Unit 1

Definition of a System and Its Parts

A **system** is an interrelated set of business procedures (or components) used within one business unit, working together for some purpose. **System is the collection of different interrelated subsystems that perform specific task.**

System is the set of components that interact to achieve common goal. **For example,** a system in the payroll department keeps track of checks, whereas an inventory system keeps track of supplies. The two systems are separate. A system has nine characteristics, seven of which are shown in the Figure 1-4. A detailed explanation of each characteristic follows, but from the figure you can see that a system exists within a larger world, an environment. A boundary separates the system from its environment. The system takes input from outside, processes it, and sends the resulting output back to its environment. The arrows in the figure show this interaction between the system and the world outside of it.

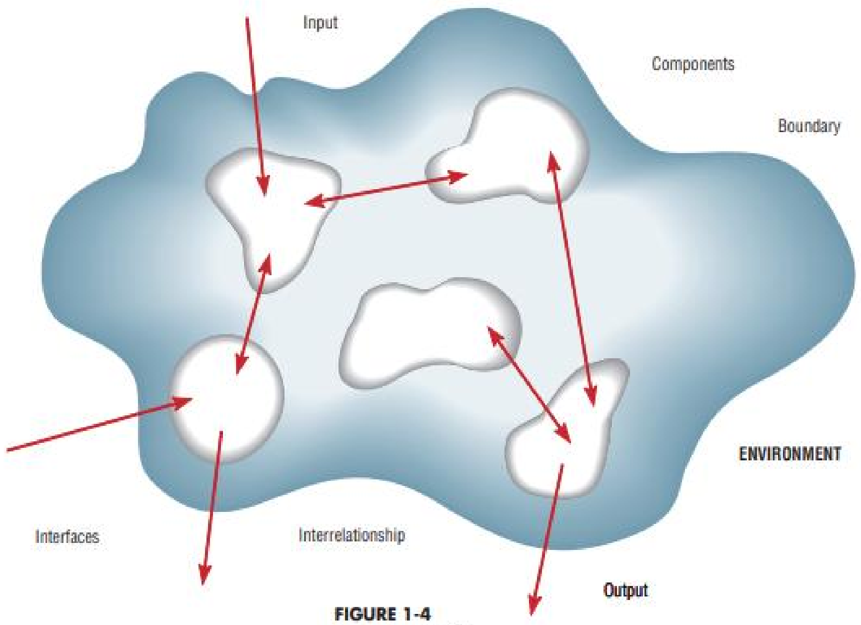


FIGURE 1-4
Seven characteristics of a system.

1. Components **2. Interrelated components**

3. Boundary **4. Purpose**

5. Environment **6. Interfaces**

7. Constraints **8. Input**

9. Output

Components:

An irreducible part or aggregation of parts that makes up a system is called the component of the system which is also called a subsystem.

Interrelated Component:

Dependence of one part of the system on one or more other system parts. If any component is dependent on any other one or more components such component is called the interrelated component.

Boundary:

The line that marks the inside and outside of a system and that sets off the system from its environment is the boundary of the system. Boundary separates the system and environment.

Purpose:

The overall goal or function of a system is its purpose.

Environment:

Everything external to a system that interacts with the system is the environment.

Interface:

Point of contact where a system meets its environment or where subsystems meet each other.

Constraint:

A limit to what a system can accomplish is the constraints for that system. We can also define the rules and regulations required for that system as the constraints.

Decomposition:
The process of breaking the description of a system down into small components also known as functional decomposition.

Input/Output: Anything given to the system from the environment to accomplish any task is called the input. The system processes those inputs to produce some results that is given to external of the boundary generally known as output.

System analysis:

Systems analysis is a **process of understanding** in detail what a system should accomplish. It is about understanding the goals and strategies of the business and defining the information requirements that support those goals and strategies. Most importantly systems analysis is not about programming but the study about the system.

System design:

It is the process of adopting/choosing the one among the many, which best accomplishes the user needs. So, simply, it is compromising mechanism. It is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. System can be defined in graphical or textual modelling languages.

