CSC 311 Systems Analysis and Design

Normally, users, who demand reliable and cost-effective systems to manage their businesses, may not be fully aware of the capabilities and limitations of contemporary computer technology. On the other hand, application developers/programmers may not also understand how to satisfy the users of their applications. In all, this brought about a communication gap (lapse or divide) between the users and the developers of systems. Thus, in order to bridge this gap, there is the need for system analysis and design.

What is a system?
System is a collection of interacting or interrelated elements or components which form a unified entity or body operating harmoniously.
Elements of a System
Input-------Processing-------output
Control

Feedback

Properties of a System
Organization
- Implies structure and order
- Arrangement of components that helps to achieve predetermined objectives

Interaction: defined by the manner in which the components operate with each other
Interdependence: how the components of a system depend on one another
Integration: how a system's components are connected together
Central objective: the objective of a system must be central, which means it must have a major goal

Types of Systems
There are various types of systems; classification of systems can be done in many ways. Some of these categories are:
● physical or abstract: systems we can touch and feel, e.g. desks and chairs are part of a computer centre which are static
● open or closed systems
● natural and manufactured: natural e.g., solar system; manufactured = man-made system
● man-made information systems: e.g., political system, educational system, transport system, etc.

Categories of Man-made Systems
● Formal Information System: it is based on the flow of information in the form of memos, instructions, etc., from top level to lower levels of management.
● Informal Information System: this is employee based system which solves the day to day work related problems.

● Computer Based System: this system is directly dependent on the computer for managing business applications. For example, automatic library system, railway reservation system, banking system, etc.

Automated Systems
This type of man made system is based on electrical/electronic technology. Some of the common components are:
Hardware CPU, RAM, HDD, Storage devices, video cards, sound cards
Software – Systems software [operating system] and applications software.
Users of the system.
Data facts required for processing by the system.
Policies and regulations governing the normal operation of the system

Information Systems
An information system accepts data as input. After processing such data, information is produced for decision making or it is fed back into the system as another form of input. The main features of an information system are data input, control/processing, storage and information output. Typically, information systems contain databases (where records of data are stored permanently on various storage devices for future reference/access).

Information System Components
Data
● Tables store data
● Linked tables work together to supply data
Processes
● Describes the tasks and functions that users, managers, and IT staff members perform to achieve specific results
People
● Stakeholders: people that are financing and funding these IT projects
● Users (end user): the person that's putting the data in, or needs the data out of a system, the person that's interacting with the IT solutions

Types of Information Systems
Depending on the area of usage, the following types of information systems are available:

Information Reporting Systems – produce reports for decision makers in organizations (for day-to-day running of such organizations).
Transaction Processing Systems – store and process records from business transactions. These systems also provide relevant information about the growth of the company.
Executive Information System – are used for strategic studies of organizations operations; thus giving the management highly critical and summarized information.
Office Automation System provide effective and efficient office communication via the use of office software products, such as Microsoft Office.
Process Control Systems – play a vital role in monitoring as well as controlling physical process in

factories. Example of such systems is the auto-pilot software in sophisticated airplanes.
Decision-Support Systems – guide managers in decision making by providing information from the collections of fragmented facts.

System Analysis And Design

Systems Analysis
A process of collecting and interpreting facts, identifying the problems, and recommending feasible suggestions for improving system functioning. It is conducted for the purpose of studying a system or its parts in order to identify its objectives.

Systems Design
It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, you need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently. System Design focuses on how to accomplish the objectives of the system.

What is System Analysis & Design?
This is a structured and systematic process of studying how a system will work a and then designing and realizing such functions.

The Project Team
By now, it s clear that most of the day-to-day transactions that are based on manual systems need to be computerized for improved results. As such, a project team must be put in place to analyze and design a new system. This team usually, consists of the management/financiers, system analysts, system designer,system users, developers/programmers and auditors.

The Management
The management most times is the financier of the project. That is, they could be the managers of the users of the system. They usually provide firsthand knowledge about the need for a new/upgraded system and also give the system analyst relevant information, such as the nature of the working environment (networked or not,), number of staff, nature of business, number of transactions per day, etc.

The System Analyst
The role of the system analyst in project realization is very crucial since he is a mediator who bridges the communication gap/lapse between the various members of the team. His role has become very vital considering the approach previously used in system development; prior to the introduction of system analysis and design.
1. System development was only about generating codes which produce the application.
2. Programs were developed based on code-as-you-think principle without adequate commentaries, documentation and algorithms; thereby, making such codes difficult to understand (even by the programmers after some time).
3. More than one programmer could not develop an application, since there is no documentation

describing the design and implementation specifications of the system. Such applications will not satisfy the needs of the user.
4 Thus, the system analyst will study the present system identifying its setbacks and then give recommendations (and alternatives) for developing a new system. He produces a document containing the logical models of what the current system does and what the new proposed system will do. Most times, the system analyst is the project manager/leader.

The System Designer
Normally, a system analyst analyses and designs a system. But to be specific, a system designer transforms the less technical specifications of the logical models to a more detailed physical model for implementation. He produces a document that shows how the new system will function to achieve desired goals.
The System User
The system users are very important in the process of development of a new system since they are the ones to use the system. The user could either be the owner of the system or a client/customer who will just purchase the system. Thus, his suggestions along the way should be noted.
Developers
Programmers transform the design specifications to codes for realization of a functional system. At this point they would have to use a particular programming paradigm (such as JAVA, C/C++, Visual Basic, etc) for system development.
Auditors
Play a vital role in quality assurance; investigating whether the standards and the goals of the system have been reached.

Systems Development Tools
- Modeling
- Prototyping
- Computer-Aided Systems Engineering (CASE) Tools

Modeling
Business model
Requirements model
Process model
Data model
Object model
Network model

Prototyping
Early working versions of information systems
Speeds up the development process significantly
Important decisions might be made too early, before business or IT issues are thoroughly understood
A prototype based on careful fact-finding and modeling techniques can be an extremely valuable tool

Computer-Aided Software Engineering (CASE) Tools

● Provides an overall framework for systems development and supports a wide variety of design methodologies such as:
● Structure analysis
● Object-Oriented analysis
● Can generate program code which speeds the implementation process

System Development Methods

Structured Analysis
• Traditional method for developing systems
• Organized into phases

Object-Oriented Analysis
• More recent method for developing systems
• Objects represent actual people, things or events

Agile/Adaptive Methods
• Latest trend in software development
• Team-based effort broken down into cycles

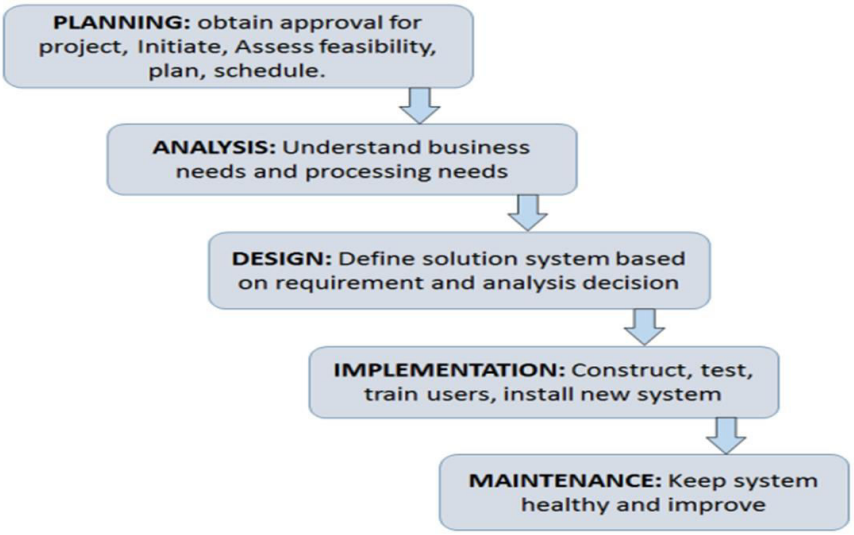
Structured Analysis

Time-tested and easy to understand
Uses phases called the Systems Development Life Cycle (SDLC)
Predictive approach
Uses process models to describe a system graphically

System Life Cycle

System 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 the various activities put together are referred to as System Development Life Cycle (SDLC) or Software Development Life Cycle (SDLC).

