Table of Contents

1 SYSTEMS DEVELOPMENT FUNDAMENTALS

2 PLANNING

3 ANALYSIS

4 DESIGN

5 IMPLEMENTATION AND MAINTENANCE

6 OBJECT ORIENTED ANALYSIS AND DESIGN

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.

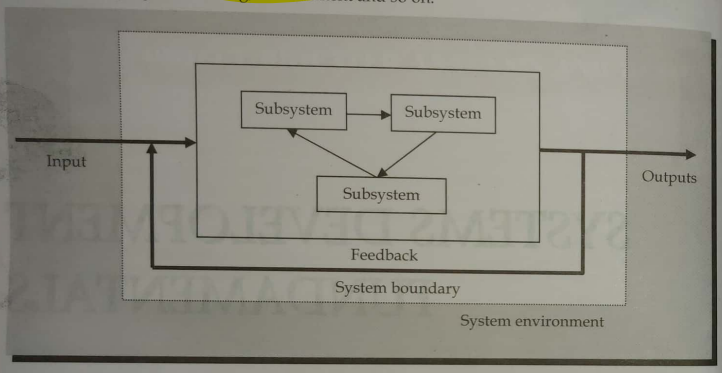


Figure 1.1: Basic System Model

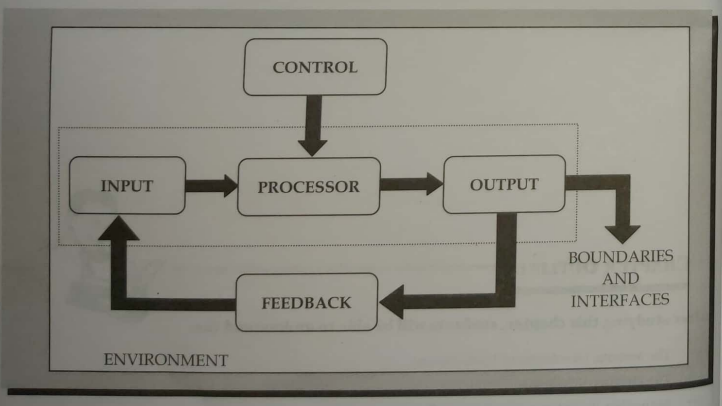


Figure 1.2: Elements of system

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:

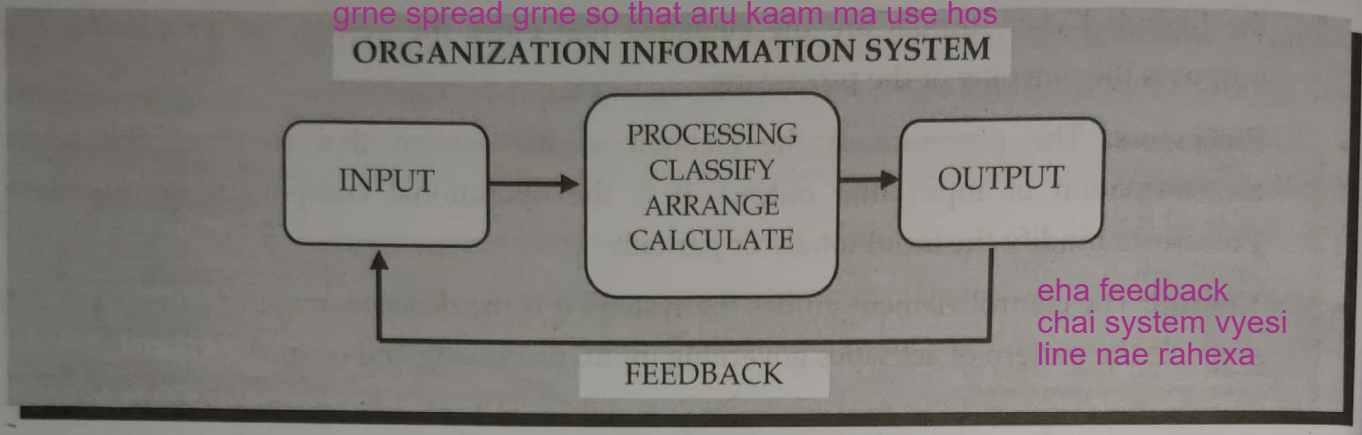
- **Organisation:** It implies structure and order. It is the arrangement of components that helps to achieve objectives.
- **Interaction:** It refers to the manner in which each component functions with other component of the system. In an organisation, for example, purchasing must interact with production, 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.

spread

data lai nae grne ho process tw ani output ma ako information lai collect grne spread grne so that aru kaam ma use hos



eha feedback chai system vyesi line nae rahexa

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.