Important System Concepts

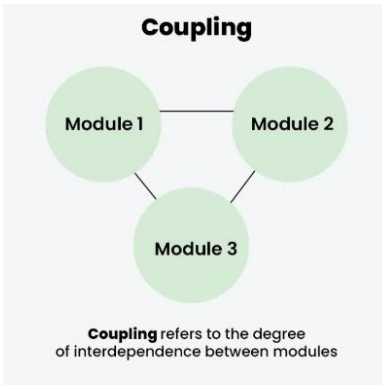
Systems analysts need to know several other important systems concepts:

- Decomposition
- Modularity
- Coupling
- Cohesion

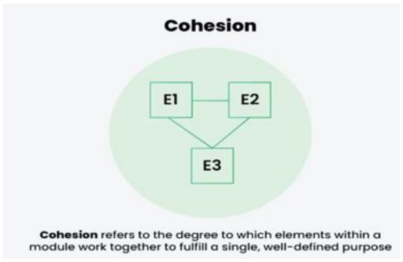
Decomposition is the process of breaking down a system into its smaller components. These components may themselves be systems (subsystems) and can be broken down into their components as well. The process of breaking the description of a system down into small components also known as functional decomposition.

Modularity is a direct result of decomposition. It refers to dividing a system into chunks or modules of a relatively uniform size. Modules can represent a system simply, making it easier to understand and easier to redesign and rebuild.

Coupling means that subsystems are dependent on each other. Subsystems should be as independent as possible. If one subsystem fails and other subsystems are highly dependent on it, the others will either fail themselves or have problems functioning. It refers to the degree of interdependence between software modules.High coupling means that module are closely connected and changes in one module may affect other modules. Low coupling means that modules are independent, and changes in one module have little impact on other modules


Coupling
Module 1
Module 2
Module 3
Coupling refers to the degree of interdependence between modules

Cohesion is the extent to which a subsystem performs a single function. It refers to the degree to which elements within a module work together to fulfill a single, well defined purpose. High cohesion means that elements are closely related and focused on a single purpose, while low cohesion means that elements are loosely related and serve multiple purposes.


Cohesion
E1
E2
E3
Cohesion refers to the degree to which elements within a module work together to fulfill a single, well-defined purpose

Coupling and **Cohesion** are two key concepts in software engineering that are used to measure the quality of software design. Low coupling and high cohesion is considered best for any s/w development.

System development life cycle:

System development life cycle is an organizational process of developing and maintaining systems. It helps in establishing a system project plan, because it gives overall list of processes and sub-processes required for developing a system. System development life cycle means combination of various activities. In other words we can say that various activities put together are referred as system development life cycle. In the system analysis and design terminology, the system development life cycle also means software development life cycle. Phases of SDLC are given as follows

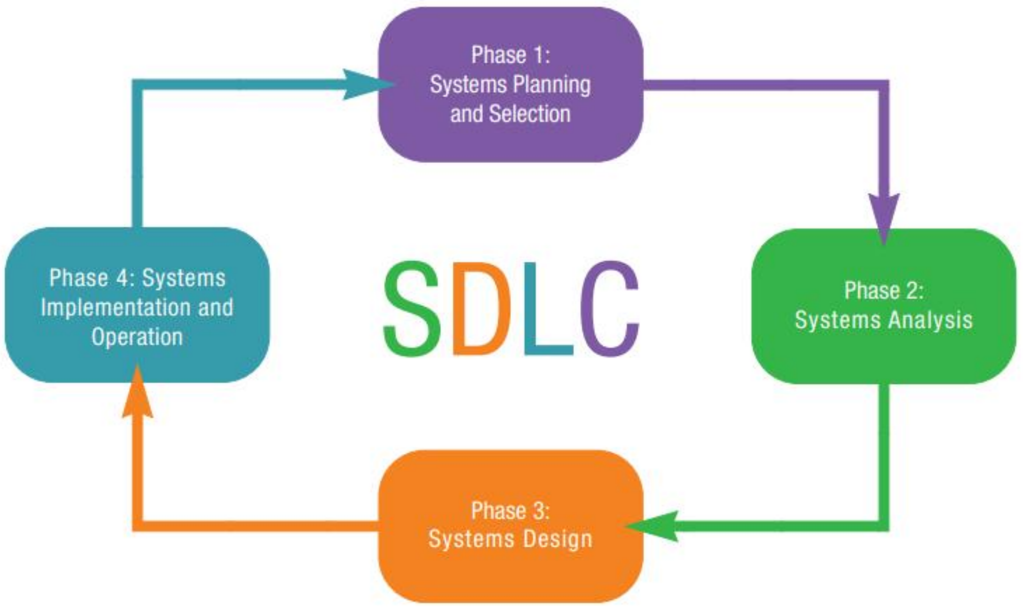

Phase 1:
Systems Planning and Selection
Phase 2:
Systems Analysis
Phase 3:
Systems Design
Phase 4: Systems Implementation and Operation
SDLC

Fig: System Development Life Cycle

•**Requirement study**

Deals with why build this system?

•Feasibility study

The feasibility analysis examines key aspects of the proposed project:

The technical feasibility (can we build it?)