The SDLC model usually includes five steps:
● Systems Planning
● Systems Analysis
● Systems Design
● Systems Implementation
● Maintenance



Each phase is itself composed of a series of steps, which rely on techniques that produce *deliverable.*

Deliverable are specific documents and files that explain various elements of the system.

Each phase refines and elaborates on the work done previously.

Planning Phase

The fundamental process of understanding why an information system should be built/improved and determining how the project team will go about building/improving it, Purpose of this phase is to perform a preliminary investigation—a critical step, Key part of preliminary investigation is a feasibility study

Deliverable A feasibility report for the entire project is created at the end of this phase.

Elements of Planning Phase

Preliminary System Study

Feasibility Study

Detailed System Study

1. ***Preliminary System Study***

This is a brief

investigation of the system under consideration and gives a clear picture of what actually the physical system is? In practice, the initial system study involves the preparation of a System Proposal which lists the Problem Definition, Objectives of the Study, Terms of reference for Study, Constraints, Expected benefits of the new system, etc. in the light of the user requirements. The system proposal is prepared by the System Analyst (who studies the system) and places it before the user management. The management may accept the proposal and the cycle proceeds to the next stage. The management may also reject the proposal or request some modifications in the proposal. In summary, we would say that system study phase passes through the following steps:

Problem identification and project initiation

Background analysis

Inference or findings (system proposal)

2. ***Feasibility Study***

The feasibility study is basically the test of the proposed system in the light

of its work ability, meeting user s requirements, effective use of resources and of course, the cost effectiveness. These are categorized as technical, operational, economic and schedule feasibility. The main goal of feasibility study is not to solve the problem but to achieve the scope. In the process of feasibility study, the cost and benefits are estimated with greater accuracy to find the Return On Investment (ROI). This also defines the resources needed to complete the detailed investigation. The result is a feasibility report submitted to the management.

3. ***Detailed System Study***

This involves detailed study of various operations performed by a system and their relationships within and outside the system. During this process, data are collected on the available files, decision points and transactions handled by the present system. Interviews, on-site observation and questionnaire are the tools used for detailed system study.

Steps in the Planning Phase

Idea: project identification and selection

Feasibility: technical, economic, organizational, etc.

Work plan: time estimation, work breakdown structure, scope management, etc.

Staff project: project sharing/tasking

Control and direct project: standards, documentation, case repository, etc.

System Analyst

Plays a key role in information systems development projects.

A person who is thoroughly aware of the system and guides the system development project by giving proper directions.

He is an expert having technical and interpersonal skills to carry out development tasks required at each phase.

Thus, the system analyst will study the present system identifying its setbacks and then give recommendations (and alternatives) for developing a new system or improving one.

He produces a document containing the logical models of:

What the current system does, and

What the new proposed system will do

The Role of a system Analyst

Key roles:

Analysts build a series of models, diagrams, and decision tables, and uses other descriptive tools and techniques

An analyst's most valuable skill is to listen

An effective analyst will involve users in every step of the development process

Other roles:

Defining and understanding the requirement of user through various Fact finding techniques.

Prioritizing the requirements by obtaining user consensus

Gathering the facts or information and acquires the opinions of users.

Maintains analysis and evaluation to arrive at appropriate system which is more user friendly.

Suggests many flexible alternative solutions, pick the best solution, and quantify cost and benefits.

Draw certain specifications which are easily understood by users and programmer in precise and detailed form.

System Analyst Skills

Knowledge, skills, and education

Technical skills

Communication skills

Business skills

Critical thinking skills

Education

Certification

Planning Phase: Development Methodologies

Methodologies: is the formalized approach in implementing SDLC

Waterfall Development

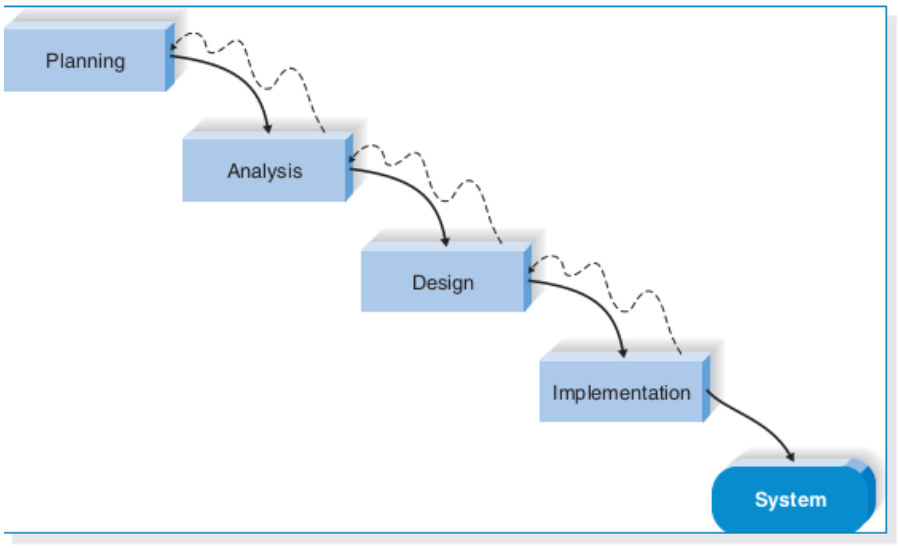
Rapid Application Development RAD

Agile Development

Fountain Development Model

Spiral Model

1. ***Waterfall Model****:* is one of the first software development model also known as the classical model, the waterfall model is intended to be irreversible but there are variations because you might need to go back in the previous phase in the waterfall.



Merits

Each phase has a deliverable

phases are processed one at a time

works well for smaller projects where requirements are clearly defined

simple and easy to understand

simple to manage due to rigidity

DE-Merits

It takes time to develop depending on how complex it is.

Once its in a testing stage it can not be possible to jump back.

Its not suitable for project where requirement its not clear.

The main theme is its good for short and clear projects

Variants of Waterfall model

- Parallel Model
- V Model

The parallel development methodologies evolved to address the lengthy time frame of waterfall development.

instead of doing the design and implementation in

sequence, a general design for the whole system is performed. Then the project is

divided into a series of sub projects that can be designed and implemented in parallel.

Once all sub projects are complete, there is a final integration of the separate pieces,

and the system is delivered.

Parallel development reduces the time required to deliver a system, so

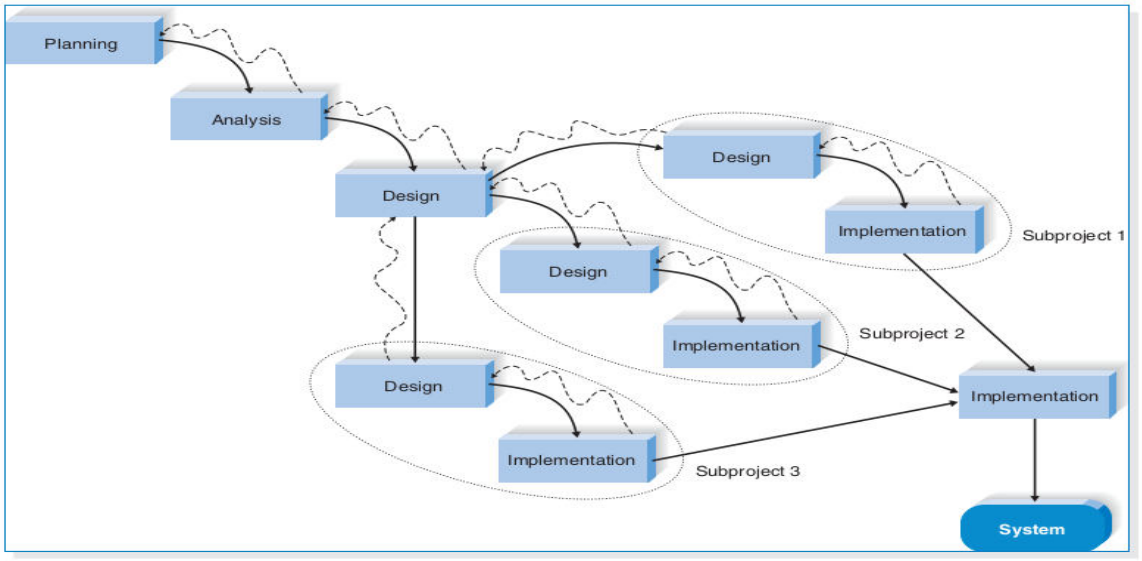
changes in the business environment are less likely to produce the need for rework.

