MODULE 3

INFORMATION SYSTEM DEVELOPMENT

Why do organizations build or improve their information systems? The starting point for building or improving an information system is the recognition of a business problem or opportunity and the belief that better information system might create benefits by improving the operation of a work system. This module deals with the different phases of information system development i.e. planning, building and maintenance of information systems.

During the first phase of the module we will concentrate on the planning of information systems. Planning is an activity of management done on an event to avoid confusions and ambiguities during its execution. Plans serve as guidelines for events that may take long duration for their completion. Information system development is not an exception. It is an event which may take months or years for its completion. So proper information systems plan is necessary to build, maintain and manage information systems. A good information system plan reflects various issues in information system development such as what will be done, who will do it, when they will do it, how it will be done and what are the desired results.

The second phase of the module deals with the building and maintenance of information systems. Systems differ in terms of their size and technological complexity and in terms of organizational problems they are meant to solve. Because there are different kinds of system, a number of methods have been developed to build systems. Each has its own advantages and disadvantages. Information system implementation and maintenance is also covered in this section.

3.1 Systems as Planned Organizational Change

Information system is a sociotechnical entity, an arrangement of both social and technical elements. The introduction of new information system involves much more than new hardware and software. It also includes changes in jobs, skills, management and organization. In the sociotechnical philosophy one cannot install new technology without considering the people who must work with it. When an information system is designed the organization is redesigned.

Systems can be technical successes but organizational failures because of a failure in the social and political process of building the system. Analysts and designers are responsible for ensuring that key members of the organization participate in the design process and are permitted to influence the system's ultimate shape.

3.1.1 Linking Information Systems to Business Plan

Planning is the process of deciding what will be done, who will do it, when they will do it, how it will be done and what are the desired results. Information system planning should be an integral part of business planning. Organizations need to develop an information systems plan that supports their overall business plan. ***Business planning*** is the process of identifying the firm's goals, objectives and priorities and developing action plans to for accomplishing those goals and objectives. ***Information system planning*** is the part of business planning concerned with deploying the firm's information system resources including people, hardware and software. This plan serves as a road map indicating the direction of systems development (the purpose of the plan), the rationale, the current systems/situation, new developments to consider, the management strategy, the implementation plan, and the budget.

There is a similarity between information system planning and planning in various functional areas. Figure 3.1 illustrates this similarity. The plan of each functional area (and the information systems) should be based on the firm's goals, objectives and priorities. The individual plans in each of these areas are part of the business plan and should be consistent with it and support it.

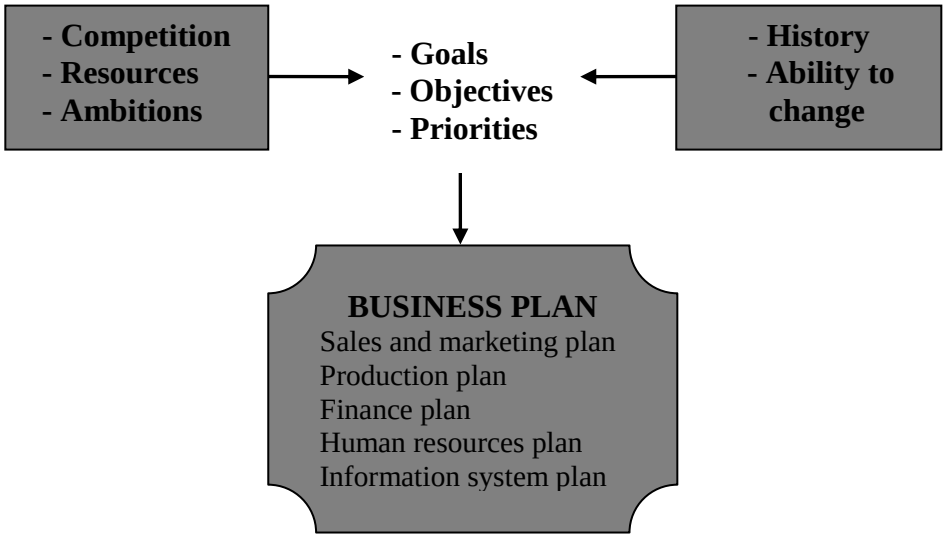

Figure 3.1: Information System plan as part of business plan

3.1.2 Establishing Organizational Information Requirements

To develop an effective information systems plan, the organization must have a clear understanding of both its long and short-term information requirements. Two

principal methodologies for establishing the essential information requirements of the organization as a whole are:

3.1.2. a Enterprise Analysis
3.1.2. b Critical Success Factors.

3.1.2.a Enterprise Analysis

Enterprise analysis (also called business systems planning) argues that the firm's information requirements can be understood only by examining the entire organization in terms of organizational units, functions, processes, and data elements. Enterprise analysis can help identify the key entities and attributes of the organization's data.

The central method used in the enterprise analysis approach is to take a large sample of managers and ask them how they use information, where they get their information, what their objectives are, how they make decisions, and what their data needs are. The results of this large survey of managers are aggregated into subunits, functions, processes, and data matrices.

The weakness of enterprise analysis is that it produces an enormous amount of data that is expensive to collect and difficult to analyze. The questions frequently focus not on management's critical objectives and where information is needed but rather on what existing information is used. The result is a tendency to automate whatever exists.

3.1.2.b Critical Success Factors

The ***critical success factors*** or strategic analysis approach argues that an organization's information requirements are determined by a small number of critical success factors (CSFs) of managers. If these goals can be attained, success of the firm or organization is assured. CSFs are shaped by the industry, the firm, the manager, and the broader environment. New information systems should focus on providing information that helps the firm meet these goals.

The principle method used in CSF analysis is personal interviews with a number of top managers identifying their goals and the resulting CSFs. These personal CSFs are aggregated to develop a picture of the firm's CSFs. Then systems are built to deliver information on these CSFs. Figure 3.2 illustrates the method of developing CSFs in an organization.