The economic feasibility (will it provide business value?)

The operational feasibility (if we build it, will it be used?)

•Analysis

Deals with the study about the system **who, why, what, when, where, when and how** for the system?

•Design

How will this system work?

•Coding

How to get output from input?

•Testing

Whether the system produces desired output?

•Implementation

Delivery & support for completed system.

•Maintenance

Maintenance is necessary to eliminate errors in the system during its working life. It has been seen that there are always some errors found in the systems that must be noted and corrected.

The system analyst:

The analyst is the person who is responsible for designing and developing information system and act as liaison (link/communication) between users and its professionals. A systems analyst, also known as business technology analyst, is an information technology (IT) professional who **specializes in analyzing, designing and implementing information systems**. They may be responsible for developing cost analysis, design considerations, staff impact amelioration (betterment), and implementation timelines.

System analyst plan and control the computer system analysis and development for assigned area; serve as Project Leader on major projects; confer with and advise use departments; evaluate user requests and needs, estimate cost and time of implementation and recommend program methodology to be followed, assuring programming compliance with established documentation standards; provide technical guidance and recommendations concerning existing computer programs and systems.

Duties and Responsibilities of system analyst:

i. Plan and control the computer system analysis and development for assigned area grant with and advise use departments; evaluate user requests and needs, estimate cost and time of implementation.
ii. Serve as Project Leader on major projects; evaluate project requirements and time lines; provide guidance and direction to assigned personnel and coordinate project phases.
iii. Assist in system studies in programming for various applications; recommend program methodology to be followed, assuring programming compliance with established documentation standards.
iv. Analyze problems outlined by users and potential users of data processing; study existing systems and procedures and the introduction of potential data processing systems.
v. Develop detailed data flow charts of existing system, documenting the work process according to installation standards; assist with the development of system objectives and comprehensive plans to organize work methodology; establish controls to assure desired output in new and modified applications.
vi. Assist department staff by advising on applications development and the best approach to system design relative to software capabilities and hardware features; review or determine application proposals and requirements as required.
vii. Provide technical support, assistance and information to users; train users in system operations as necessary; coordinate communication and activities with

users to review and analyze user problems and needs; provide work direction to assigned programmers.

viii. Compile information and data and prepare various reports related to computer systems and functions; document system software and hardware as necessary.

ix. Maintain a variety of records, accounts, logs and files related to systems; prepare and purge records as necessary.

x. Write or modify programs as necessary to meet user needs; prepare block diagrams and flow charts; write or modify program source code; prepare sample test data; test, correct and revise programs as necessary.

xi. Perform other duties as assigned.

xii. Maintain regular attendance.

What is the key role of a systems analyst?

A systems analyst serves as a business professional who uses analysis and design techniques to solve business problems using information technology.

System analyst can perform

- examining current systems.
- talking to users (requirements gathering)
- producing specifications for new or modified systems.
- cooperate with other IT staff such as programmers to produce new systems.
- implementing new systems.

The three primary roles of the systems analyst are **consultant, supporting expert, and agent of change**.

The skills of the system analyst:

- Critical thinking ability.
- Strong problem-solving capacity.
- High-level written and verbal communication skills.
- Project management skills.
- Ability to work under pressure and to tight deadlines.

Knowledge and abilities of system analyst:

Knowledge:

Advanced principles and techniques of systems analysis, design and programming.

Advanced principles of internal system maintenance.

Analysis project coordination requirements.

Database structures, on-line applications and system capabilities of the District.

Database telecommunications design. Programming languages such as COBOL and Job control language concepts.

Principles and concepts involved in computer programming and maintenance.

Structured programming practices and techniques.

Programming and computer operation documentation.

Interpersonal skills using tact, patience and courtesy.

Technical aspects of field of specialty.

Abilities:

Coordinate, oversee, analyze and maintain computer systems.

Direct and coordinate major analysis and programming projects.

Provide technical guidance and recommendations concerning existing computer programs and systems. Apply principles and techniques of computer programming to specific problems and processes. Research, analyze and recommend new system software and hardware.

Write or modify programs to meet user needs.

Code data into machine language.

Initiate procedural modifications.

Demonstrate proficiency in appropriate program languages.

Anticipate system space capacity requirements.

Provide assistance to other data processing personnel regarding technical problems.

Plan and organize work.

Meet schedules and time lines.

Work independently with little direction.

Establish and maintain cooperative and effective working relationships with others.

Central Repository

CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place ofstorage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.

The Systems Development Environment

Introduction

Information Systems Analysis and Design (ISAD)

Complex organizational process.

Used to develop and maintain computer-basedinformation systems.