The approach still suffers from problems caused by voluminous deliverable. It also

adds a new problem: If the sub projects are not completely independent, design

decisions in one sub project may affect another, and at the project end, integrating

the sub projects may be quite challenging.

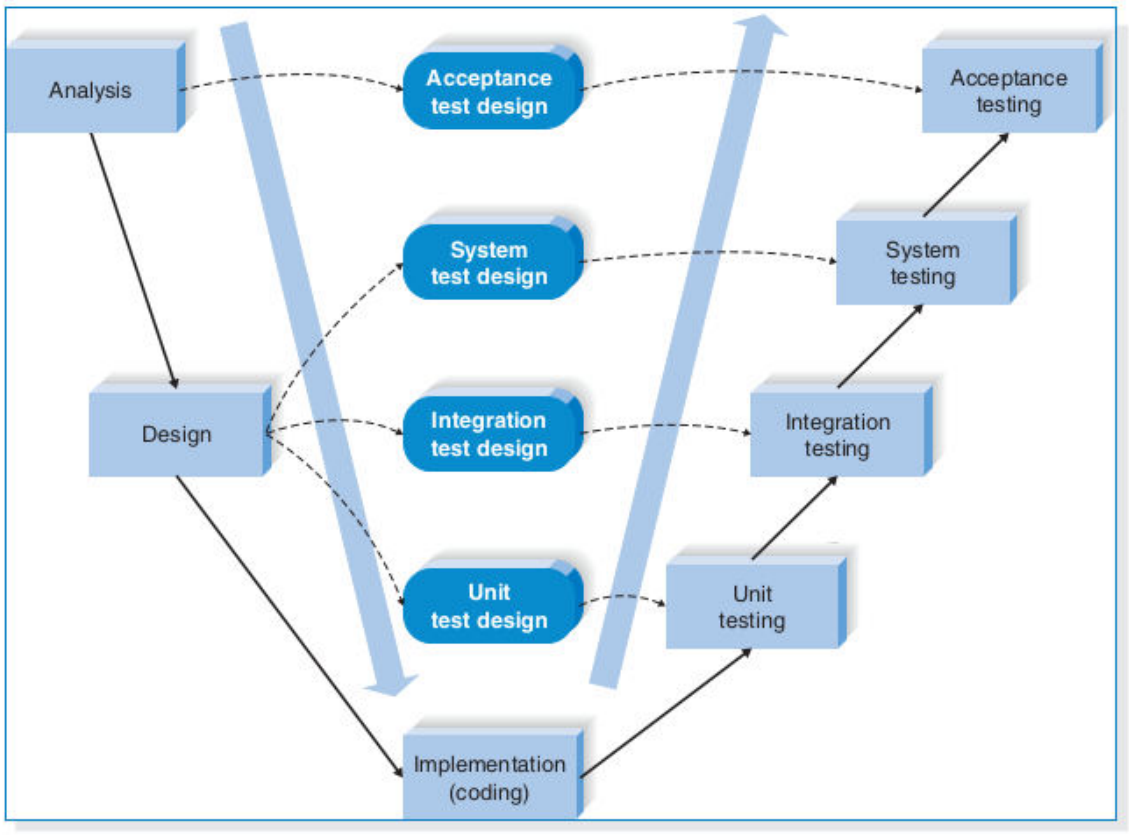


DE-Merits

Creating project requires careful design decision

V- Model

The V-model is another variation of waterfall development that pays more explicit attention to testing. The development process proceeds down the left-hand slope of the V, defining requirements and designing system components. At the base of the V, the code is written. On the upward-sloping right side of the model, testing of components, integration testing, and, finally, acceptance testing are performed. A key concept of this model is that as requirements are specified and components designed, testing for those elements is also defined. In this manner, each level of testing is clearly linked to a part of the analysis or design phase, helping to ensure high quality and relevant testing and maximize test effectiveness .



Rapid Application Development (RAD):- The aim of this model is to speed up the development process of waterfall model with the help of CASE tools, and JAD.

The goal is to get a prototype of the system and hand them to the users to get feedback and generate evaluations.

Approaches in RAD

- ***Iterative Development***

  breaks the overall project into a series of versions that are developed sequentially. The most important and fundamental requirements are bundled into the first version of the system. This version is developed quickly by a mini-waterfall process, and once implemented, the users can provide valuable feedback to be incorporated into the next version of the system

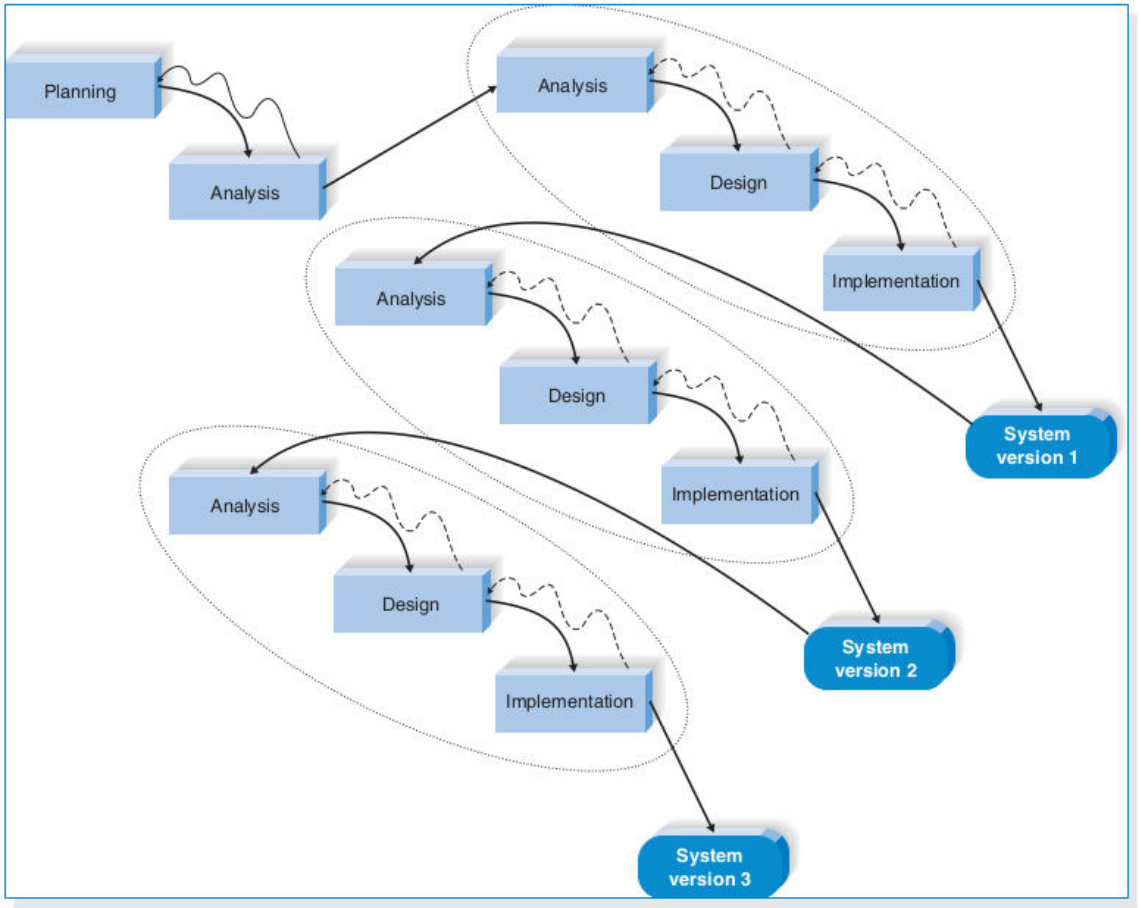
- ***System prototyping***

  performs the analysis, design, and implementation phases concurrently in order to quickly develop a simplified version of the proposed system and give it to the users for evaluation and feedback

