occurs at the end of the analysis phase because you cannot evaluate packaged software until user requirements have been gathered and structured. Purchasing application software is not a substitute for conducting the systems analysis phase.

As we said earlier, **vendor support** refers to whether the vendor can provide support, and how much. Support includes assistance to install the software, to train user and systems staff on the software, and to provide help as problems arise after installation. **Flexibility** refers to how easy it is for you, or the vendor, to customize the software. If the software is not sufficiently flexible, your users may have to adapt the way they work to fit the software. Are they likely to adapt in this manner? Purchased software can be modified in several ways. Sometimes, the vendor will make custom changes for you if you are willing to pay for the redesign and programming. Some vendors design the software for customization. For example, the software may include several different ways of processing data and, at installation time, the customer chooses which to initiate.

Documentation includes the user's manual as well as technical documentation. How understandable and up to date is the documentation? What is the cost for multiple copies, if required? **Response time** refers to how long it takes the software package to respond to the user's requests in an interactive session. Another measure of time would be how long it takes the software to complete running a job. Finally, **ease of installation** is a measure of the difficulty of loading the software and making it operational.

Validating Purchased Software Information

One way to get all of the information you want about a software package is to collect it from the vendor. Some of this information may be contained in the software documentation and technical marketing literature. Other information can be provided upon request. For example, you can send prospective vendors a questionnaire asking specific questions about their packages. This questionnaire may be part of a **request for proposal (RFP)** or request for quote (RFQ) process your organization requires when major purchases are made. If you decide that new hardware or system software is a strong possibility, you may want to issue a request for proposal (RFP) to vendors. The RFP will ask the vendors to propose hardware and system software that will meet the requirements of your new system. Issuing an RFP gives you the opportunity to have vendors conduct the research you need in order to decide among various options.

Of course, actually using the software yourself and running it through a series of tests based on the criteria for selecting software may provide the best route for evaluation. Remember to test not only the software, but also the documentation, the training materials, and even the technical support facilities. One requirement on prospective software vendors as part of the bidding process is that they install demo version of their software for a limited amount of time on corresponding system.

One of the most reliable and insightful sources of feedback is other users of the software. Vendors will usually provide a list of customers. Here is where your personal network of contacts, developed through professional groups, college friends, trade associations, or local business clubs, can be a resource; do not hesitate to find some contacts on your own. Such current or former customers can provide a depth of insight on the use of a package at their organizations. To gain a range of opinions about possible packages, you can use independent software testing services that periodically evaluate software and collect user opinions. Such surveys are available for a fee either as subscription services or on demand. Occasionally, unbiased surveys appear in trade publications. Often, however, articles in trade publications, even software reviews, are actually seeded by the software manufacturer and are not unbiased. If you are comparing several software packages, you can assign scores for each package on each criterion and compare the scores using the quantitative method for comparing alternative system design strategies.

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 tends to result in higher-quality software with lower defect rates, decreasing maintenance costs. Some of the components that can be reuse are **Source code, Design and interfaces, User manuals, Software Documentation, Software requirement specifications,** and **many more. Commercial-off-the-shelf** is ready-made software and components are ready-made

components that can be reused for new software. **Reuse software engineering** is based on guidelines and principles for reusing the existing software. Stages of reuse-oriented software engineering are shown in figure 1.22 and are explained below:

- **Requirement specification:** First of all, specify the requirements. This will help to decide that we have some existing software components for the development of software or not.
- **Component analysis:** Helps to decide that which component can be reused where.
- **Requirement updations / modifications:** If the requirements are changed by the customer, then still existing components are helpful for reuse or not.
- **Reuse System design:** If the requirements are changed by the customer, then still existing system designs are helpful for reuse or not.
- **Development:** Existing components are matching with new software or not.
- **Integration:** Can we integrate the new systems with existing components?
- **System Validation:** To validate the system that it can be accepted by the customer or not.

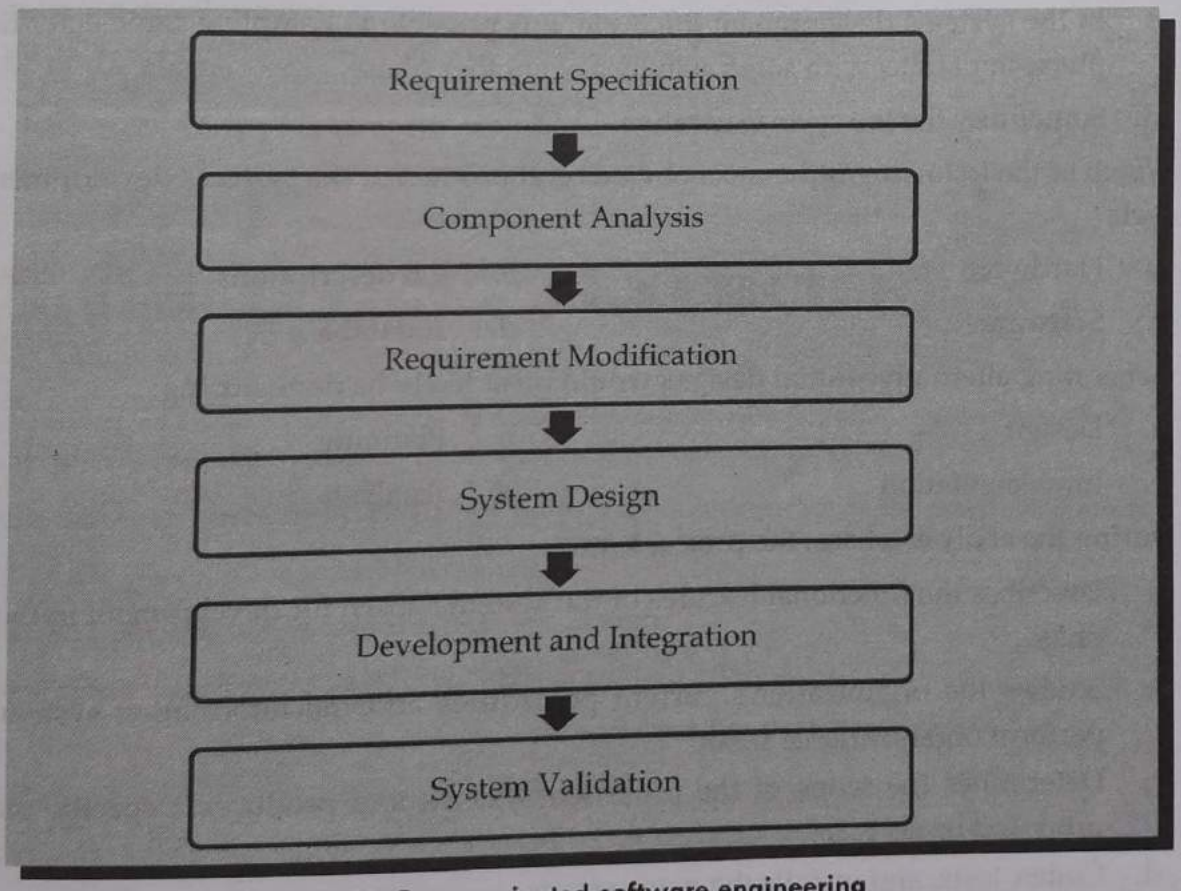


Figure 1.22: Reuse-oriented software engineering

Advantages of reuse

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

Chapter 2

PLANNING

CHAPTER OUTLINE

After studying this chapter, students will be able to understand the:

- System Development Projects: Identification and Selection
- System Development Projects: Initiation and Planning

IDENTIFYING AND SELECTING SYSTEMS DEVELOPMENT PROJECTS

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. Nonintegrated 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.

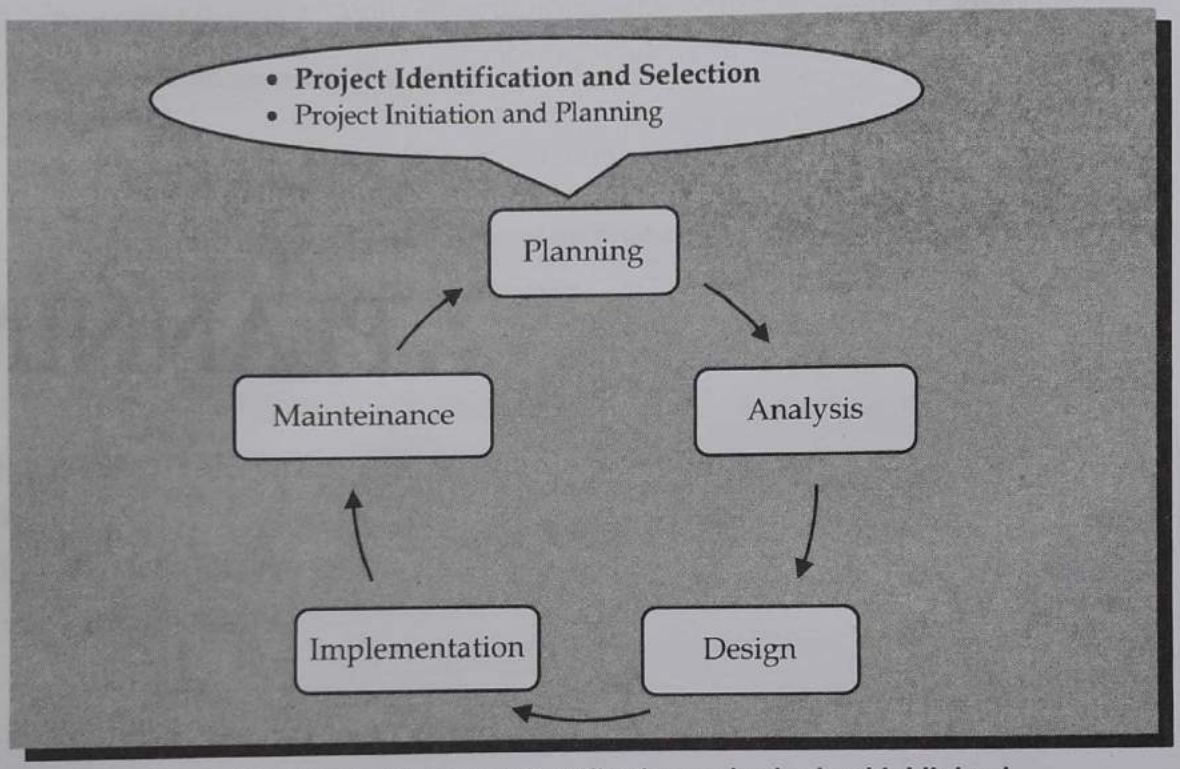


Figure 2.1: SDLC with project identification and selection highlighted

The first phase of the SDLC is planning, consisting of project identification and selection, and project initiation and planning as illustrated in figure 2.1. During project identification and selection, a senior manager, a business group, an IS manager, or a steering committee identifies and assesses all possible systems development projects that an organization unit could undertake. Next, those projects deemed most likely to yield significant organizational benefits, given available resources, are selected for subsequent development activities. Organizations vary in their approach to identifying and selecting projects. In some organizations, project identification and selection is a very formal process in which projects are outcomes of a larger overall planning process.

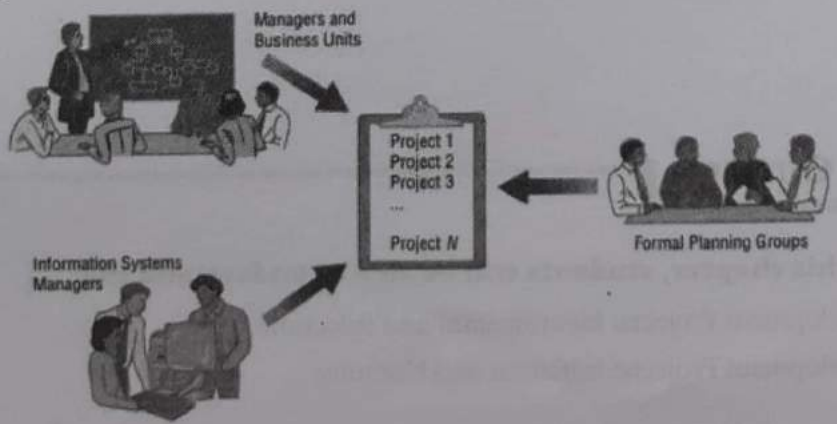


Figure 2.2: Three key source for information systems projects

Information systems development requests come from a variety of sources which is illustrated in figure 2.2. One source is requests by managers and business units for replacing or extending an existing system to gain needed information or to provide a new service to customers. Another source for requests is IS managers who want to make a system more efficient and less costly to operate, or want to move it to a new operating environment. A final source of projects is a formal planning group that identifies projects for improvement to help the organization meet its corporate objectives (e.g., a new system to provide better customer service). Regardless of how a given organization actually executes the project identification and selection process, a common sequence of activities occurs. In the following sections, we describe a general process for identifying and selecting projects and producing the deliverables and outcomes of this process.

Process of Identifying and Selecting IS Development Projects

Project identification and selection consists of three primary activities as illustrated in figure 2.3:

1. **Identifying potential development projects**
2. **Classifying and ranking projects**
3. **Selecting projects for development**

Each of these activities is described below.

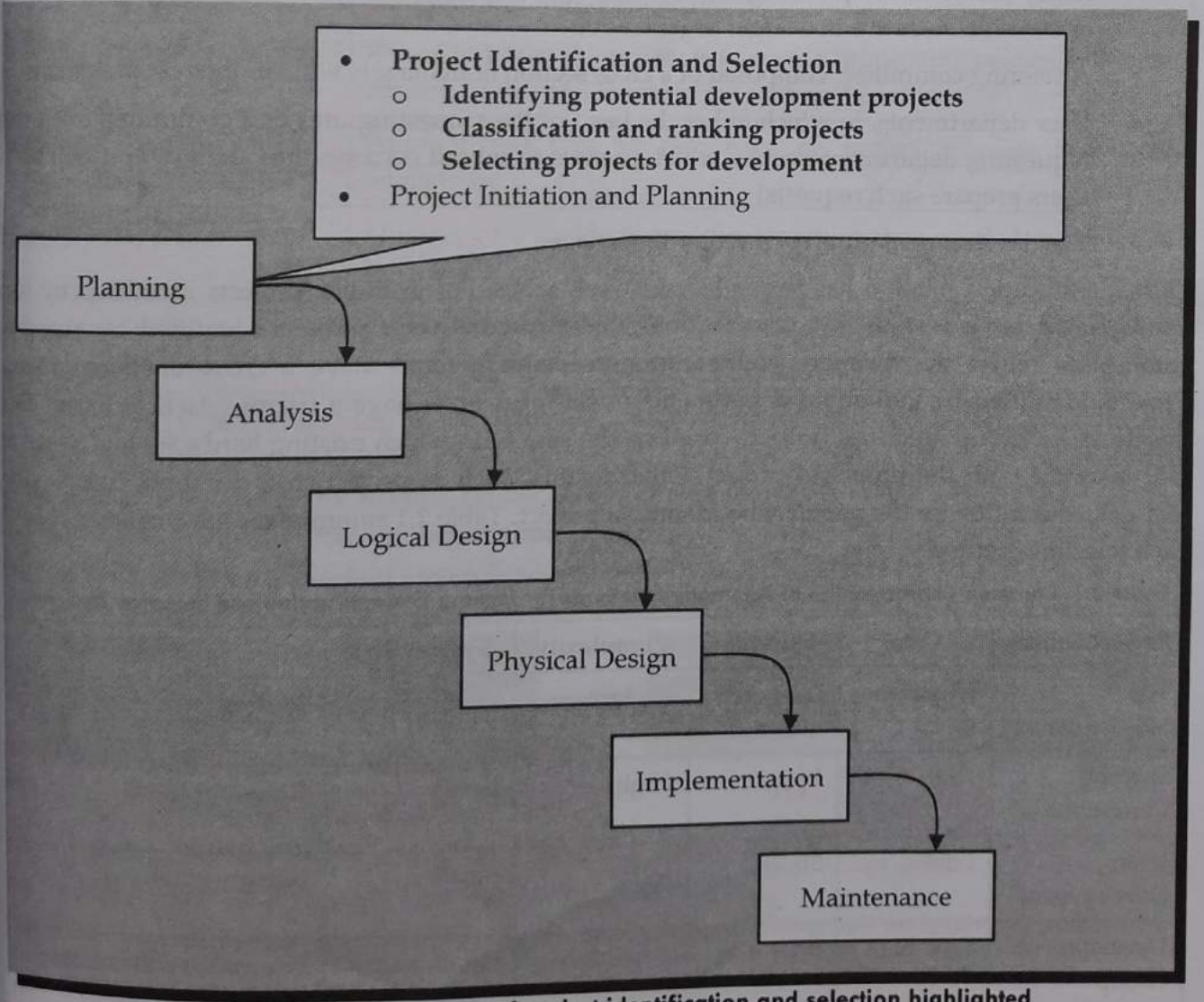


Figure 2.3: SDLC with process of project identification and selection highlighted

1. **Identifying Potential Development Projects**

Organizations vary as to how they identify projects.

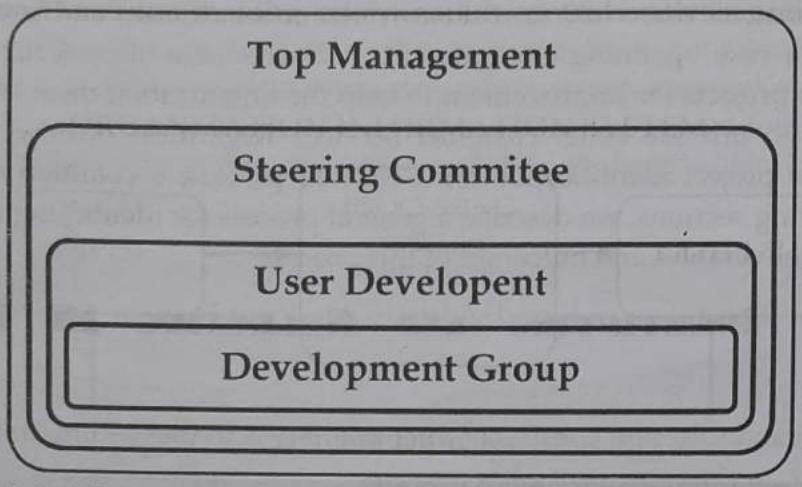


Figure 2.4: Selection Methods for Identifying Potential Development Projects

This process can be performed by:

- A key member of top management, either the CEO of a small or medium-size 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 (as a systems analyst, you will help users prepare such requests)
- The development group or a senior IS manager

Each identification method has strengths and weaknesses. For example, projects identified by top management have a strategic organizational focus. Alternatively, projects identified by steering committees reflect the diversity of the committee and therefore have a cross-functional focus. Projects identified by individual departments or business units have a narrow, tactical focus. The development group identifies projects based on the ease with which existing hardware and systems will integrate with the proposed project. Other factors, such as project cost, duration, complexity, and risk, also influence the people who identify a project. Table 2.1 summarizes the characteristics of each selection method.

Table 2.1: Common Characteristics of Alternative Methods for Making IS Identification and Selection Decisions

Project Source	Cost	Duration	Complexity	System Size	Focus
Top Management	Highest	Longest	Highest	Largest	Strategic
Steering Committee	High	Long	High	Large	Cross-functional
User Development	Low	Short	Low	Small	Departmental
Development Group	Low-high	Shot-long	Low-high	Small-large	Integration with existing systems

2. Classifying and Ranking Projects

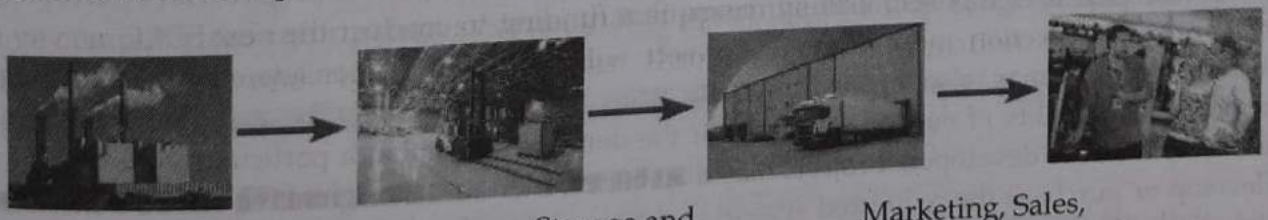
Assessing the merit of potential projects is the second major activity in the project identification and selection phase. As with project identification, classifying and ranking projects can be performed by top managers, a steering committee, business units, or the IS development group. The criteria used to assign the merit of a given project can vary based on the size of the organization. Table 2.2 summarizes the criteria commonly used to evaluate projects. In any given organization, one or several criteria might be used during the classifying and ranking process.

As with project identification, the criteria used to evaluate projects will vary by organization. If, for example, an organization uses a steering committee, it may choose to meet monthly or quarterly to review projects and use a wide variety of evaluation criteria. At these meetings, new project requests are reviewed relative to projects already identified, and ongoing projects are monitored. The relative ratings of projects are used to guide the final activity of this identification process – project selection.

Table 2.2: Possible evaluation criteria when classifying and ranking projects

Evaluation Criteria	Description
Value Chain Analysis	Extent to which activities add value and costs when developing products and/or services
Strategic Alignment	Extent to which the project is viewed as helping the organization activities and long-term goals
Potential Benefits	Extent to which the project is viewed as improving profits, customer service, ... etc., and the duration of these benefits
Resource Availability	Amount and type of resources the project requires and their availability
Project Size/Duration	Number of individuals and the length of time needed to complete the project
Technical Difficulty/Risks	Level of technical difficulty to successfully complete the project within given time and resource constraints

An important project evaluation method that is widely used for assessing information systems development projects is called **value chain analysis**. Value chain analysis is the process of analyzing an organization's activities for making products and/or services to determine where value is added and costs are incurred. Once an organization gains a clear understanding of its value chain, improvements in the organization's operations and performance can be achieved.



Transform Raw Materials into Products

Storage and Distribution of Products

Marketing, Sales, and Customers Support

Figure 2.5: Organizations can be thought of as a value chain, transforming raw materials into products for customers

3. Selecting IS Development Projects.

The selection of projects is the final activity in the project identification and selection phase. The short- and long-term projects most likely to achieve business objectives are considered. As business conditions change over time, the relative importance of any single project may substantially change. Thus, the identification and selection of projects is an important and ongoing activity.