Used by a team of businessand systems professionals.

An organizational improvement process (responds to and anticipate problems).

Uses technology (Internet, WWW marketing,online business, eBay, Amazon.com etc).

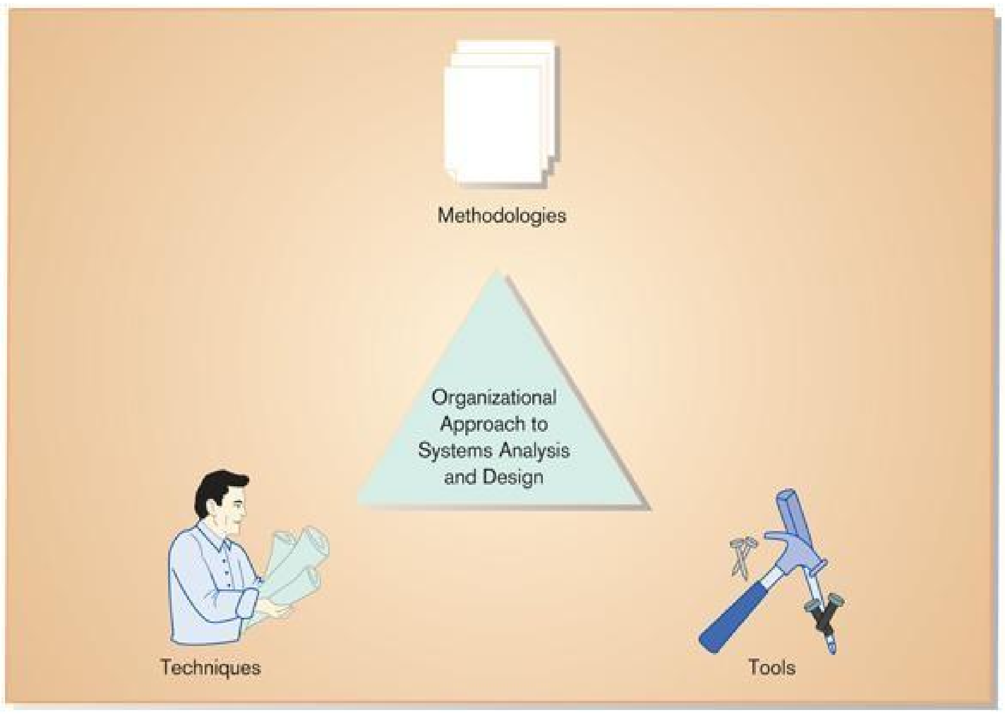


Figure 1-1 An organizational approach to systems analysis and design is driven by methodologies, techniques, and tools

An output of system analysis and design is **application software**. **Methodologies** are comprehensive, multiple-step approaches to system development. Methodologies uses different techniques. **Techniques** are particular processes that you as an analyst, will follow to ensure your work is well thought out, complete, and comprehensive to others on your project team. Eg: conducting interviews to determine what your system should do, diagramming the system's logic, designing the reports your system will generate. **Tools** are computer programs that make it easy to use and benefit from techniques.

A Modern Approach to Systems Analysis and Design

1950s: All applications had to be developed in machine language or assembly language. They had to be developed from scratch because due to the absence of software industry.

1960s: Smaller, faster, less expensive computers, beginning of the software industry, use in-house development.

1970s: Realized how expensive to develop customized information system for every application started development of database management system.

1980s: The software industry expended greatly, CASE(computer aided software engineering) tools. Started writing application software in oop languages, graphics were used, developed less software in-house and bought more from software vendors.

1990s: Focus on system integration,GUI (Graphical user interface) applications, client/server platforms, Internet.

The new century: Web application development, wireless PDAs (personal digital assistants, eg pocket PCs), ASP(application service provider).

Application Software

Computer software designed to support organizational functions or processes.

Systems Analyst

Organizational role most responsible for analysisand design of information systems.

Types of Information Systems and Systems Development

Transaction Processing Systems (TPS)

- Automate handling of data about businessactivities (transactions)
- Process orientation

Management Information Systems (MIS)

- Converts raw data from transaction processing system into meaningful form
- Data orientation

Decision Support Systems (DSS)

- Designed to help decision makers
- Provides interactive environment for decisionmaking
- Involves data warehouses, executive informationsystems (EIS)
- Database, model base, user dialogue

Summary of Information Systems Types

Developing Information Systems

System Development Methodology is a standard process followed in an organization to conduct all the steps necessary to analyze, design, implement, and maintain information systems.