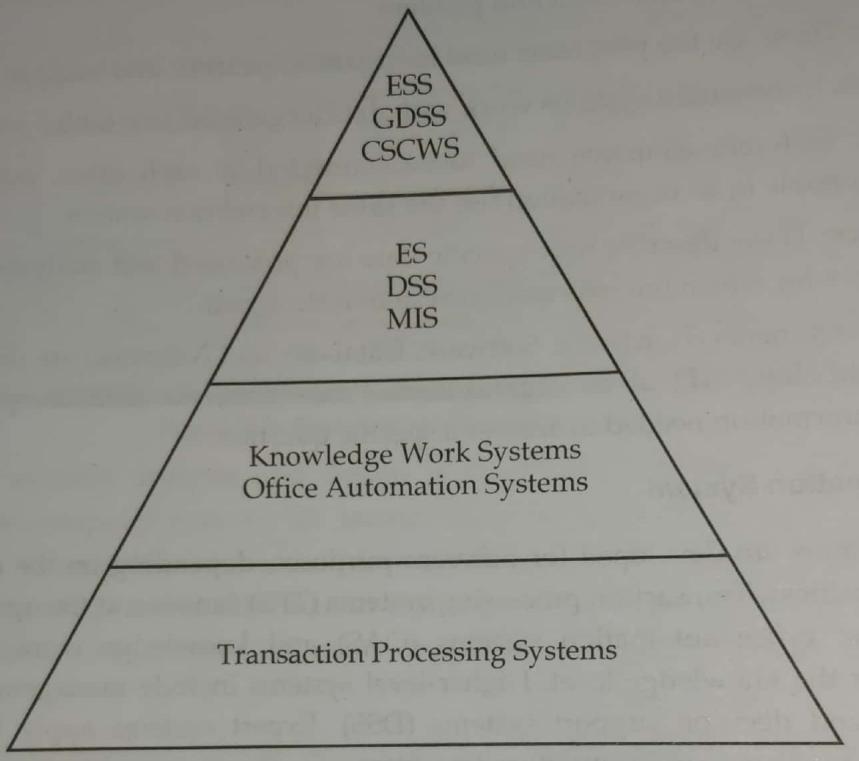


Figure 1.4: A systems analyst may be involved with any or all of these systems

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.

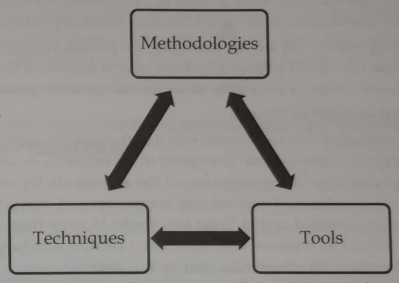


Figure 1.5: An organizational approach to systems analysis and design is driven by methodologies, techniques, and tools.

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.

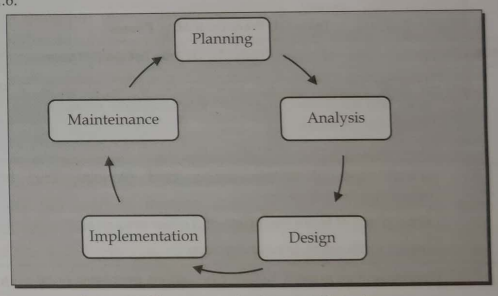


Figure 1.6: The systems development life cycle

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.