Numerous factors must be considered when selecting a project, as illustrated in figure 2.6. These factors include:

- Perceived needs of the organization
- Existing systems and ongoing projects
- Resource availability
- Evaluation criteria
- Current business conditions
- Perspectives of the decision makers

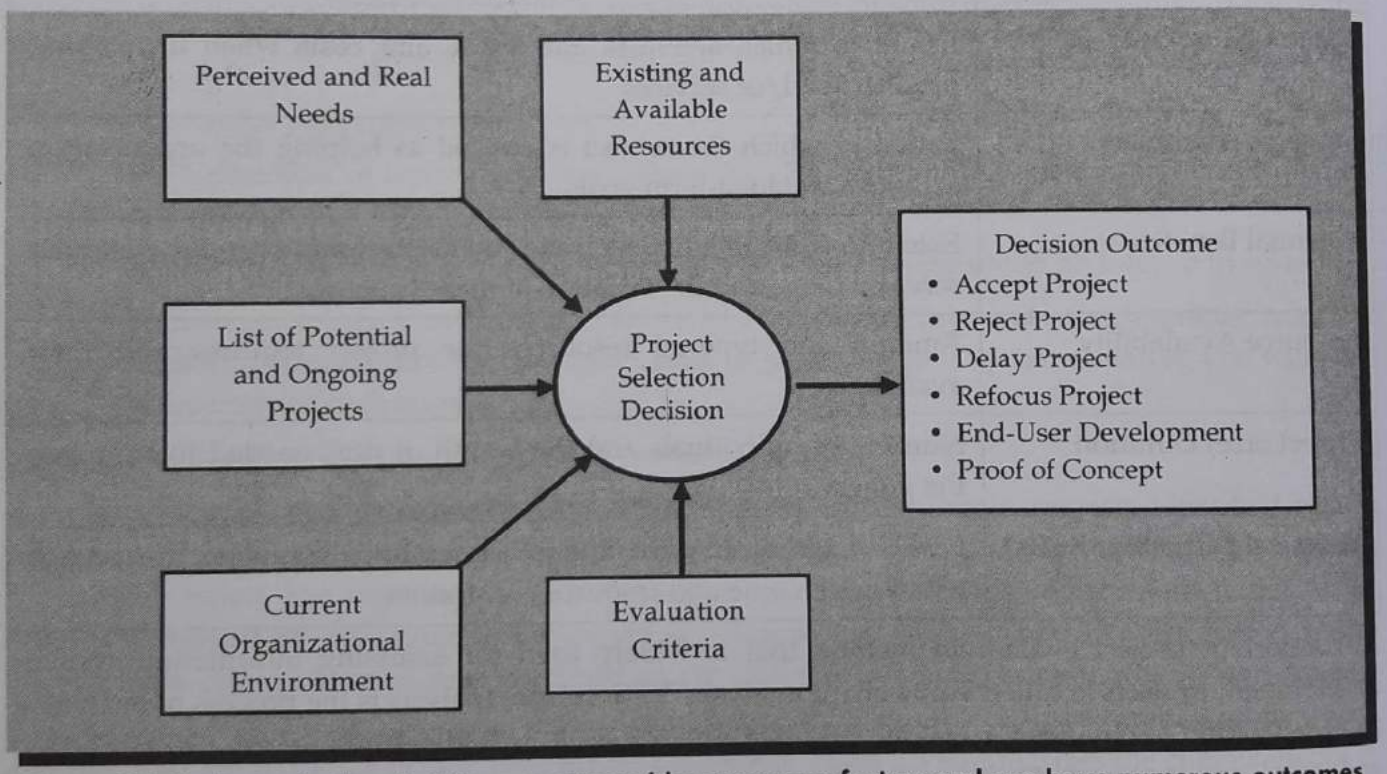


Figure 2.6: Project selection decisions must consider numerous factors and can have numerous outcomes

This decision-making process can lead to numerous outcomes. Of course, projects can be accepted or rejected. Acceptance of a project usually means that funding to conduct the next SDLC activity has been approved. Rejection means that the project will no longer be considered for development. However, projects may also be conditionally accepted; projects 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 themselves. Finally, the requesters of a project may be asked to modify and resubmit their request after making suggested changes or clarifications.

Deliverables and Outcomes

The primary deliverable, or end product, from the project identification and selection phase is a schedule of specific IS development projects. These projects come from both top-down and bottom-up sources, and once selected they move into the second activity within this SDLC phase—project

initiation and planning. This sequence of events is illustrated in figure 2.7. An outcome of this activity is the assurance that people in the organization gave careful consideration to project selection and clearly understood how each project could help the organization reach its objectives. Because of the principle of incremental commitment, a selected project does not necessarily result in a working system. **Incremental commitment** is a strategy in system analysis and design in which the project is reviewed after each phase and continuation of the project is rejustified in each of these reviews. This reassessment will determine whether the business conditions have changed or whether a more detailed understanding of a system's costs, benefits, and risks would suggest that the project is not as worthy as previously thought.

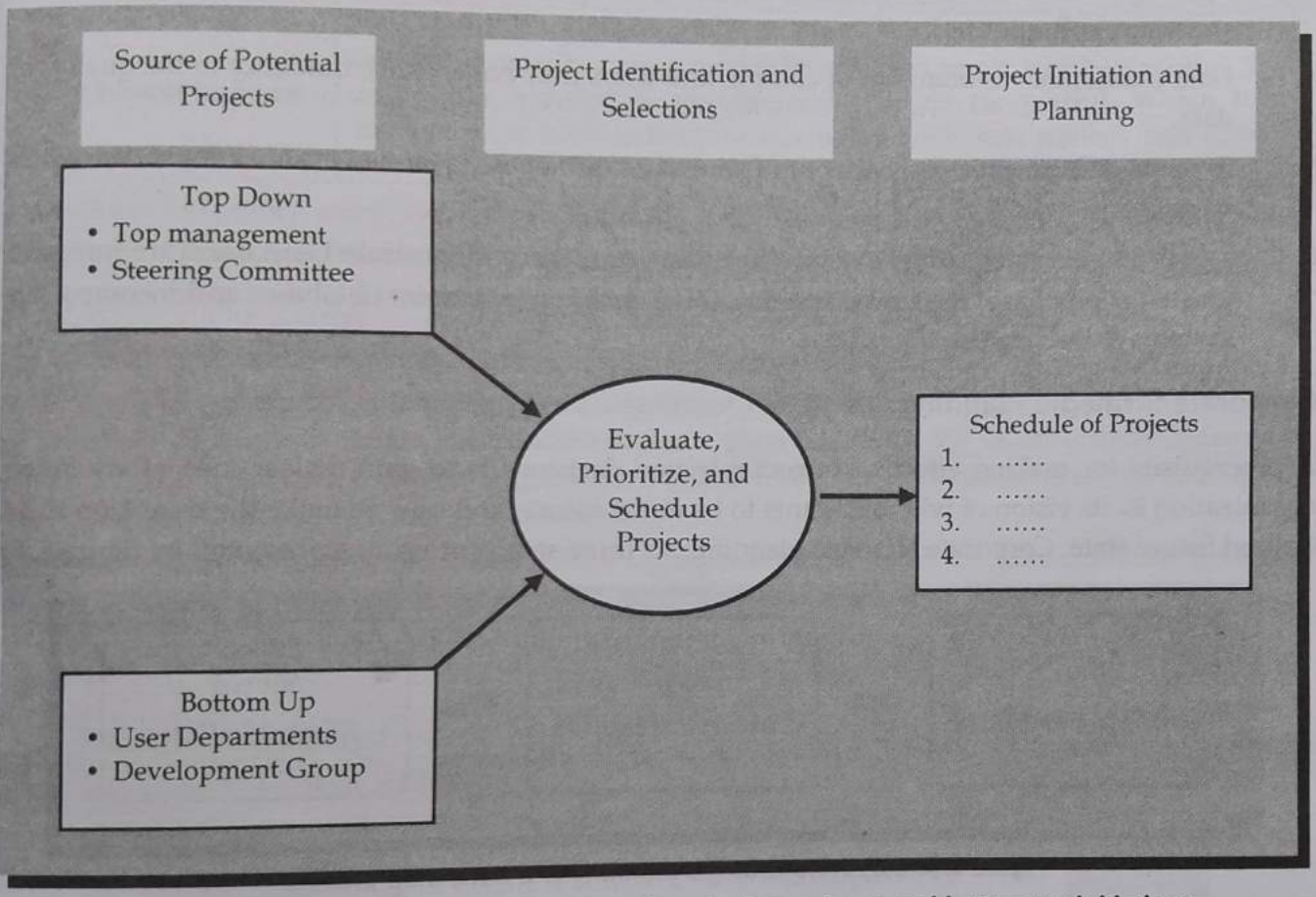


Figure 2.7: IS development projects come from both top-down and bottom-up initiatives

Many Organizations have found in order to make good project selection decisions and provide sound guidance as issue arise in your work as a systems analyst on a project, a clear understanding of overall organizational business strategy and objectives is required. This means that a clear understanding of the business and the desired role of information systems in achieving organizational goals is a precondition to improving the identification and selection process.

Corporate and Information Systems Planning

Although there are numerous motivations for carefully planning the identification and selection of projects, organizations have not traditionally used 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.

The need for improved information systems projects identification and selection is readily apparent when we consider factors such as:

1. The cost of information systems has risen steadily and approaches 40 percent of total expenses in some organizations.
2. Many systems cannot handle applications that cross organizational boundaries.
3. Many systems often do not address the critical problems of the business as a whole or support strategic applications.
4. Data redundancy is often out of control, and users may have little confidence in the quality of data.
5. Systems maintenance costs are out of control as old, poorly planned systems must constantly be revised.
6. Application backlogs often extend three years or more, and frustrated end users are forced to create (or purchase) their own systems, often creating redundant databases and incompatible systems in the process.

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. Corporate strategic planning is a three-step process as represented by figure 2.8.

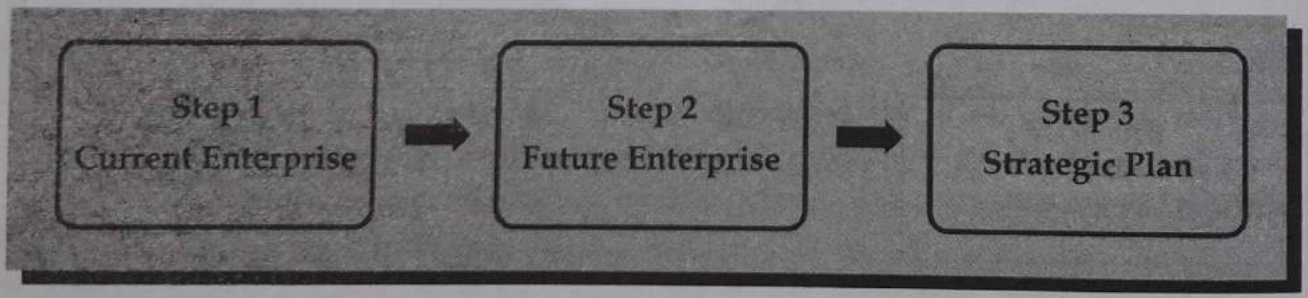


Figure 2.8: Corporate strategic planning is a three-step process

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. **Corporate strategic planning** is an ongoing process that defines the mission, objectives, and strategies of an organization. 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. The **mission statement** of a company typically states in very simple terms what business the company is in. 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 express an organization's qualitative and quantitative goals for reaching a desired future position. A **competitive strategy** is the method by which an organization attempts to achieve its mission and objectives. Table 2.3 summarizes the generic competitive strategies.

Table 2.3: 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.
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.

To make the long story short, we say: To build the most effective information systems, it is only through the clear understanding of organizational mission, objectives, and strategies that IS development projects should be identified and selected.

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.

The three key activities of this modeling process are represented in figure 2.9. 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 it 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. These three activities parallel those of corporate strategic planning, and this relationship is shown in figure 2.10.

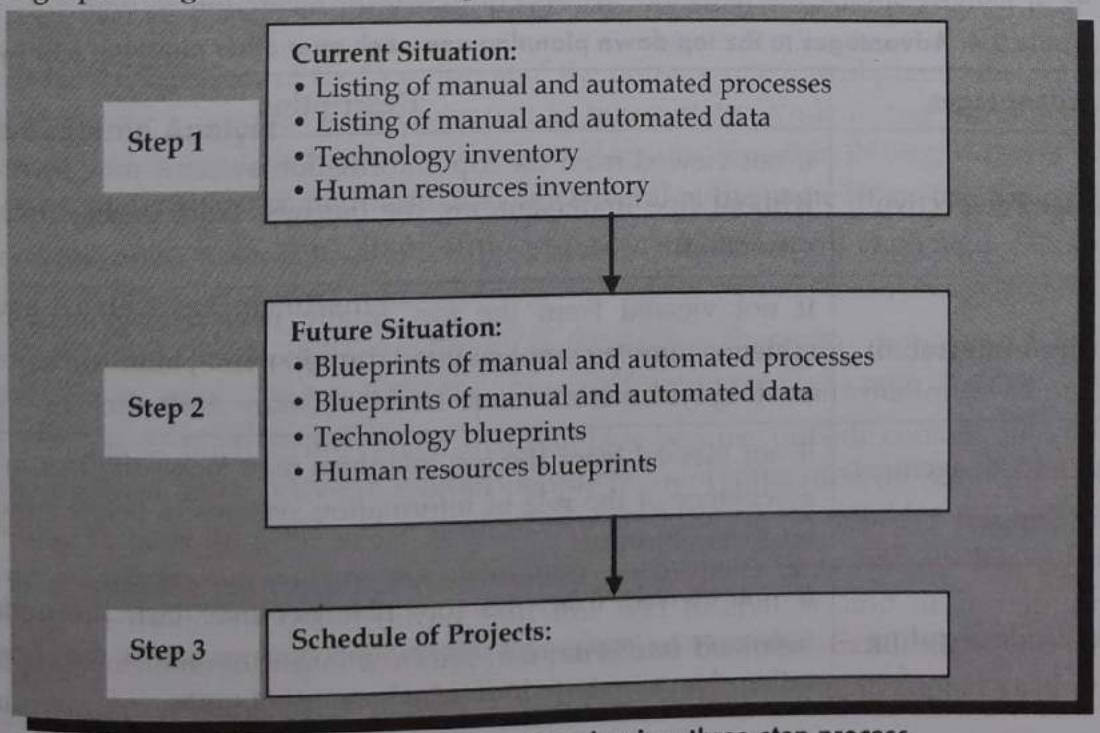


Figure 2.9: Information systems planning is a three-step process

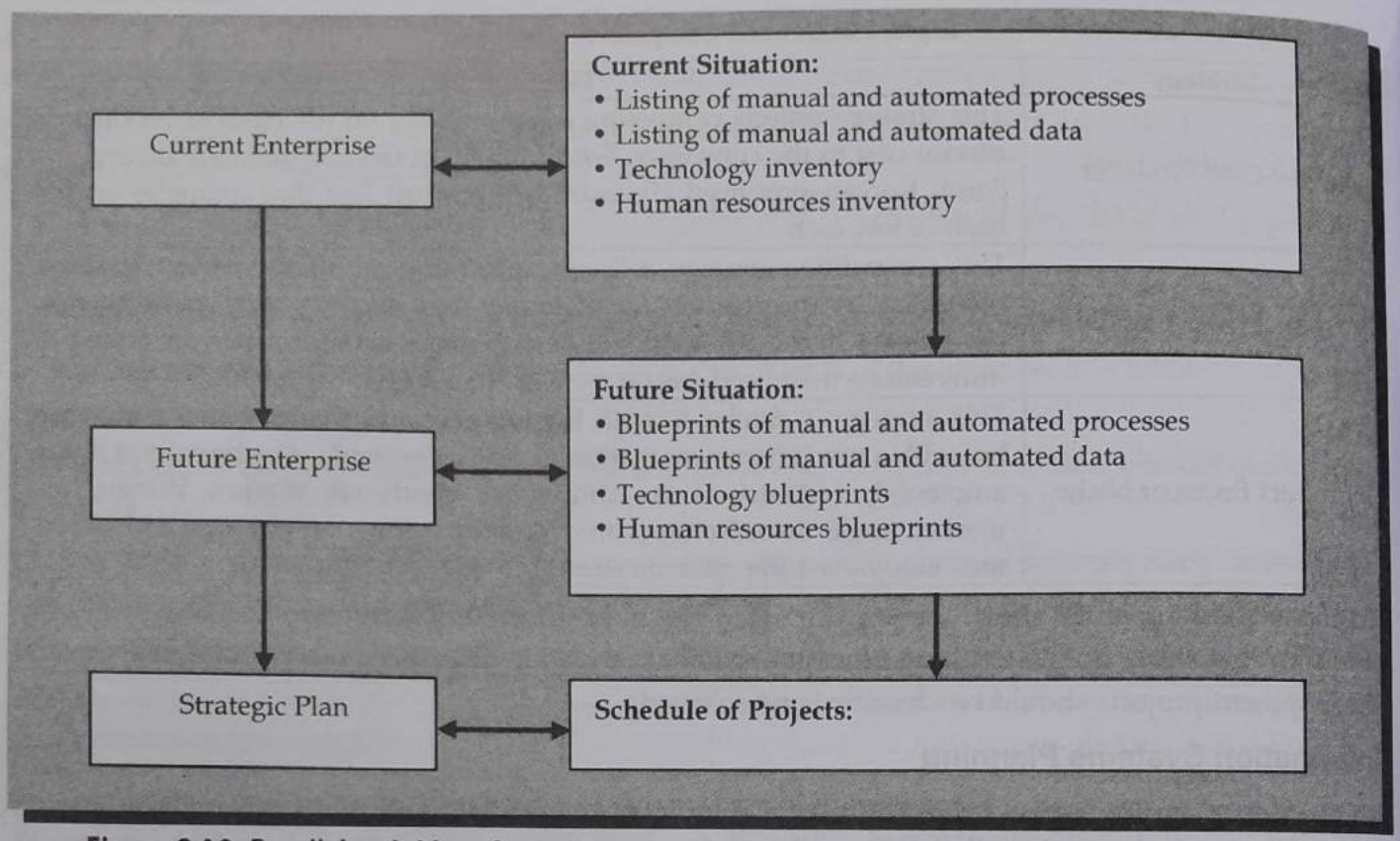


Figure 2.10: Parallel activities of corporate strategic planning and information systems planning

Numerous methodologies such as Business Systems Planning (BSP) and Information Engineering (IE) have been developed to support the ISP process; most contain the following three key activities:

1. Describing Current Situation.

The most widely used approach for describing the current organizational situation is generically referred to as top-down planning. **Top-down planning** is a generic ISP methodology that attempts to gain a broad understanding of the informational needs of the entire organization. The top-down approach to ISP has several advantages over other planning approaches, which are summarized in table 2.4.

Table 2.4: Advantages to the top-down planning approach over other planning approaches

Advantages	Description
Broader Perspective	If not viewed from the top, information systems may be implemented without first understanding the business from general management's viewpoint.
Improved Integration	If not viewed from the top, totally new management information systems may be implemented rather than planning how to evolve existing systems.
Improved Management Support	If not viewed from the top, planners may lack sufficient management acceptance of the role of information systems in helping them achieve business objective.
Better Understanding	If not viewed from the top, planners may lack the understanding necessary to implement information systems across the entire business rather than simply to individual operating units.

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. **Bottom-up planning** is a generic information systems planning methodology that identifies and defines IS development projects based upon solving operational business problems or taking advantage of some business opportunities.

2. Describing the Target Situation, trends, and Constraints

After describing the current situation, the next step in the ISP 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 (see figure 2.9).

In summary, to create the target situation, planners must first edit their initial lists and record the desired locations, units, functions, processes, data, and information systems within the constraints and trends of the organization environment (e.g., time, resources, technological evolution, competition, and so on). Next, matrices are updated to relate information in a manner consistent with the desired future state. Planners then focus on the differences between the current and future lists and matrices to identify projects and transition strategies.

3. 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, long-range 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 components of a typical information systems plan are explained below:

- **Organizational Mission, Objectives, and Strategy:** Briefly describes the mission, objectives, and strategy of the organization. The current and future views of the company are also briefly presented (i.e., where we are, where we want to be).
- **Informational Inventory:** This section provides a summary of the various business processes, functions, data entities, and information needs of the enterprise. This inventory will view both current and future needs.
- **Mission and Objectives of IS:** Description of the primary role IS will play in the organization to transform the enterprise from its current to future state. While it may later be revised, it represents the current best estimate of the overall role for IS within the organization. This role may be as a necessary cost, an investment, or a strategic advantage, for example.
- **Constraints on IS Development:** Briefly describes limitations imposed by technology and current level of resources within the company—financial, technological, and personnel.
- **Overall Systems Needs and Long-Range IS Strategies:** Presents a summary of the overall systems needed within the company and the set of long-range (2–5 years) strategies chosen by the IS department to fill the needs.
- **The Short-Term Plan:** Shows a detailed inventory of present projects and systems and a detailed plan of projects to be developed or advanced during the current year. These projects may be the result of the long-range IS strategies or of requests from managers that have already been approved and are in some stage of the life cycle.
- **Conclusions:** Contains likely but not-yet certain events that may affect the plan, an inventory of business change elements as presently known, and a description of their estimated impact on the plan.

INITIATING AND PLANNING SYSTEMS DEVELOPMENT PROJECTS

During the first phase of the SDLC planning, two primary activities are performed as illustrated in figure 2.11. The first, project identification and selection, focuses on the activities during which the need for a new or enhanced system is recognized but does not deal with a specific project but rather identifies the portfolio of projects to be under taken by the organization. Thus, project identification and selection is often thought of as a "pre-project" step in the life cycle. Regardless of how a project is identified and selected, 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. It is crucial that organizations understand whether resources should be devoted to a project; otherwise, very expensive mistakes can be made. Project initiation and planning is where projects are accepted for development, rejected, or redirected. This is also where a systems analyst, begin to play a major role in the systems development process.

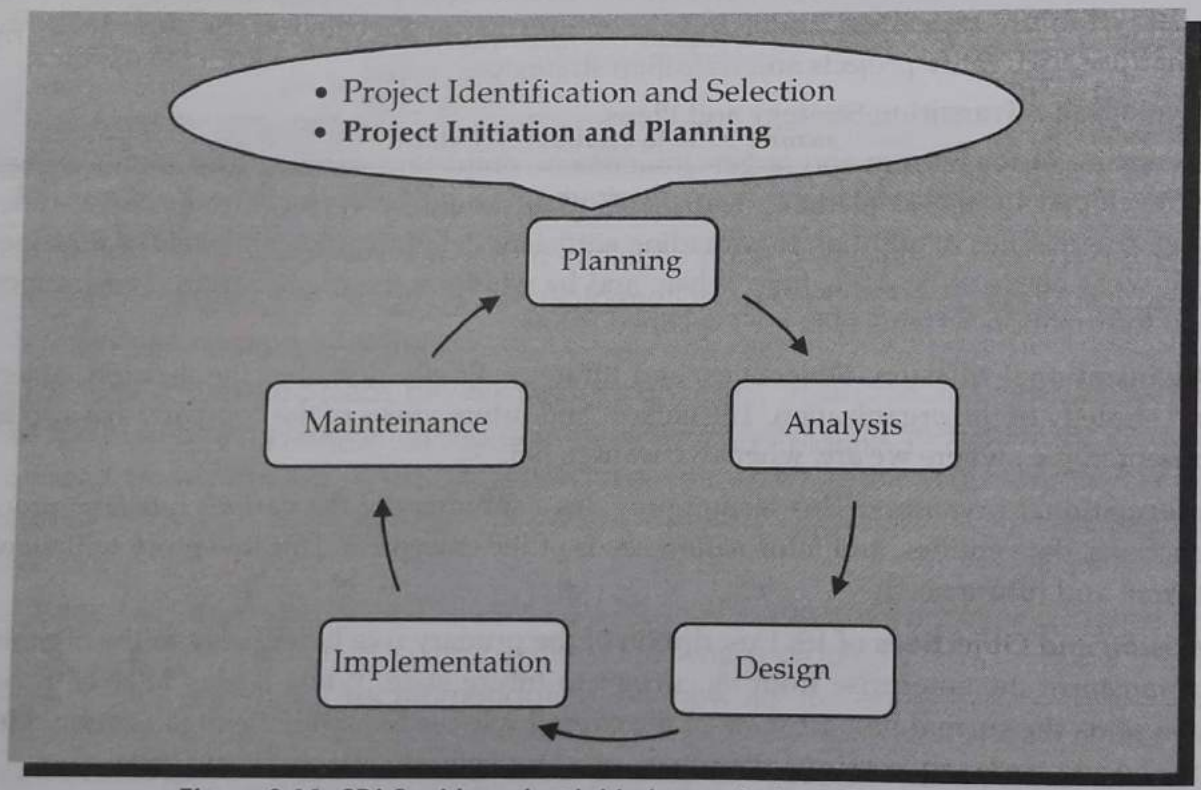


Figure 2.11: SDLC with project initiation and planning highlighted

Most organizations assign an experienced systems analyst, or team of analysts for large projects, to perform project initiation and planning. The analyst will work with the proposed customers—managers and users in a business unit—of the system and other technical development staff in preparing the final plan. Experienced analysts working with customers who well understand their information services needs should be able to perform a detailed analysis with relatively little effort. Less experienced analysts with customers who only vaguely understand their needs will likely expend more effort in order to be certain that the project scope and work plan are feasible. The objective of project initiation and planning is to transform a vague system request document into a tangible project description, as illustrated in figure 2.12. Effective communication among the systems analysts, users, and management is crucial to the creation of a meaningful project plan. Getting all parties to agree on the direction of a project may be difficult for cross-department projects when

different parties have different business objectives. Projects at large, complex organizations require systems analysts to take more time to analyze both the current and proposed systems.