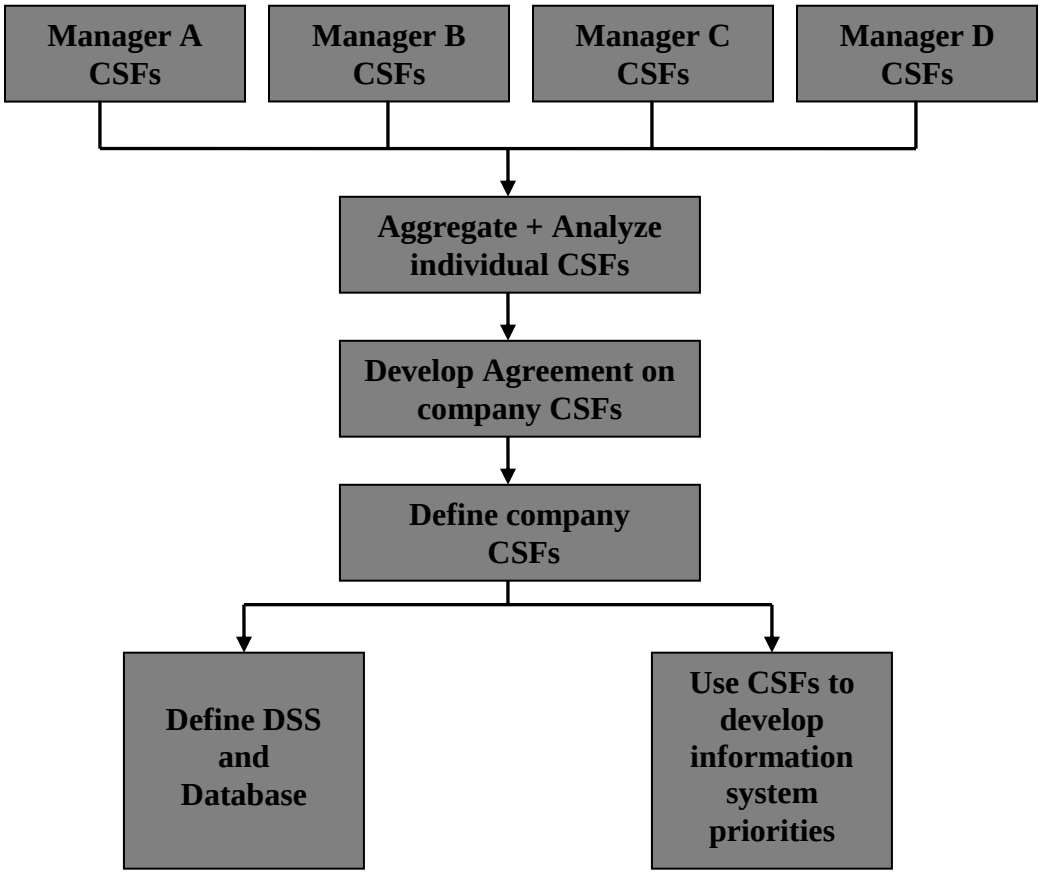

Figure 3.2: Method of developing CSFs in an Organization

The strength of the CSF method is that it produces less data to analyze than does enterprise analysis. Only top managers are interviewed, and the questions focus on a small number of CSFs rather than requiring a broad inquiry into what information is used in the organization.

The method's primary weakness is that the aggregation process and the analysis of the data are art forms. There is no particularly rigorous way in which individual CSFs can be aggregated into a clear company pattern. Second, interviewees (and interviewers) often become confused when distinguishing between individual and organizational CSFs. These types of CSFs are not necessarily the same. What may be considered critical to a manager may not be important for the organization as a whole.

3.1.3 Systems Development and Organizational Change

New information systems can be powerful instruments for organizational change, enabling organizations to redesign their structure, scope, power relationships, work flows, products and services. Information technology can promote various degrees of organizational change, ranging from incremental to far-reaching. Figure 3.3 shows four kinds of structural organizational change that are enabled by information technology. They are:

3.1.3.a Automation
3.1.3.b Rationalization
3.1.3.c Reengineering
3.1.3.d Paradigm shifts

Each carries different rewards and risks. The most common forms of organizational change are automation and rationalization. These relatively slow-moving and slow-changing strategies present modest returns but little risk. Faster and more comprehensive change—such as reengineering and paradigm shifts—carries high rewards but offers substantial chances of failure.

3.1.3.a Automation

Automation is the most common form of IT-enabled organizational change. The first applications of information technology involved assisting employees with performing their tasks more efficiently and effectively. ***Automation*** simply refers to the use computers to speed up the performance of existing tasks. Calculating paychecks and payroll registers, giving bank tellers instant access to customer deposit records, and developing a nationwide network of airline reservation terminals for airline reservation agents are all examples of early automation.

3.1.3.b Rationalization

Automation is the process of just computerizing existing operations of an organization. Direct automation of existing operating procedures of an organization might not provide the full benefit of computerization. So it is often necessary to streamline the existing arrangement of operating procedures and structures before the automation process to obtain the optimal benefit of information technology. ***Rationalization of procedures*** is the streaming of standard operating procedures, eliminating obvious bottlenecks, so that automation makes operation procedures more efficient.

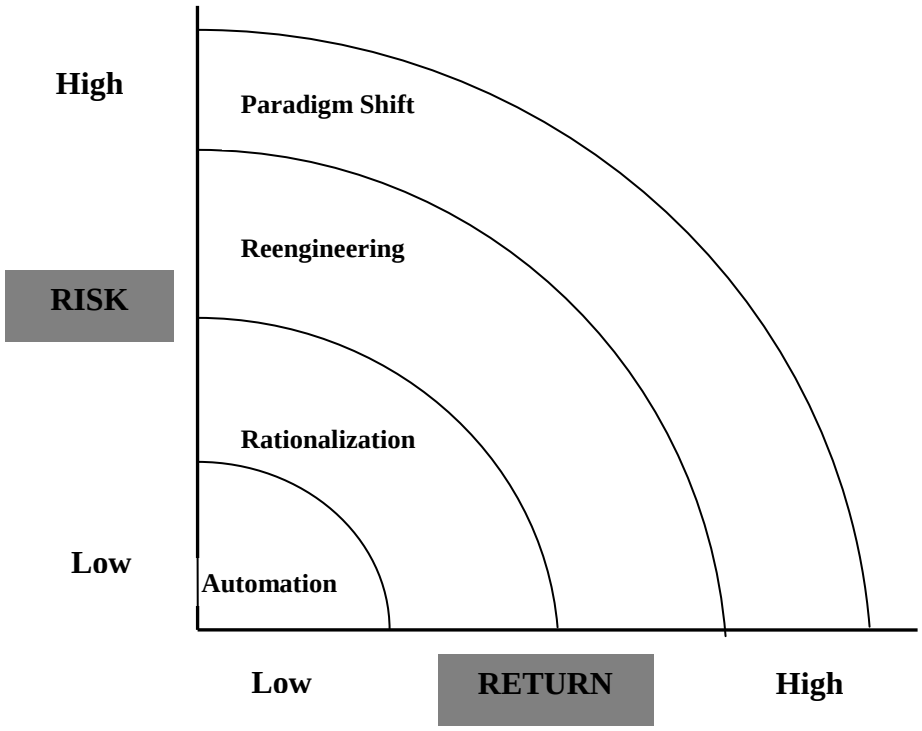

Figure 3.3: Organizational change carries risks and rewards

3.1.3.c Reengineering

A more powerful type of organizational change is ***business process reengineering***, in which business processes are radically redesigned, combining steps to cut waste and eliminating repetitive paper intensive tasks in order to improve cost, quality and service and to maximize the benefit of information technology. It is much more ambitious than rationalization of procedures, requiring a new vision of how the process is to be organized. A detailed discussion of business process reengineering will be considered in the next section of this module.

3.1.3.d Paradigm Shifts

Rationalizing procedures and business process reengineering are limited to specific parts of a business. New information systems can ultimately affect the design of the entire organization by transforming how the organization carries out its business or even the nature of the business. ***Paradigm shifts*** are radical reconceptualization of the nature of business and the nature of organization.

Paradigm shifts and reengineering often fail because extensive organizational change is so difficult to orchestrate. Why, then, do so many corporations entertain such

radical change? Because the rewards are equally high. In many instances firms seeking paradigm shifts and pursuing reengineering strategies achieve stunning, order-of-magnitude increases in their returns on investment (or productivity).

3.1.4 Business Process Reengineering and Process Improvement

Many companies today are focusing on building new information systems that will improve their business processes. Business process reengineering is a highly publicized idea that became an umbrella term for a variety of internal corporate initiatives in the early 1990s.

Business process reengineering is the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical contemporary measures of performance, such as cost, quality, service and speed.

[Michael Hammer]

If organizations rethink and radically redesign their business processes before applying computing power, they can potentially obtain very large payoffs from their investments in information technology. Common outcomes of business process reengineering include combining several jobs into one, permitting workers to make more decisions themselves, defining simple versions of processes for simple cases versus complex ones and reorganizing jobs to give individuals more understanding and more responsibility. Many reengineering efforts also result in significant staff reductions.

3.1.4.1 Steps in Effective Reengineering

One of the most important strategic decisions that a firm can make is not deciding how to use computers to improve business processes but rather understanding what business processes need improvement. When systems are used to strengthen the wrong business model or business processes, the business can become more efficient at doing what it should not do. As a result, the firm becomes vulnerable to competitors who may have discovered the right business model. Considerable time and cost can also be spent improving business processes that have little impact on overall firm performance and revenue. Managers need to determine what business processes are the most important to focus on when applying new information technology and how improving these processes will help the firm execute its strategy.

Management must understand and measure the performance of existing processes as a baseline. If, for example, the objective of process redesign is to reduce

time and cost in developing a new product or filling an order, the organization needs to measure the time and cost consumed by the unchanged process. For example, before reengineering, it cost C. R. England & Sons $5.10 to send an invoice; after processes were reengineered, the cost per invoice dropped to 15 cents.