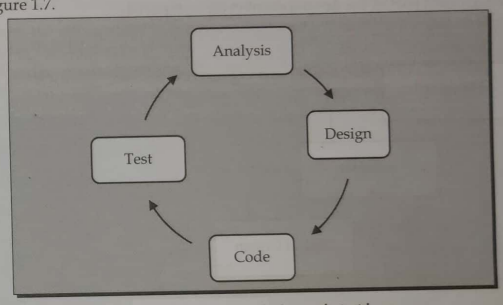


Figure 1.7: The analysis-design-code-test loop

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.

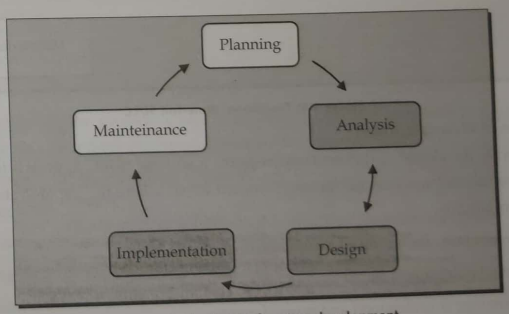


Figure 1.8: The heart of system development

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.

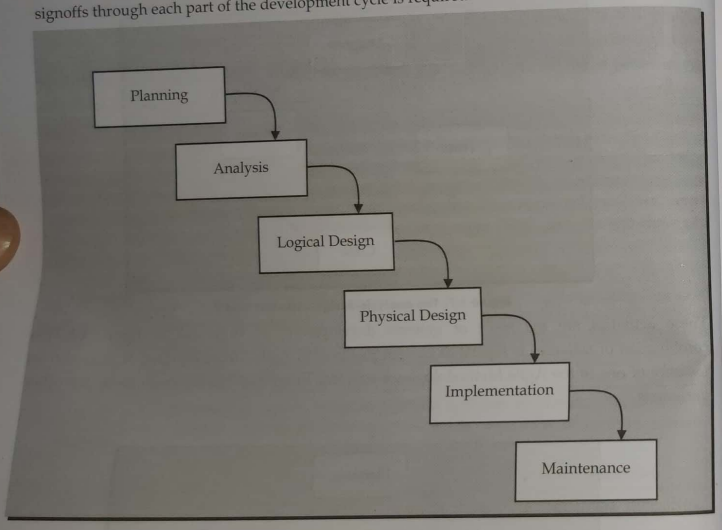


Figure 1.9: Traditional Waterfall SDLC

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.