- ***Throw away prototyping***

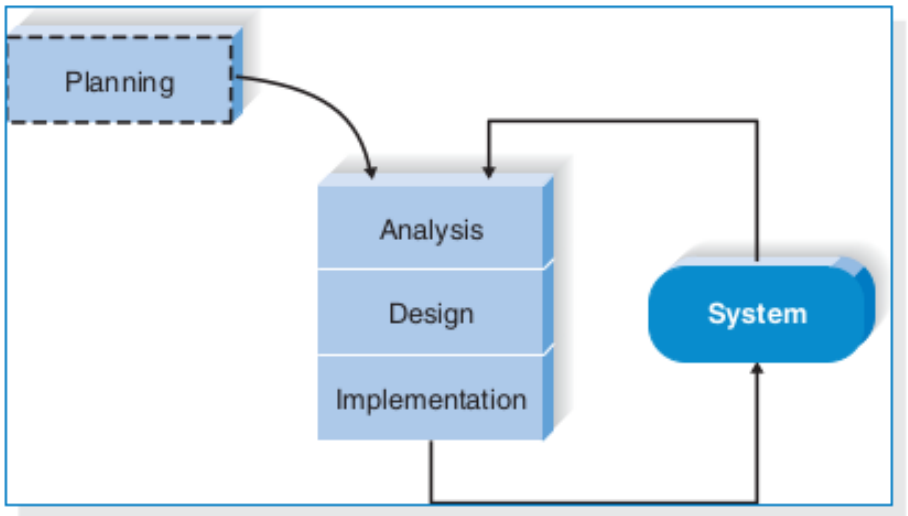
  includes the development of prototypes, but uses the prototypes primarily to explore design alternatives rather than as the actual new system (as in system prototyping) Many of the feature suggested by the users may not be well understood, however, and there may be challenging

technical issues to be solved. Each of these issues is examined by analyzing, designing, and building a design prototype. A design prototype is not intended to be a working system. It contains only enough detail to enable users to understand the issues under consideration.



Agile Development

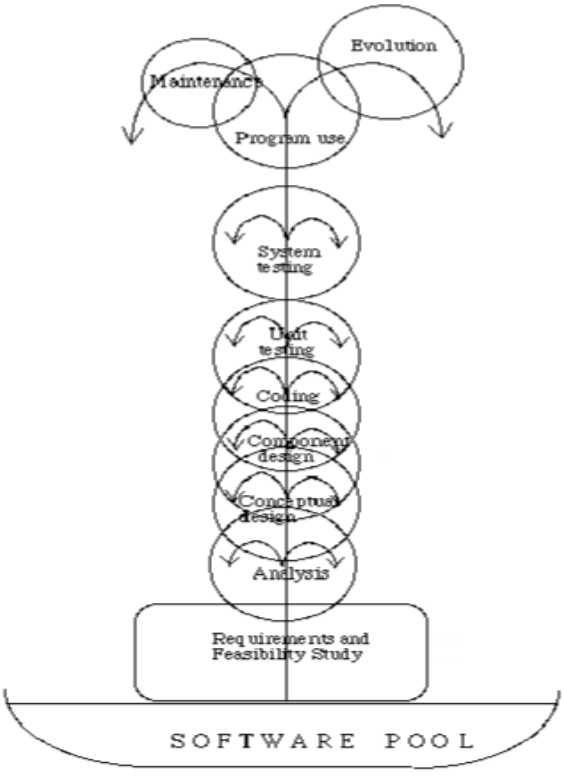
Agile development is a group of programming-eccentric methodologies that focus on streamlining the SDLC. Much of the modeling and documentation overhead is eliminated; instead, face-to-face communication is preferred. A project emphasizes simple, iterative application development in which every iteration is a complete software project, including planning, requirements analysis, design, coding, testing, and documentation. (Agile Development methodology uses OOP concept)



Fountain Model

This model is specifically useful for object-oriented system development; where each phase is regarded as a complete entity (to be analyzed, designed, and implemented separately from other phases). Note that he phases overlap ensuring parallelism of development process.

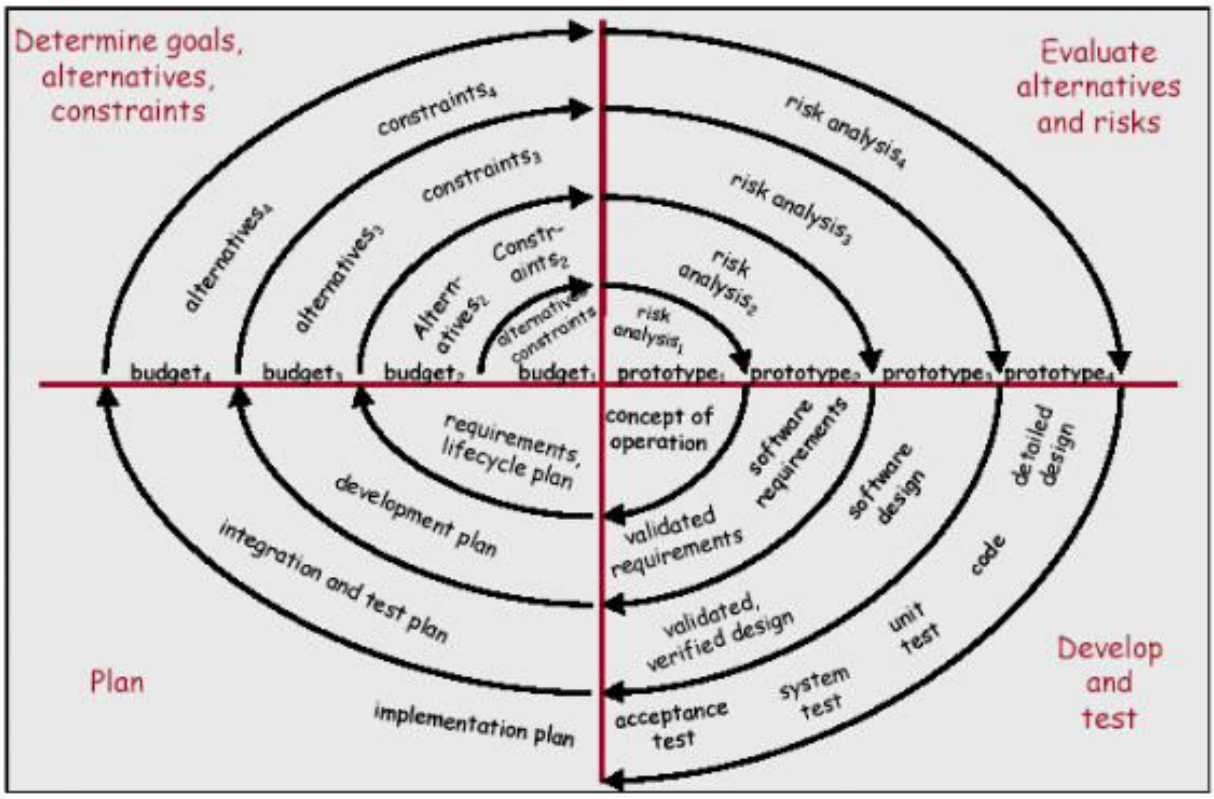
The main disadvantage of this model is unclear development process; since it requires code-a-bit, test-a-bit [CABTAB] principle.



Spiral Model

This is an evolutionary software development process which combines the prototype with the linear sequential waterfall model. It involves evaluation and risk analysis of each phase with the aim of discovering alternatives to implementation.