The conventional method of designing systems establishes the information requirements of a business function or process and then determines how they can be supported by information technology. However, information technology can create new design options for various processes because it can be used to challenge long-standing assumptions about work arrangements that used to inhibit organizations. For example, it is no longer necessary for people to be in the same physical location to work together on a document. Using networks and document management technology, people can access and work on the same document from many different locations. *Information technology should be allowed to influence process design from the start.*

Following these steps does not automatically guarantee that reengineering will always be successful. The majority of reengineering projects do not achieve breakthrough gains in business performance because the organizational changes are often very difficult to manage. Managing change is neither simple nor intuitive, and companies committed to reengineering need a good change management strategy.

3.1.5 Process Improvement: Total Quality Management and Six Sigma

Business process reengineering (BPR) is primarily a one-time effort, focusing on identifying one or two strategic business processes that need radical change. BPR projects tend to be expensive and organizationally disruptive. But organizations have many business processes and support processes that must be constantly revised to keep the business competitive.

Quality management is another area of continuous process improvement. In addition to increasing organizational efficiency, companies must fine-tune their business processes to improve the quality in their products, services, and operations. Many are using the concept of ***Total Quality Management (TQM)*** to make quality the responsibility of all people and functions within an organization. TQM holds that the achievement of quality control is an end in itself. Everyone is expected to contribute to the overall improvement of quality—the engineer who avoids design errors, the production worker who spots defects, the sales representative who presents the product properly to potential customers, and even the secretary who avoids typing mistakes.

Another quality concept that is being widely implemented today is six sigma. ***Six sigma*** is a specific measure of quality, representing 3.4 defects per million opportunities. Most companies cannot achieve this level of quality but use six sigma as a goal to implement a set of methodologies and techniques for improving quality and reducing costs. Studies have repeatedly shown that the earlier in the business cycle a problem is eliminated, the less it costs the company. Thus, quality improvements not only raise the level of product and service quality, but they can also lower costs.

3.1.5.1 Information system support and Quality Improvements

TQM and six sigma are considered to be more incremental than business process reengineering. TQM typically focuses on making a series of continuous improvements rather than dramatic bursts of change. Six sigma uses statistical analysis tools to detect flaws in the execution of an existing process and make minor adjustments. Sometimes, however, processes may have to be fully reengineered to achieve a specified level of quality. Information systems can help firms achieve their quality goals by helping them simplify products or processes, make improvements based on customer demands, reduce cycle time, improve the quality and precision of design and production, and meet benchmarking standards.

3.2 Overview of Systems Development

A new information system is built as a solution to some type of problem or set of problems the organization perceives it is facing. The problem may be one in which managers and employees realize that the organization is not performing as well as expected, or it may come from the realization that the organization should take advantage of new opportunities to perform more successfully.

The activities that go into producing an information system solution to an organizational problem or opportunity are called ***systems development***. Systems development is a structured kind of problem solving with distinct activities. Building a system can be broken down into six core activities as shown in figure 3.4. These activities consist of systems analysis, systems design, programming, testing, conversion, and production and maintenance.

3.2.1 Systems Analysis

Systems analysis is the analysis of the problem that the organization will try to solve with an information system. It consists of defining the problem, identifying its causes, specifying the solution, and identifying the information requirements that must be met by a system solution.

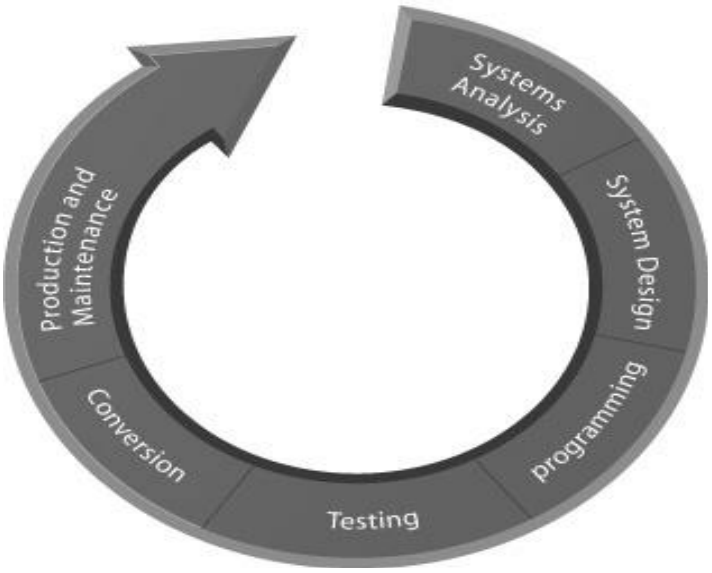

Figure 3.4: The system development process

Initially the system analyst, a person responsible for conducting systems analysis, performs an organizational analysis. During ***organizational analysis*** the analyst creates a road map of the existing organization and systems, identifying the primary owners and users of data and the existing hardware and software that serve the organization. By examining documents, work papers, and procedures; observing system operations; and interviewing key users of the systems, the analyst can identify the problem areas and objectives a solution would achieve. Often the solution requires building a new information system or improving an existing one.

The systems analysis would include a ***feasibility study*** to determine whether that solution was feasible, or achievable, from a financial, technical, and organizational standpoint. The feasibility study would determine whether the proposed system was a good investment, whether the technology needed for the system was available and could be handled by the firm's information systems specialists, and whether the organization could handle the changes introduced by the system.

Normally, the systems analysis process identifies several alternative solutions that the organization can pursue. The process then assesses the feasibility of each. A written systems proposal report describes the costs and benefits, advantages and disadvantages of each alternative. It is up to management to determine which mix of

costs, benefits, technical features, and organizational impacts represents the most desirable alternative.

Perhaps the most challenging task of the systems analyst is to define the specific information requirements that must be met by the system solution selected. At the most basic level, the ***information requirements*** of a new system involve identifying who needs what information, where, when, and how. Requirements analysis carefully defines the objectives of the new or modified system and develops a detailed description of the functions that the new system must perform.

3.2.2 Systems Design

During the systems analysis phase the problem of the existing system is identified and defined. Different solutions are proposed for the problem. Feasibility assessments of these solutions are made to choose the best solution. Information requirements for the new system are specified.

While systems analysis describes what a system should do to meet information requirements, ***systems design*** shows how the system will fulfill this objective. Requirements are translated into design specifications. The design of an information system is the overall plan or model for that system.

System design involves first the conceptual design (general design) phase and then a physical design (detailed design) phase. The ***conceptual design*** emphasizes the system as seen by those who will operate or use the outputs of the system i.e. a more user – oriented design for the system. It describes the inputs and outputs, files and databases, functions and procedures to be performed by the system from a higher level of abstraction i.e. without going into the technical details of the design and thus making it referable with users.

The ***physical design*** consists of activities to prepare the detailed technical design of the application system. It is based on information requirements and the conceptual design. The work of the physical design is to take the fairly high level user - oriented requirements of the conceptual design phase and produce a specific technical design. This work is performed by the systems analyst and other technical personnel. Users may participate in this phase but much of the work requires data processing expertise instead of user function expertise.

3.2.3 Programming

During the programming stage, system specifications that were prepared during the design stage are translated into software program code. Organizations write software programs themselves or purchase application software packages for this purpose.

3.2.4 Testing

Exhaustive and thorough testing must be conducted to ascertain whether the system produces the right results. Testing answers the question, "Will the system produce the desired results under known conditions?" Testing is time consuming: Test data must be carefully prepared, results reviewed, and corrections made in the system. In some instances parts of the system may have to be redesigned.

Testing an information system can be broken down into three types of activities: unit testing, system testing, and acceptance testing. ***Unit testing***, or ***program testing***, consists of testing each program separately in the system. It is widely believed that the purpose of such testing is to guarantee that programs are error free, but this goal is realistically impossible. Testing should be viewed instead as a means of locating errors in programs, focusing on finding all the ways to make a program fail. Once they are pinpointed, problems can be corrected.