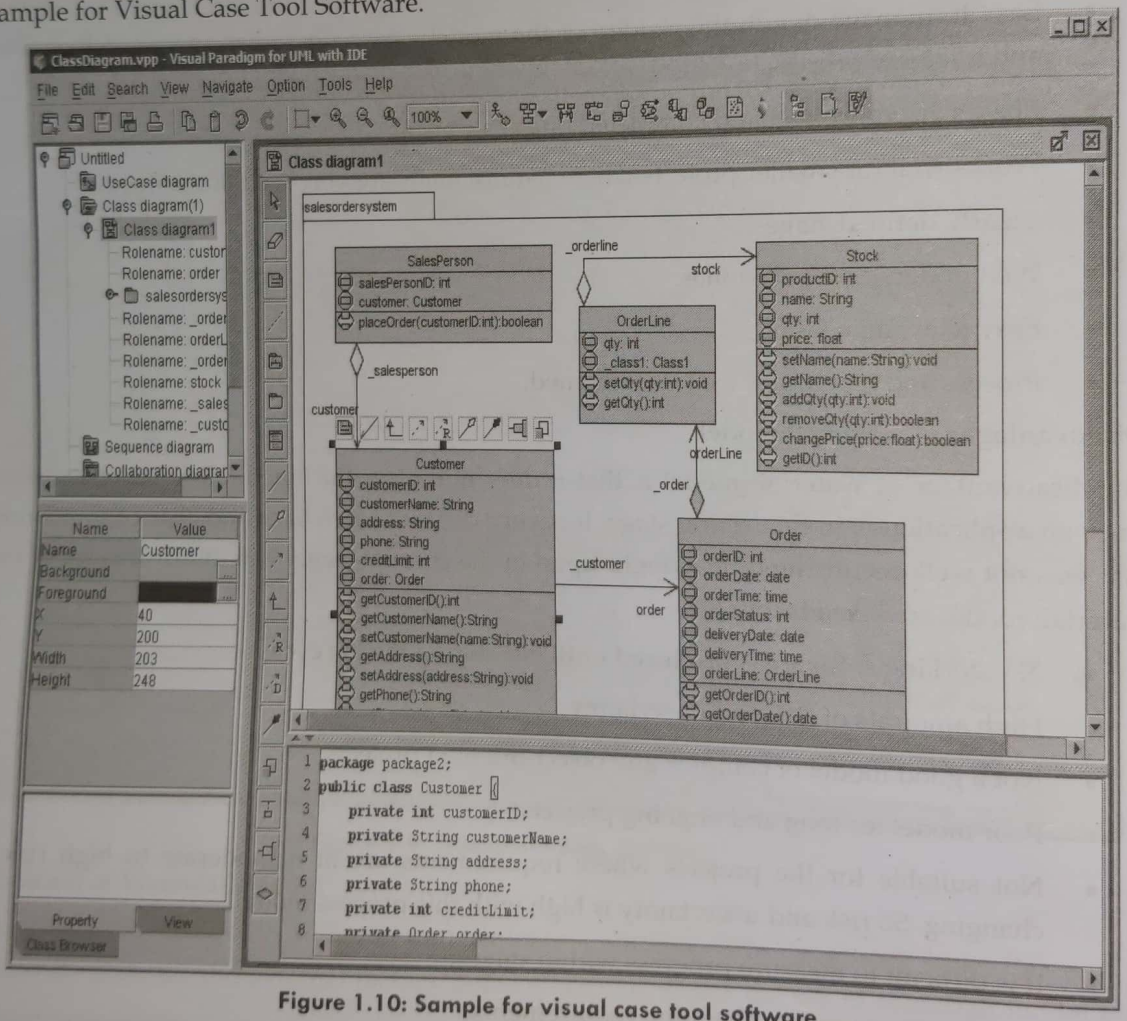


Figure 1.10: Sample for visual case tool software

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.

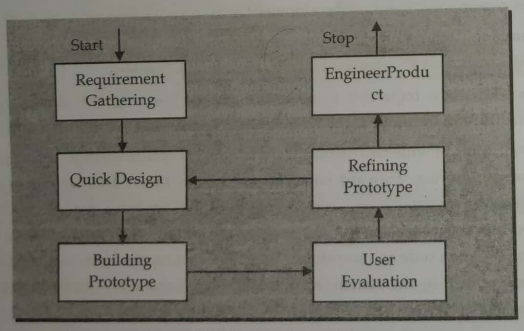


Figure 1.11: Prototyping model

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.