Thus, this model has an upper hand over most of the models because it provides checkpoints for project cancellation (if the risks cannot be resolved). However, to effectively implement this model, training, skill, manpower and considerable expenses are required. This model is normally used for the development of large-scale systems



Usefulness in Developing Systems	Waterfall	Parallel	V-Model	Iterative	System Prototyping	Throwaway Prototyping	Agile Development
with unclear user requirements	Poor	Poor	Poor	Good	Excellent	Excellent	Excellent
with unfamiliar technology	Poor	Poor	Poor	Good	Poor	Excellent	Poor
that are complex	Good	Good	Good	Good	Poor	Excellent	Poor
that are reliable	Good	Good	Excellent	Good	Poor	Excellent	Good
with short time schedule	Poor	Good	Poor	Excellent	Excellent	Good	Excellent
with schedule visibility	Poor	Poor	Poor	Excellent	Excellent	Good	Good

Analysis Phase

The analysis phase answers the questions o who will use the system, hat the system will do, and here and when it will be used, All of the deliverable are combined into a system proposal, which is presented to management, ho decides whether the project should continue to move forward.

System Analysis

-Requirement determination

-Use Case Analysis

-Process Modeling

-Data Modeling

The goal is to get to know the system requirements, requirements are what the system must do.

AS-IS system- Understand the existing situation of the system.

Identify improvements.

Define requirements for the new system (the to-be system). And concept of the systematic

Requirement Determination

Is to know and understand what are the requirement of the system and how do you record them. A precise list of what the system must do to provide the needed value to the business

There are three types of requirements

1. Business Requirement:- are what the business needs
2. User Requirement:- there are what the users do (use Case Analysis)
3. System Requirement:- is the statement of what system must do there are of two types
    -Functional system requirement
    -Non Functional system requirement

Functional Requirement:- are requirement which specifies the support that will provide to the user in fulfilling his/her task.

Process oriented / information oriented

Non-Functional requirement:- are requirement which specifies the performance requirement, cultural,political,security(access restrictions),legal, operational requirements of the systems without non functional requirements the system is likely to fail.

Requirement/ Facts gathering Techniques

Thus, in order to achieve the goal of this phase, facts gathering thru interviews, questionnaire/ surveys observation,document Analysis, Experiments, Joint Application Development.

Interviews: - The interview is the most commonly used requirements elicitation technique. After

all, it is natural—usually, if you need to know something, you ask someone. In general, interviews are conducted one on one (one interviewer and one interviewee),

but sometimes, due to time constraints, several people are interviewed at the same

time. Structure top-down and bottom up

Question types are open ended, close ended,open ended broad details, close ended simple detail.

Probing Questions:-

Strengths of Interviews

Interviewee can respond freely

Interviewee have free feedback

Interviewee non verbal communication can be observed

Weakness of Interview

It is time consuming

Its is very costly

Success can be determined by by the human resource communication skills

Questionnaires/ Surveys :- are special document that allow facts to be collected from large number of people.

Types of Surveys

Fixed Format and Free Format

Fixed Format:- objectives and subjective surveys

Free Format:- Essay surveys

Tips for designing questionnaire to develop a good questionnaire you need to determine what fact are to be collected and from whom it should be collected

Strengths of Questionnaires

- Can be answered quickly and easier.
- Users can maintain anonymity while filling questionnaire

Weakness of Survey

- Response might be often low
- Body language cannot be observed
- You cannot clarify vague answers

Joint Application Development:- its an expertise structured grouped process, to produce complete requirements of the system by bringing all the bodies together, manager, user , developer analyst together it needs a vibrant facilitator and a facility for the JAD.

JAD is expensive but valuable.

Weakness is that people might be hesitant to share decisions, many people opinions are involved.
How to overcome this is by using Electronic JAD (e-JAD)
Strength of JAD
Understand multiple perspective

Observations:- Observation, the act of watching processes being performed, is a powerful tool to gain insight into the as-is system. Observation enables the analyst to see the reality of a situation, rather than listening to others describe it in interviews or JAD sessions.
Strengths of observations
Data gathering is highly reliable
Inexpensive
Weakness of Observation
people can perform differently when being observed
difficult and boring

Document Analysis:- Project teams often use document analysis to understand the as-is system. Under ideal circumstances, the project team that developed the existing system will have produced documentation, which was then updated by all subsequent projects. In this case, the project team can start by reviewing the documentation and examining the system itself.
Experiments

Use Case Analysis
Use-cases as the means of expressing user requirement.
Use case is the description of how a system interact with environment by illustrating the activities of the user in the system. The goal is to create a set of use cases that express and clarify all the user requirements.
Note: *Use case are event driven;*

Elements Of Use Cases
Basic Information Each use case has a name and number. The name should be as simple, yet descriptive, as possible. The number is simply a sequential number that serves to reference each use case
The priority may be assigned to indicate the relative significance of the use

case in the overall system.

Preconditions Use cases are often performed in a sequence in order to accomplish an overall business task. While it might be possible to describe everything in one very large use case, that use case could become unwieldy.

Normal Course The next major part of a use case is the description of the major steps that are performed to execute the response to the event, the inputs used for the steps, and the outputs produced by the steps.

Alternative Courses In this section, the steps followed for alternative paths through the use case are outlined. Alternative courses are included to depict branches in logic that also will lead to a successful conclusion of the use case.

Post conditions As we explained in the preconditions section, use cases may be performed in a series in order to accomplish the overall user goal. In this section of the use case, we define the final products of this use case.

Exceptions In order to be complete, a use case should describe any error conditions or exceptions that may occur as the use case steps are performed. These are not normal branches in decision logic, but are unusual occurrences or errors that could potentially be encountered and will lead to an unsuccessful result.

Summary Inputs and Outputs The final section of the use case summarizes the set of major inputs and outputs to the steps of the use case. Each of the major inputs and outputs to the use case are listed, along with its source or destination.

When creating a use case you identify events that the users perform,
you identify the exceptions. Use Case is in sequence.
Use gradual refinement in building use cases.
Use cases convey the users point of view they are useful tools to clarify the requirements of a system.
Use Cases are represented in a two types of diagrams.
Text base and figure.
Casual and Complex use case format;

Use Case Name: Request a chemical	ID: UC-2	Priority: High

Actor: Lawn Chemical Applicator (LCA)

Description: The Lawn Chemical Applicator (LCA) specifies the lawn chemical needed for a job by entering its name or ID number. The system satisfies the request by reserving the quantity requested or the quantity available and notifying the Chemical Supply Warehouse of the pick-up.

Trigger: A Lawn Chemical Applicator (LCA) needs a chemical for a job.

Type: ☑ External ☐ Temporal

Preconditions:

1. The LCA identity is authenticated.
2. The LCA has necessary training and credentials on file.
3. The Chemical Supply datastore is up-to-date and on-line.

Normal Course:

1.0 Request a lawn chemical from the chemical supply warehouse.

1. The LCA specifies a chemical needed and the quantity needed
2. The system lists chemical and quantity on hand from Chemical Supply datastore
   a. If the quantity on hand is less than the quantity needed, the LCA specifies the quantity he will take
   b. Purchasing is notified of chemical shortage
3. The system gives the LCA a Chemical Pick-up Authorization for the quantity requested
4. The system notifies the Chemical Supply Warehouse of the chemical pick-up
5. The system stores the Lawn Chemical Request in the Chemical Request datastore

Postconditions:

1. The Lawn Chemical Request is stored in the Chemical Management System.
2. The Chemical Pick-up Authorization is produced for the LCA.
3. The Chemical Supply Warehouse is notified of the chemical pick-up.
4. Purchasing is notified of chemical outage.