System testing tests the functioning of the information system as a whole. It tries to determine whether discrete modules will function together as planned and whether discrepancies exist between the way the system actually works and the way it was conceived. Among the areas examined are performance time, capacity for file storage and handling peak loads, recovery and restart capabilities, and manual procedures.

Acceptance testing provides the final certification that the system is ready to be used in a production setting. Systems tests are evaluated by users and reviewed by management. When all parties are satisfied that the new system meets their standards, the system is formally accepted for installation. Acceptance testing is generally conducted in the conversion phase of system development.

The systems development team works with users to devise a systematic test plan. The test plan includes all of the preparations for the series of tests we have just described.

3.2.5 Conversion

Conversion is the process of changing from the old system to the new system. Four main conversion strategies can be employed: the parallel strategy, the direct cutover strategy, the pilot study strategy, and the phased approach strategy.

In a ***parallel strategy*** both the old system and its potential replacement are run together for a time until everyone is assured that the new one functions correctly. This is the safest conversion approach because, in the event of errors or processing disruptions, the old system can still be used as a backup. However, this approach is very expensive, and additional staff or resources may be required to run the extra system.

The ***direct cutover*** strategy replaces the old system entirely with the new system on an appointed day. It is a very risky approach that can potentially be more costly than running two systems in parallel if serious problems with the new system are found. There is no other system to fall back on. Dislocations, disruptions, and the cost of corrections may be enormous.

The ***pilot study*** strategy introduces the new system to only a limited area of the organization, such as a single department or operating unit. When this pilot version is complete and working smoothly, it is installed throughout the rest of the organization, either simultaneously or in stages.

The ***phased approach*** strategy introduces the new system in stages, either by functions or by organizational units. If, for example, the system is introduced by functions, a new payroll system might begin with hourly workers who are paid weekly, followed six months later by adding salaried employees (who are paid monthly) to the system. If the system is introduced by organizational units, corporate headquarters might be converted first, followed by outlying operating units four months later.

File building is a major activity conducted during the conversion phase. File building refers to the collection and conversion to machine-readable form of all the new data required by the application. Moving from an old system to a new one requires that end users be trained to use the new system. ***Training*** is the process of ensuring that system participants know what they need to know about the work system and information system. Detailed ***documentation*** showing how the system works from both a technical and end-user standpoint is finalized during conversion time for use in training and everyday operations. Lack of proper training and documentation

contributes to system failure, so this portion of the systems development process is very important.

3.2.5 Production and Maintenance

After the new system is installed and conversion is complete, the system is said to be in ***production***. During this stage, the system will be reviewed by both users and technical specialists to determine how well it has met its original objectives and to decide whether any revisions or modifications are in order. In some instances, a formal post implementation audit document is prepared. After the system has been fine-tuned, it must be maintained while it is in production to correct errors, meet requirements, or improve processing efficiency. Changes in hardware, software, documentation, or procedures to a production system to correct errors, meet new requirements, or improve processing efficiency are termed ***maintenance***.

Building successful information systems requires close cooperation among end users and information systems specialists throughout the systems development process. Table 3.1 summarizes the systems development activities.

Core Activity	Description
Systems analysis	Identify problem(s) Specify solutions Establish information requirements
Systems design	Create design specifications
Programming	Translate design specifications into program code
Testing	Unit test Systems test Acceptance test
Conversion	Plan conversion Prepare documentation Train users and technical staff
Production and maintenance	Operate the system Evaluate the system Modify the system

TABLE 3.1: Systems Development

3.3 Modeling and Designing Systems: Structured and Object-Oriented Methodologies

There are alternative methodologies for modeling and designing systems. Structured methodologies and object-oriented development are the most prominent.

3.3.1 Structured Methodologies

Structured methodologies have been used to document, analyze, and design information systems since the 1970s. ***Structured*** refers to the fact that the techniques are step by step, with each step building on the previous one. Structured methodologies are ***top-down***, progressing from the highest, most abstract level to the lowest level of detail—from the general to the specific.

Structured development methods are ***process-oriented***, focusing primarily on modeling the processes, or actions that capture, store, manipulate, and distribute data as the data flow through a system. These methods separate data from processes. A separate programming procedure must be written every time someone wants to take an action on a particular piece of data. Structured methodologies include structured analysis, structured design and structured programming.

3.3.1.1 Structured Analysis

Structured Analysis is widely used to define system inputs, processes and outputs. It offers a logical graphic model of information flow called ***Data Flow Diagram (DFD)***, partitioning a system into modules that show manageable levels of detail. It rigorously specifies the processes or transformations that occur within each module and the interfaces that exist between them.

Figure 3.5 shows a simple data flow diagram for a mail-in university course registration system. The ***rounded boxes*** represent processes, which portray the transformation of data. The ***square box*** represents an external entity, which is an originator or receiver of data, located outside the boundaries of the system being modeled. The ***open rectangles*** represent data stores, which are either manual or automated inventories of data. The ***arrows*** represent data flows, which show the movement between processes, external entities, and data stores. They always contain packets of data with the name or content of each data flow listed beside the arrow.

This data flow diagram shows that students submit registration forms with their name, identification number, and the numbers of the courses they wish to take. In process 1.0 the system verifies that each course selected is still open by referencing the university's course file. The file distinguishes courses that are open from those that have been canceled or filled. Process 1.0 then determines which of the student's selections can be accepted or rejected. Process 2.0 enrolls the student in the courses for which he or she has been accepted. It updates the university's course file with the

student's name and identification number and recalculates the class size. Process 2.0 also updates the university's student master file with information about new students or changes in address. Process 3.0 then sends each student applicant a confirmation-of-registration letter listing the courses for which he or she is registered and noting the course selections that could not be fulfilled.

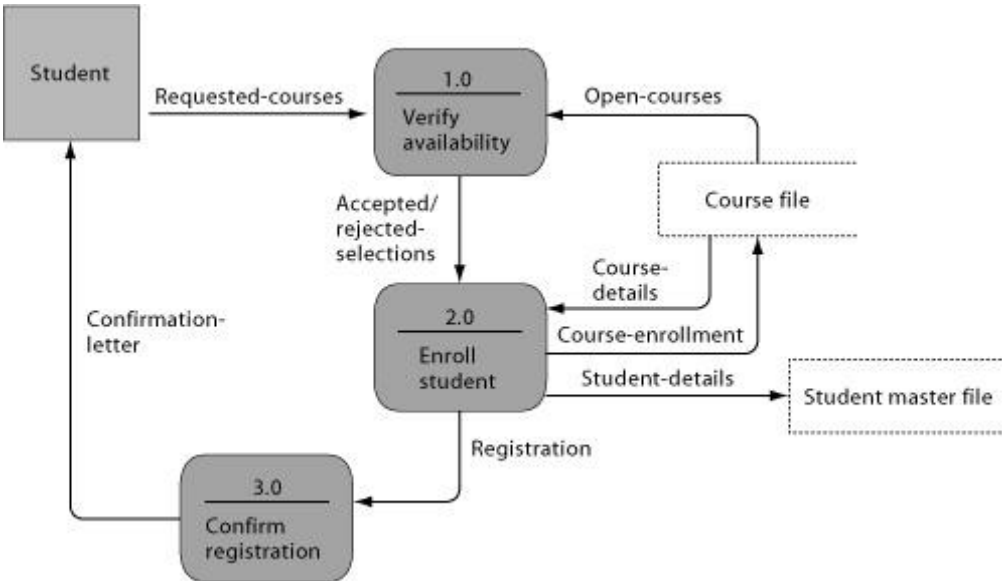

Figure 3.5 Data flow diagram for mail-in university registration system

Another tool for structured analysis is a ***data dictionary***, which contains information about individual pieces of data and data groupings within a system. The data dictionary defines the contents of data flows and data stores so that systems builders understand exactly what pieces of data they contain. ***Process specifications*** describe the transformation occurring within the lowest level of the data flow diagrams. They express the logic for each process.