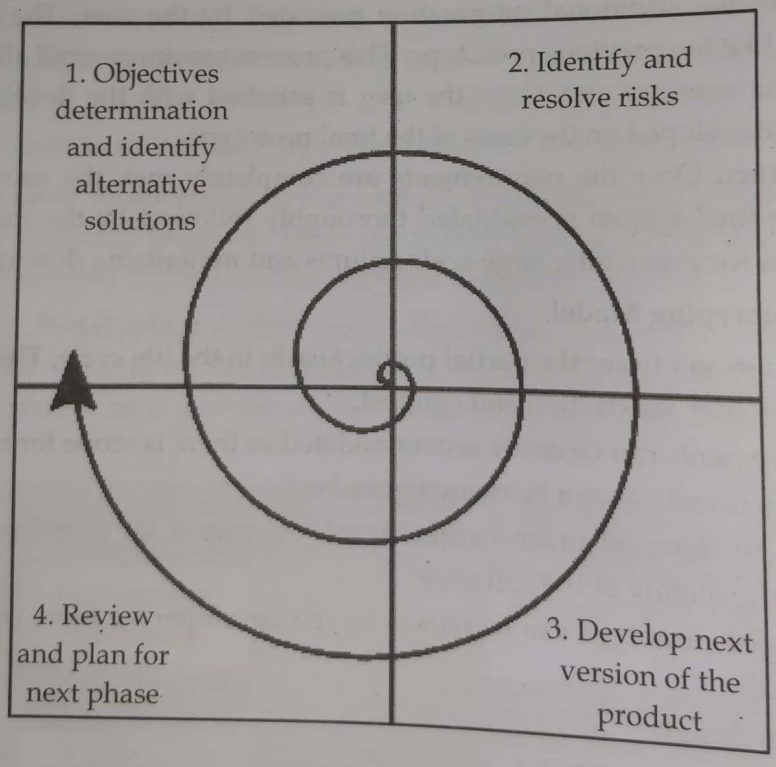


Figure 1.12: Different phases of spiral model

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.

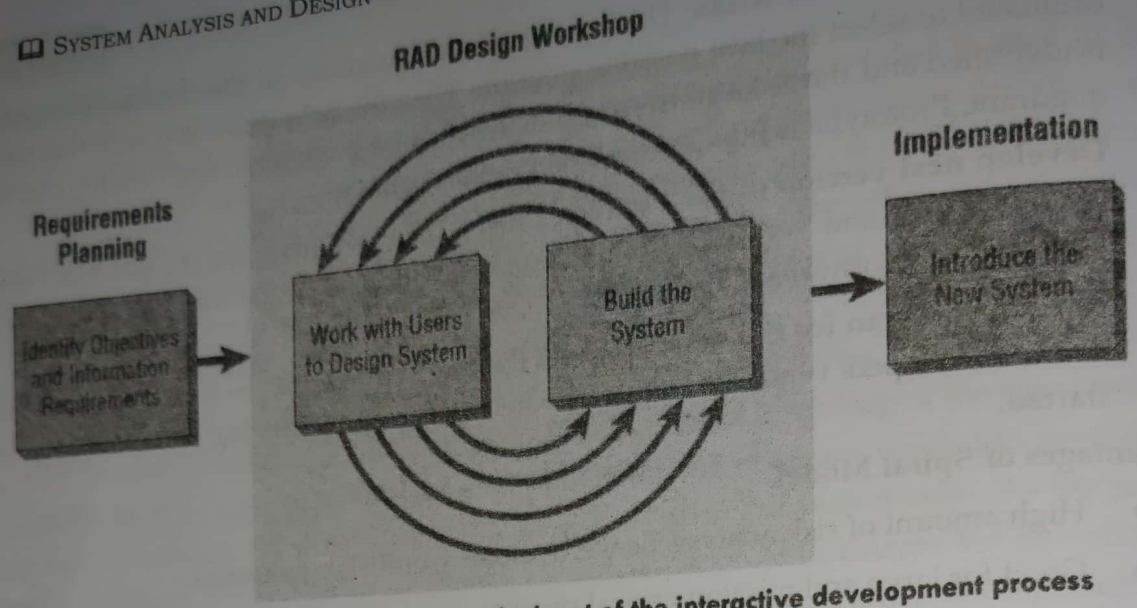


Figure 1.13: The RAD design workshop is the heart of the interactive development process

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.

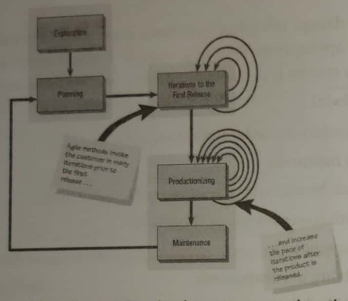


Figure 1.14: The five stages of the agile modeling development process show that frequent iterations are essential to successful system development

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.