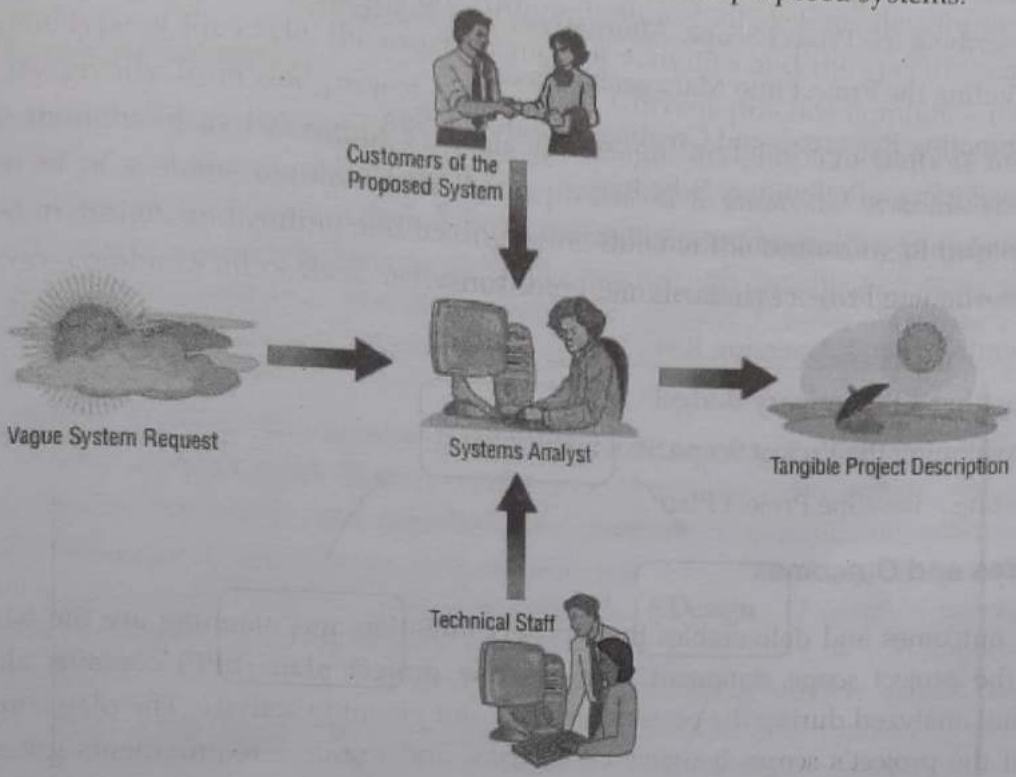


Figure 2.12: The systems analyst transforms a vague system request into a tangible project description during project initiation and planning

Process of Initiating and Planning IS Development Projects

As its name implies, two major activities occur during project initiation and project planning. Project initiation focuses on activities that will help organize a team to conduct project planning. During initiation, one or more analysts are assigned to work with a customer to establish work standards and communication procedures. Summary of six activities performed during project initiation are listed below.

- 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

The second activity, project planning, focuses on defining clear, discrete tasks and the work needed to complete each task. The objective of the project planning process is to produce two documents: a baseline project plan (BPP) and the project scope statement (PSS). The BPP becomes the foundation for the remainder of the development project. It is an internal document used by the development team but not shared with customers. The PSS, produced by the project team, clearly outlines the objectives of the project for the customer. As with the project initiation process, the size, scope, and complexity of a project dictate the comprehensiveness of the project planning process and the 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 listed below.

- 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

Deliverables and Outcomes

The major outcomes and deliverables from project initiation and planning are the baseline project plan and the project scope statement. The **baseline project plan (BPP)** contains all information collected and analyzed during the project initiation and planning activity. The plan contains the best estimate of the project's scope, benefits, costs, risks, and resource requirements given the current understanding of the project. The project selection committee uses the BPP to help decide whether to continue, redirect, or cancel a project. If selected, the BPP becomes the foundation document for all subsequent SDLC activities; however, it is updated as new information is learned during subsequent SDLC activities.

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. 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.

Assessing Project Feasibility

Most information systems projects have budgets and deadlines. Assessing project feasibility is a required task that can be a large undertaking because it requires a systems analyst, to evaluate a wide range of factors. Although the specifics of a given project will dictate which factors are most important, most feasibility factors fall into the following categories:

- Economic
- Operational
- Technical
- Schedule
- Legal and contractual
- Political

Together, the culmination of these feasibility analyses forms the business case that justifies the expenditure of resources on the project. In the remainder of this section, we will examine various feasibility issues.

Assessing Economic Feasibility

A study of economic feasibility is required for the baseline project plan. The purpose for assessing **economic feasibility** is to identify the financial benefits and costs associated with the development project. Economic feasibility is often referred to as **cost-benefit analysis.** During project initiation and planning, it will be impossible for you to define precisely all benefits and costs related to a particular project. Yet, it is important that you identify and quantify benefits and costs, or it will be impossible for you to conduct a sound economic analysis and determine whether one project is more feasible than another.

Determining Project Benefits: Information systems can provide many benefits to an organization. For example, a new or renovated IS can automate monotonous jobs, reduce errors, provide innovative services to customers and suppliers, and improve organizational efficiency, speed, flexibility, and morale. These benefits are both tangible and intangible. **Tangible benefits** refer to items that can be measured in dollars and with certainty. Examples of tangible benefits 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. For example, a tangible benefit that allows a company to perform a task 50 percentage of the time may be difficult to quantify in terms of hard dollar savings. Most tangible benefits fit in one or more of 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 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. Potential tangible benefits may have to be considered intangible during project initiation and planning because you may not be able to quantify them in dollars or with certainty at this stage in the life cycle. During later stages, such intangibles can become tangible benefits as you better understand the ramifications of the system you are designing. Intangible benefits include:

- Competitive necessity
- More timely information
- Improved organizational planning
- Increased organizational flexibility
- Promotion of organizational learning and understanding
- Availability of new, better, or more information
- Ability to investigate more alternatives
- Faster decision making
- More confidence in decision quality
- Improved processing efficiency

- Improved asset utilization
- Improved resource control
- Increased accuracy in clerical operations
- Improved work process that can improve employee morale or customer satisfaction
- Positive impacts on society
- Improved social responsibility
- Better usage of resources ("greener")

After determining project benefits, project costs must be identified.

Determining Project Costs: An information system can have both tangible and intangible costs. A **tangible cost** refers to an item that you can easily measure in dollars and with certainty. From a systems development perspective, tangible costs include items such as hardware costs, labor costs, and operational costs from employee training and building renovations. Alternatively, an intangible cost refers to an item 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** refers to the cost of owning and operating a system, including the total cost of acquisition, as well as all costs associated with its ongoing use and maintenance. Besides tangible and intangible costs, you can distinguish system-related development costs as either one-time or recurring. A **one-time cost** refers to a cost associated with project initiation and development and the start-up of the system. These costs typically encompass the following activities:

- System development
- New hardware and software purchases
- User training
- Site preparation
- Data or system conversion

When conducting an economic cost-benefit analysis, you should create a worksheet for capturing these expenses. This worksheet can be a two-column document or a multicolumn spreadsheet. For large projects, one-time costs may be staged over one or more years. In these cases, a separate one-time cost worksheet should be created for each year. This separation would make it easier to perform present-value calculations. A **recurring cost** refers to a cost resulting from the ongoing evolution and use of the system. Examples of these costs typically include:

- Application software maintenance
- Incremental data storage expense
- Incremental communications
- New software and hardware leases
- Consumable supplies and other expenses (e.g., paper, forms, datacenter personnel)

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

The Time Value of Money: Most techniques used to determine economic feasibility encompass the concept of the **time value of money (TVM)** which refers to comparing present cash outlays to future expected returns. As we've seen, 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, rather than future value when comparing investment options. 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. Some of the costs of the system will be accrued in after implementation. Before cost-benefit analysis, these costs should be brought back to the current dollars. **Present value** is the current value of a dollar at any time in the future. It is calculated using the formula:

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

Where PV_n is the present value of Y dollars n years for now when i is the discount rate. To calculate **net present value (NVP),** simply add the present values calculated previously.

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.

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

A graphical representation of this analysis is shown in figure 2.13. Using the information from the economic analysis, PVF's Systems Priority Board will be in a much better position to understand the potential economic impact of the CTS. It should be clear from this analysis that, without such information, it would be virtually impossible to know the cost–benefits of a proposed system and impossible to make an informed decision regarding approval or rejection of the service request. A system analyst can use many techniques to compute a project's economic feasibility. Because most information systems have a useful life of more than one year and will provide benefits and incur expenses for more than one year, most techniques for analyzing economic feasibility employ the concept of the TVM. Some of these cost–benefit analysis techniques are quite simple; whereas others are more sophisticated. Table 2.5 describes three commonly used techniques for conducting economic feasibility analysis.

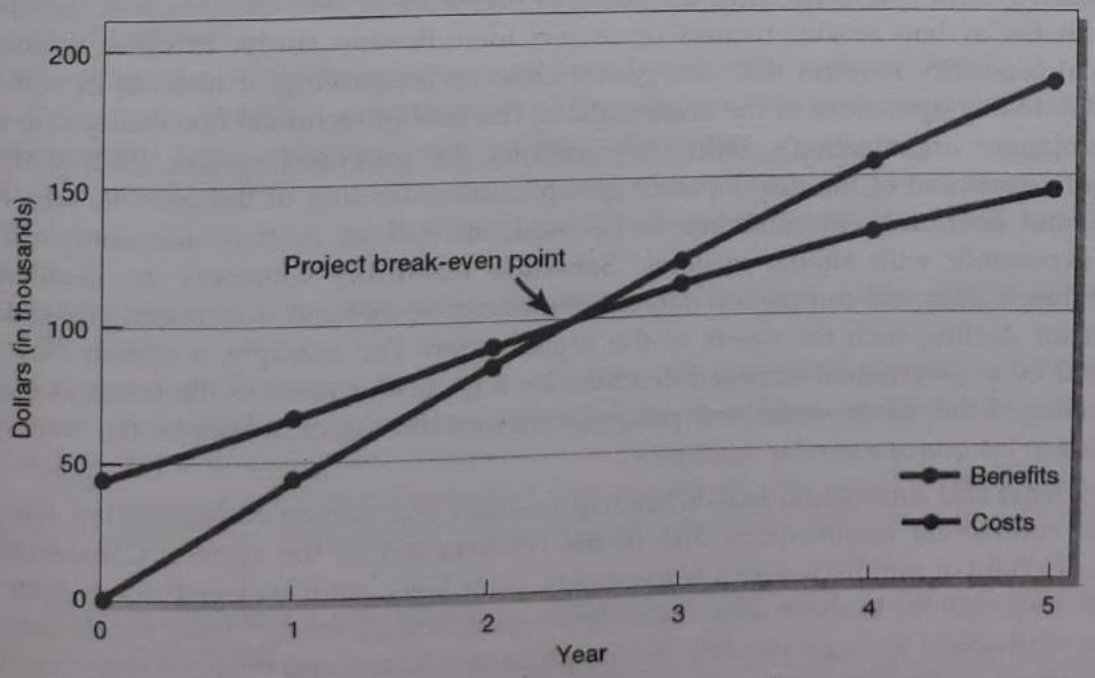


Figure 2.13: Break-even analysis example

Table 2.5: Commonly used economical cost-benefit analysis techniques

Analysis Technique	Description
Net Present Value (NPV)	NPV uses a discount rate determined from the company's cost of capital to establish the present value of a project. The discount rate is used to determine the present value of both cash receipts and outlays.
Return of Investment (ROI)	ROI is the ratio of the net cash receipts of the project divided by the cash outlays of the project. Trade-off analysis can be made among projects competing for investment by comparing their representative ROI ratios.
Break-Even Analysis (BEA)	BEA finds the amount of time required for the cumulative cash flow from a project to equal its initial and ongoing investment.

A systems project, to be approved for continuation, a systems project may not have to achieve break-even or have an ROI greater than estimated during project initiation and planning. Because you may not be able to quantify many benefits or costs at this point in a project, such financial hurdles for a project may be unattainable. In this case, simply doing as thorough an economic analysis as possible, including producing a long list of intangibles, may be sufficient for the project to progress. One other option is to run the type of economic analysis shown in figure 2.13 using pessimistic, optimistic, and expected benefit and cost estimates during project initiation and planning. This range of possible outcomes, along with the list of intangible benefits and the support of the requesting business unit, will often be enough to allow the project to continue to the analysis-phase. Systems analyst must, be as precise as can be with the economic analysis, especially when investment capital is scarce. In this case, it may be necessary to conduct some typical analysis-phase activities during project initiation and planning in order to clearly identify inefficiencies and shortcomings with the existing system and to explain how a new system will overcome these problems.

Assessing Other Feasibility Concerns

You may need to consider other feasibility studies when formulating the business case for a system during project planning. **Operational feasibility** is the process of examining the likelihood that the project will attain its desired objectives. The goal of this study is to understand 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. In other words, assessing operational feasibility requires that you gain a clear understanding of how an IS will fit into the current day-to-day operations of the organization. The goal of **technical feasibility** is to understand the development 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. **Schedule feasibility** considers the likelihood that 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. For example, a system may have to be operational by a government-imposed deadline by a particular point in the business cycle (such as the beginning of the season when new products are introduced), or at least by the time a competitor is expected to introduce a similar system.

Assessing **legal and contractual feasibility** requires that you gain an understanding of any potential legal and contractual ramifications due to the construction of the system. Considerations might include copyright or nondisclosure infringements, 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. Typically, legal and contractual feasibility is a

greater consideration if your organization has historically used an outside organization for specific systems or services that you now are considering handling yourself. Assessing **political feasibility** involves understanding how key stakeholders within the organization view the proposed system. Because an information system may affect the distribution of information within the organization, and thus the distribution of power, the construction of an IS can have political ramifications. Those stakeholders not supporting the project may take steps to block, disrupt, or change the project's intended focus.

Building and Reviewing the Baseline Project Plan

Building the Baseline Project Plan

All the information collected during project initiation and planning is collected and organized into a document called the baseline project plan. Once the BPP is completed, a formal review of the project can be conducted with customers. The focus of the walkthrough is to verify all information and assumptions in the baseline plan before moving ahead with the project. An outline of a baseline project plan, shown in table 2.6, contains four major sections:

1. Introduction
2. System description
3. Feasibility assessment
4. Management issues

Table 2.6: Outline of a baseline project plan

BASELINE PROJECT PLAN REPORT
1.INTRODUCTION • Project Overview: Provides an executive summary that specifies the project's scope, feasibility, justification, resource requirements, and schedules. Additionally, a brief statement of the problem, the environment in which the system is to be implemented, and constraints that affect the project are provided. • Recommendation: Provides a summary of important findings from the planning process and recommendations for subsequent activities.
2. SYSTEM DESCRIPTION • Alternatives: Provides a brief presentation of alternative system configurations. • System Description: Provides a description of the selected configuration and a narrative of input information, tasks performed, and resultant information.
3. FEASIBILITY ASSESSMENT • Economic Analysis: Provides an economic justification for the system using cost-benefit analysis. • Technical Analysis: Provides a discussion of relevant technical risk factors and an overall risk rating of the project. • Operational Analysis: Provides an analysis of how the proposed system solves business problems or takes advantage of business opportunities in addition to an assessment of how current day-to-day activities will be changed by the system. • Legal and Contractual Analysis: Provides a description of any legal or contractual risks related to the project (e.g., copyright or nondisclosure issues, data capture or transferring, and so on). • Political Analysis: Provides a description of how key stakeholders within the organization view the proposed system. • Schedules, Time Line, and Resource Analysis—Provides a description of potential timeframe and completion date scenarios using various resource allocation schemes.

4. MANAGEMENT ISSUES
- Team Configuration and Management: Provides a description of the team member roles and reporting relationships.
- Communication Plan: Provides a description of the communication procedures to be followed by management, team members, and the customer.
- Project Standards and Procedures: Provides a description of how deliverables will be evaluated and accepted by the customer.
- Other Project-Specific Topics: Provides a description of any other relevant issues related to the project uncovered during planning.

The Introduction Section of the Baseline Project Plan: The purpose of the introduction is to provide a brief overview of the entire document and outline a recommended course of action for the project. The entire introduction section is often limited to only a few pages. Although the Introduction section is sequenced as the first section of the BPP, it is often the final section to be written. It is only after performing most of the project planning activities that a clear overview and recommendation can be created. One activity that should be performed initially is the definition of project scope.

The System Description Section of the Baseline Project Plan: The second section of the BPP is the System Description, which contains an outline of possible alternative solutions in addition to the one deemed most appropriate for the given situation. Note that this description is at a very high level and mostly narrative in form. The following examples demonstrate that alternatives may be stated simply:

1. Web-based online system
2. Mainframe with central database
3. Local area network with decentralized databases
4. Batch data input with online retrieval
5. Purchasing of a prewritten package

If the project is approved for construction or purchase, you will need to collect and structure information in a more detailed and rigorous manner during the analysis phase and evaluate in greater depth these and other alternative directions for the system.

The Feasibility Assessment Section of the Baseline Project Plan: In the third section, Feasibility Assessment, issues related to project costs and benefits, technical difficulties, and other such concerns are outlined. This is also the section where high level project schedules are specified using network diagrams and Gantt charts. During project initiation and planning, task and activity estimates are not generally detailed. An accurate work breakdown can be done only for the next one or two life cycle activities. After defining the primary tasks for the project, an estimate of the resource requirements can be made. As with defining tasks and activities; this activity is primarily concerned with gaining rough estimates of the human resource requirements because people are the most expensive resource element. Once systems analyst defines the major tasks and resource requirements, a preliminary schedule can be developed. Defining an acceptable schedule may require that you find additional or different resources or that you change the scope of the project. The greatest amount of project planning effort is typically expended on these Feasibility Assessment activities.

The Management Issues Section of the Baseline Project Plan: The final section, Management Issues, outlines a number of managerial concerns related to the project. This will be a very short section if the proposed project is going to be conducted exactly as prescribed by the organization's standard systems development methodology. Most projects, however, have some unique characteristics that require minor to major deviation from the standard methodology. In the Team Configuration and Management portion, you identify the types of people to work on the project, who will be responsible for which tasks, and how work will be supervised and reviewed. In the Communications Plan portion, you explain how the user will be kept informed about project progress (such as periodic review meetings or even a newsletter) and what mechanisms will be used to foster sharing of ideas among team members, such as some form of computer-based conference facility. An example of the type of information contained in the Project Standards and Procedures portion would be procedures for submitting and approving project change requests and any other issues deemed important for the project's success.

Reviewing the Baseline Project Plan

Before the next phase of the SDLC can begin, the users, management, and development group must review the BPP in order to verify that it makes sense. This review takes place before the BPP is submitted or presented to a project approval body, such as an IS steering committee or the person who must fund the project. The objective of this review is to ensure that the proposed system conforms to organizational standards and that all relevant parties understand and agree with the information contained in the BPP. A common method for performing this review (as well as reviews during subsequent life cycle phases) is called a **structured walk-through**. **Walk-throughs** are peer group reviews of any product created during the systems development process and are widely used by professional development organizations. Experience has shown that walk-throughs are a very effective way to ensure the quality of an information system and have become a common day-to-day activity for many systems analysts.

Most walk-throughs are not rigidly formal or exceedingly long in duration. It is important, however, to establish a specific agenda for the walk-through so that all attendees understand what is to be covered and the expected completion time. At walk-through meetings, there is a need to have individuals play specific roles. These roles are as follows:

- **Coordinator.** This person plans the meeting and facilitates a smooth meeting process. This person may be the project leader or a lead analyst responsible for the current life cycle step.
- **Presenter.** This person describes the work product to the group. The presenter is usually an analyst who has done all or some of the work being presented.
- **User.** This person (or group) makes sure that the work product meets the needs of the project's customers. This user would usually be someone not on the project team.
- **Secretary.** This person takes notes and records decisions or recommendations made by the group. This may be a clerk assigned to the project team or it may be one of the analysts on the team.
- **Standards bearer.** The role of this person is to ensure that the work product adheres to organizational technical standards. Many larger organizations have staff groups within the unit responsible for establishing standard procedures, methods, and documentation formats.

These standards bearers validate the work so that it can be used by others in the development organization.

- **Maintenance oracle.** This person reviews the work product in terms of future maintenance activities. The goal is to make the system and its documentation easy to maintain.

Walk-through meetings are a common occurrence in most systems development groups and can be used for more activities than reviewing the BPP, including the following:

- System specifications
- Logical and physical designs
- Code or program segments
- Test procedures and results
- Manuals and documentation

One of the key advantages in using a structured review process is it ensures that formal review points occur during the project. At each subsequent phase of the project, a formal review should be conducted (and shown on the project schedule) to make sure all aspects of the project are satisfactorily accomplished before assigning additional resources to the project. This conservative approach of reviewing each major project activity with continuation contingent on successful completion of the prior phase is called incremental commitment. It is much easier to stop or redirect a project at any point when using this approach.

Walk-throughs are used throughout the duration of the project for briefing team members and external stakeholders. These presentations can provide many benefits to the team, but, unfortunately, are often not well done. With the proliferation of computer technology and the availability of powerful software to assist in designing and delivering presentations, making an effective presentation has never been easier. Microsoft's Power Point has emerged as the de facto standard for creating computer-based presentations. Although this program is relatively easy to use, it can also be misused such that the "bells and whistles" added to a computer-based presentation actually detract from the presentation. Like any project, to make an effective presentation it must be well-planned, well-designed, and well-delivered. Planning and designing your presentation is equally important as delivering it. If your slides are poorly laid out, hard to read, or inconsistent, it won't matter how good your delivery is; your audience will think more about the poor quality of the slides than about what you are saying. Fortunately, with a little work it is easy to design a high-quality presentation if you follow a few simple steps, which are outlined in table 2.7.

Table 2.7: Guideline for making an effective presentation

	Presentation Planning
Who is the audience?	To design the most effective presentation, you need to consider the audience (e.g., What do they know about your topic? What is their education level?).
What is the message?	Your presentation should be designed with a particular objective in mind.
What is the presentation environment?	Knowledge of the room size, shape, and lighting is valuable information for designing an optimal presentation.