3.3.1.2 Structured Design

In structured methodology, software design is modeled using hierarchical structure charts. The ***structure chart*** is a top-down chart, showing each level of design, its relationship to other levels, and its place in the overall design structure. The design first considers the main function of a program or system, then breaks this function into sub functions, and decomposes each sub function until the lowest level of detail has been reached. Figure 3.6 shows a high-level structure chart for a payroll system. A

structure chart may document one program, one system (a set of programs), or part of one program.

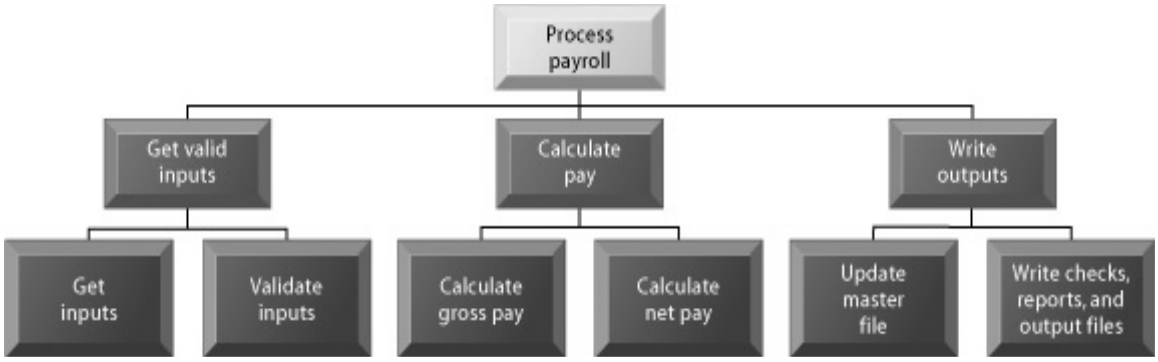

Figure 3.6: High-level structure chart for a payroll system

3.3.1.3 Structured Programming

Structured programming extends the principles governing structured design to the writing of programs to make software programs easier to understand and modify. It is based on the principle of modularization, which follows from top-down analysis and design. A ***module*** is a logical unit of a program that performs one or several functions.

In structured programming, the basic components that make a software program easier to understand and modify are the structured constructs. Sequence construct, selection construct and iteration construct are the three basic control constructs that are used in a structured program.

The ***sequence construct*** executes statements in the order in which they appear, with control passing unconditionally from one statement to the next. ***The selection construct*** tests a condition and executes one of the two alternative instructions based on the result of the test. The ***iteration construct*** repeats a segment of code as long as a conditional test remains true.

3.3.2 Object-Oriented Methodologies

Structured methods are useful for modeling processes, but do not handle the modeling of data well. They also treat data and processes as logically separate entities.

Object-oriented development tries to deal with these issues. Object-oriented development uses the object as the basic unit of systems analysis and design. An ***object***

combines data and the specific processes that operate on those data. Data encapsulated in an object can be accessed and modified only by the operations, or methods, associated with that object. Instead of passing data to procedures, programs send a message for an object to perform an operation that is already embedded in it. The system is modeled as a collection of objects and the relationships among them. Because processing logic resides within objects rather than in separate software programs, objects must collaborate with each other to make the system work.

Object-oriented modeling is based on the concepts of class and inheritance. Objects belonging to a certain class, or general categories of similar objects, have the features of that class. Classes of objects in turn can inherit all the structure and behaviors of a more general class and then add variables and behaviors unique to each object. New classes of objects are created by choosing an existing class and specifying how the new class differs from the existing class, instead of starting from scratch each time.

We can see how class and inheritance work in Figure 3.7, which illustrates the relationships among classes concerning employees and how they are paid. Employee is the common ancestor, or super class, for the other three classes. Salaried, Hourly, and Temporary are subclasses of Employee. The class name is in the top compartment, the attributes for each class are in the middle portion of each box, and the list of operations is in the bottom portion of each box. The features that are shared by all employees (id, name, address, date hired, position, and pay) are stored in the Employee super class, whereas each subclass stores features that are specific to that particular type of employee. Specific to Hourly employees, for example, are their hourly rates and overtime rates. A solid line from the subclass to the super class is a generalization path showing that the subclasses Salaried, Hourly, and Temporary have common features that can be generalized into the super class Employee.

The phases of object-oriented development are similar to those of conventional systems development, consisting of analysis, design, and implementation. However, object-oriented development is more iterative and incremental than traditional structured development. During analysis, systems builders document the functional requirements of the system, specifying its most important properties and what the proposed system must do. Interactions between the system and its users are analyzed to identify objects, which include both data and processes. The object-oriented design phase describes how the objects will behave and how they will interact with one other.

Similar objects are grouped together to form a class, and classes are grouped into hierarchies in which a subclass inherits the attributes and methods from its super class.

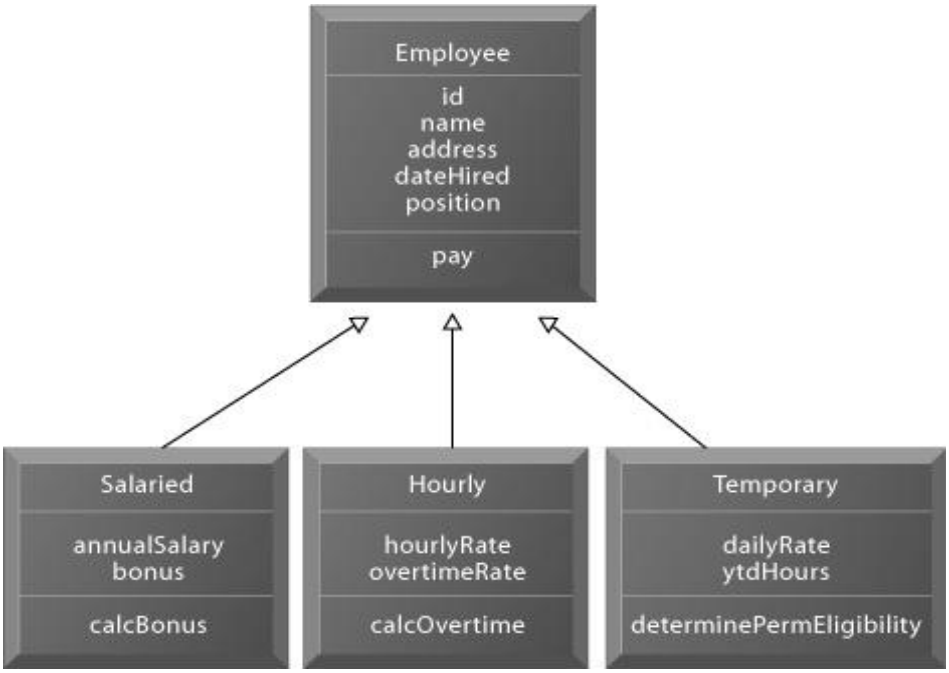

Figure 3.7: Class and inheritance

The information system is implemented by translating the design into program code, reusing classes that are already available in a library of reusable software objects and adding new ones created during the object-oriented design phase. Implementation may also involve the creation of an object-oriented database. The resulting system must be thoroughly tested and evaluated.

Because objects are reusable, object-oriented development could potentially reduce the time and cost of writing software because organizations can reuse software objects that have already been created as building blocks for other applications. New systems can be created by using some existing objects, changing others, and adding a few new objects. Object-oriented frameworks have been developed to provide reusable, semi complete applications that the organization can further customize into finished applications.

Unified Modeling Language (UML) has become the industry standard for representing various views of an object-oriented system using a series of graphical diagrams. The underlying model integrates these views to promote consistency during

analysis, design, and implementation. UML uses two principal types of diagrams: structural diagrams and behavioral diagrams.