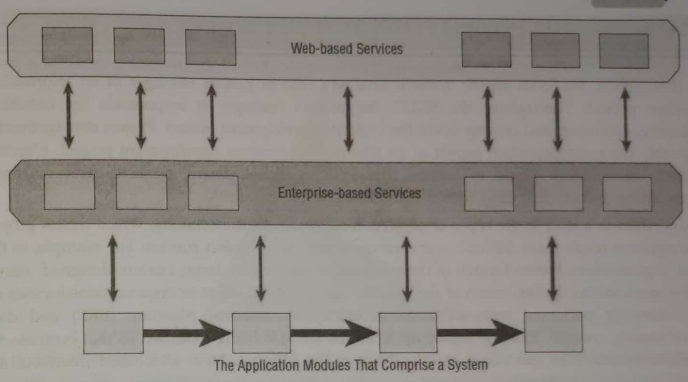


Figure 1.15: Modules in SOA are independent and can be ubiquitous

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.

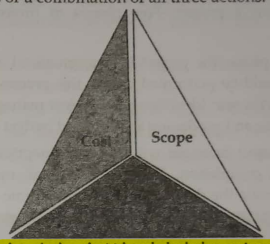


Figure 1.16: A typical project triangle includes cost, scope, and time.

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.

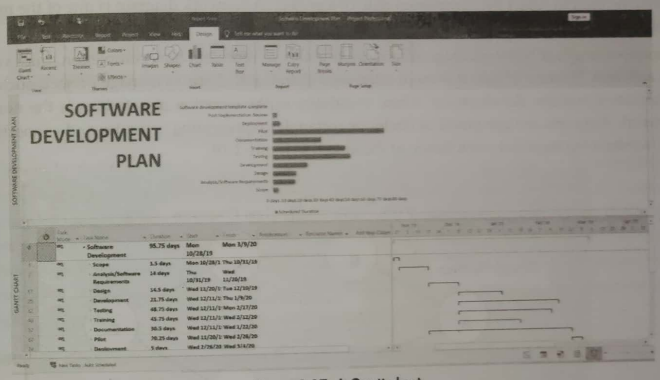


Figure 1.17: A Gantt chart

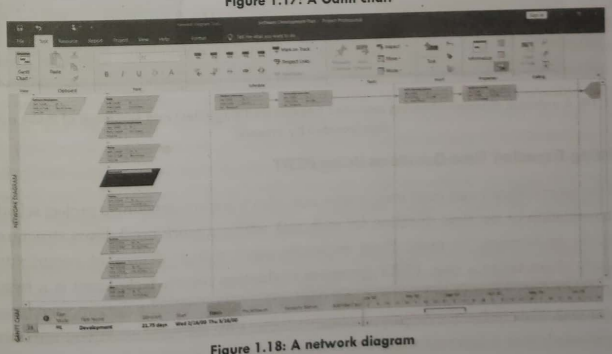


Figure 1.18: A network diagram

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.

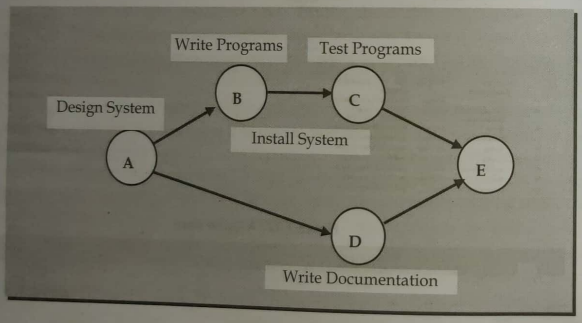


Figure 1.19: A network diagram showing activities (represented by circles) and sequence of those activities (represented by arrows)

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.

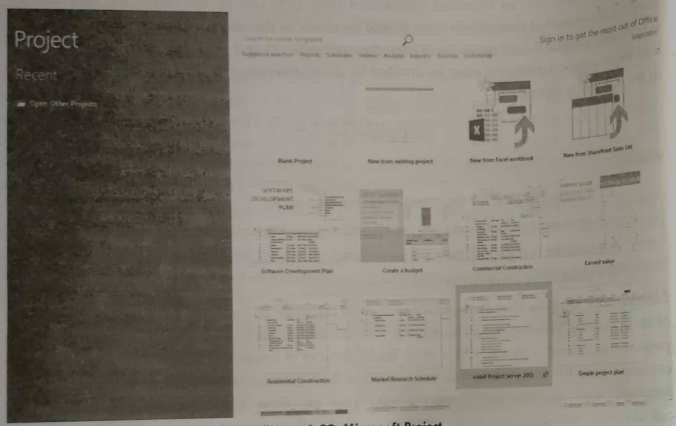


Figure 1.20: Microsoft Project

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.