Presentation Design	
Organize the sequence.	Organize your presentation so that like elements or topics are found in one place, instead of scattered throughout the material in random fashion.
Keep it simple.	Make sure that you don't pack too much information onto a slide so that it is difficult to read. Also, work to have as few slides as possible; in other words, only include information that you absolutely need.
Be consistent.	Make sure that you are consistent in the types of fonts, font sizes, colors, design approach, and backgrounds.
Use variety.	Use both textual and graphical slides to convey information in the most meaningful format.
Don't rely on the spellchecker alone.	Make sure you carefully review your presentation for typographical and wording errors.
Use bells and whistles sparingly.	Make sure that you use familiar graphical icons to guide and enhance slides; don't lose sight of your message as you add bells and whistles. Also, take great care when making transitions between slides and elements so that "special effects" don't take away from your message.
Use supplemental materials appropriately.	Take care when using supplemental materials so that they don't distract the audience. For example, don't provide handouts until you want the audience to actually read this material.
Have a clear beginning and end.	At the beginning, introduce yourself and your teammates (if any), thank your audience for being there, and provide a clear outline of what will be covered during the presentation. At the conclusion, have a concluding slide so that the audience clearly sees that the presentation is over.
Presentation Delivery	
Practice.	Make sure that you thoroughly test your completed work on yourself and others to be sure it covers your points and presents them in an effective manner within the time frame required.
Arrive early and cue up your presentation.	It is good practice, when feasible, to have your presentation ready to go prior to the arrival of the audience.
Learn to use the "special" software keys.	Using special keys to navigate the presentation will allow you to focus on your message and not on the software.
Have a backup plan.	Have a backup plan in case technology fails or your presentation is lost when traveling.
Deliver the information effectively.	To make an effective presentation, you must become an effective public speaker through practice.
Personal appearance matters.	Your appearance and demeanor can go a long way toward enhancing how the audience receives your presentation.

Chapter 3

ANALYSIS

CHAPTER OUTLINE

After studying this chapter, students will be able to understand the:

- System Requirements
- System Process Requirements
- System Data Requirements

INTRODUCTION

Analysis is the first systems development life cycle (SDLC) phase where we begin to understand, in depth, the need for system changes. Systems analysis involves a substantial amount of effort and cost, and is therefore undertaken only after management has decided that the systems development project under consideration has merit and should be pursued through this phase. The analysis team should not take the analysis process for granted or attempt to speed through it. Most observers would agree that many of the errors in developed systems are directly traceable to inadequate efforts in the analysis and design phases of the life cycle. Because analysis is a large and involved process, we divide it into two main activities to make the overall process easier to understand:

- **Requirements determination.** This is primarily a fact finding activity.
- **Requirements structuring.** This activity creates a thorough and clear description of current business operations and new information processing services.

DETERMINING SYSTEM REQUIREMENTS

Collection of information is at the core of systems analysis. Information requirement determination (IRD) is frequently and convincingly presented as the most critical phase of information system (IS) development, and many IS failures have been attributed to incomplete and inaccurate information requirements. System analysts must collect the information about the current system and how users would like to improve their performance with new information system. Accurately understanding the users' requirements will help the system developing team deliver a proper system to the end users in limited time and limited budget.

Information on what the new system should do is gathered from as many sources as possible: users of current system, from observing users, from the reports, forms and procedures. All this is documented and made ready for structuring. Main characteristics of a good systems analyst are listed below:

- **impertinence** (ask questions about everything that exists and also about what may exist in the future),
- **impartiality** (find the best solution to a business problem or opportunity - consider issues raised by all parties),
- **relax constraints** (eliminate unfeasibility, assume that anything is possible, traditions may not always be reasonable),
- **attention to details** (everything must fit together so that the system works properly),
- **Reframing** (every system is different and needs a new creative approach).

The primary deliverables from requirements determination are various forms of information gathered during the determination process: transcripts of interviews, notes from observations and analysis of documents, analyzed responses from questionnaires, sets of forms, reports, job descriptions, or computer-generated output such as system prototypes. We could group all this information in three groups:

- Information collected from conversations with or observations of users (interview transcripts, notes form 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)

TRADITIONAL METHODS FOR DETERMINING REQUIREMENTS

At the core of systems analysis is the collection of information. At the outset, we must collect information about the information systems that are currently being used and how users would like to improve the current systems and organizational operations with new or replacement information systems. 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.

INTERVIEWING AND QUESTIONNAIRES

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. Others are interviewed to understand organizational direction, policies, and expectations that managers have of the units they supervise. During interviewing, you gather facts, opinions, and speculation and observe body language, emotions, and other signs of what people want and how they assess current systems. Interviewing someone effectively can be done in many ways, and no one method is necessarily better than another. There are many ways to arrange an effectively interview and no one is superior to others. However, experience analysts commonly accept some following best practices for an effective interview:

- Prepare the interview carefully, including appointment, priming question, checklist, agenda, and questions.
- Listen carefully and take note during the interview (tape record if possible)
- Review notes within 48 hours after interview
- Be neutral
- Seek diverse views

The general nature of the interview should be explained to the interviewee in advance. You may ask the interviewee to think about specific questions or issues, or to review certain documentation to prepare for the interview. Spend some time thinking about what you need to find out, and write down your questions. Do not assume that you can anticipate all possible questions. You want the interview to be natural and, to some degree, you want to direct the

interview spontaneously as you discover what expertise the interviewee brings to the session. Prepare an interview guide or checklist so that you know in which sequence to ask your questions and how much time to spend in each area of the interview. The checklist might include some probing questions to ask as follow-up if you receive certain anticipated responses.

Questionnaires

Questionnaires have the advantage of gathering information from many people in a relatively short time and of being less biased in the interpretation of their results. Choosing right questionnaires respondents and designing effective questionnaires are the critical issues in this information collection method. People usually are only use a part of functions of a system, so they are always just familiar with a part of the system functions or processes. In most situations, one copy of questionnaires obviously cannot fit to all the users. To conduct an effective survey, the analyst should group the users properly and design different questionnaires for different group. Moreover, the ability to build good questionnaires is a skill that improves with practice and experience. When designing questionnaires, the analyst should concern the following issues at least:

- The ambiguity of questions.
- Consistence of respondents' answers.
- What kind of question should be applied, open-ended or close-ended?
- What is the proper length of the questionnaires?

Interview Outline	
Interviewee: Name of person being interviewed	**Interviewer**: Name of person leading interview
Location/Medium: Office, conference room, or phone number	**Appointment Date:** **Start Time:** **End Time:**
Agenda: Introduction Background Project Overview of Interview Topics to be covered Permission to Tape Record Topic 1 Questions Topic 2 Questions Summary of Major Points Questions from Interviewee Closing	**Approximate Time:**
General Observations: Interviewee seemed busy - probably need to call in a few days for follow-up questions since he gave only short answers. PC was turned off- probably not a regular PC user.	

Unresolved Issues, Topics not covered: He needs to look up figures of previous years. He raised the issue of how to handle the problem, but he did not have to discuss.	
When to ask question, if conditional Question number 1 Have you used current system? If so, how often? If yes, go to Question number 2.	**Answer** Yes, I ask for report weekly **Observation** Seemed anxious - may be over-estimating usage frequency
Question 2 What do you like least about this system?	**Answer** Using wrong units Observations: Options determine the right units, but user chose the wrong one.

Figure 3.1: A typical interview guide

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. Respondents must feel free to state their true opinions and perspectives and trust that their ideas will be considered. Avoid questions such as "Should the system continue to provide the ability to override the default value, even though most users now do not like the feature?" because such wording predefines a socially acceptable answer.
- Second, listen carefully to what is being said. Take careful notes or, if possible, record the interview on a tape recorder (be sure to ask permission first!). The answers may contain extremely important information for the project. Also, this may be your only chance to get information from this particular person. If you run out of time and still need more information from the person you are talking to, ask to schedule a follow-up interview.
- Third, once the interview is over, go back to your office and key in your notes within forty-eight hours with a word processing program such as Microsoft Word. For numerical data, you can use a spreadsheet program such as Microsoft Excel. If you recorded the interview, use the recording to verify your notes. After forty-eight hours, your memory of the interview will fade quickly. As you type and organize your notes, write down any additional questions that might arise from lapses in your notes or ambiguous information. Separate facts from your opinions and interpretations. Make a list of unclear points that need clarification. Call the person you interviewed and get answers to these new questions. Use the phone call as an opportunity to verify the accuracy of your notes. You may also want to send a written copy of your notes to the person you interviewed to check your notes for accuracy. Finally, make sure to thank the person for his or her time. You may need to talk to your respondent again. If the interviewee will be a user of your system or is involved in some other way in the system's success, you want to leave a good impression.

- 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. Let the interviewee know that there are many steps to the project. Many people will have to be interviewed. Choices will have to be made from among many technically possible alternatives. Let respondents know that their ideas will be carefully considered. Because of the repetitive nature of the systems development process, however, it is premature to say now exactly what the ultimate system will or will not do.
- Fifth, seek a variety of perspectives from the interviews. Talk to several different people: potential users of the system, users of other systems that might be affected by this new system, managers and superiors, information systems staff, and others. Encourage people to think about current problems and opportunities and what new information services might better serve the organization. You want to understand all possible perspectives so that later you will have information on which to base a recommendation or design decision that everyone can accept.

Comparison between Interviews and Questionnaires		
Characteristics	**Interviews**	**Questionnaires**
Information Richness	High (many channel)	Medium to low (only responses)
Time Required	Can be extensive	Low or moderate
Expense	Can be high	Moderate
Confidentiality	Interviewee is known to interviewer	Respondent can be unknown
Chance of Follow-up & Probing	Good: probing and clarification questions can be asked by either interviewer or interviewee	Limited: probing and follow-up done after original data collection
Involvement of Subject	Interviewee is involved and committed	Respondent is passive, no clear commitment
Potential Audience	Limited numbers, but complete responses from those interviewed	Can be quite large, but lack of response from some can bias results

DIRECTLY OBSERVING USERS

People are not always very reliable informants, even when they try to be reliable and tell what they think is the truth. People often do not have a completely accurate appreciation of what they do or how they do it. This we especially true concerning infrequent events, issues from the past, or issues for which people have considerable passion. Since people cannot always be trusted to reliably interpret and report their own actions, analyst can supplement and corroborate what people say by watching what they do or by obtaining relatively objective measures of how people behave in work situation. However, observation can cause people to change their normal operation behavior. It will make the gathered information biased.

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 support. 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.

However, when analyzing those official documentations, analysts should pay attention to the difference between the systems described on the official documentations and the practical systems in real world. For the reason of inadequacies of formal procedures, individual work habits and preferences, resistance to control, and other factors, the difference between so called formal system and informal system universally exists.

In documents you can find information about:

- Problems with existing systems (e.g., missing information or redundant steps)
- Opportunities to meet new needs if only certain information or information processing were available (e.g., analysis of sales based on customer type)
- Organizational direction that can influence information system requirements (e.g., trying to link customers and suppliers more closely to the organization)
- Titles and names of key individuals who have an interest in relevant existing systems (e.g., the name of a sales manager who has led a study of buying behavior of key customers)
- Values of the organization or individuals who can help determine priorities for different capabilities desired by different users (e.g., maintaining market share even if it means lower short-term profits)
- Special information-processing circumstances that occur irregularly that may not be identified by any other requirements determination technique (e.g., special handling needed for a few large-volume customers who require use of customized customer ordering procedures)
- The reason why current systems are designed as they are, which can suggest features left out of current software that may now be feasible and desirable (e.g., data about a customer's purchase of competitors' products not available when the current system was designed; these data now available from several sources)
- Data, rules for processing data, and principles by which the organization operates that must be enforced by the information system (e.g., each customer assigned exactly one sales department staff member as primary contact if customer has any questions)

CONTEMPORARY METHODS FOR DETERMINING SYSTEM REQUIREMENTS

Even though we called interviews, questionnaires, observation, and document analysis traditional methods for determining a system's requirements, all of these methods are still used by analysts to collect important information. Today, however, additional techniques are available 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 learn about two modern information-gathering techniques for analysis: joint application design (JAD) and prototyping. These techniques can support effective information collection and structuring while reducing the amount of time required for analysis.

JOINT APPLICATION DESIGN

JAD was developed by Chuck Morris and Tony Crawford of IBM in 1977 and has since been proven successful on thousands of software projects across industry sectors and application boundaries. Joint Application Development (JAD) is a user requirements elicitation process that involves the system owner and end users in the design and development of an application through a succession of collaborative workshops called JAD sessions. The JAD approach leads to shorter development lifecycles and greater client satisfaction because it draws users and information systems analysts together to jointly design systems in facilitated group sessions. JAD can also be adapted to developing business processes that do not involve computer automation.

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 and structured, but highly effective, process. Having all the key people together in one place at one time allows analysts to see the areas of agreement and the areas of conflict. Meeting with all 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. JAD sessions are usually conducted in a location away from where the people involved normally work, in order to limit distractions and help participants' better 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. Other expenses include the costs associated with flying people to a remote site and putting them up in hotels and feeding them for several days. The following is a list of typical JAD participants:

- **JAD session leader:** The JAD 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 all ideas.
- **Users:** The key users of the system under consideration are vital participants in a JAD. They are the only ones who clearly understand 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, because of its expense, a JAD must be sponsored by someone at a relatively high level in the company such as a vice president or chief executive officer. If the sponsor attends any sessions, it is usually only at the 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, usually on a personal computer or laptop.
- **IS staff:** Besides systems analysts, other IS staff, such as programmers, database analysts, IS planners and data-center personnel may attend the session. Their purpose is to learn from the discussion and possibly to contribute their ideas on the technical feasibility of proposed ideas or on the technical limitations of current systems.

JAD sessions are usually held in special-purpose rooms where participants sit around horseshoe-shaped tables, as in Figure below. These rooms are typically equipped with whiteboards (possibly electronic, with a printer to make copies of what is written on the board). Other audiovisual tools may be used, such as magnetic symbols that can be easily rearranged on a whiteboard, flip charts, and computer-generated displays. Flip-chart paper is typically used for keeping track of issues that cannot be resolved during the JAD, or for those issues requiring additional information that can be gathered during breaks in the proceedings. Computers may be used to create and display form or report designs or to diagram existing or replacement systems. In general, however, most JADs do not benefit much from computer support. The end result of a completed JAD is a set of documents that detail the workings of the current system and the features of a replacement system. Depending on the exact purpose of the JAD, analysts may gain detailed information on what is desired of the replacement system.

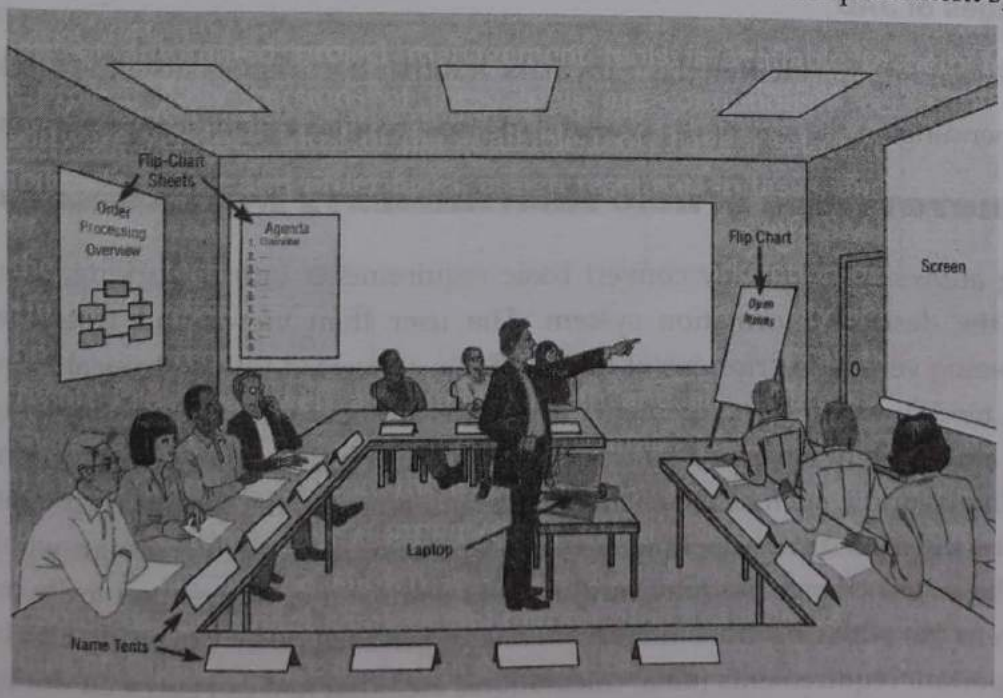


Figure 3.2: A typical room layout for a JAD session. Source: Based on Wood and Silver, 1989

When a JAD is completed, the final result is a set of documents that detail the workings of the current system related to study of a replacement system. These requirements definition document generally includes business activity model and definitions, data model and definition, data input and output requirements. It may also include interface requirements, screen and report layouts, ad hoc query specifications, menus, and security requirements. When used at a later point in the system development life cycle, a JAD session can also be used to refine a system prototype, develop new job profiles for system users, or develop an implementation plan. However, to exploit full potential of JAD, the groupware tools should be applied in JAD workshop sessions. The use of groupware tools to support the joint Application Development technique increases the value of this technique dramatically. When groupware tools are used in an automated JAD workshop, they greatly facilitate the generation, analysis, and documentation of information. This is particularly valuable for JAD workshops conducted to define and build consensus on the requirements for new systems.

Advantages of JAD

- 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

Disadvantages of JAD

- 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

Using Prototyping during Requirements Determination

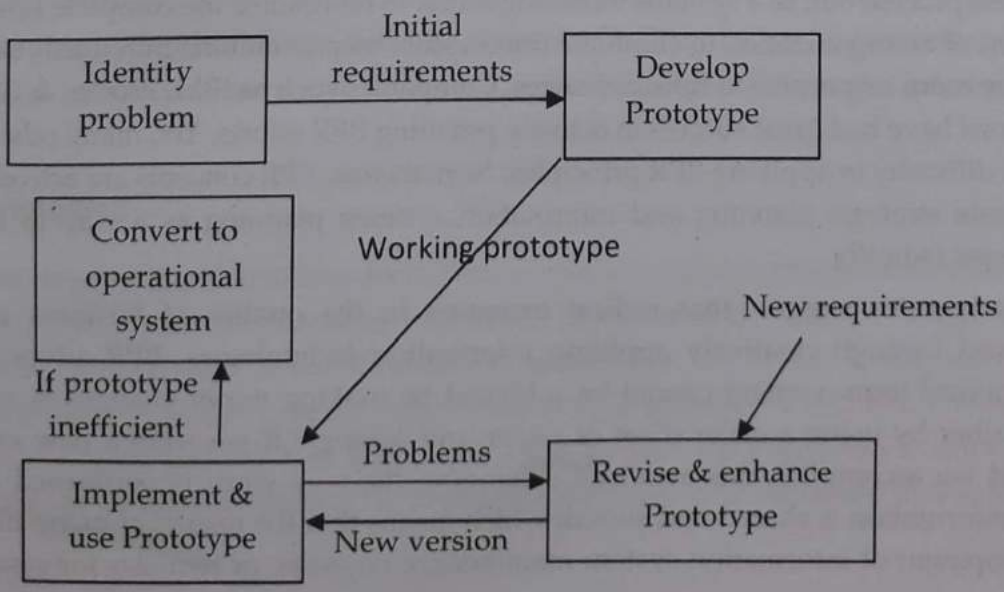
Prototyping allows us to quickly convert basic requirements into a working, though limited, version of the desired information system. The user then views and tests the prototype. Typically, seeing verbal descriptions of requirements converted into a physical system prompts the user to modify existing requirements and generate new ones. For example, in the initial interviews, a user might have said he wanted all relevant utility billing information on a single computer display form, such as the client's name and address, the service record, and payment history. 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 for the information to be 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. You would then redesign the prototype to incorporate the suggested changes. Once modified, users would again view and test the prototype. Once again, you would incorporate their suggestions for change. Through such a repetitive process, the chances are good that you will be able to better capture a system's requirements. The goal with using prototyping to support requirements determination is to develop concrete specifications for the ultimate system, not to build the ultimate system. Prototyping is most useful for requirements determination when:

- User requirements are not clear or well understood, which is often the case for totally new systems or systems that support decision making.
- One or a few users and other stakeholders are involved with the system.
- Possible designs are complex and require concrete form to evaluate fully.
- Communication problems have existed in the past between users and analysts, and both parties want to be sure that system requirements are as specific as possible.
- Tools (such as form and report generators) and data are readily available to rapidly build working systems.

Prototyping also has some drawbacks as a tool for requirements determination. They include the following:

- A tendency to avoid creating formal documentation of system requirements, which can then make the system more difficult to develop into a fully working system.
- Prototypes can become idiosyncratic to the initial user and difficult to diffuse or adapt to other potential users.
- Prototypes are often built as stand-alone systems, thus ignoring issues of sharing data and interactions with other existing systems.
- Checks in the SDLC are bypassed so that some more subtle, but still important, system requirements might be forgotten (e.g., security, some data-entry controls, or standardization of data across systems).



RADICAL METHODS FOR DETERMINING SYSTEM REQUIREMENTS

Whether traditional or modern, the methods for determining system requirements that you have read about in this chapter apply to any requirements determination effort, regardless of its motivation. Yet, most of what you have learned has traditionally been applied to systems development projects that involve automating existing processes. Analysts use system requirements determination to understand current problems and opportunities, as well as what are needed and desired in future systems. Typically, the current way of doing things has a large impact on the new system. In some organizations, though, management is looking for new ways to perform current tasks. These ways may be radically different from how things are done now, but the payoffs may be enormous: Fewer people may be needed to do the same work; relationships with customers may improve dramatically; and processes may become much more efficient and effective, all of which can result in increased profits. The overall process by which current methods are replaced with radically new methods is 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 U.S. tour, you discover that improving your putting, driving accuracy, and sand shots will help, but the new competitive environment is simply not suited to your playing style. 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 press. If you are good enough, you may survive, but without reengineering, you will never become a winner. Just as the competitiveness of golf forces good players to adapt their games to changing conditions, the competitiveness of our global economy has driven most companies into a mode of continuously improving the quality of their products and services. Organizations realize that creatively using information technologies can significantly improve most business processes. The idea behind BPR is not just to improve each business process but, in a systems modeling sense, to reorganize the complete flow of data in major sections of an organization to eliminate unnecessary steps, combine previously separate steps, and become more responsive to future changes. Companies such as IBM, Procter & Gamble, Wal-Mart, and Ford have had great success in actively pursuing BPR efforts. Yet, many other companies have found difficulty in applying BPR principles. Nonetheless, BPR concepts are actively applied in both corporate strategic planning and information systems planning as a way to improve business processes radically.

BPR advocates suggest that radical increases in the quality of business processes can be achieved through creatively applying information technologies. BPR advocates also suggest that radical improvement cannot be achieved by making minor changes in existing processes but rather by using a clean sheet of paper and asking, "If we were a new organization, how would we accomplish this activity?" Changing the way work is performed also changes the way information is shared and stored, which means that the results of many BPR efforts are the development of information system maintenance requests, or requests for system replacement.

You likely have encountered or will encounter BPR initiatives in your own organization. A recent survey of IS executives found that they view BPR to be a top IS priority for the coming years.