Structural diagrams are used to describe the relationship between classes. Figure 3.7 is an example of one type of structural diagram called a ***class diagram***. It shows classes of employees and the relationships between them.

Behavioral diagrams are used to describe interactions in an object-oriented system. Figure 3.8 illustrates one type of behavioral diagram called a ***use case diagram***. A use case diagram shows the relationship between an actor and a system. The ***actor*** (represented in the diagram as a stick man) is an external entity that interacts with the system, and the use case represents a series of related actions initiated by the actor to accomplish a specific goal. Several interrelated use cases are represented as ovals within a box.

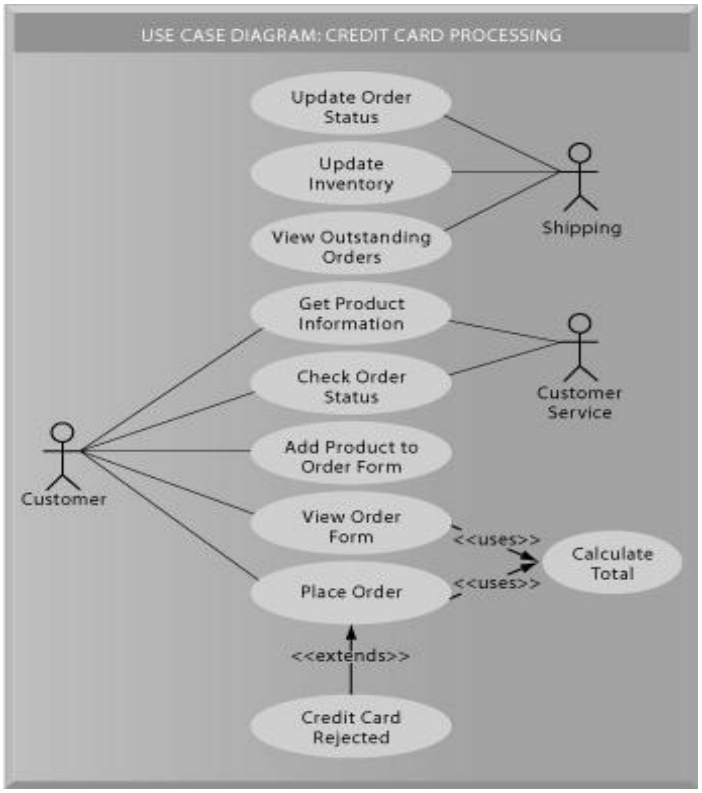

Figure 3.8: A UML use case diagram

Use case diagrams model the functions of a system, showing how objects interact with each other and with the users of the system. Illustrated above is a use case diagram for credit card processing.

3.4 Computer-Aided Software Engineering

Computer-Aided Software Engineering (CASE)—sometimes called computer-aided systems engineering—provides software tools to automate the methodologies we have just described to reduce the amount of repetitive work the developer needs to do. CASE tools also facilitate the creation of clear documentation and the coordination of team development efforts. Team members can share their work easily by accessing each other's files to review or modify what has been done. Modest productivity benefits can also be achieved if the tools are used properly. Many CASE tools are PC-based, with powerful graphical capabilities.

Many CASE tools have been classified in terms of whether they support activities at the front end or the back end of the systems development process. ***Front-end CASE tools*** focus on capturing analysis and design information in the early stages of systems development, whereas ***back-end CASE tools*** address coding, testing, and maintenance activities. Back-end tools help convert specifications automatically into program code.

3.5 Alternative Systems-Building Approaches

Systems differ in terms of their size and technological complexity and in terms of the organizational problems they are meant to solve. A number of systems-building approaches have been developed to deal with these differences. This section describes these alternative methods:

3.5.1 Traditional systems life cycle
3.5.2 Prototyping
3.5.3End-user development
3.5.4 Application software packages
3.5.5 Outsourcing

3.5.1 Traditional systems life cycle

The systems life cycle is the oldest method for building information systems. The life cycle methodology is a phased approach to building a system, dividing systems development into formal stages.

The systems life cycle methodology maintains a very formal division of labor between end users and information systems specialists. Technical specialists, such as system analysts and programmers, are responsible for much of the systems analysis,

design, and implementation work; end users are limited to providing information requirements and reviewing the technical staff 's work. The life cycle also emphasizes formal specifications and paperwork; so many documents are generated during the course of a systems project.

The systems life cycle is still used for building large complex systems that require a rigorous and formal requirements analysis, predefined specifications, and tight controls over the systems-building process. However, the systems life cycle approach can be costly, time consuming, and inflexible.

3.5.2 Prototyping

Prototyping consists of building an experimental system rapidly and inexpensively for end users to evaluate. By interacting with the prototype, users can get a better idea of their information requirements. The prototype endorsed by the users can be used as a template to create the final system.

The ***prototype*** is a working version of an information system or part of the system, but is meant to be only a preliminary model. Once operational, the prototype will be further refined until it conforms precisely to users' requirements. Once the design has been finalized, the prototype can be converted to a polished production system.

The process of building a preliminary design, trying it out, refining it, and trying again has been called an **iterative process of systems development** because the steps required to build a system can be repeated over and over again. Prototyping is more explicitly iterative than the conventional life cycle, and it actively promotes system design changes.

3.5.2.1 Steps in Prototyping

Figure 3.9 shows a four-step model of the prototyping process, which consists of the following:

Step 1: Identify the user's basic requirements. The system designer (usually an information systems specialist) works with the user only long enough to capture the user's basic information needs.

Step 2: Develop an initial prototype. The system designer creates a working prototype quickly, using tools for rapidly generating software.

Step 3: Use the prototype. The user is encouraged to work with the system to determine how well the prototype meets his or her needs and to make suggestions for improving the prototype.

Step 4: Revise and enhance the prototype. The system builder notes all changes the user requests and refines the prototype accordingly. After the prototype has been revised, the cycle returns to step 3. Steps 3 and 4 are repeated until the user is satisfied.

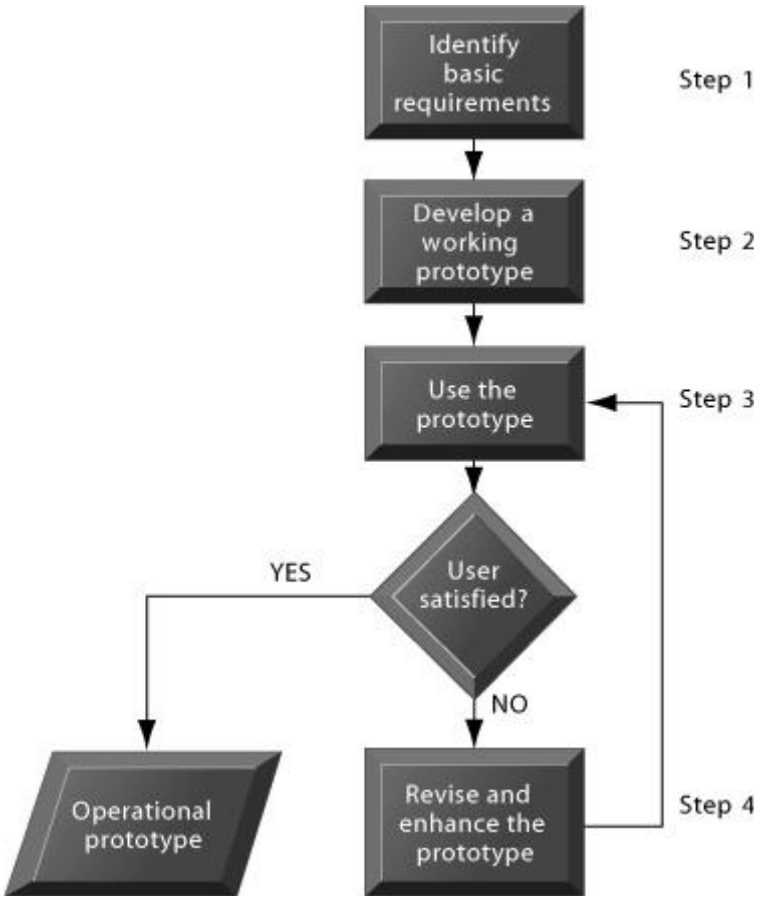

Figure 3.9: The prototyping process

When no more iterations are required, the approved prototype then becomes an operational prototype that furnishes the final specifications for the application. Sometimes the prototype is adopted as the production version of the system.