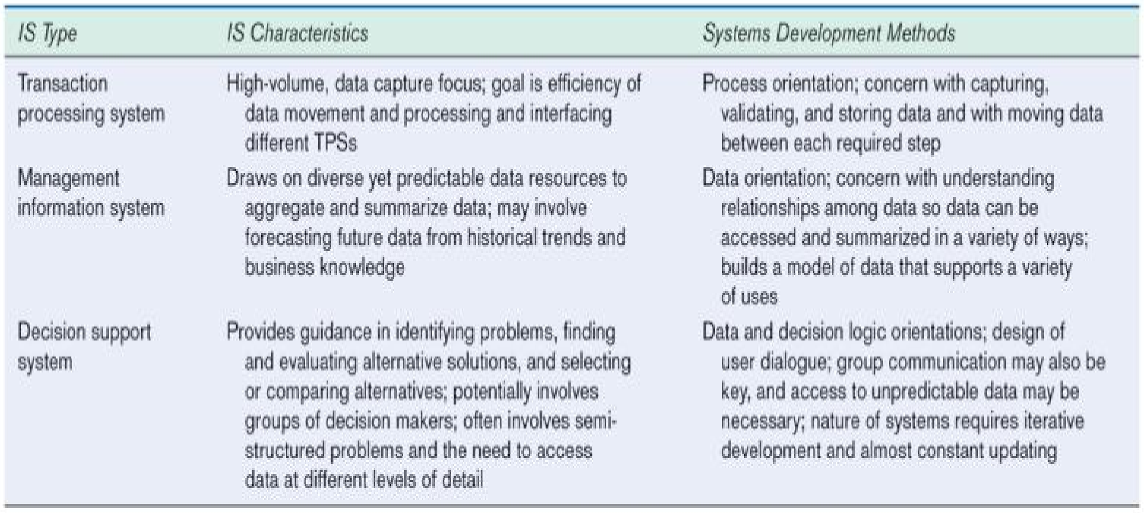


Table 1-1 Systems Development for Different IS Types

IS Type	*IS Characteristics*	*Systems Development Methods*
Transaction processing system	High-volume, data capture focus; goal is efficiency of data movement and processing and interfacing different TPSs	Process orientation; concern with capturing, validating, and storing data and with moving data between each required step
Management information system	Draws on diverse yet predictable data resources to aggregate and summarize data; may involve forecasting future data from historical trends and business knowledge	Data orientation; concern with understanding relationships among data so data can be accessed and summarized in a variety of ways; builds a model of data that supports a variety of uses
Decision support system	Provides guidance in identifying problems, finding and evaluating alternative solutions, and selecting or comparing alternatives; potentially involves groups of decision makers; often involves semi-structured problems and the need to access data at different levels of detail	Data and decision logic orientations; design of user dialogue; group communication may also be key, and access to unpredictable data may be necessary; nature of systems requires iterative development and almost constant updating

Systems Development Life Cycle (SDLC)

Traditional methodology used to develop, maintain,and replace information systems.

Phases in SDLC:

Planning

Analysis

Design

Implementation

Maintenance

Standard and Evolutionary Views of SDLC

Figure 1-3 The systems development life cycle

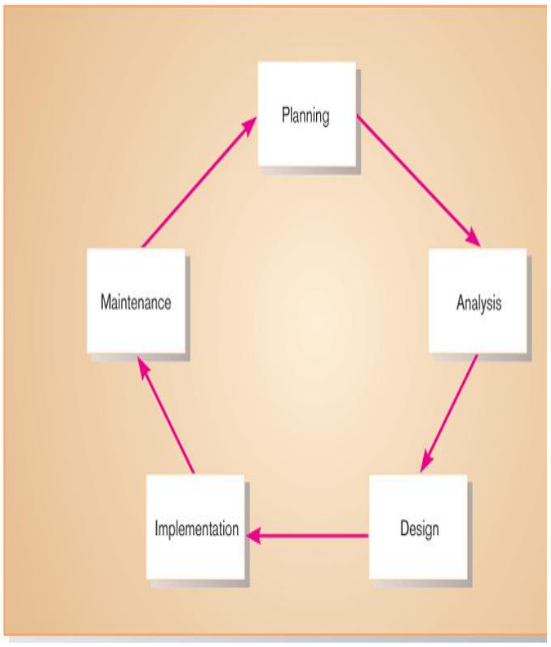
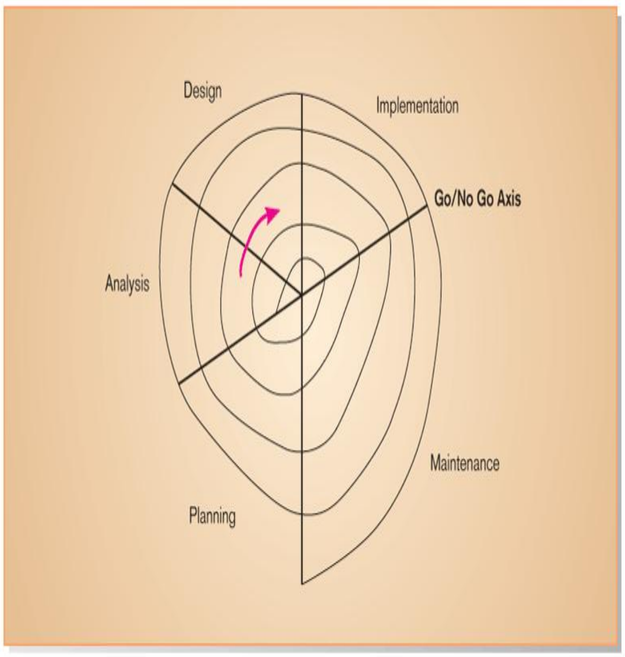