Identifying Processes to Reengineer

A first step in any BPR effort is to understand what processes need to change, what are 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. BPR, therefore, requires you first to understand those activities that are part of the organization's key business processes and then to alter the sequence and structure of activities to achieve radical improvements in speed, quality, and customer satisfaction. The same techniques you learned to use for system requirements determination can be applied to discovering and understanding key business processes: interviewing key individuals, observing activities, reading and studying organizational documents, and conducting JAD sessions. After identifying key business processes, the next step is to identify specific activities that can be radically improved through reengineering. Michael Hammer and James Champy, two academics who coined the term BPR, suggest systems analysts ask three questions to identify activities for radical change:

1. How important is the activity to delivering an outcome?
2. How feasible is changing the activity?
3. How dysfunctional is the activity?

The answers to these questions provide guidance for selecting which activities to change. Those activities deemed important, changeable, yet functional, are primary candidates for alteration. To identify dysfunctional activities, Hammer and Champy suggest you look for activities that involve excessive information exchanges between individuals, information that is redundantly recorded or needs to be rekeyed, excessive inventory buffers or inspections, and a lot of rework or complexity.

Disruptive Technologies

Once key business processes and activities have been identified, information technologies must be applied to improve business processes radically. Hammer and Champy 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 about the power of new technologies and think of innovative ways to alter the way work is done. This approach is contrary to deductive thinking, in which problems are first identified and solutions then formulated.

Hammer and Champy suggest that managers especially consider disruptive technologies when applying deductive thinking. Disruptive technologies are those that enable the breaking of long-held business rules that inhibit organizations from making radical business changes. For example, Toyota is using production schedule databases and electronic data interchange (EDI)—an information system that allows companies to link their computers directly to suppliers—to work with its suppliers as if they and Saturn were one company. Suppliers do not wait until Saturn sends them a purchase order for more parts but simply monitor inventory levels and automatically send shipments as needed.

STRUCTURING SYSTEM PROCESS REQUIREMENTS

Here we understand the logical modeling of processes by studying examples of data flow diagrams.

PROCESS MODELING

Process modeling involves graphically representing the processes, or actions, that capture, manipulate, store, and distribute data between a system and its environment and among components within a system. A common form of a process model is a data-flow diagram (DFD). A data-flow diagram is a graphic that illustrates the movement of data between external entities and the processes and data stores within a system. Although several different tools have been developed for process modeling, we focus solely on data-flow diagrams because they are useful tools for process modeling. Data-flow diagramming is one of several structured analysis techniques used to increase software development productivity. Although not all organizations use each structured analysis technique, collectively, these techniques, like dataflow diagrams, have had a significant impact on the quality of the systems development process.

MODELING A SYSTEM'S PROCESS

The analysis team begins the process of structuring requirements with an abundance of information gathered during requirements determination. As part of structuring, you and the other team members must organize the information into a meaningful representation of the information system that exists and of the requirements desired in a replacement system. In addition to modeling the processing elements of an information system and transformation of data in the system, you must also model the structure of data within the system. Analysts use both process and data models to establish the specification of an information system. With a supporting tool, such as a CASE tool, process and data models can also provide the basis for the automatic generation of an information system.

Deliverables and Outcomes

In structured analysis, the primary deliverables from process modeling are a set of coherent, interrelated data-flow diagrams. The following table lists the progression of deliverables that result from studying and documenting a system's process. First, a context data-flow diagram

shows the scope of the system, indicating which elements are inside and outside the system. Second, data-flow diagrams of the current system specify which people and technologies are used in which processes to move and transform data, accepting inputs and producing outputs. The detail of these diagrams allows analysts to understand the current system and eventually to determine how to convert the current system into its replacement. Third, technology-independent, or logical, data-flow diagrams show the dataflow, structure, and functional requirements of the new system. Finally, entries for all of the objects in all diagrams are included in the project dictionary or CASE repository.

1. Context DFD
2. DFDs of current physical system
3. DFDs of new logical system
4. Thorough description of each DFD component

Table: Deliverables for Process Modeling

This logical progression of deliverables helps you to understand the existing system. You can then reduce this system into its essential elements to show the way in which the new system should meet its information processing requirements, as they were identified during requirements determination. In later steps in the systems development life cycle, you and other project team members make decisions on exactly how the new system will deliver these new requirements in specific manual and automated functions. Because requirements determination and structuring are often parallel steps, data-flow diagrams evolve from the more general to the more detailed as current and replacement systems are better understood. Even though data-flow diagrams remain popular tools for process modeling and can significantly increase software development productivity, they are not used in all systems development methodologies. Some organizations, such as EDS, have developed their own type of diagrams to model processes. Some methodologies, such as rapid application development (RAD), do not model processes separately at all. Instead, RAD builds processes—the work or actions that transform data so that they can be stored or distributed-into the prototypes created as the core of its development life cycle. However, even if you never formally use data-flow diagrams in your professional career, they remain a part of systems development's history. DFDs illustrate important concepts about the movement of data between manual and automated steps and are a way to depict work flow in an organization. DFDs continue to benefit information systems professionals as tools for both analysis and communication. For that reason, we devote this entire chapter to DFDs.

Data Flow Diagrams

DFD graphically representing the [illegible] processes which capture, manipulate, store, and distribut[illegible] n components of a system. The visu[illegible] between User and System designer[illegible] l expands it to a hierarchy of detail[illegible] asons:

Refer teacher note for DFD, E-R-diagram.

- Logical DFD is based on business events and independent of particular technology or physical arrangement, which makes the resulting system more stable.
- Logical DFD allows analyst to understand the business being studied and to identify the reason behind implementation plans.
- Systems implemented based on logical DFD will be easier to maintain because business functions are not subject to frequent change.
- Very often, logical DFD does not contain data stores other than files or a database, making less complex than physical DFD and is easier to develop.
- Physical DFD can be easily formed by modifying a logical DFD.

Benefits of Physical Data Flow Diagram

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

LOGIC MODELING

A logic model is a graphic depiction (road map) that presents the shared relationships among the resources, activities, outputs, outcomes, and impact for your program. It depicts the relationship between your program's activities and its intended effects.

Data flow diagrams do not show the logic inside the processes – what occurs within a process? How input data is converted into output information. Logic modeling involves representing internal structure and functionality of processes depicted on a DFD. Processes must be clearly described before translating them into programming language. Logic modeling can also be used to show when processes on a DFD occur. Logic modeling will be generic without taking syntax of a particular programming language. A logic model has four components:

1. **Needs** are about the problem and why it's important to address it. What issue are we trying to address?
2. **Inputs** are the things that contribute to addressing the need (usually a combination of money, time, expertise and resources). What resources are we investing?
3. **Activities** describe the things that the inputs allow to happen. What are we doing with these resources?
4. **Outcomes** are usually expressed in terms of measures of success. What difference are we hoping to make?

Each process on the lowest level DFD will be represented by one or more of the following:

- Structured English
- Decision Tables
- Decision Trees
- State-transition diagrams
- Sequence diagrams
- Activity diagrams

Modeling Logic with Decision Tables

Decision tables are a concise visual representation for specifying which actions to perform depending on given conditions. They are algorithms whose output is a set of actions. A decision table is an excellent tool to use in both testing and requirements management. Essentially it is a structured exercise to formulate requirements when dealing with complex business rules. Decision tables are used to model complicated logic. They can make it easy to see that all possible combinations of conditions have been considered and when conditions are missed, it is easy to see this.

Let's take an example scenario for an ATM where a decision table would be of use.

A customer requests a cash withdrawal. One of the business rules for the ATM is that the ATM machine pays out the amount if the customer has sufficient funds in their account or if the customer has the credit granted. Already, this simple example of a business rule is quite complicated to describe in text. A decision table makes the same requirements clearer to understand:

Conditions	**R1**	**R2**	**R3**
Withdrawal amount<=balance	T	F	F
Credit granted	-	T	F
Actions			
Withdrawal granted	T	T	F

In a decision table, conditions are usually expressed as true (T) or false (F). Each column in the table corresponds to a rule in the business logic that describes the unique combination of circumstances that will result in the actions. The table above contains three different business rules, and one of them is the "withdrawal is granted if the requested amount is covered by the balance." It is normal to create at least one test case per column, which results in full coverage of all business rules.

Decision tables can be used in all situations where the outcome depends on the combinations of different choices, and that is usually very often. In many systems there are tons of business rules where decision tables add a lot of value.

Steps to create decision tables

- **Step 1 – Analyze the requirement and create the first column**

  Requirement: "Withdrawal is granted if requested amount is covered by the balance or if the customer is granted credit to cover the withdrawal amount". Express conditions and resulting actions in a list so that they are either TRUE or FALSE. In this case there are two conditions, "withdrawal amount ≤ balance" and "credit granted". There is one result, the withdrawal is granted.

Conditions
Withdrawal amount<=balance
Credit granted
Actions
Withdrawal granted

- **Step 2: Add Columns**

  Calculate how many columns are needed in the table. The number of columns depends on the number of conditions and the number of alternatives for each condition. If there are two conditions and each condition can be either true or false, you need 4 columns. If there are three conditions there will be 8 columns and so on. Mathematically, the number of columns is $2^{conditions}$. In this case $2^2 = 4$ columns.

  Number of columns that is needed:

Number of Conditions	Number of Columns
1	2
2	4
3	8
4	16
5	32

The bottom line is that you should create more smaller decision tables instead of fewer larger ones, otherwise you run the risk of the decision tables being so large as to be unmanageable. Test the technique by picking areas with fewer business rules.

Now is the time to fill in the T (TRUE) and F (FALSE) for the conditions. How do you do that? The simplest is to say that it should look like this:

- Row 1: TF
- Row 2: TTFF
- Row 3: TTTTFFFF

For each row, there is twice as many T and F as the previous line.

Repeat the pattern above from left to right for the entire row. In other words, for a table with 8 columns, the first row will read TFTFTFTF, the second row will read TTFFTTFF and the third row will read TTTTFFFF.

Conditions				
Withdrawal amount<=balance	T	F	F	F
Credit granted	T	T	F	F
Actions				
Withdrawal granted				

- **Step 3: Reduce the table**

Mark insignificant values with "-". If the requested amount is less than or equal to the account balance it does not matter if credit is granted. In the next step, you can delete the columns that have become identical.

Conditions				
Withdrawal amount<=balance	T	F	T	F
Credit granted	-	T	-	F
Actions				
Withdrawal granted				

Check for invalid combinations. Invalid combinations are those that cannot happen, for example, that someone is both an infant and senior. Mark them somehow, e.g. with "X". In this example, there are no invalid combinations.

Finish by removing duplicate columns. In this case, the first and third column are equal, therefore one of them is removed.

- **Step 4: Determine actions**

Enter actions for each column in the table. You will be able to find this information in the requirement. Name the columns (the rules). They may be named R1/Rule 1, R2/Rule 2 and so on, but you can also give them more descriptive names.

Conditions			
Withdrawal amount<=balance	T	F	F
Credit granted	-	T	F
Actions			
Withdrawal granted	T	T	F

- **Step 5: Write test cases**

  Write test cases based on the table. At least one test case per column gives full coverage of all business rules.

  - **Test case for R1:** balance = 200, requested withdrawal = 200. Expected result: withdrawal granted.
  - **Test case for R2:** balance = 100, requested withdrawal = 200, credit granted. Expected result: withdrawal granted.
  - **Test case for R3:** balance = 100, requested withdrawal = 200, no credit. Expected Result: withdrawal denied.

Advantages and disadvantages of decision tables

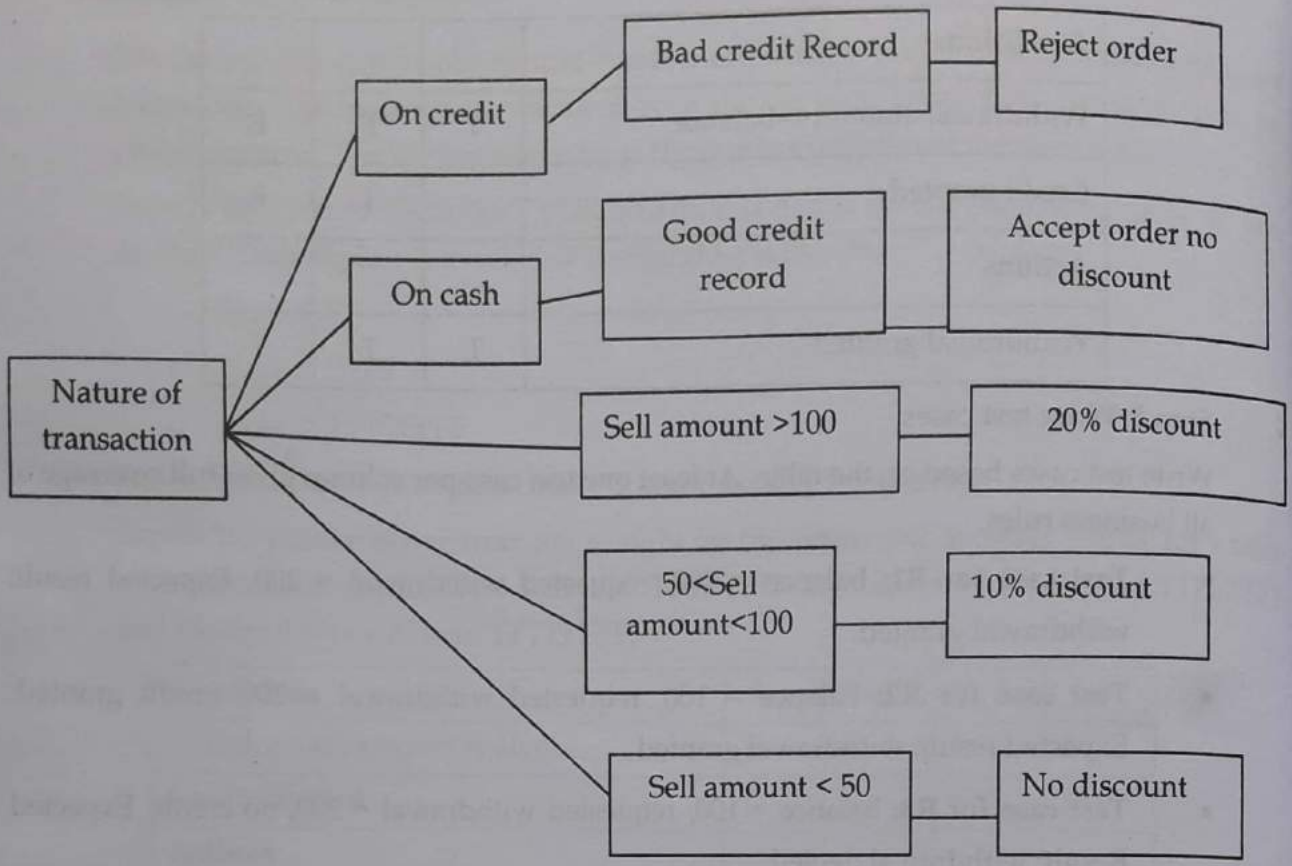
One advantage of using decision tables is that they make it possible to detect combinations of conditions that would otherwise not have been found and therefore not tested or developed. The requirements become much clearer and you often realize that some requirements are illogical, something that is hard to see when the requirements are only expressed in text.

A disadvantage of the technique is that a decision table is not equivalent to complete test cases containing step-by-step instructions of what to do in what order. When this level of detail is required, the decision table has to be further detailed into test cases.

Decision Trees

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Decision trees are a method for defining complex relationships by describing decisions and avoiding the problems in communication. A decision tree is a diagram that shows alternative actions and conditions within horizontal tree framework. Thus, it depicts which conditions to consider first, second, and so on. Decision trees depict the relationship of each condition and their permissible actions. A square node indicates an action and a circle indicates a condition. It forces analysts to consider the sequence of decisions and identifies the actual decision that must be made.



Advantages of Decision tree

- It expresses the logic of if then else in pictorial form.
- It is useful to express the logic when a value is variable or an action is dependent on nested decision i.e. the outcome of another decision.
- It helps the analyst to identify the actual decision to be made.
- It is used to verify logic and problems that involve a few complex decision and limited number of actions.

Disadvantages of Decision tree

- The lack of decision tree is that there is absence of information in its format to take what other combinations of conditions to test.
- A large number of branches with many paths will confuse rather than help in analysis.

PSEUDO-CODES

Pseudo-code is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations. It is used for creating an outline or a rough draft of a program. Pseudo-code summarizes a program's flow, but excludes underlying details. System designers write pseudo-code to ensure that programmers understand a software project's requirements and align code accordingly.

It's simply an implementation of an algorithm in the form of annotations and informative text written in plain English. It has no syntax like any of the programming language and thus can't be compiled or interpreted by the computer.

Example: Program to print Fibonacci up to n numbers.

```
Void function Fibonacci
Get value of n;
Set value of a to 1;
Set value of b to 1;
Initialize i to 0
For (i=0; i< n; i++)
{
        If a greater than b
        {
                Increase b by a;
                Print b;
        }
        Else if b greater than a
        {
                Increase a by b;
                Print a;
        }
}
```

Advantages of Pseudo-code

- Improves the readability of any approach. It's one of the best approaches to start implementation of an algorithm.
- Acts as a bridge between the program and the algorithm or flowchart. Also works as a rough documentation, so the program of one developer can be understood easily when a pseudo code is written out. In industries, the approach of documentation is essential. And that's where a pseudo-code proves vital.
- The main goal of a pseudo code is to explain what exactly each line of a program should do, hence making the code construction phase easier for the programmer.

STRUCTURING SYSTEM DATA REQUIREMENTS

INTRODUCTION

Structuring system and database requirements concentrates on the definition, structure and relationships within data. The characteristics of data captured during data modeling are crucial in the design of databases, programs, computer screens and printed reports. This information is essential in ensuring data integrity in an information system. Moreover data are often the most complex aspects of many modern information systems and are reasonably permanent (in contrast to data flows, which are rather dynamic).

The most common format used for data modeling is entity-relationship diagramming. Data modeling explains the characteristics and structure of data independent of how the data may be stored in computer memories and is usually developed iteratively.

CONCEPTUAL DATA MODELING

A conceptual schema or conceptual data model is a map of concepts and their relationships used for databases. This describes the semantics of an organization and represents a series of assertions about its nature. A conceptual data model is a representation of organizational data. Its purpose is to show as many rules about the meaning and interrelationships among data as possible. It is important that the process, logic and data model descriptions of a system are consistent and complete since each describes different but complementary views of the same information system.

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 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. Conceptual data modeling, using either the ER or UML approach, is particularly useful in the early steps of the database life cycle, which involve requirements analysis and logical design. These two steps are often done simultaneously, particularly when requirements are determined from interviews with end users and modeled in terms of data-to-data relationships and process-to-data relationships. The conceptual data modeling step (ER approach) involves the classification of entities and attributes first, then identification of generalization hierarchies and other abstractions, and finally the definition of all relationships among entities. Relationships may be binary (the most common), ternary, and higher-level n-ary. Data modeling of individual requirements typically involves creating a different view for each end user's requirements. Then the designer must integrate those views into a global schema so that the entire database is pictured as an integrated whole. This helps to eliminate needless redundancy—such elimination is particularly important in logical design. Controlled redundancy can be created later, at the physical design level, to enhance database performance.

The process of conceptual data modeling

This process usually begins with developing a conceptual data model for the existing system. Then a new CDM, which includes all of the data requirements for the new system is built. E-R diagrams can be translated into wide variety of technical architectures for data, such as relational, network and hierarchical.

Deliverables and outcomes

The primary deliverable from the conceptual data modeling is the entity-relationship diagram. Another deliverable from CDM is a full set of entries about data objects to be stored in the project dictionary or repository. The repository is the mechanism to link data, process and logic models of an information system.

GATHERING INFORMATION FOR CONCEPTUAL DATA MODELING

A data model explains what the organization does and what rules govern how work is performed in the organization. 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 specific information requirements in the computer displays, reports or business forms. The bottom-up approach is on the contrary a process of gaining an understanding of data by reviewing specific business documents handled within the system.

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. In later sections of this chapter, we will introduce some specific terminology and constructs used in data modeling. Even without this specific data modeling language, you can begin to understand the kinds of questions that must be answered during requirements determination. These questions relate to understanding the rules and policies by which the area to be supported by the new information system operates. That is, a data model explains what the organization does and what rules govern how work is performed in the organization. You do not, however, need to know (and often can't fully anticipate) how or when data are processed or used to do data modeling.

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. It is this perspective that is typically the basis for a purchased data model.

INTRODUCTION TO E-R MODELING

The basic E-R modeling notation uses three main constructs: data entities, relationships and their associated attributes. An E-R data model is a detailed, logical representation of the data for an organization or for a business area. An entity is a person, place, object, event, or concept in the user environment about which the organization wishes to maintain data. An entity has its own identity, which distinguishes it from other entities. There is an important distinction between entity types and entity instances. An entity type is a collection of entities that share common properties or characteristics. An entity instance is a single occurrence of an entity type. For example there is usually one EMPLOYEE type but there may be hundreds of instances of this entity type stored in a database. Each entity type has a set of attributes associated with it. For example: for an entity STUDENT we have such attributes like: STUDENT NO, NAME, ADDRESS, PHONE NO. Every entity must have an attribute or set of attributes that distinguishes one instance from other instances of the same type - and that is called a candidate key. A primary key is a candidate key that has been selected to be used as the identifier for the entity type. For each entity the name of the primary key is underlined on an E-R diagram.

Chapter 4

DESIGN

Chapter Outline

After studying this chapter, students will be able to understand the:

- Designing Databases
- Designing Forms and Reports
- Designing Interfaces and Dialogues

INTRODUCTION

Systems development is systematic process which includes phases such as planning, analysis, design, deployment, and maintenance. Here we discuss about the designing phase only. It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, we need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently.

System design is the phase that bridges the gap between problem domain and the existing system in a manageable way. This phase focuses on the solution domain, i.e. "how to implement?". It is the phase where the SRS document is converted into a format that can be implemented and decides how the system will operate.

DATABASE DESIGN

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model. Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. It helps produce database systems:

- That meets the requirements of the users
- Have high performance.

The main objectives of database designing are to produce logical and physical designs models of the proposed database system.

- The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.
- The physical data design model involves translating the logical design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

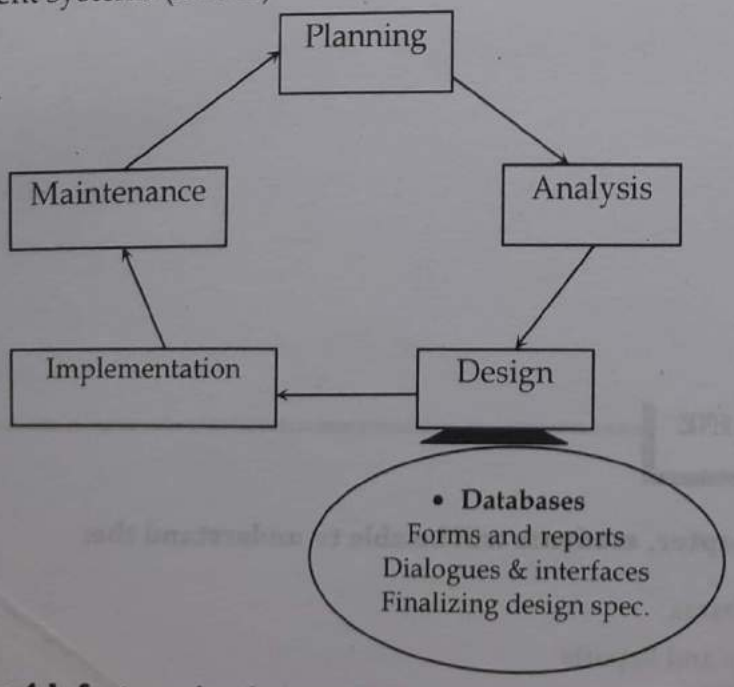


Figure 4.1: Systems development life cycle with design phase highlighted

THE PROCESS OF DATABASE DESIGN

Figure below shows that database modeling and design activities occur in all phases of the systems development process. Here, we discuss methods that help us finalize logical and physical database designs during the design phase. In logical database design, we use a process called normalization, which is a way to build a data model that has the properties of simplicity, non-redundancy, and minimal maintenance.