3.5.2.2 Advantages and Disadvantages of Prototyping

Prototyping is most *useful when there is some uncertainty about requirements or design solutions*. Prototyping is especially *useful in designing an information system's*

end-user interface (the part of the system with which end users interact, such as online display and data-entry screens, reports, or Web pages). Because prototyping encourages intense enduser involvement throughout the systems development life cycle, *it is more likely to produce systems that fulfill user requirements.*

However, rapid prototyping can gloss over essential steps in systems development. *If the completed prototype works reasonably well, management may not see the need for reprogramming, redesign, or full documentation and testing to build a polished production system.* Some of these hastily constructed systems may not easily accommodate large quantities of data or a large number of users in a production environment.

3.5.3 End-User Development

Some types of information systems can be developed by end users with little or no formal assistance from technical specialists. This phenomenon is called end-user development. A series of software tools categorized as fourth-generation languages makes this possible. ***Fourth-generation languages*** are software tools that enable end users to create reports or develop software applications with minimal or no technical assistance. Some of these fourth-generation tools also enhance professional programmers' productivity.

Fourth-generation languages tend to be nonprocedural, or less procedural, than conventional programming languages. Procedural languages require specification of the sequence of steps, or procedures, that tell the computer what to do and how to do it. Nonprocedural languages need only specify what has to be accomplished rather than provide details about how to carry out the task.

On the whole, end-user-developed systems can be completed more rapidly than those developed through the conventional systems life cycle. Allowing users to specify their own business needs improves requirements gathering and often leads to a higher level of user involvement and satisfaction with the system. However, fourth-generation tools still cannot replace conventional tools for some business applications because they cannot easily handle the processing of large numbers of transactions or applications with extensive procedural logic and updating requirements. End-user computing also poses organizational risks because it occurs outside of traditional mechanisms for information systems management and control. When systems are created rapidly, without a formal development methodology, testing and documentation may be

inadequate. Control over data can be lost in systems outside the traditional information systems department.

3.5.4 Application software packages

Many applications are common to all business organizations—for example, payroll, accounts receivable, general ledger, or inventory control. For such universal functions with standard processes that do not change a great deal over time, a generalized system will fulfill the requirements of many organizations.

If a software package can fulfill most of an organization's requirements, the company does not have to write its own software. The company can save time and money by using the prewritten, predesigned, pretested software programs from the package. Package vendors supply much of the ongoing maintenance and support for the system, including enhancements to keep the system in line with ongoing technical and business developments.

If an organization has unique requirements that the package does not address, many packages include capabilities for customization. Customization features allow a software package to be modified to meet an organization's unique requirements without destroying the integrity of the package software. If a great deal of customization is required, additional programming and customization work may become so expensive and time consuming that they negate many of the advantages of software packages.

3.5.5 Outsourcing

If a firm does not want to use its internal resources to build or operate information systems, it can outsource the work to an external organization that specializes in providing these services. Application service providers (ASPs), which we describe in module 2, are one form of outsourcing. In another form of outsourcing, a company could hire an external vendor to design and create the software for its system. The outsourcing vendor might be domestic or in another country.

Outsourcing has become popular because some organizations perceive it as providing more value than an in-house computer center or information systems staff. Not all organizations benefit from outsourcing, and the disadvantages of outsourcing can create serious problems for organizations if they are not well understood and managed. Many firms underestimate costs for identifying and evaluating vendors of information technology services, for transitioning to a new vendor, and for monitoring

vendors to make sure they are fulfilling their contractual obligations. These hidden costs can easily undercut anticipated benefits from outsourcing. Table 3.2 compares the advantages and disadvantages of each of the systems-building alternatives.

Approach	Features	Advantages	Disadvantages
Systems life cycle	Sequential step-by-step formal process Written specification and approvals Limited role of users	Useful for large, complex systems and projects	Slow and expensive Discourages changes Massive paperwork to manage
Prototyping	Requirements specified dynamically with experimental system Rapid, informal, and iterative process Users continually interact with the prototype	Rapid and relatively inexpensive Useful when requirements uncertain or when end-user interface is very important Promotes user participation	Inappropriate for large, complex systems Can gloss over steps in analysis, documentation, and testing
Application software package	Commercial software eliminates need for internally developed software programs	Design, programming, installation, and maintenance work reduced Can save time and cost when developing common business applications Reduces need for internal information systems resources	May not meet organization's unique requirements May not perform many business functions well Extensive customization raises development costs
End-user development	Systems created by end users using fourth-generation software tools Rapid and informal Minimal role of information systems specialists	Users control systems-building Saves development time and cost Reduces application backlog	Can lead to proliferation of uncontrolled information systems and data Systems do not always meet quality assurance standards
Outsourcing	Systems built and sometimes operated by external vendor	Can reduce or control costs Can produce systems when internal resources are not available or technically deficient	Loss of control over the information systems function Dependence on the technical direction and prosperity of external vendors

Table 3.2: Comparison of Systems-Development Approaches

3.6 Rapid Application Development

Object-oriented software tools, reusable software, prototyping, and fourth-generation language tools are helping systems builders create working systems much more rapidly than they could using traditional systems-building methods and software tools. The term ***Rapid Application Development (RAD)*** is used to describe this process of creating workable systems in a very short period of time. RAD can include the use of visual programming and other tools for building graphical user interfaces, iterative prototyping of key system elements, the automation of program code generation, and close teamwork among end users and information systems specialists.

Sometimes a technique called ***Joint Application Design (JAD)*** is used to accelerate the generation of information requirements and to develop the initial systems design. JAD brings end users and information systems specialists together in an interactive session to discuss the system's design. Properly prepared and facilitated, JAD sessions can significantly speed up the design phase and involve users at an intense level.

3.7 The concept of Implementation

To manage the organizational change surrounding the introduction of a new information system effectively, the process of implementation must be examined. ***Implementation*** refers to all organizational activities working toward the adoption, management, and routinization of an innovation, such as a new information system.

3.7.1 Causes of Implementation Success and Failure

An implementation process whether it is a success or failure is determined by a set of factors. These factors that determines implementation outcome are the following:

- The role of users in the implementation process
- The degree of management support for and commitment to the implementation effort
- The level of complexity and risk of the implementation project
- The quality of management of the implementation process

Figure 3.10 illustrates the different information system success and failure factors and the different areas of information system (design, cost, operations and data) that are reflected by the information system success or failure.

3.7.1.1 User Involvement and Influence

User involvement in the design and operation of information systems has several positive results. First, if users are heavily involved in systems design, they have more opportunities to mold the system according to their priorities and business requirements, and more opportunities to control the outcome. Second, they are more likely to react positively to the completed system because they have been active participants in the change process.