Figure 1-4 Evolutionary model SDLC



Planning – an organization's total information system needs are identified, analyzed, prioritized, and arranged. The outcome of the project identification and selection process is a determination of which system development project should be undertaken by the organization, at least in terms of initial study.

Major activities during planning are:

i) Investigation of the system problem or opportunity.

ii) Presentation of reasons why the system should or should not be developed by the organization.

iii) Determining the **scope** of the proposed system.

Also produce a specific plan for the proposed project the team will follow. This specifies the time and resources needed for execution.

Also identify whether the costs of developing system would give benefit.

Analysis: The second phase in the SDLC is **analysis.** During this phase, the analyst thoroughly studies the organization'scurrent procedures and the information systems used to perform organizational tasks. Analysis has **two sub phases**. The **first** is requirements determination. In this sub phase, analysts work with users to determine what the users want from a proposed system. The requirements determination process usually involves a careful study of any current systems, manual and computerized, that might be replaced or enhanced as part of the project. In the **second** part of analysis, analysts study the requirements and structure them according to their interrelationships and eliminate any redundancies. The output of the analysis phase is a description of (but not a detailed design for) the alternative solution recommended by the analysis team. Once the recommendation is accepted by those with funding authority, the analysts can begin to make plans to acquire any hardware and system software necessary to buildor operate the system as proposed.

Design: The third phase in the SDLC is **design. During design, analysts convert the** description of the recommended alternative solution into **logical** and then **physical** system specifications. The analysts must design all aspects of the system, from input and output screens to reports, databases,and computer processes. The analysts must then provide the physical specifics of the system they have designed, either asa model or as detailed documentation, to guide those who will build the new system. That part of the design process that is independent of any specific hardware or software platform is referred to as **logical design. Theoretically, the system could** be implemented on any hardware and systems software. The idea is to make sure that the system functions as intended. Once the overall high-level design of the system is worked out, the analysts begin turning **logical specifications into physical ones**. This process is referred to as **physical design.** As part of physical design, analysts design the various parts of the system to perform the physical operations necessary to facilitate data capture, processing, and information output. This can be done in many ways, from creating a working model of the system to be implemented to writing detailed specifications describing all the different parts of the system and how they should be built. In many cases, the working model becomes the basis for the actual system to be used. During physical design, the analyst team must determine many of the physical details necessary to build the final system, from the programming language the system will be written in, to the database system that will storethe data, to the hardware platform on which the system will run.

Often the choices of language, database, and platform are already decided by the organization or by the client, and at this point these information technologies must be taken into account in the physical design of the system. The final product of the design phase is the physical system specifications in a form ready to be turned over to programmers and other system builders for construction. Figure 1-6 illustrates the differences between logical and physical design.