Exceptions:

E1: Chemical is no longer approved for use (occurs at step 1)

1. The system displays message. "That chemical is no longer approved for use"
2. The system asks the LCA if he wants to request another chemical or to exit

3a. The LCA asks to request another chemical

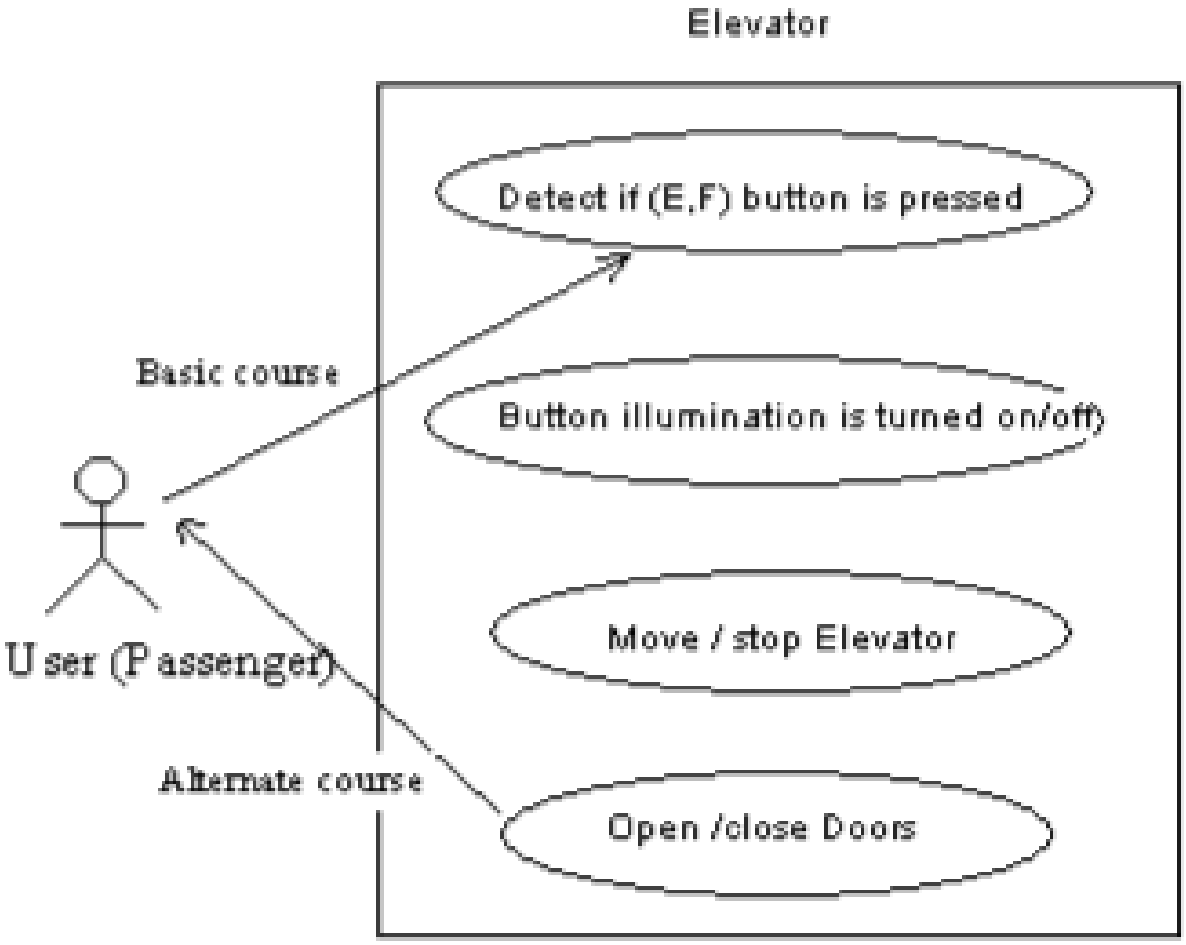
4a. The system starts Normal Course again

3b. The LCA asks to exit

4b. The system terminates the use case

FIGURE 4-3
Request a Chemical Use Case—Casual Format

The complex example of a use case diagram



Process Modeling

process model describes business processes—the activities that people do.

Process models are developed for the as-is system and/or the to-be system. This

chapter describes data flow diagramming, one of the most commonly used process modeling techniques.

A process model is a graphical way of representing how a business system should operate. It illustrates the processes or activities that are performed and how data move among them. A process model can be used to document the current system (i.e., as-is system) or the new system being developed (i.e., to-be system), whether computerized or not.

Process Modeling Techniques

Data Flow Diagramming DFD

Data flow diagramming is a technique that diagrams the business processes and the

data that pass among them

Elements of Data Flow Diagrams

which includes a set of symbols, naming conventions, and syntax rules. There are four

symbols in the DFD language (processes, data flows, data stores, and external entities)

Process A process is an activity or a function that is performed for some specific

business reason. Processes can be manual or computerized.

Data Flow:- Data Flow A data flow is a single piece of data (sometimes called a data element), or a logical collection of several pieces of information ,Every data flow should be named with a noun**.**

Data Store A data store is a collection of data that is stored in some way (which

is determined later when creating the physical model). Every data store is named

with a noun and is assigned an identification number and a description.

External Entity An external entity is a person, organization, organization unit, or

system that is external to the system, but interacts with it (e.g., customer, clearing-

house, government organization, accounting system).

Context Diagram As the name suggests, the context diagram shows the entire system in context with its environment. All process models have one context diagram.

The context diagram shows the overall business process as just one process

(i.e., the system itself) and shows the data flows to and from external entities.

Process Descriptions

The purpose of the process descriptions is to explain what the process does and pro-

vide additional information that the DFD does not provide. As we move through the

SDLC, we gradually move from the general text descriptions of requirements into

more and more precise descriptions that are eventually translated into very precise

programming languages

DFD are created using by project team using CASE TOOLS

PROGRAM DESIGN

Moving from Logical to Physical Process Models

As the application logic of the information system is designed, the implementation decisions describing how the system will work are made. Physical data flow diagrams show these implementation details, including data stores that refer to files and database tables, programs or human actions that perform processes, and the physical transfer media for the data flows. The automated parts of the system are distinguished from the manual parts by the human-machine boundary.

Structure Chart

The structure chart shows all the functional components that must be included in the program at a high level, arranged in a hierarchical format that implies order and control. Lines that connect modules can contain a loop, which signifies that the subordinate module is repeated before other modules to its right are invoked. A diamond is placed over subordinate modules that are invoked conditionally. An arrow with a filled circle represents a control couple or flag, which passes system messages from one module to another. The data couple, an arrow with an empty circle, denotes the passing of records or fields

Building Structure Charts

Creating a structure chart is usually a four-step process. First, the analyst identifies the top-level modules and then decomposes them into lower levels. Second, the analyst adds the control connections among modules, such as loops and conditional lines that show when modules call subordinates. Third, the analyst adds couples, the information that modules pass among themselves. Finally, the analyst reviews the structure chart and revises it again and again until it is complete.

Structure Chart Design Guidelines

There are several design guidelines that you should follow when designing structure charts. First, build modules with high cohesion so that each module performs only one function. There are seven kinds of cohesion, ranging from good to bad, and the instances of bad cohesion should be removed from the structure chart. Second,modules should not be interdependent, but rather should be loosely coupled, using good types of coupling. There are five kinds of coupling, ranging from good to bad; as with cohesion, bad instances should be avoided. Finally, structure charts should display high fan-in and low fan-out, which means that modules should have many control modules but limited subordinates.

Program Specification

Program specifications provide more detailed instructions to the programmers about how to code the modules. The program specification contains several components that communicate basic module information (e.g., a name, calculations that must be performed, and the target programming language), inputs and outputs, special instructions for the programmer, and pseudocode.

Pseudocode

Pseudocode is a technique similar to structured English that communicates the code written with the use of programming structures and a generic language that is not program language specific. Pseudocode is much more like real code than is structured English, and its audience is the programmer as opposed to the analyst.

Data Storage Design

File Data Storage Formats

There are two basic types of data storage formats: files and databases. Files are electronic lists of data that have been optimized to perform a particular transaction, and there are five different types: master, look-up, transaction, audit, and history. Master files typically are kept for long periods because they store important business information, such as order information or customer mailing information.

Look-up files contain static values that are used to validate fields in the master files, and transaction files temporarily hold information that will be used for a master file update. An audit file records "before" and "after" images of data as they are altered so that an audit can be performed if the integrity of the data is questioned. Finally, the history file stores past transactions (e.g., old customers, past orders) that are no longer needed by the system.

Database Data Storage Formats

A database is a collection of groupings of information that are related to each other in some way, and a DBMS (database management system) is software that creates and manipulates these databases. There are four types of databases that are likely to be encountered during a project: legacy, relational, object, and multidimensional. The legacy databases (e.g., hierarchical databases and network databases) use older, sometimes outdated technology and are rarely used to develop new applications. The relational database is the most popular kind of database for application development today, and it is based on collections of tables that are related to each other

through common fields known as foreign keys. Object databases contain data and processes that are represented by object classes, and relationships between object classes are shown by encapsulating one object class within another and are mainly used in multimedia applications (e.g., graphics, video, and sound). One of the newest members in the database arena is the multidimensional database, which has become popular with the increase in data warehousing. It stores precalculated quantitative information (e.g., totals, averages) at the intersection of dimensions (e.g., time, salesperson, product) to support applications that require data to be sliced and diced.