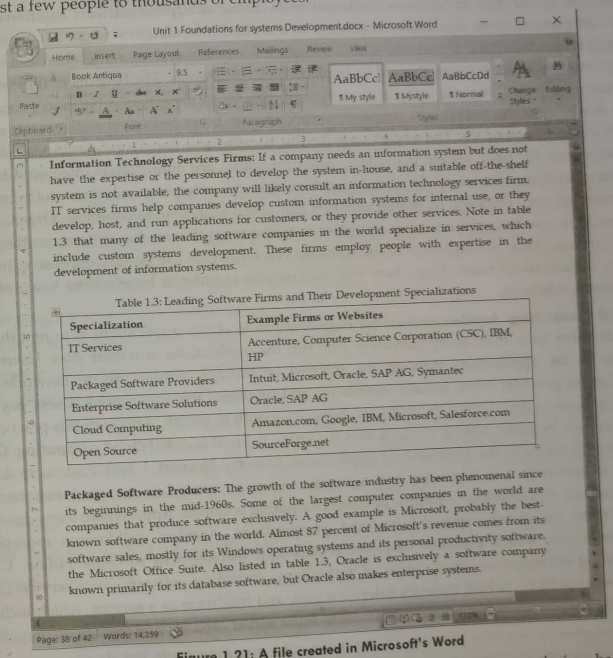


Figure 1.21: A file created in Microsoft's Word

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.

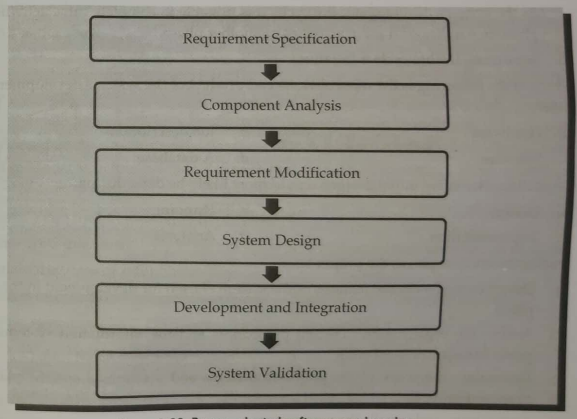


Figure 1.22: Reuse-oriented software engineering

Advantages of reuse

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

c. In-house development

d. All of the above

EXERCISE

1. What do you mean by system? Explain the characteristics of system.
2. Define Information System. What are the typical components of Information Systems?
3. Explain the types of Information System.
4. What is information systems analysis and design?
5. Who is system analyst? Discuss the roles of system analyst.
6. How has systems analysis and design changed over the past four decades?
7. List and explain the different phases in the SDLC.
8. List and explain some of the problems with the traditional waterfall SDLC.
9. Define CASE tools. Explain the types of CASE tools. What are the pros and cons of CASE tools?
10. Explain the prototyping approach of SDLC in detail.
11. Define spiral model of SDLC and explain its phases along with diagram.
12. Explain the RAD model. State the advantages and disadvantages of RAD approach.
13. Explain Agile Methodologies. When would you use Agile methodologies?
14. Discuss the reason why organizations undertake information systems projects.
15. Define project. Explain the project triangle.

16. List and describe the common skills and activities of project manager. Explain the phases of Project Management.
17. Explain the calculation process of Expected Time Duration using PERT with example.
18. Describe and compare the various sources of software.
19. Why do companies restore to outsourcing?
20. Write short notes on:
    a. Modern Approach to SAD.
    b. Project Manager
    c. Project Management Process
    d. Gantt chart
    e. Network diagram
    f. PERT (program evaluation review technique)
    g. Using Project Management Software
    h. Cloud Computing
    i. Enterprise Resource Planning (ERP)
    j. Request for proposal (RFP)
    k. Reuse
    l. In-house Development
    m. Packaged Software Producers

❑❑❑