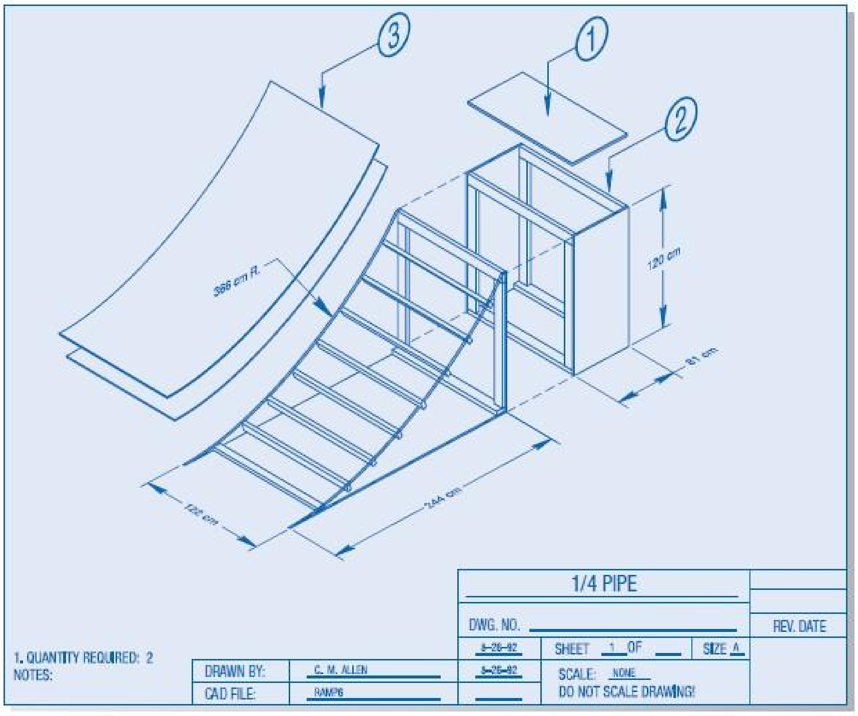
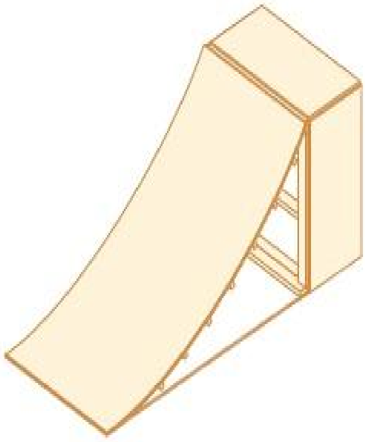


FIGURE 1-6
Difference between logical design and physical design
(a) A skateboard ramp blueprint (logical design)

(*Sources: www.tumyeto.com/tydu/skatebrd/organizations/plans/14pipe.jpg; www.tumyeto.com/tydu/skatebrd/organizations/plans/iuscblue.html.* Accessed September 16, 1999. Reprinted by permission of the International Association of Skateboard Companies.)


(b) A skateboard ramp (physical design)

Implementation: The fourth phase in the SDLC is **implementation.** The physical system specifications, whether in the form of a detailed model or as detailed written specifications, are turned over to programmers as the first part of the implementation phase. During implementation, analysts turn system specifications into a working system that is tested and then put into use. **Implementation includes coding, testing, and installation**.

During **coding**, programmers write the programs that make up the system. Sometimes the code is generated by the same system used to build the detailed model of the 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 part of the dailyactivities of the organization. Application software is installed, or loaded, on existing or new hardware, and users are introduced to the new system and trained. Testing and installation should be planned for as early as the project initiation and planning phase; both testing and installation require extensive analysis in order to develop exactly the right approach.

Finalization of documentation, training programs.

Note that documentation and training programs are finalized during implementation; documentation is produced throughout the life cycle.

Maintenance: The fifth and final phase in the SDLC is **maintenance. When asystem (including** its training, documentation, and support) 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 thesystem change over time. **In maintenance, programmers make the changes that users ask for and modify the system** to reflect evolving business conditions. These changes are necessary to keep the system running and useful. In a sense, maintenance is not a separate phase but a repetition of the other life cycle phases required to study and implement the needed changes. The amount of time and effort devoted to maintenance dependsa great deal on the performance of the previous phases of the life cycle. When an information system is no longer performing as desired, when maintenance costs become prohibitive, or when an organization's needs have changed substantially. Such problems indicate that it is time to begin designing the system's replacement, there by completing the loop and starting the life cycle over again.