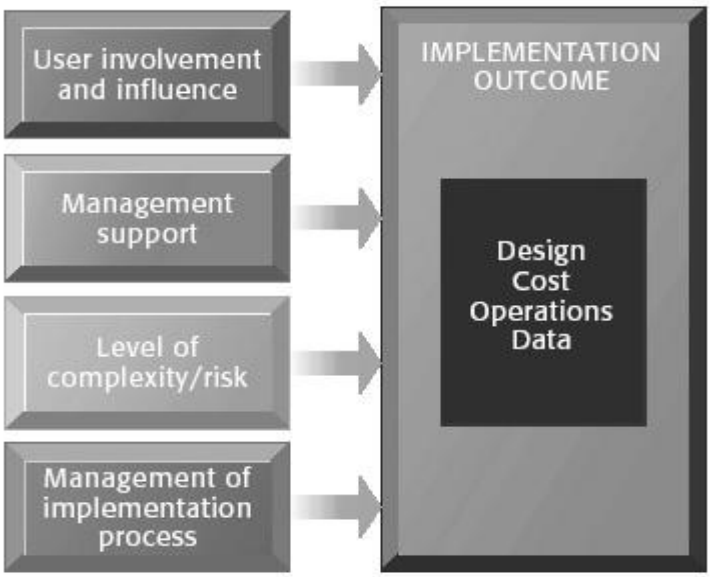

Figure 3.10: Information systems success or failure factors

Systems development projects run a very high risk of failure when there is a pronounced communication gap between users and technicians and when these groups continue to pursue different goals. Under such conditions, users are often driven out of the implementation process. Because they cannot comprehend what the technicians are saying, users conclude that the entire project is best left in the hands of the information specialists alone. With so many implementation efforts guided by purely technical considerations, it is no wonder that many systems fail to serve organizational needs.

3.7.1.2 Management Support and Commitment

If an information systems project has the backing and commitment of management at various levels, it is more likely to be perceived positively by both users and the technical information services staff. Both groups will believe that their participation in the development process will receive higher-level attention and priority. They will be recognized and rewarded for the time and effort they devote to implementation. Management backing also ensures that a systems project receives sufficient funding and resources to be successful.

3.7.1.3 Level of Complexity and Risk

Systems differ dramatically in their size, scope, level of complexity, and organizational and technical components. Some systems development projects are more likely to fail or suffer delays because they carry a much higher level of risk than others.

The level of project risk is influenced by project size, project structure, and the level of technical expertise of the information systems staff and project team.

3.7.1.4 Management of the Implementation Process

The development of a new system must be carefully managed and orchestrated, and the way a project is executed is likely to be the most important factor influencing its outcome. Training to ensure that end users are comfortable with the new system and fully understand its potential uses is often sacrificed or forgotten in systems development projects. If the budget is strained at the very beginning, toward the end of a project there will likely be insufficient funds for training and documentation.

A systems development project without proper management will most likely suffer these consequences:

- Costs that vastly exceed budgets
- Unexpected time slippage
- Technical shortfalls resulting in performance that is significantly below the estimated level
- Failure to obtain anticipated benefits

3.7.2 Managing Implementation

All aspects of the implementation process cannot be easily controlled or planned. However, anticipating potential implementation problems and applying appropriate corrective strategies can increase the chances for system success. Various project management, requirements gathering, and planning methodologies have been developed for specific categories of problems. Strategies have also been devised for ensuring that users play appropriate roles throughout the implementation period and for managing the organizational change process.

3.7.2.1 Controlling Risk Factors

The first step in managing project risk involves identifying the nature and level of risk confronting the project. Then these risks associated with the project must be controlled. There are different ways of controlling risks.

3.7.2.1.1 Managing Technical Complexity

Projects with challenging and complex technology to master benefit from internal integration tools. The success of such projects depends on how well their technical complexity can be managed. Project leaders need both heavy technical and administrative experience. They must be able to anticipate problems and develop smooth working relationships among a predominantly technical team. The team should

be under the leadership of a manager with a strong technical and project management background and team members should be highly experienced. Team meetings should take place frequently. Essential technical skills or expertise not available internally should be secured from outside the organization.

3.7.2.1.2 Formal Planning and Control tools

Large projects benefit from appropriate use of formal planning tools and formal control tools. With project management techniques, such as Program Evaluation and Review Technique (PERT) or Gantt charts, a detailed plan can be developed. ***PERT*** lists the specific activities that make up a project, their duration, and the activities that must be completed before a specific activity can start. A ***Gantt chart*** visually represents the sequence and timing of different tasks in a development project as well as their resource requirements.

3.7.2.1.3 Increasing User Involvement and Overcoming User Resistance

Projects with relatively little structure and many undefined requirements must involve users fully at all stages. Users must be mobilized to support one of many possible design options and to remain committed to a single design. External integration tools consist of ways to link the work of the implementation team to users at all organizational levels.

Participation in implementation activities may not be enough to overcome the problem of user resistance. The implementation process demands organizational change. Such change may be resisted because different users may be affected by the system in different ways. Whereas some users may welcome a new system because it brings changes they perceive as beneficial to them, others may resist these changes because they believe the shifts are detrimental to their interests.

If the use of a system is voluntary, users may choose to avoid it; if use is mandatory, resistance will take the form of increased error rates, disruptions, turnover, and even sabotage. Therefore, the implementation strategy must not only encourage user participation and involvement, but it must also address the issue of counter implementation. ***Counter implementation*** is a deliberate strategy to thwart the implementation of an information system or an innovation in an organization.

3.7.3 Designing for the Organization

Because the purpose of a new system is to improve the organization's performance, the systems development process must explicitly address the ways in

which the organization will change when the new system is installed, including installation of intranets, extranets, and Web applications. In addition to procedural changes, transformations in job functions, organizational structure, power relationships, and behavior should be carefully planned.

Areas where users interface with the system should be carefully designed, with sensitivity to ergonomic issues. ***Ergonomics*** refers to the interaction of people and machines in the work environment. It considers the design of jobs, health issues, and the end-user interface of information systems. The impact of the application system on the work environment and job dimensions must be
carefully assessed.

Most contemporary systems-building approaches tend to treat end users as essential to the systems-building process but as playing a largely passive role relative to other forces shaping the system, such as the specialist systems designers and management. A ***sociotechnical design*** plan establishes human objectives for the system that lead to increased job satisfaction. Designers set forth separate sets of technical and social design solutions. The proposed technical solutions are compared with the proposed social solutions. Social and technical solutions that can be combined are proposed as sociotechnical solutions. The alternative that best meets both social and technical objectives is selected for the final design. The resulting sociotechnical design is expected to produce an information system that blends technical efficiency with sensitivity to organizational and human needs, leading to high job satisfaction.

3.8 Pitfalls in MIS Development

There are different issues that adversely affect the MIS development process. The following are the main hindering factors to the development of MIS.

1. ***Lack of proper objectives and goals for many MIS projects***. Too many MIS projects are embarked upon by organizations without clear objectives and goals, which lead them into not being able to design and implement a good MIS. With a lack of objectives many organizations have failed to coordinate their information systems.
2. ***Lack of management participation and development***. There has been a general lack of top management participation and involvement in many MIS projects. Too many MIS projects are left to project leaders who neither have the authority nor know much about the management process.

3. ***Control measures and evaluative criteria are inadequate or absent.*** Lack of control is a direct result of lack of management involvement and results in a lack of proper MIS objectives. Many organizations that have control measures have found them dysfunctional to the MIS development. For example, tight controls have put pressure on MIS executives, forcing them only to seek short cuts.
4. ***False assumptions are used in designing* MIS**. MIS information analysts must make many assumptions, some of which hold while others do not. False assumptions often lead to improperly defined system objectives.
5. ***Inability of many system designers to identify information needs for managers***. It is very difficult to determine the information needs for managers. Because managers, by the nature of some of their decisions, cannot determine their own information needs. System designers, not being sure what information the managers actually require, go ahead and design systems they hope will satisfy the managers. Sometimes systems designers are able to determine the information needs of the operating management and they try to apply those needs to all the other management levels, which is of little use.
6. ***Lack of flexibility in many MIS.*** Some MISs fails the test of time when the environment they were designed for changes. Any MIS requires a degree of flexibility to meet changing and diverse information needs of managers.
7. ***Poor implementation program.*** Just as a good MIS design program is required, a good implementation program is also of high priority. Many well designed MISs have failed because of lack of proper implementation programs.
8. ***Too much emphasis is placed on technical aspects while placing relatively little emphasis on human factors.*** Many MISs have been designed around only technical factors giving little or no consideration at all to the humans who will use the system. Many good technically designed systems have failed to work because the designers have failed to recognize this important man/machine relationship.