Selecting a Data Storage Format

The application's data should drive the storage format decision. Relational databases support simple data types very effectively, whereas object databases are best for complex data. Multidimensional databases are tuned to store aggregated, quan-titative information. The type of system also should be considered when choosing among data storage formats. Relational databases have matured to support transactional systems, whereas multidimensional databases have been designed to perform best in decision support environments. Although less critical to the format selection decision, the project team needs to consider the kind of technology that exists within the organization and the kind of technology likely to be used in the future.

Physical Entity Relationship Diagrams

One important aspect of design is the movement from logical to physical entity relationship diagrams. Physical ERDs contain references to how data will be stored in a file or database table, and metadata are included to describe the data model components. The model reflects design decisions that will affect the physical implementation of the system.

The CRUD matrix should be modified to show exactly how data in the physical data model are created and used in the physical process model. The CRUD matrix helps ensure balance between the physical process and data models prior to implementation.

Optimizing Data Storage

There are two primary dimensions in which to optimize a relational database: for storage efficiency and for speed of access. The most efficient relational database tables in terms of data storage are those that have no redundant data and very few

null values. Normalization is the process whereby a series of rules are applied to the logical data model to determine how well optimized it is for storage efficiency. Once you have optimized your logical data design for storage efficiency, the data may be spread out across a number of tables. To improve speed, the project team may decide to denormalize—or add redundancy back into—the design that is depicted in the physical data model. Denormalization reduces the number of joins that must be performed in a query, thus speeding up data access. Denormalization is best in situations in which data are accessed frequently and updated rarely. There are three modeling situations that are good candidates for denormalization: look-up tables, entities that share one-to-one (1:1) relationships, and entities that share one-to-many (1:M) relationships. In all three cases, attributes from one entity are moved or repeated in another entity to reduce the joins that must occur during data access.
Clustering occurs when similar records are stored close together on the storage medium to speed up data retrieval. In intrafile clustering, similar records in the table are stored together in some way, such as in sequence. Interfile clustering combines records from more than one table that typically are retrieved together. Indexes also can be created to improve the access speed of a system. An index is a minitable that contains values from one or more columns in a table and information about where the values can be found. Instead of performing a table scan, which is the most inefficient way to retrieve data from a table, an index points directly to the records that match the requirements of a query.
Finally, the speed of the system can be improved if the right hardware is purchased to support its data. Analysts can use volumetrics to estimate the current and future size of data in the database and then share these numbers with the people who are responsible for buying and configuring the database hardware.

Implementation Phase

Managing Programming

Programming is done by programmers, so systems analysts have other responsibilities during this stage. The project manager, however, is usually very busy. The first step is to assign tasks to the programmers—ideally, the fewest possible to complete the project, because coordination problems increase as the size of the programming team increases. Coordination can be improved by having regular meetings, ensuring that standards are followed, implementing change control, and using computer-

aided software engineering (CASE) tools effectively. One of the key functions of the project manager is to manage the schedule and adjust it for delays. Two common causes of delays are scope creep and minor slippages that go unnoticed.

Testing

Tests must be carefully planned because the cost of fixing one major bug after the system is installed can easily exceed the annual salary of a programmer. A test plan contains several tests that examine different aspects of the system. A test, in turn, specifies several test cases that will be examined by the testers. A unit test examines a module or program within the system; test cases come from the program specifications or the program code itself. An integration test examines how well several modules work together; test cases come from the interface design, use scenarios, and the physical data flow diagrams (DFDs). A system test examines the system as a whole and is broader than the unit and integration tests; test cases come from the system design, the infrastructure design, the unit tests, and the integration. Acceptance testing is done by the users to determine whether the system is acceptable to them; it draws on the system test plans (alpha testing) and the real work the users perform (beta testing).

Documentation

Documentation, both user documentation and system documentation, is moving away from paper-based documents to online documentation. There are three types of user documentation: Reference documents are designed to be used when the user needs to learn how to perform a specific function (e.g., an online help system); procedures manuals describe how to perform business tasks; and tutorials teach people how to use the system. Documentation navigation controls (e.g., a table of contents, index, a “find” function, intelligent agents, or links between pages) enable users to find documentation topics (e.g., how to perform a function, how to use an interface command, an explanation of a term).

Transitioning to the new System

Making the Transition to the New System

Transitioning to the new system is facilitated by following Lewin’s three-step model of organizational change: Unfreeze, move, and refreeze. Activities during systems analysis and design will help unfreeze attachment to the existing system. The migration plan guides the movement from the as-is system to the to-be system. The support and maintenance provided for the new system helps to refreeze the new

system into everyday use in the organization.

The Migration Plan

The migration plan encompasses a number of elements that will guide the transition from the old to the new system. The organization's readiness can be developed by the conversion strategy that is selected and business contingency planning. The technology is prepared through installation of the hardware and software and conversion of the data. The people are prepared to accept and use the new system through many channels, including management policies, adoption motivation techniques, and training. Understanding the sources of resistance to change and the costs and benefits that the users perceive will help analysts develop a successful migration plan.

Postimplementation Activities

System support is performed by the operations group, which provides online and help-desk support to the users. System support has both a level 1 support staff, which answers the phone and handles most of the questions, and level 2 support staff, which follows up on challenging problems and sometimes generates change requests for bug fixes. System maintenance responds to change requests (from the system support staff, users, other development project teams, and senior management) to fix bugs and improve the business value of the system. The goal of project assessment is to understand what was successful about the system and the project activities (and therefore should be continued in the next system or project) and what needs to be improved. Project team review focuses on the way in which the project team carried out its activities and usually results in documentation of key lessons learned. System review focuses on understanding the extent to which the proposed costs and benefits from the new system that were identified during project initiation were actually recognized from the implemented system.

The Movement to Objects

BASIC CHARACTERISTICS OF OBJECT-ORIENTED SYSTEMS

Object-oriented systems focus on capturing the structure and behavior of information systems in little modules that encompass both data and processes. These little modules are known as objects. In this section of the chapter, we describe the basic characteristics of object-oriented systems.

Classes and Objects

A class is the general template we use to define and create specific instances, or objects. Every object is associated with a class. For example, all of the objects that capture information about patients could fall into a class called Patient, because there are attributes (e.g., names, addresses, and birth dates) and methods (e.g., insert new instances, maintain information, and delete entries) that all patients share

Methods and Messages

Methods implement an object's behavior. A method is nothing more than an action that an object can perform. Methods are very much like a function or procedure in a traditional programming language such as C, COBOL, or Pascal. Messages are information sent to objects to trigger methods. A message is essentially a function or procedure call from one object to another object. For example, if a customer is new to the organization, the system will send an insert message to the customer object.

Encapsulation and Information Hiding

The ideas of encapsulation and information hiding are interrelated in object-oriented systems. Neither of the concepts is new, however. Encapsulation is simply the combining of process and data into a single entity. Object-oriented approaches combine process and data into holistic entities (objects).

Inheritance

Inheritance, as an information systems development characteristic, was proposed in data modeling in the late 1970s and early 1980s. The data modeling literature suggests using inheritance to identify higher level, or more general, classes of objects. Common sets of attributes and methods can be organized into superclasses. Typically, classes are arranged in a hierarchy whereby the superclasses, or general classes, are at the top, and the subclasses, or specific classes, are at the bottom

Polymorphism and Dynamic Binding

Polymorphism means that the same message can be interpreted differently by different classes of objects. For example, inserting a patient means something different than inserting an appointment. As such, different pieces of information need to be collected and stored. Luckily, we do not have to be concerned with how something is done when using objects. We can simply send a message to an object, and that object will be responsible for interpreting the message appropriately. For example, if we sent the message "Draw yourself " to a square object, a circle object, and

a triangle object, the results would be very different, even though the message is the same.