The Heart of the Systems Development Process

Figure 1-8 The analysis–design–code–test loop

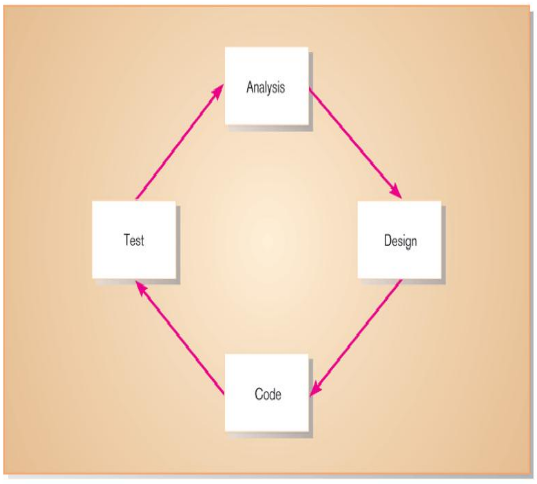
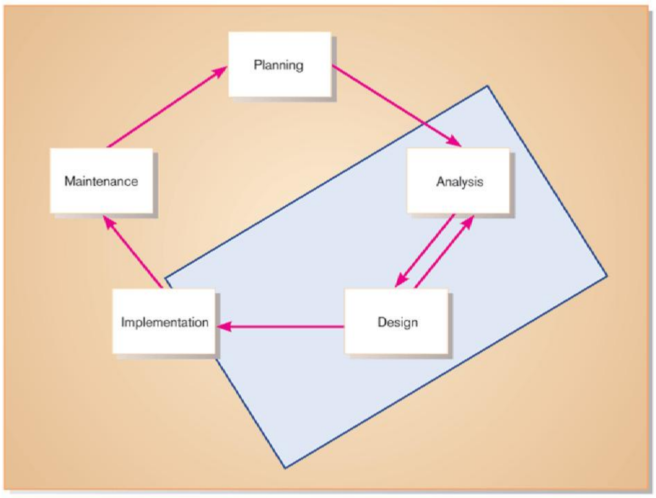


Figure 1-9 The heart of systems development



Current practice combines analysis, design, and implementation into asingle iterative and parallel process of activities

Traditional Waterfall SDLC

- Treat each phase as complete unto itself, never to be revisitedonce finished.
- Feedback came to be ignored in implementation.
- Traditionally,one phase ended and another began once a milestone had been reached.
- **System requirements** "locked in" after being determined (can't change). Once the milestone had been reached and the new phase initiated, it became difficult to go back. The enormous amount

of effort and time necessary to implement a specific design meant that it would be very expensive to make changes in a system once it was developed.

- Limited user involvement (only in requirements phase).

- In addition, under the traditional waterfall approach, nebulous and intangible processes such as analysis and design are given hard- and-fast dates for completion, and success is overwhelmingly measured by whether those dates are met.
- The focus on milestone deadlines, instead of on obtaining and interpreting feedback from the development process, leads to too little focus on doing good analysis and design. The focus on deadlines leads to systems that do not match users' needs and that require extensive maintenance, unnecessarily increasing development costs. Finding and fixing a software problem after thedelivery of the system is often far more expensive than finding andfixing it during analysis and design (Griss, 2003). The result of focusing on deadlines rather than on good practice is unnecessary rework and maintenance effort, both of which are expensive. According to some estimates, maintenance costs account for 40 to 70 percent of systems development costs (Dorfman and Thayer, 1997). Given these problems, people working in systems development began to look for better ways to conduct systems analysis and design.

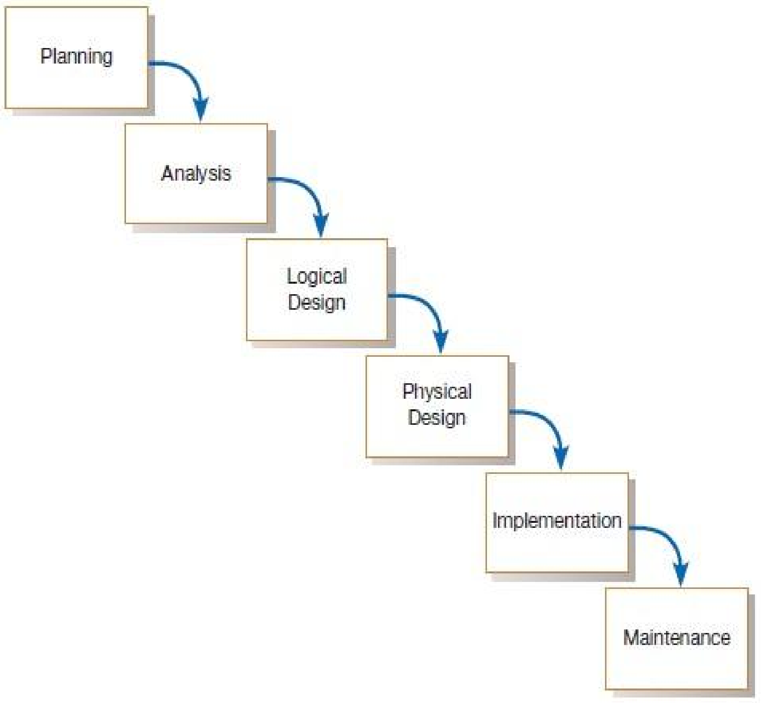


FIGURE 1-10
Traditional waterfall SDLC