- Enterprise wide data model (ER with only attributes
- Conceptual data model (ER with only attributes for specific project)

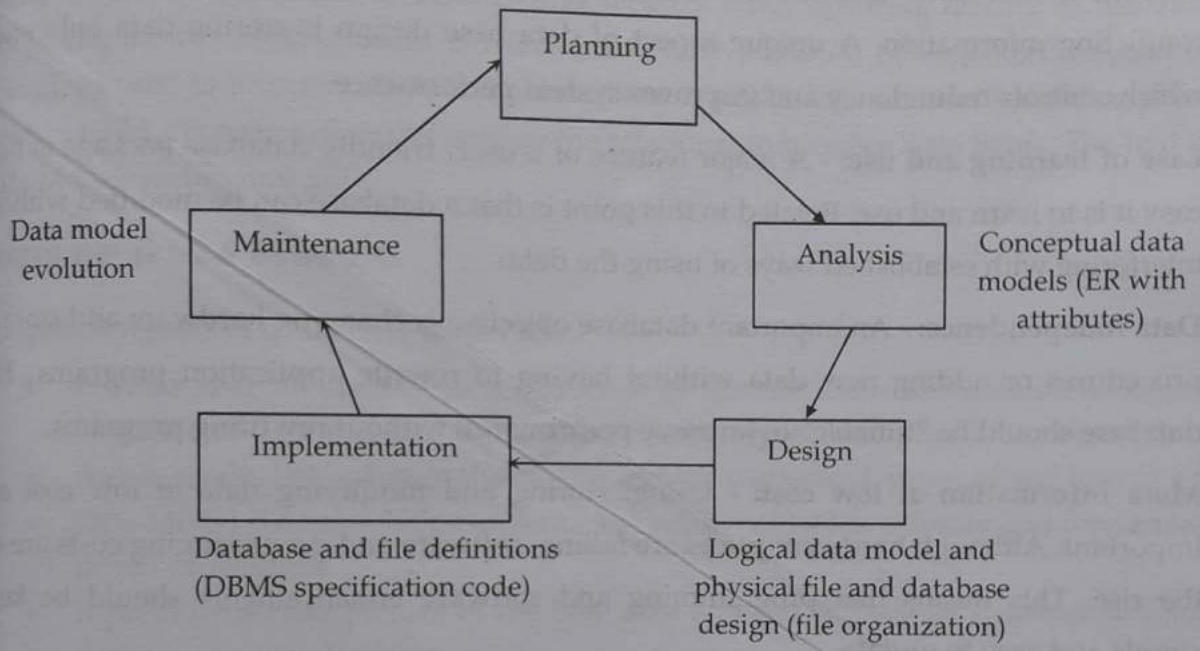


Figure 4.2: Relationship between data modeling and the SDLC

In most situations, many physical database design decisions are implicit or eliminated when we choose the data management technologies to use with the application. We concentrate on those decisions that will make most frequently and use Oracle to illustrate the range of physical database design parameters you must manage. There are four key steps in logical database modeling and design:

1. Develop a logical data model for each known user interface (form and report) for the application using normalization principles.
2. Combine normalized data requirements from all user interfaces into one con-solidated logical database model; this step is called view integration.
3. Translate the conceptual E-R data model for the application or enterprise, developed without explicit consideration of specific user interfaces, into normalized data requirements.
4. Compare the consolidated logical database design with the translated E-R model and produce, through view integration, one final logical database model for the application.

OBJECTIVES OF DATA BASE

The general theme behind a database is to handle information as an integrated whole. There is none of the artificiality that is normally embedded in separate file or applications. A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make information access easy, quick, inexpensive and flexible for the user. In data base design, several specific objectives are considered:

1. **Controlled redundancy: -** Redundant data occupies space and, therefore, is wasteful. If versions of the same data are in different phases of updating, the system often gives conflicting information. A unique aspect of data base design is storing data only once, which controls redundancy and improves system performance.
2. **Ease of learning and use: -** A major feature of a user- friendly database package is how easy it is to learn and use. Related to this point is that a database can be modified without interfering with established ways of using the data.
3. **Data independence: -** An important database objective is changing hardware and storage procedures or adding new data without having to rewrite application programs. The database should be "tunable" to improve performance without rewriting programs.
4. **More information at low cost: -** Using, storing and modifying data at low cost are important. Although hardware prices are falling, software and programming costs are on the rise. This means that programming and software enhancements should be kept simple and easy to update.
5. **Accuracy and integrity: -** The accuracy of a database ensures that data quality and content remain constant. Integrity controls detect data inaccuracies where they occur.
6. **Recovery from failure: -** With multi-user access to a database, the system must recover quickly after it is down with no loss of transactions. This objective also helps maintain data accuracy and integrity.
7. **Privacy and security: -** For data to remain private, security measures must be taken to prevent unauthorized access. Database security means that data are protected from various forms of destruction; users must be positively identified and their actions monitored.
8. **Performance: -** This objective emphasizes response time to inquiries suitable to the use of the data. How satisfactory the response time is depends on the nature of the user-data base dialogue. For example, inquiries regarding airline seat availability should be handled in a few seconds. On the other extreme, inquiries regarding the total sale of a product over the past two weeks may be handled satisfactorily in 50 seconds.

DATA MODELING

Data modeling is the process of creating a data model for the data to be stored in a database. This data model is a conceptual representation of

- Data objects
- The associations between different data objects
- The rules.

Data modeling helps in the visual representation of data and enforces business rules, regulatory compliances, and government policies on the data. Data Models ensure consistency in naming conventions, default values, semantics, and security while ensuring quality of the data. Data model emphasizes on what data is needed and how it should be organized instead of what operations need to be performed on the data. Data Model is like architect's building plan which helps to build a conceptual model and set the relationship between data items. The two types of Data Models techniques are

USES OF DATA MODEL

The primary goals of using data model are:

- Ensures that all data objects required by the database are accurately represented. Omission of data will lead to creation of faulty reports and produce incorrect results.
- A data model helps design the database at the conceptual, physical and logical levels.
- Data Model structure helps to define the relational tables, primary and foreign keys and stored procedures.
- It provides a clear picture of the base data and can be used by database developers to create a physical database.
- It is also helpful to identify missing and redundant data.
- Though the initial creation of data model is labor and time consuming, in the long run, it makes your IT infrastructure upgrade and maintenance cheaper and faster.

TYPES OF DATA MODELS

There are mainly three different types of data models:

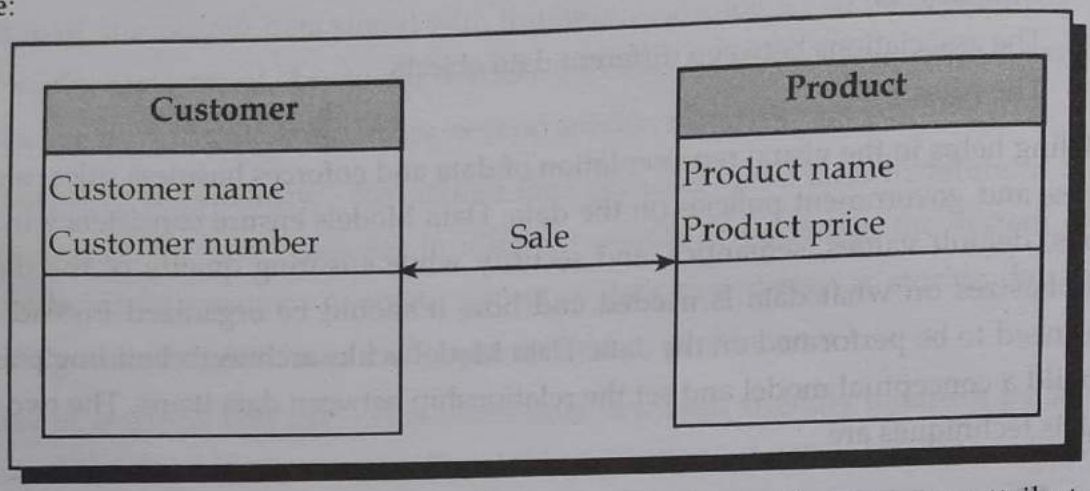
- Conceptual model
- Logical data model
- Physical data model

Conceptual data model

This data model defines **what** the system contains. This model is typically created by business stakeholders and Data Architects. The purpose is to organize scope and define business concepts and rules. The main aim of this model is to establish the entities, their attributes, and their relationships. In this Data modeling level, there is hardly any detail available of the actual database structure. The 3 basic tenants of data model are

- **Entity**: It is real-world thing.
- **Attribute**: It is characteristics or properties of an entity.
- **Relationship**: It is dependency or association between two entities.

Example:



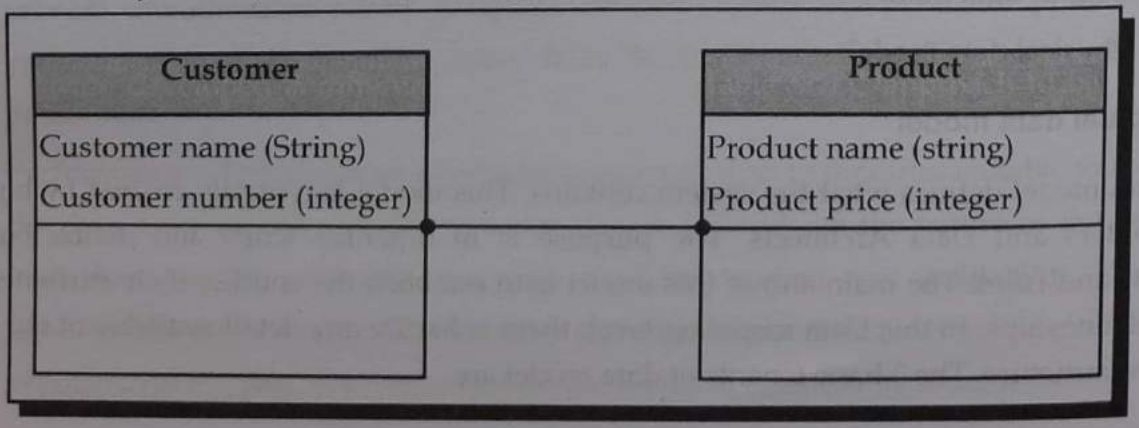
Customer and Product are two entities. Customer number and name are attributes of the Customer entity. Product name and price are attributes of product entity. Sale is the relationship between the customer and product.

Characteristics of a conceptual data model

- Offers organization-wide coverage of the business concepts.
- This type of Data Models are designed and developed for a business audience.
- The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and technology. The focus is to represent data as a user will see it in the "real world."

Logical data model

It defines **how** the system should be implemented regardless of the DBMS. This model is typically created by Data Architects and Business Analysts. The purpose is to developed technical map of rules and data structures. Logical data models add further information to the conceptual model elements. It defines the structure of the data elements and set the relationships between them. The advantage of the Logical data model is to provide a foundation to form the base for the Physical model. However, the modeling structure remains generic. At this Data Modeling level, no primary or secondary key is defined. At this Data modeling level, you need to verify and adjust the connector details that were set earlier for relationships.



Characteristics of a Logical data model

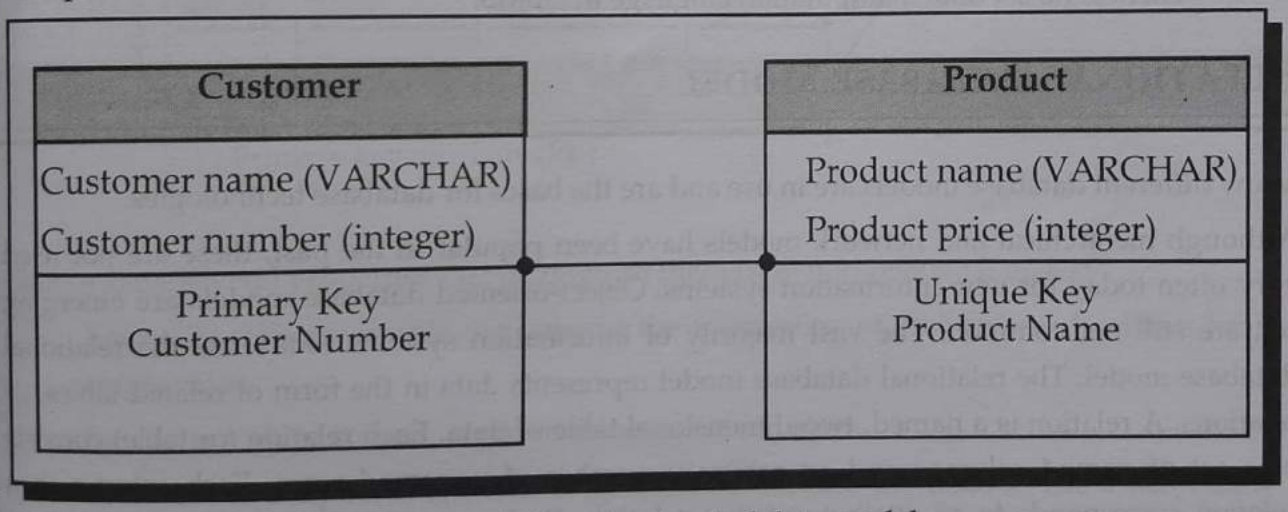
- Describes data needs for a single project but could integrate with other logical data models based on the scope of the project.
- Designed and developed independently from the DBMS.
- Data attributes will have data types with exact precisions and length.
- Normalization processes to the model is applied typically till 3NF.

Physical data model

This data model describes **how** the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database. A Physical Data Model describes the database specific implementation of the data model. It offers an abstraction of the database and helps generate schema. This is because of the richness of meta-data offered by a Physical Data Model.

This type of Data model also helps to visualize database structure. It helps to model database columns keys, constraints, indexes, triggers, and other RDBMS features.

Example:



Characteristics of a physical data model

- The physical data model describes data need for a single project or application though it may be integrated with other physical data models based on project scope.
- Data Model contains relationships between tables that which addresses cardinality and null ability of the relationships.
- Developed for a specific version of a DBMS, location, data storage or technology to be used in the project.
- Columns should have exact data types, lengths assigned and default values.
- Primary and Foreign keys, views, indexes, access profiles, and authorizations, etc. are defined.

Advantages and Disadvantages of Data Model

Advantages of Data model

- The main goal of a designing data model is to make certain that data objects offered by the functional team are represented accurately.
- The data model should be detailed enough to be used for building the physical database.
- The information in the data model can be used for defining the relationship between tables, primary and foreign keys, and stored procedures.
- Data Model helps business to communicate the within and across organizations.
- Data model helps to documents data mappings in ETL process
- Help to recognize correct sources of data to populate the model

Disadvantages of Data model

- To develop Data model one should know physical data stored characteristics.
- This is a navigational system produces complex application development, management. Thus, it requires knowledge of the biographical truth.
- Even smaller changes made in structure require modification in the entire application.
- There is no set data manipulation language in DBMS.

RELATIONAL DATABASE MODEL

Many different database models are in use and are the bases for database technologies.

Although hierarchical and network models have been popular in the past, these are not used very often today for new information systems. Object-oriented database models are emerging but are still not common. The vast majority of information systems today use the relational database model. The relational database model represents data in the form of related tables, or relations. A relation is a named, two-dimensional table of data. Each relation (or table) consists of a set of named columns and an arbitrary number of unnamed rows. Each column in a relation corresponds to an attribute of that relation. Each row of a relation corresponds to a record that contains data values for an entity.

In relational model, the data and relationships are represented by collection of inter-related tables. Each table is a group of column and rows, where column represents attribute of an entity and rows represents records. Relational data model represents the logical view of how data is stored in the relational databases. There exist some concepts related to this, which includes the following terms.

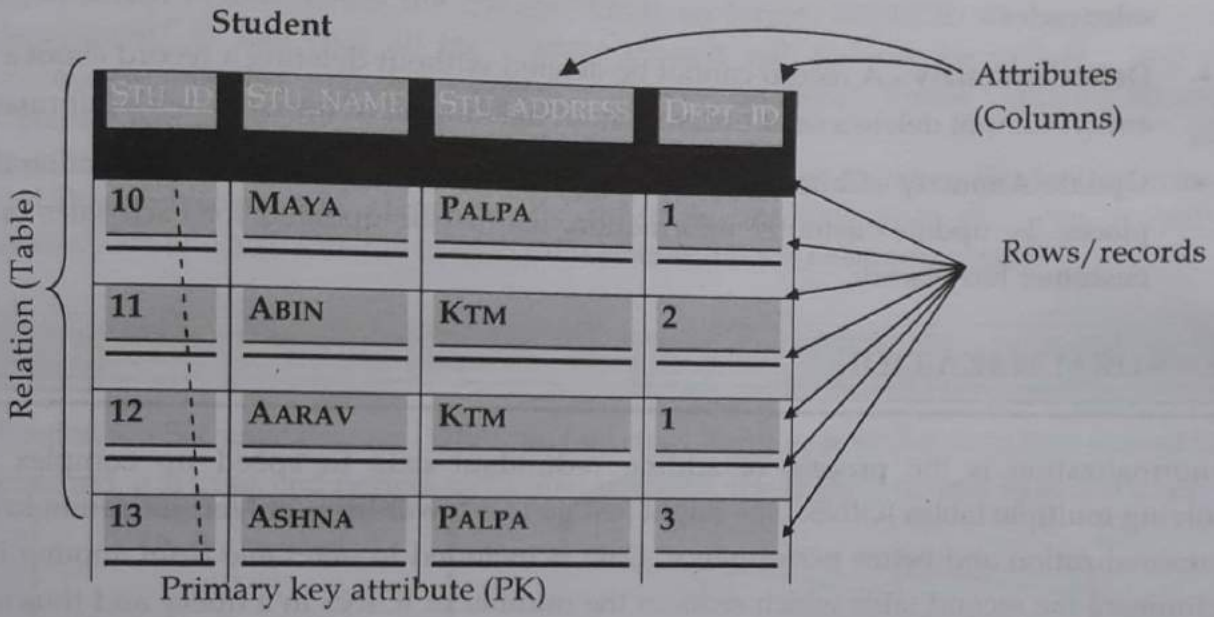
- **Table:** In relational data model, data is stored in the tables. The table consists of a number of rows and columns. Thus, table is used because it can represent the data in the simplest form possible making data retrieval very easy.

  Attribute: Any relation have definite properties that are called as attributes.

- **Tuple:** Rows of table represents the tuple which contains the data records.
- **Domain:** Domain is a set of values which is indivisible i.e. value for each attribute present in the table contains some specific domain in which the value needs to lie. For

Example: The value of date of birth must be greater than zero. As, it cannot be negative. This is called domain of an attribute.

- **Relation:** A relation in relational data model represents the respective attributes and the correlation among them.



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

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

NORMALIZATION

Normalization is the process of or ... it also eliminates redundancy and i... ory. To normalize a databa... en the tables.

Refer teacher note for Normalization

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 name. In general, we will represent each attribute from each normalized relation as one or more fields. The basic decisions we 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.

Calculated Fields: It is common for an attribute to be mathematically related to other data. For example, an invoice may include a total due field, which represents the sum of the amount due on each item on the invoice. A field that can be derived from other database fields is called a calculated field (or a computed field or a derived field). Recall that a functional dependency between attributes does not imply a calculated field. Some database technologies allow you to explicitly define calculated fields along with other raw data fields. 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.

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. In contrast, a physical table is a named set of rows and columns that specifies the fields in each row of the table. A physical table may or may not correspond to one relation. Whereas normalized relations possess properties of well-structured relations, the design of a physical table has two goals different from those of normalization: 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 are stored close together on disk. De-normalization is the process of splitting or combining normalized relations into physical tables based on affinity of use of rows and fields.

DESIGNING FORMS AND REPORTS: INTRODUCTION

The forms are used to present or collect information on a single item, such as a customer, product, or event. Forms can be used for both input and output. Reports, on the other hand, are used to convey information on a collection of items. Form and report design is a key ingredient for successful systems. Because users often equate the quality of a system with the quality of its input and output methods, you can see that the design process for forms and reports is an especially important activity. And because information can be collected and formatted in many ways, gaining an understanding of design dos and don'ts and the tradeoffs between various formatting options is useful for all systems analysts.

Both forms and reports are the product of input and output design and are business document consisting of specified data. The main difference is that forms provide fields for data input but reports are purely used for reading.

DESIGNING FORMS AND REPORTS

Forms are used to present or collect information on a single item such as a customer, product, or event. Forms can be used for both input and output. Reports, on the other hand, are used to convey information on a collection of items. Forms and report design is a key ingredient for successful systems. As users often equate the quality of a system to the quality of its input and out methods, the design process of for forms and reports is an especially important activity.

Forms and reports are identified during requirements structuring. The kinds of forms and reports the system will handle are established as a part of the design strategy formed at the end of the analysis phase of the system development process. Forms and reports are integrally related to various diagrams developed during requirements structuring. For example, every input form will be associated with a data flow entering a process on a DFD, and every output form or report will be a data flow produced by a process on a DFD. Further, the data on all forms and reports must consists of data elements on data stores and on the E-R data model for the application, or must be computed from these data elements.

The Process of Designing Forms and Reports

Designing forms and reports is a user-focused activity that typically follows a prototyping approach. First, you must gain an understanding of the intended user and task objectives by collecting initial requirements during requirements determination. During this process, several questions must be answered. These questions attempt to answer the "who, what, when, where, and how" related to the creation of all forms and reports as given below.

1. Who will use the form or report?
2. What is the purpose of the form or report?
3. When is the form or report needed or used?
4. Where does the form or report need to be delivered and used?
5. How many people need to use or view the form or report?

Gaining an understanding of these questions is a required first step in the creation of any form or report. After collecting the initial requirements, you structure and refine this information into an initial prototype. Structuring and refining the requirements are completely independent of the users, although you may need to occasionally contact users in order to clarify some issue overlooked during analysis. Finally, you ask users to review and evaluate the prototype. After reviewing the prototype, users may accept the design or request that changes be made. If changes are needed, you will repeat the construction-evaluate-refinement cycle until the design is accepted. Usually, several iterations of this cycle occur during the design of a single form or report. As with any prototyping process, you should make sure that these iterations occur rapidly in order to gain the greatest benefits from this design approach.

The initial prototype may be constructed in numerous environments. The obvious choice is to use CASE tool or the standard development tools used within your organization. Often, initial prototypes are simply mock screens that are not working modules or systems. Mock screens can be produced from a word processor; computer graphics design package, or electronic spreadsheet.

Deliverables and Outcomes

Design specifications are the major deliverables and are inputs to the system implementation phase. Design specifications have three sections:

1. **Narrative overview:** This section contains general overview of the characteristics of the target users, tasks, system, and environmental factors in which the form or report will be used. The purpose is to explain to those who will actually develop the final form, why this form exists, and how it will be used so that they can make the appropriate decisions.
2. **Sample design:** This section provides a sample design of the form. This design may be hand drawn using a coding sheet although, in most instances, it is developed using CASE or standard development tool. Using actual development tools allows the design to be more thoroughly tested and assessed.
3. **Testing and usability assessment:** This section provides all testing and usability assessment information. Assessing usability depends on speed, accuracy, and satisfaction.