Polymorphism is made possible through dynamic binding. Dynamic, or late, binding is a technique that delays identifying the type of object until run-time. As such, the specific method that is actually called is not chosen by the object-oriented system until the system is running. This is in contrast to static binding. In a statically bound system, the type of object would be determined at compile time. Therefore, the developer would have to choose which method should be called, instead of allowing the system to do it

OBJECT-ORIENTED SYSTEMS ANALYSIS AND DESIGN

Object-oriented approaches to developing information systems, technically speaking, can use any of the traditional methodologies presented in Chapter 2 (waterfall development, parallel development, V-model, iterative development, system prototyping, and throwaway prototyping). The object-oriented approaches are most associated with an iterative development RAD methodology, however. The primary difference between a traditional approach like structured design and an object-oriented approach is how a problem is decomposed. In traditional approaches, the problem decomposition is either process-centric or data-centric. When modeling real-world systems is involved, however, processes and data are so closely related that it is difficult to pick one or the other as the primary focus. Based on this lack of congruence with the real world, new object-oriented methodologies have emerged that use the RAD-based sequence of SDLC phases, but attempt to balance the emphasis between process and data. This is done by focusing the decomposition of problems on objects that contain both data and processes.

Use Case Driven

Use case driven means that use cases are the primary modeling tool employed to define the behavior of the system. A use case describes how the user interacts with the system to perform some activity, such as placing an order, making a reservation, or searching for information. The use cases are used to identify and to communicate the requirements for the system to the programmers who must write the system. Use cases are inherently simple because they focus on only one activity at a time. In contrast, the process models we have learned about in this book are far

more complex because they require the developers to create models of the entire system. In traditional process models, each business activity is decomposed into a set of subprocesses, which are further decomposed into more subprocesses, and so on. In contrast, use cases focus on only one activity at a time, so developing models is much simpler.

Architecture Centric

Any modern approach to systems analysis and design should be architecture centric. Architecture centric means that the underlying architecture of the evolving system drives the specification, construction, and documentation of the system. There are three separate, but interrelated, architectural views of a system: functional, static, and dynamic. The functional view describes the external behavior of the system from the perspective of the user. On the surface, this view is closely related to process-modeling approaches in structured analysis and design. There are important differences between them, however. The static view describes the structure of the system in terms of attributes, methods, classes, relationships, and messages. This view is very similar to data-modeling approaches in structured analysis and design. The dynamic view describes the internal behavior of the system in terms of messages passed between objects and state changes within an object. This view in many ways combines the process and data-modeling approaches, because the execution of a method can cause state changes in the receiving object.

Iterative and Incremental

Object-oriented approaches emphasize iterative and incremental development that undergoes continuous testing throughout the life of the project. Each iteration of the system brings the system closer and closer to the final needs of the users.

Benefits of Object-Oriented Systems Analysis and Design

So far, we have described several major concepts that permeate any object-oriented approach to systems development, but you may be wondering how these concepts affect the performance of a project team. The answer is simple. Concepts like512 polymorphism, encapsulation, and inheritance taken together allow analysts to break a complex system into smaller, more manageable components, to work on the components individually, and to more easily piece the components back together to

form a system. This modularity makes system development easier to grasp, easier to share among members of a project team, and easier to communi-cate to
users who are needed throughout the SDLC to provide requirements and confirm how well the system meets the requirements.

By modularizing system development, the project team is actually creating reusable pieces that can be plugged into other systems efforts, or used as starting points for other projects. Ultimately, this can save time, because new projects do not have to start from scratch and learning curves are not as steep.

UNIFIED MODELING LANGUAGE, VERSION 2.0

The objective of the Unified Modeling Language is to provide a common vocabulary of object-based terms and diagramming techniques that is rich enough to model any systems development project from analysis to design. The current version of UML, version 2.0, was accepted by the Object Management Group (OMG) in 2003. This version of UML defines a set of 14 diagramming techniques for modeling a system.

The diagrams are broken into two major groupings: one for modeling the structure of a system and one for modeling behavior. Structure diagrams are used for representing the data and static relationships that are in an information system. Behavior diagrams provide the analyst with a way to depict the dynamic relationships among the instances or objects that represent the business information system. They also allow the modeling of dynamic behavior of individual objects throughout their lifetime. The behavior diagrams support the analyst in modeling the functional requirements of an evolving information system.

t. In other words, the diagrams move from documenting the requirements to laying out the design. Overall, the consistent notation, integration among the diagramming techniques, and the application of the diagrams across the entire development process make the UML a powerful and flexible language for analysts and developers.

Note that UML is not a methodology; it does not formally mandate how to apply the diagramming techniques. Many organizations are experimenting with UML and trying to understand how to incorporate its techniques into their systems analysis and design methodologies. In many cases, the UML diagrams simply replace the older structured techniques (e.g., class diagrams replace ERD diagrams). The basic SDLC stays the same, but one step is performed by a different diagramming technique. Methodologies that are suited to the object-oriented

approach are available, however, such as the rational unified process (RUP).

The Rational Unified Process (RUP)

Rational Software Corporation has created a methodology called the rational unified process (RUP) that defines how to apply UML. RUP is a rapid application development approach to building systems that is similar to the iterative development approach or extreme programming. The first step of the methodology is building use cases for the system, which identify and communicate the high-level business requirements for thesystem. This step drives the rest of the SDLC. Next, analysts draw analysis diagrams, later building on the analysis efforts through design and development. The UML diagrams start off conceptual and abstract, and then include details that ultimately will lead to code generation and development. The diagrams move from showing the what to showing the how.

RUP emphasizes iterative, incremental development and prototyping that undergo continuous testing throughout the life of the project.

Use Case Diagram

A use case diagram illustrates the main functions of a system and the different kinds of users that interact with it. The diagram includes actors, which are people or things that derive value from the system, and use cases that represent the functionality of the system. The actors and use cases are separated by a system boundary and connected by lines representing associations. At times, actors are specialized versions of more general actors. Similarly, use cases can extend or include other use cases. Building use case diagrams is a five-step process whereby the analyst identifies the use cases, draws the system boundary, adds the use cases to the diagram, identifies the actors, and, finally, adds appropriate associations to connect use cases and actors together.

Class Diagram

The class diagram shows the classes and relationships among classes that remain constant in the system, over time. The main building block of the class diagram is a class, which stores and manages information in the system. Classes have attributes that capture information about the class and about operations, which are actions that a class can perform. There are three kinds of operations: constructor, query, and update. The classes are related to each other through an association, which has a name and a multiplicity that denotes the minimum and maximum instances that participate in the relationship. Two special associations, aggregation and generalization, are used when classes comprise other classes or when one sub-

class inherits properties and behaviors from a superclass, respectively. Class diagrams are created by first identifying classes, along with their attributes and operations. Then relationships are drawn among classes to show associations. Special notations are used to depict the aggregation and generalization associations.

Sequence Diagram

The sequence diagram is a dynamic model that illustrates instances of classes that participate in a use case and messages that pass between them over time. Objects are placed horizontally across the top of a sequence diagram, each having a dotted line, called a lifeline, vertically below it. The focus of control, represented by a thin rectangle, is placed over the lifeline to show when the objects are sending or receiving messages. A message is a communication between objects that conveys information with the expectation that activity will ensue, and the messages are shown by an arrow connecting two objects that points in the direction that the message is being passed. To create a sequence diagram, first identify the classes that participate in the use case and then add the messages that pass among them. Finally, you will need to add the lifeline and focus of control. Sequence diagrams are helpful for understanding real-time specifications and for complex scenarios of a use case.

Behavioral State Machine Diagram

The behavioral state machine diagram shows the different states that a single instance of a class passes through during its life in response to events, along with responses and actions. A state is a set of values that describes an object at a specific point in time, and it represents a point in an object's life in which it satisfies some condition, performs some action, or waits for something to happen. An event is something that takes place at a certain point in time and changes a value(s) that describes an object, which in turn changes the object's state. As objects move from state to state, they undergo transitions. When drawing a behavioral state machine diagram, rectangles with rounded corners are first placed on the model to represent the various states that the class will take. Next, arrows are drawn between the rectangles to denote the transitions, and event labels are written above the arrows to describe the event that causes the transition to occur.