FORMATTING FORMS AND REPORTS

A wide variety of information can be provided to users of information systems and, as technology continues to evolve, a greater variety of data types will be used. There are numerous guidelines for formatting information.

General Formatting Guidelines

Over the past several years, industry and academic researchers have spent considerable effort investigating how information formatting influences individual task, performance, and perceptions of usability. Through this work, several guidelines for formatting information have emerged as given below:

1. **Meaningful titles:** Titles should be clear and specific describing content and use. They should include version information and current date.
2. **Meaningful information:** Forms should include only necessary information with no need to modification.
3. **Balance of layout:** Adequate spacing and margins should be used. All data and entry fields should be clearly labeled.
4. **Easy navigation system**: Clearly show how to move forward and backward. Clearly show where you are currently. Notify the user when on the last page of a multi-paged sequence.

Assessing Usability

There are many factors to consider when you design forms and reports. The objective for designing forms, reports, and all human-computer interactions is usability. Usability typically refers to the following three characteristics:

1. **Speed.** Can you complete a task efficiently?
2. **Accuracy.** Does the system provide what you expect?
3. **Satisfaction.** Do you like using the system?

In other words, usability means that your designs should assist, not hinder, user performance. Thus, usability refers to an overall evaluation of how a system performs in supporting a particular user for a particular task. In the remainder of this section, we describe numerous factors that influence usability and several techniques for assessing the usability of a design.

General Design Guidelines for Usability of Forms and Reports

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 key strokes, and the system should remember where the user stopped during the last use of the system.

DESIGNING INTERFACES AND DIALOGUES: INTRODUCTION

Interface design focuses on how information is provided to and captured from users; dialogue design focuses on the sequencing of interface displays. Dialogues are analogous to a conversation between two people. The grammatical rules followed by each person during a conversation are analogous to the interface. Thus, the design of interfaces and dialogues is the process of defining the manner in which humans and computers exchange information. A good human computer interface provides a uniform structure for finding, viewing, and invoking the different components of a system.

Measures of Usability

User-friendliness is a term often used, and misused, to describe system usability. Although the term is widely used, it is too vague from a design standpoint to provide adequate information because it means different things to different people. Consequently, most development groups use several methods for assessing usability, including the following considerations:

- **Learnability:** How difficult is it for a user to perform a task for the first time?
- **Efficiency:** How quickly can users perform tasks once they know how to perform them?
- **Error rate:** How many errors might a user encounter, and how easy it is to recover from those errors?
- **Memorability:** How easy is it to remember how to accomplish a task when revisiting the system after some period of time?
- **Satisfaction and aesthetics:** How enjoyable is the system's visual appeal and how enjoyable is the system to use?

In assessing usability, you can collect information by observation, interviews, keystroke capturing, and questionnaires. Time to learn simply reflects how long it takes the average system user to become proficient using the system. Equally important is the extent to which users remember how to use inputs and outputs over time.

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.

Designing Interfaces and Dialogues

Similar to designing forms and reports, the process of designing interfaces and dialogues is a user-focused activity. This means that you follow a prototyping methodology of iteratively collecting information, constructing a prototype, assessing usability, and making refinements. To design usable interfaces and dialogues, you must answer the same who, what, when, where, and how questions used to guide the design of forms and reports. Thus, this process parallels that of designing forms and reports.

Deliverables and outcomes

The deliverable and outcome from system interface and dialogue design is the creation of a design specification. This specification is also similar to the specification produced for form and report designs—with one exception:

1. Narrative overview
2. Sample design
3. Testing and usability assessment

For interface and dialogue designs, one additional subsection is included: a section outlining the dialogue sequence—the ways a user can move from one display to another.

INTERACTION METHODS AND DEVICES

The human-computer interface defines the ways in which users interact with an information system. All human-computer interfaces must have an interaction style and use some hardware devices for supporting this interaction.

METHODS OF INTERACTING

When designing the user interface, the most fundamental decision you make relates to the methods used to interact with the system. Given that there are numerous approaches for designing the interaction, we briefly provide a review of those most commonly used. Our review will examine the basics of five widely used styles: command language, menu, form, object, and natural language. We will also describe several devices for interacting, focusing primarily on their usability for various interaction activities.

COMMAND LANGUAGE INTERACTION

In command language interaction, the user enters explicit statements to invoke operations within a system. This type of interaction requires users to remember command syntax and semantics. For example, to rename a copy of a file called "file.doc" in the current directory as "newfile.doc" at the command prompt within Linux, you would type:

```
$ cp file.doc newfile.doc
```

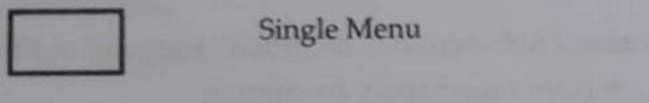
Command language interaction places a substantial burden on the user to remember names, syntax, and operations. Most newer or large-scale systems no longer rely entirely on a command language interface. Yet command languages are good for experienced users, for systems with a limited command set, and for rapid interaction with the system.

MENU INTERACTION

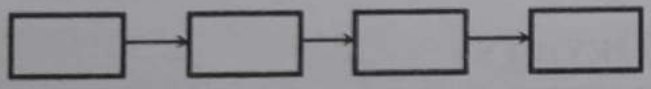
A menu is simply a list of options; when an option is selected by the user, a specific command is invoked or another menu is activated. Menus have become the most widely used interface method because the user only needs to understand simple signposts and route options to effectively navigate through a system.

Menus can differ significantly in their design and complexity. The variation of their design is most often related to the capabilities of the development environment, the skills of the developer, and the size and complexity of the system. For smaller and less complex systems with limited system options, you may use a single menu or a linear sequence of menus. A single menu has obvious advantages over a command language but may provide little guidance beyond invoking the command.

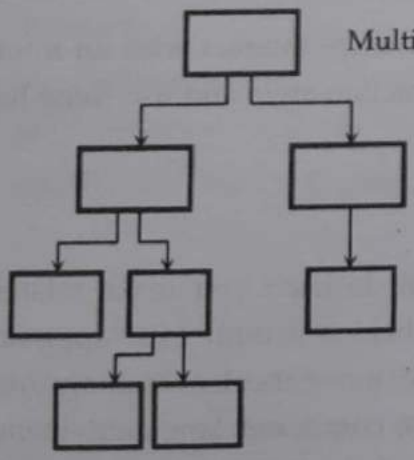
For large and more complex systems, you can use menu hierarchies to provide navigation between menus. These hierarchies can be simple tree structures or variations wherein children menus have multiple parent menus. Some of these hierarchies may allow multilevel traversal. Variations as to how menus are arranged can greatly influence the usability of a system.



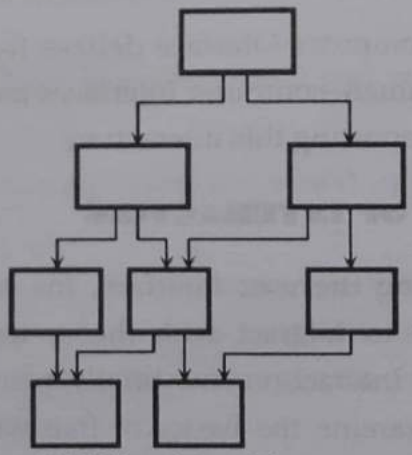
Single Menu



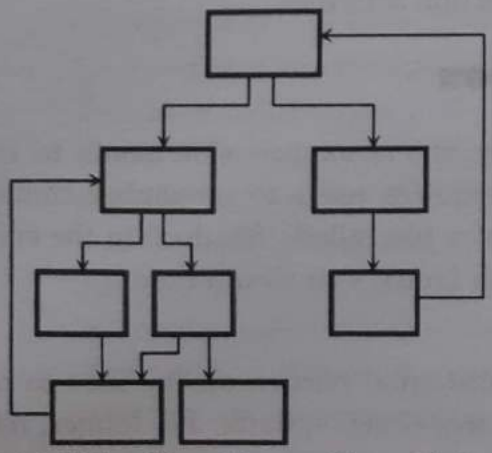
Linear Sequence Menu



Multi-level tree Menu



Multi-level tree Menu with multiple parents



Multi-level tree Menu with multiple parents & multi-level traversal

Guidelines for Menu Design

Working	• Each menu should have a meaningful title • Command verbs should clearly and specifically describe operations • Menu items should be displayed in mixed uppercase and lowercase letters and have a clear, unambiguous interpretation.
Organization	• A consistent organizing principle should be used that relates to the tasks the intended users perform; for example, related options should be grouped together, and the same option should have the same wording and codes each time it appears.
Length	• The number of menu choices should not exceed the length of the screen • Submenus should be used to break up exceedingly long menus
Selection	• Selection and entry methods should be consistent and reflect the size of the application and sophistication of the users. • How the user is to select each option and the consequences of each option should be clear (e.g., whether another menu will appear)
Highlighting	• Highlighting should be minimized and used only to convey selected options (e.g., a check mark) or unavailable options (e.g., dimmed text)

Form Interaction

The premise of form interaction is to allow users to fill in the blanks when working with a system. Form interaction is effective for both the input and presentation of information. An effectively designed form includes a self-explanatory title and field headings, has fields organized into logical groupings with distinctive boundaries, provides default values when practical, displays data in appropriate field lengths, and minimizes the need to scroll windows. Form interaction is the most commonly used method for data entry and retrieval in business-based systems. Using interactive forms, organizations can easily provide all types of information to web surfers.

Object-Based Interaction

The most common method for implementing object based interaction is through the use of icons. Icons are graphic symbols that look like the processing option they are meant to represent. Users select operations by pointing to the appropriate icon with some type of pointing device. The primary advantages to icons are that they take up little screen space and

can be quickly understood by most users. An icon may also look like a button that, when selected or depressed, causes the system to take an action relevant to that form, such as cancel, save, edit a record, or ask for help.

NATURAL LANGUAGE INTERACTION

One branch of artificial intelligence research studies techniques for allowing systems to accept inputs and produce outputs in a conventional language such as English. This method of interaction is referred to as natural language interaction. Presently, natural language interaction is not as viable an interaction style as the other methods presented. Current implementations can be tedious, frustrating, and time consuming for the user and are often built to accept input in narrowly constrained domains (e.g., database queries). Natural language interaction is being applied within both keyboard and voice entry systems.

DESIGNING INTERFACES

User interface is the front-end application view to which user interacts in order to use the software. User can manipulate and control the software as well as hardware by means of user interface. Today, user interface is found at almost every place where digital technology exists, right from computers, mobile phones, cars, music players, airplanes, ships etc. User interface is part of software and is designed such a way that it is expected to provide the user insight of the software. UI provides fundamental platform for human-computer interaction. UI can be graphical, text-based, audio-video based, depending upon the underlying hardware and software combination. UI can be hardware or software or a combination of both. The software becomes more popular if its user interface is:

- Attractive
- Simple to use
- Responsive in short time
- Clear to understand
- Consistent on all interfacing screens

DESIGNING LAYOUTS

To ease user training and data recording, you should use standard formats for computer-based forms and reports similar to those used on paper-based forms and reports for recording or reporting information. This form has several general areas common to most forms:

- Header information
- Sequence and time-related information
- Instruction or formatting information
- Body or data details
- Totals or data summary
- Authorization or signatures
- Comments

In many organizations, data are often first recorded on paper-based forms and then later recorded within application systems. When designing layouts to record or display information on paper-based forms, you should try to make both as similar as possible. Additionally, data entry displays should be consistently formatted across applications to speed data entry and reduce errors. Another concern when designing the layout of computer-based forms is the design of between-field navigation. Because you can control the sequence for users to move between fields, standard screen navigation should flow from left to right and top to bottom just as when you work on paper-based forms.

When designing the navigation procedures within your system, flexibility and consistency are primary concerns. Users should be able to freely move forward and backward or to any desired data entry fields. Users should be able to navigate each form in the same way or in as similar a manner as possible. Additionally, data should not usually be permanently saved by the system until the user makes an explicit request to do so. This allows the user to abandon a data entry screen, back up, or move forward without adversely affecting the contents of the permanent data.

Designing Dialogues

The process of designing the overall sequences that users follow to interact with an information system is called dialogue design. A dialogue is the sequence in which information is displayed to and obtained from a user. As the designer, our role is to select the most appropriate interaction methods and devices (described earlier) and to define the conditions under which information is displayed to and obtained from users. The dialogue design process consists of three major steps:

- Designing the dialogue sequence
- Building a prototype
- Assessing usability

For a dialogue to have high usability, it must be consistent in form, function, and style. All other rules regarding dialogue design are mitigated by the consistency guideline. For example, the effectiveness of how well errors are handled or feedback is provided will be significantly influenced by consistency in design. If the system does not consistently handle errors, the user will often be at a loss as to why certain things happen.

Guidelines for the Design of Human–Computer Dialogues

Guideline	Explanation
Consistency	Dialogues should be consistent in sequence of actions, keystrokes, and terminology (e.g., the same labels should be used for the same operations on all screens, and the location of the same information should be the same on all displays).
Shortcuts and Sequence	Allow advanced users to take shortcuts using special keys (e.g., CTRL-C to copy highlighted text). A natural sequence of steps should be followed (e.g., enter first name before last name, if appropriate).

Feedback	Feedback should be provided for every user action (e.g., confirm that a record has been added, rather than simply putting another blank form on the screen).
Closure	Dialogues should be logically grouped and have a beginning, middle, and end (e.g., the last in the sequence of screens should indicate that there are no more screens).
Error Handling	All errors should be detected and reported; suggestions on how to proceed should be made (e.g., suggest why such errors occur and what user can do to correct the error). Synonyms for certain responses should be accepted (e.g., accept either "t," "T," or "TRUE").
Reversal	Dialogues should, when possible, allow the user to reverse actions (e.g., undo a deletion); data should not be deleted without confirmation (e.g., display all the data for a record the user has indicated is to be deleted).
Control	Dialogues should make the user (especially an experienced user) feel in control of the system (e.g., provide a consistent response time at a pace acceptable to the user).
Ease	It should be a simple process for users to enter information and navigate between screens (e.g., provide means to move forward, backward, and to specific screens, such as first and last).

DESIGNING THE DIALOGUE SEQUENCE

Your first step in dialogue design is to define the sequence. In other words, you must first gain an understanding of how users might interact with the system. This means that you must have a clear understanding of user, task, technological, and environmental characteristics when designing dialogues. Suppose that the marketing manager at Pine Valley Furniture (PVF) wants sales and marketing personnel to be able to review the year-to-date transaction activity for any PVF customer. After talking with the manager, you both agree that a typical dialogue between a user and the Customer Information System for obtaining this information might proceed as follows:

- Request to view individual customer information
- Specify the customer of interest
- Select the year-to-date transaction summary display
- Review customer information
- Leave system

As a designer, once you understand how a user wishes to use a system, you can then transform these activities into a formal dialogue specification.

DESIGNING INTERFACES AND DIALOGUES IN GRAPHICAL ENVIRONMENTS

Graphical user interface (GUI) environments have become the de facto standard for human-computer interaction. Although all of the interface and dialogue design guidelines presented previously apply to designing GUIs, additional issues that are unique to these environments must be considered. Here, we briefly discuss some of these issues.

Graphical interface Design issues

When designing GUIs for an operating environment such as Microsoft Windows or the Apple OSX, numerous factors must be considered. Some factors are common to all GUI environments, whereas others are specific to a single environment. We will not, however, discuss the subtleties and details of any single environment. Instead, our discussion will focus on a few general truths that experienced designers mention as critical to the design of usable GUIs. In most discussions of GUI programming, two rules repeatedly emerge as composing the first step to becoming an effective GUI designer:

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

The first step should be an obvious one. The greatest strength of designing within a standard operating environment is that standards for the behavior of most system operations have already been defined. For example, how you cut and paste, set up your default printer, design menus, or assign commands to functions have been standardized both within and across applications. This allows experienced users of one GUI-based application to easily learn a new application. Thus, in order to design effective interfaces in such environments, you must first understand how other applications have been designed so that you will adopt the established standards for "look and feel." Failure to adopt the standard conventions in a given environment will result in a system that will likely frustrate and confuse users.

MULTIPLE CHOICE QUESTIONS

1. Who are the people that actually use the system to perform or support the work to be completed?
   a. System analysts
   b. system designers
   c. System builders
   d. none of the above
2. Form of dependency in which set of attributes that is neither a subset of any of keys nor candidate key is classified as .....
   a. Transitive dependency
   b. full functional dependency
   c. Partial dependency
   d. prime functional dependency

Chapter 5

IMPLEMENTATION AND MAINTENANCE

CHAPTER OUTLINE

After studying this chapter, students will be able to understand the:

- System Implementation
- System Maintemtnce

INTRODUCTION

System Implementation is the process of converting the physical system specification into working and reliable software and hardware document the work that has been done and provide help for current an 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. As each program module is produced, it can be tested individually, then as part of a larger program, and then as part of a larger system.

Although testing is done during implementation, we must begin planning for testing earlier in the project.

Planning involves determining what needs to be tested and collecting test data. This is often done during the analysis phase because testing requirements are related to system requirements.

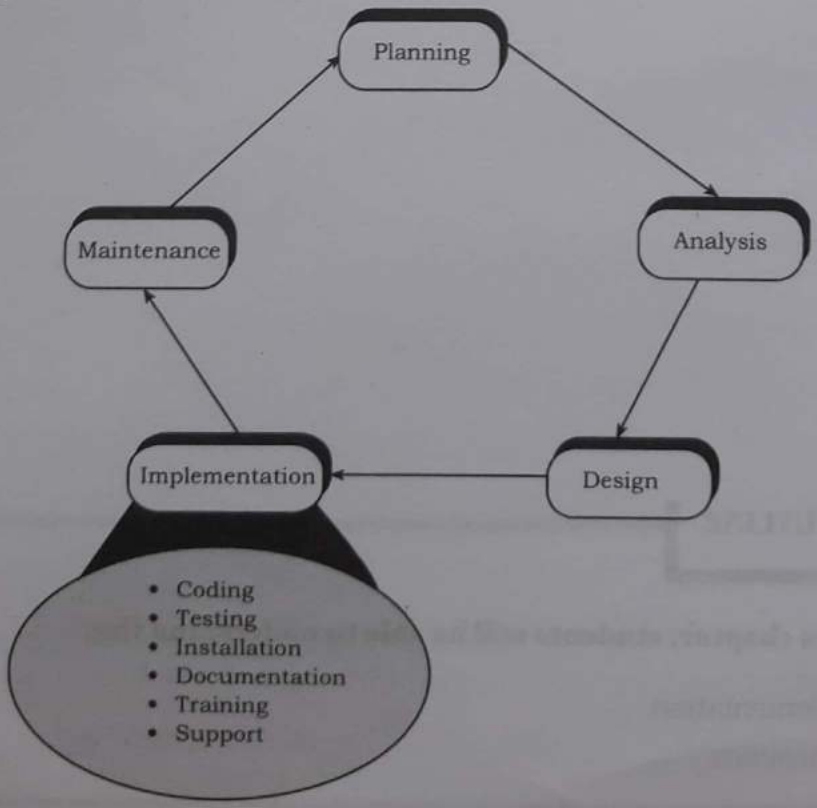
Installation is the process during which the current system is replaced by the new system. This includes conversion of existing data, software, documentation, and work procedures to those consistent with the new system. Users must give up the old way of doing their jobs, whether manual or automated, and adjusted to accomplishing the same task with the new system.

Although 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 tom the new projects, analyst need to prepare documents that will reveal all of the important information that are accumulated about this system during its development and its implementation.

Larger organizations also tend to provide training and support to computer users throughout the organization. Some of the training and support is very specific to particular application systems, whereas the rest is general to particular operating systems or off-the-shelf software packages.



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 it's design and development and meets the user requirements effectively and efficiently with handling all the exceptional and boundary cases. Software testing is a 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.

APPLICATIONS OF SOFTWARE TESTING

- Cost Effective Development - Early testing saves both time and cost in many aspects, however reducing the cost without testing may result in improper design of a software application rendering the product useless.
- Product Improvement - During the SDLC phases, testing is never a time-consuming process. However diagnosing and fixing the errors identified during proper testing is a time-consuming but productive activity.
- Test Automation - Test Automation reduces the testing time, but it is not possible to start test automation at any time during software development. Test automaton should be started when the software has been manually tested and is stable to some extent. Moreover, test automation can never be used if requirements keep changing.
- Quality Check - Software testing helps in determining following set of properties of any software such as
  - Functionality
  - Reliability
  - Usability
  - Efficiency
  - Maintainability
  - Portability

DIFFERENT TYPES OF TESTS

1. **Inspections:** A testing technique in which participants examine program code for predictable language specific errors. Syntax, grammar, and some other routine errors can be checked by automated inspection software. 60 to 90 percent of all software effects as well as provide programmers with feedback that enables them to avoid making the same types of errors in future work.
2. **Desk checking:** Desk checking is an informal manual test that programmers can use to verify coding and algorithm logic before a program launch. This enables them to spot errors that might prevent a program from working as it should. Modern debugging tools

make desk checking less essential than it was in the past, but it can still be a useful way of spotting logic errors.

It refers to the manual approach of reviewing source code (sitting a desk), rather than running it through a debugger or another automated process. In some cases, a programmer may even use a pencil and paper to record the process and output of functions within a program. For example, the developer may track the value of one or more variables in a function from beginning to end. Manually going through the code line-by-line may help a programmer catch improper logic or inefficiencies that a software debugger would not.

While desk checking is useful for uncovering logic errors and other issues within a program's source code, it is time-consuming and subject to human error. Therefore, an IDE or debugging tool is better suited for detecting small problems, such as syntax errors. It is also helpful to have more than one developer desk check a program to reduce the likelihood of overlooking errors in the source code.

A testing technique in which the programmer or someone else who understands the logic of the program works through the code with a paper and pencil. The reviewer acts as the computer, mentally checking each step and its results for the entire set of computer instructions.

3. **Unit testing:** It focuses on smallest unit of software design. In this we test an individual unit or group of inter related units. It is often done by programmer by using sample input and observing its corresponding outputs. Automated technique whereby each module is tested alone in an attempt to discover any errors that may exist in the module's code.

4. **Integration testing:** The process of bringing together more than one modules that a program comprises for testing purposes. Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

5. **System testing:** The process of bringing together of all of the programs that a system comprises for testing purposes. System testing is the testing to ensure that by putting the software in different environments (e.g., Operating Systems) it still works. System testing is done with full system implementation and environment. Programs are typically integrated in a top-down incremental fashion. The system can be tested in two ways:

   i. **Black box testing**: Black box testing 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 called white box testing because the test cases are totally visible to the general users and they can also make test cases.

S.N.	Black Box Testing	White Box Testing
1	Black box testing is the Software testing method which is used to test the software without knowing the internal structure of code or program.	White box testing is the software testing method in which internal structure is being known to tester who is going to test the software.
2	This type of testing is carried out by testers.	Generally, this type of testing is carried out by software developers.
3	Implementation Knowledge is not required to carry out Black Box Testing.	Implementation Knowledge is required to carry out White Box Testing.
4	Programming Knowledge is not required to carry out Black Box Testing.	Programming Knowledge is required to carry out White Box Testing.
5	Testing is applicable on higher levels of testing like System Testing, Acceptance testing.	Testing is applicable on lower level of testing like Unit Testing, Integration testing.
6	Black box testing means functional test or external testing.	White box testing means structural test or interior testing.
7	In Black Box testing is primarily concentrate on the functionality of the system under test.	In White Box testing is primarily concentrate on the testing of program code of the system under test like code structure, branches, conditions, loops etc.
8	The main aim of this testing to check on what functionality is performing by the system under test.	The main aim of White Box testing to check on how System is performing.
9	Black Box testing can be started based on Requirement Specifications documents.	White Box testing can be started based on Detail Design documents.
10	The Functional testing, Behavior testing, Close box testing is carried out under Black Box testing, so there is no required of the programming knowledge.	The Structural testing, Logic testing, Path testing, Loop testing, Code coverage testing, Open box testing is carried out under White Box testing, so there is compulsory to know about programming knowledge.

6. **Stub testing:** A technique used in testing modules, especially where modules are written and tested in a top down fashion, where a few lines of codes are used to substitute for subordinate modules. Top-level modules contain many call to subordinate modules, we may wonder how they can be tested if the lower level modules haven't been written yet. This is called stub testing.

7. **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.

   i. **Alpha testing:** User testing of a completed information system using simulated data. The types of tests performed during alpha testing include the following:

      a. **Recovery testing:** Forces the software to fail in order to verify that recovery is properly performed.

      b. **Security testing:** Verifies that protection mechanisms built into the system will protect it from improper penetration.

      c. **Stress testing:** Tries to break the system (eg: what happens when a record is written to the database with incomplete information or what happens under extreme online transaction loads or with a large number of concurrent users).

      d. **Performance testing:** Determines how the system performs in the range of possible environments in which it may be used (eg: different hardware configurations, networks, operating systems).

   iii. **Beta testing:** User testing of a completed information system using real data in the real user environment. The intent of the beta test is to determine whether the software, documentation, technical support and training activities work as intended. Beta testing can be viewed as a rehearsal of the installation phase.

INSTALLATION

The process of moving from the current information system to the new one is called installation. In this activity, all the users must give up their reliance on the current system and begin to rely on new system. There are different ways and an 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 factor. Different ways

of installation are:

1. **Direct installation**: Changing over from the old information system to a new one by turning off the old system when the new one is turned on. Any errors resulting from the new system will have a direct impact on the users. 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. Direct installation can be very risky. Direct installation requires a complete installation of the whole system.

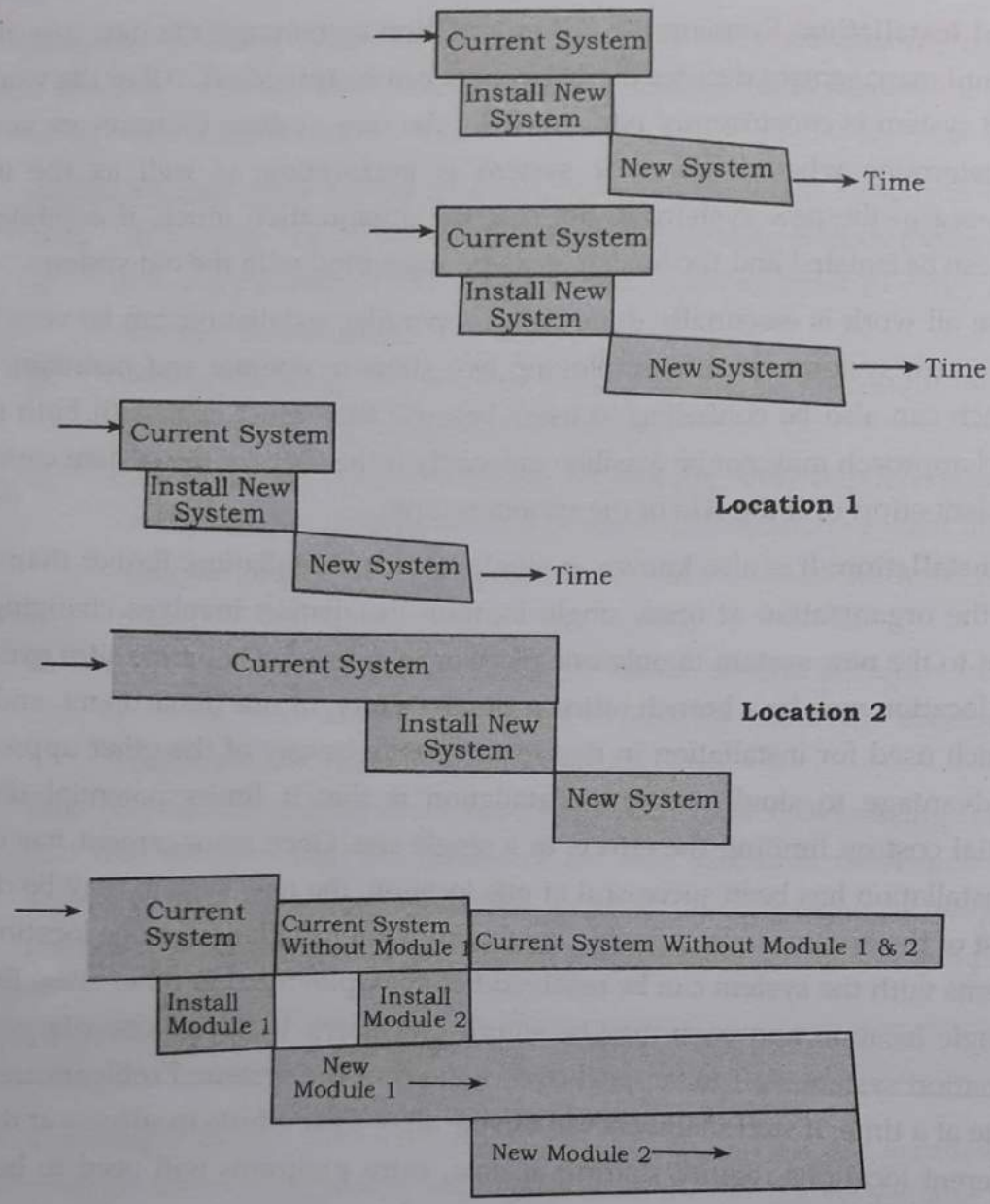
   For a large system, this may mean a long time until the new system can be installed, thus delaying system benefits or even missing the opportunities that motivated the system request. It is the least expensive installation method, and it creates considerable interest in making the installation a success.

2. **Parallel installation:** Running the old information system and the new one at the same time until management decides the old system can be turned off. All of the work done by the old system is concurrently performed by the new system. Outputs are compared 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 two staffs to operate and maintain. A parallel approach can also be confusing to users because they must deal with both systems. A parallel approach may not be feasible, especially if the users of the system cannot tolerate redundant effort or if the size of the system is large.

3. **Pilot installation:** It is also known as single-location installation. Rather than converting all of the organization at once, single location installation involves changing form 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. Problems with the system can be resolved before deployment to other sites. Even though the single location approach may be simpler for users, it still places a large burden on information system staff to support two versions of the system. Problems are isolated at one site at a time. If staff members can devote all of their efforts to success at the pilot site. If different locations require sharing of data, extra programs will need to be written to synchronize the current and new systems.

4. **Phased installation:** It is also called staged installation. Different parts of the old and new systems are used in cooperation until the whole new system is installed. By converting gradually, the organization's risk is spread out over time and place. Also phased installation allows for some benefits from the new system before the whole system is ready. For example, a 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 that pieces of the old system cannot be incrementally replaced so this strategy is not feasible. 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.



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:

1. It solves the problem of indispensability 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.
2. 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.

3. 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

1. **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 requirement, and the form and type 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 program.
   - 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 techniques such as dataflow and entity-relationship diagrams.
2. **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.

TRAINING AND SUPPORTING USERS

The type of training needed will vary by system type and user expertise. Types of training methods are:

- Resident expert
- 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 provide users access to information on new releases, bugs and tips for more effective 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.

SYSTEM MAINTENANCE

Once software system is put into use or installed, new requirements emerges and existing requirements change as the business running that system changes and 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 results obtained from the evaluation process help the organization to determine whether its information systems are effective and efficient or otherwise. The process of monitoring, evaluating, and modifying of existing information systems to make required or desirable improvements may be termed as System Maintenance. System maintenance is an ongoing activity, which covers a wide variety of activities, including removing program and design errors, updating documentation and test data and updating user support.

System maintenance is the general process of changing a system after it has been delivered. The changes may be Simple changes to correct coding errors, more extensive changes to correct design errors or significant enhancement to correct specification errors or accommodate new requirements.

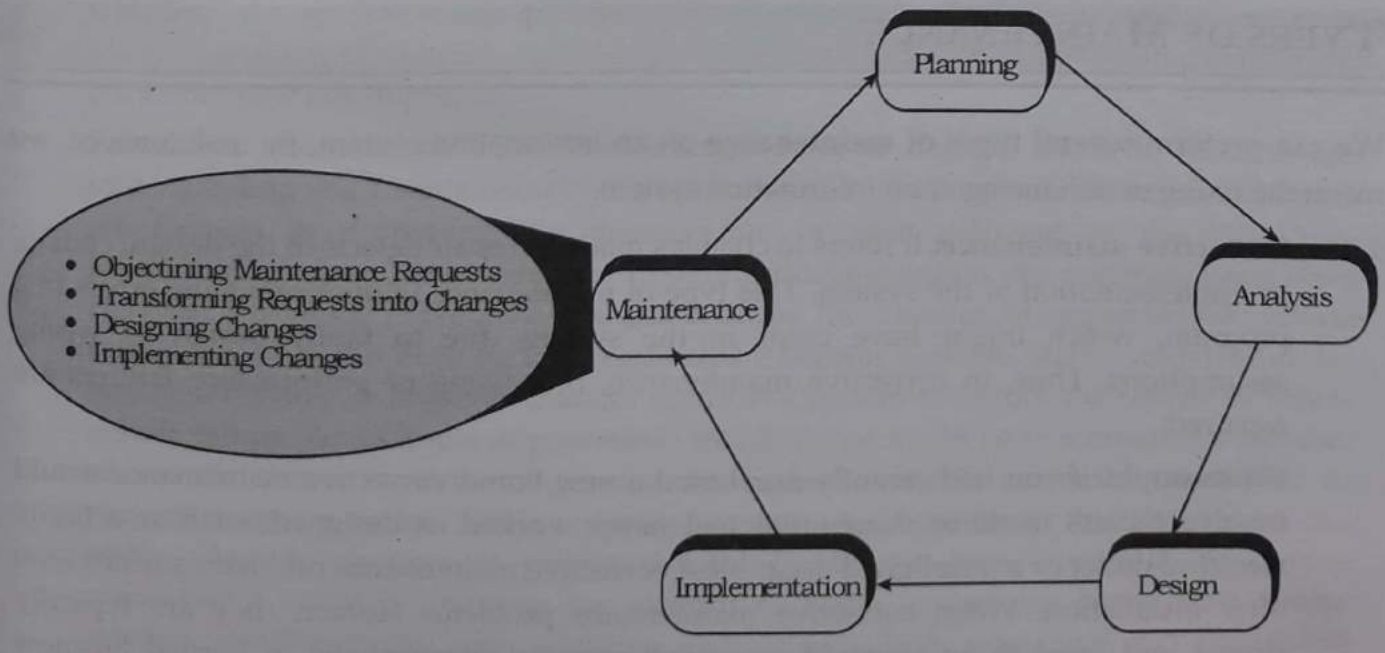
There are four major activities occur within maintenance:

1. Obtaining maintenance requests
2. Transforming requests into changes
3. Designing changes
4. Implementing changes

1. **Obtaining Maintenance Requests:** In this step a formal process be established whereby users can submit system change requests. 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.
2. **Transforming Request into Changes:** Once a request is received, analysis must be performed to identify the of the scope of the request. It must be determined how the request will affect the current system and how long such a project will take. As with the initial development of a system, the size of a maintenance request can be analyzed for risk and feasibility.
3. **Designing changes:** Next, a change request can be transformed into a formal design change, which can then be fed into the maintenance implementation phase.
4. **Implementing Changes:** In this activity once the change design is approved, proposed changes are implemented in respective components of the system.

ANALOGY BETWEEN SDLC AND SYSTEM MAINTENANCE

Many similarities exist between the SDLC and the activities within the maintenance process. The first phase of the SDLC—planning—is analogous to the maintenance process of obtaining a maintenance request. The SDLC analysis phase is analogous to the maintenance process of transforming requests into a specific system change. The SDLC design phase, of course, equates to the designing changes process (step 3). Finally, the SDLC phase implementation equates to step 4, implementing changes. This similarity between the maintenance process and the SDLC is no accident. The concepts and techniques used to initially develop a system are also used to maintain it.

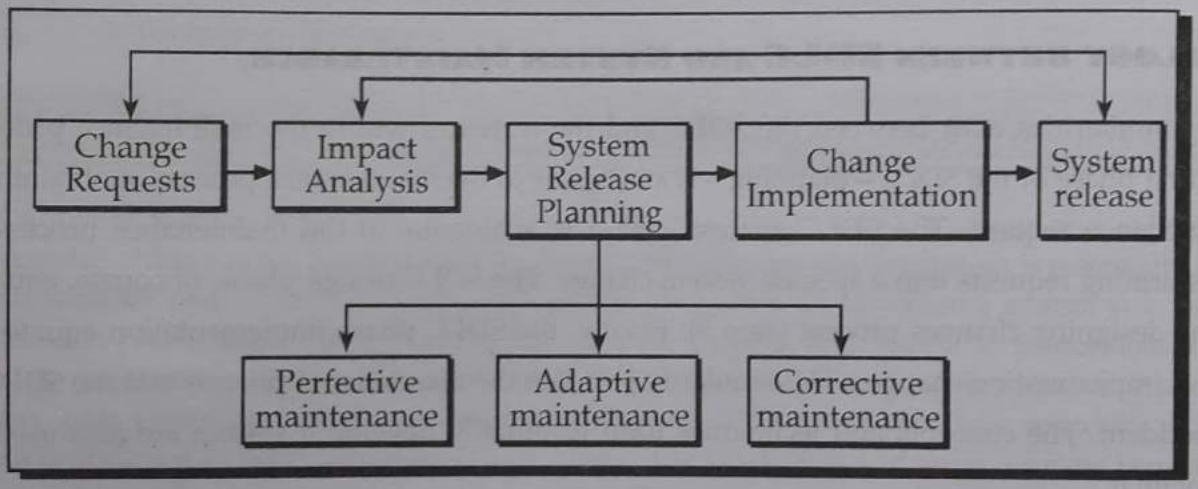


CONDUCTING MAINTENANCE

A significant portion of the expenditures for information systems 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.

Maintenance processes vary considerably depending on the type of system being maintained, the development process used in the organization and the people involved in the process. However, the maintenance process is triggered by a set of change requests from system users, management or customers. The cost and impact of these changes are assed to see how much f the system is affected by the change and how much it might costs to implement the change. If the proposed changes are accepted, new release of the system is planned. During release planning, all the proposed changes are considered. A decision is then made on which changes to implement in the next version of the system. The changes are implemented and validated.

The Maintenance Process



TYPES OF MAINTENANCE

We can perform several types of **maintenance** on an information system. By maintenance, we mean the fixing or enhancing of an information system.

1. **Corrective maintenance:** It refers to changes made to repair defects in the design, coding, or implementation of the system. This type of maintenance implies removing errors in a program, which might have crept in the system due to faulty design or wrong assumptions. Thus, in corrective maintenance, processing or performance failures are repaired.

   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.

2. **Adaptive maintenance:** In adaptive maintenance, program functions are changed to enable the information system to satisfy the information needs of the user. It 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. Contrary to corrective maintenance, adaptive maintenance is generally a small part of an organization's maintenance effort, but it adds value to the organization.

   This type of maintenance may become necessary because of organizational changes which may include:

   a. Change in the organizational procedures,
   b. Change in organizational objectives, goals, policies, etc.
   c. Change in forms,
   d. Change in information needs of managers.
   e. Change in system controls and security needs, etc.

3. **Perfective maintenance:** Perfective maintenance means adding new programs or modifying the existing programs to enhance the performance of the information system. This type of maintenance undertaken to respond to user's additional needs which may be due to the changes within or outside of the organization. Outside changes are primarily environmental changes, which may in the absence of system maintenance, render the information system ineffective and inefficient. These environmental changes include:

   a) Changes in governmental policies, laws, etc.,
   b) Economic and competitive conditions, and
   c) New technology.

   It 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.

4. **Preventive maintenance:** Preventive changes refer to changes made to increase the understanding and maintainability of your software in the long run. Preventive changes are focused in decreasing the deterioration of your software in the long run. Restructuring, optimizing code and updating documentation are common preventive changes. Executing preventive changes reduces the amount of unpredictable effects software can have in the long term and helps it become scalable, stable, understandable and maintainable. It **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. In our home example, preventive maintenance could be painting the exterior to better protect the home from severe weather conditions. As with adaptive maintenance, both perfective and preventive maintenance are typically a much lower priority than corrective maintenance.

THE COST OF MAINTENANCE

The cost of maintenance represent a large proportion of the budget of most organization that use the software system. 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 more older legacy systems that require more and more maintenance. More maintenance means more maintenance work for programmers. In addition, about one-third of the costs of establishing and keeping a presence on the Web go to programming maintenance.

Maintenance cost as a proportion cost of development costs vary from one application domain to another. For business application system, maintenance cost were broadly comparable with system development cost. For embedded real time systems, maintenance cost may be up to four times higher than development cost.

So while developing system, good software engineering techniques such as precise specification, use of object oriented development and configuration management are used that contribute to maintenance cost reduction.

FACTORS INFLUENCING MAINTENANCE COST

Numerous factors influence the **maintainability** of a system. These factors, or cost elements, determine the extent to which a system has high or low maintain- ability. Of these factors, three are most significant: the number of latent defects, the number of customers, and documentation quality. The others—personnel, tools, and software structure—have noticeable, but less, influence.

- **Latent defects:** This is the number of unknown errors existing in the system after it is installed. Because corrective maintenance accounts for most maintenance activity, the number of latent defects in a system influences most of the costs as- sociated with maintaining a system.
- **Number of customers for a given system:** In general, the greater the number of customers, the greater the maintenance costs. For example, if a system has only one customer, problem and change requests will come from only one source. Also, training, error reporting, and support will be simpler. Maintenance requests are less likely to be contradictory or incompatible.
- **Quality of system documentation:** Without quality documentation, maintenance efforts can increase exponentially. High-quality documentation leads reduction in the sys- tem maintenance effort when compared with average-quality documentation. In other words, quality documentation makes it easier to find code that needs to be changed and to understand how the code needs to be changed. Good documentation also explains why a system does what it does and why alternatives were not feasible, which saves wasted maintenance efforts.

- **Maintenance personnel:** In some organizations, the best programmers are assigned to maintenance. Highly skilled programmers are needed because the maintenance programmer is typically not the original programmer and must quickly understand and carefully change the software.
- **Tools:** Tools that can automatically produce system documentation where none exists can also lower maintenance costs. Also, tools that can automatically gen- erate new code based on system specification changes can dramatically reduce maintenance time and costs.
- **Well-structured programs:** Well-designed system are easier to understand and fix.

MANAGING MAINTENANCE

As maintenance activities consume more and more of the systems development bud- get, 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. We will address this concern by dis- cussing several topics related to the effective management of systems maintenance.

MANAGING MAINTENANCE PERSONNEL

One concern with managing maintenance relates to personnel management. Historically, many organizations had a "maintenance group" that was separate from the "development group." With the increased number of maintenance personnel, the development of formal methodologies and tools, changing organizational forms, end-user computing, and the widespread use of very high-level languages for the development of some systems, organizations have rethought the organization of maintenance and development personnel. In other words, should the maintenance group be separated from the development group? Or should the same people who build the system also maintain it? A third option is to let the primary end users of the system in the functional units of the business have their own maintenance personnel.

MEASURING MAINTENANCE EFFECTIVENESS

A second management issue is the mea- surement of maintenance effectiveness. As with the effective management of person- nel, the measurement of maintenance activities is fundamental to understanding the quality of the development and maintenance efforts. To measure effectiveness, you 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 system.

Tracking the types of failures also provides important management information for future projects. For example, if a higher frequency of errors occurs when a partic- ular development environment is used, such information can help guide personnel assignments; training courses; or the avoidance of a particular package, language, or environment during future development. The primary lesson here is that without measuring and tracking maintenance activities, you cannot gain the knowledge to improve or know how well you are doing relative to the past. To effectively manage and to continuously improve, you must measure and assess performance over time.

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.

Configuration Management An aspect of managing maintenance is **configuration management**, which is the process of ensuring that only authorized changes are made to a system. Configuration management (CM) is a systems engineering process for establishing and maintaining consistency of a product's performance, functional, and physical attributes with its requirements, design, and operational information throughout its life. A configuration management is used to keep track of an organization's hardware, software and related information. This includes software versions and updates installed on the organization's computer systems. CM also involves logging the network addresses belonging to the hardware devices used. Software is available for all of these tracking tasks. 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 pro- cess can reside in the library. Before any code can be checked back in to the librar- ian, the code must pass the quality control procedures, testing, and documentation standards established by the organization.

When various maintenance personnel working on different maintenance tasks complete each task, the librarian notifies those still working that updates have been made to the baseline modules. This means that all tasks being worked on must now incorporate the latest baseline modules before being approved for check-in. Following a formal process of checking modules out and in, a system librarian helps to ensure that only tested and approved modules become part of the baseline System. It is also the responsibility of the librarian to keep copies of all prior versions of all system modules, including the **build routines** needed to construct *any version of*

the system that has *ever* existed. It may be important to reconstruct old versions of the system if new ones fail or to support users that cannot run newer versions on their computer system.

Special software systems have been created to manage system configuration and version control activities (see the box "Configuration Management Tools"). This software is becoming increasingly necessary as the change control process becomes ever more complicated in organizations deploying several different networks, oper-ating systems, languages, and database management systems in which there may be many concurrent versions of an application, each for a different platform. One func-tion of this software is to control access to modules in the system library. Each time a module is checked out or in, this activity is recorded, after being authorized.

Role of CASE in Maintenance

In traditional systems development, much of the time is spent on coding and testing. When software changes are approved, code is first changed and then tested. Once the functionality of the code is assured, the documentation and specification documents are updated to reflect system changes. Over time, the process of keeping all system documentation "current" can be a very boring and time-consuming activity that is often neglected. This neglect makes future maintenance by the same or *differ- ent* programmers difficult at best.

A primary objective of using automated tools for systems development and maintenance is to radically change the way in which code and documentation are modified and updated. When using an integrated development environment, ana- lysts maintain design documents such as data flow diagrams and screen designs, not source code. In other words, design documents are modified and then code generators automatically create a new version of the system from these updated designs. Also, because the changes are made at the design specification level, most documentation changes, such as an updated data flow diagram, will have already been completed during the maintenance process itself. Thus, one of the biggest advantages to using automated tools, for example, is its usefulness in system maintenance.

In addition to using general automated tools for maintenance, two special purpose tools, reverse engineering and reengineering tools, are used to maintain older systems that have incomplete documentation. These tools are often referred to as *design recovery* tools because their primary benefit is to create high-level design documents of a program by reading and analyzing its source code.

Reverse engineering tools are those that can create a representation of a system or program module at a design level of abstraction. For example, reverse engineering tools read program source code as input; perform an analysis; and extract information such as program control structures, data structures, and data flow. Once a program is represented at a design level using both graphical and textual representations, the design can be more effectively restructured to current business needs or programming practices by an analyst. For example, Microsoft's Visual Studio.NET can be used to reverse engineer applications into UML or other development diagrams.