System Analysis and Design

What is a System?

"An orderly grouping of interrelated and interdependent components linked together according to a plan to achieve a specific objective".

Elements of a system

- **Components**
  A component or a subsystem is an irreducible part or a collection of parts that makes up a system.

- **Boundary**
  The limits that identify a system's components, processes, and interrelationships when it interfaces with the environment. Boundary marks the inside and outside of a system.

- **Environment**
  Everything external to a system which has a relationship with or an impact on the system.

Characteristics of a system

- **Central Objective**
  A specific result that a system aims to achieve. It's the Overall goal or function of a system. All the components should work together to achieve the common goal.

- **Organization**
  The arrangement of components that helps to achieve predetermined objectives. Organization implies the structure and the order.

- **Interaction**
  The manner in which the components operate with each other. It is concerned with the interrelationships between various components of a system.

- **Interdependence**

  The extent to which the components of a system depend on each other. In a system, what happens to one component can affect the other parts.

- **Integration**

  Integration is concerned with how system components are connected together. Successful integration means that components of the system will work together within the system even if each part performs a unique function.

Types of Systems

- **Open and Closed systems**

  An open system has many interfaces with its environment. It permits interaction across system boundary. The system receives inputs from its environment and delivers outputs to its environment.

  Ex. Human Respiratory System

  A closed system does not interact with its environment. It is self-contained and isolated from environmental influences.

  Ex. Human Blood Circulation System

- **Natural and manmade systems**

  Natural systems are created by the nature.

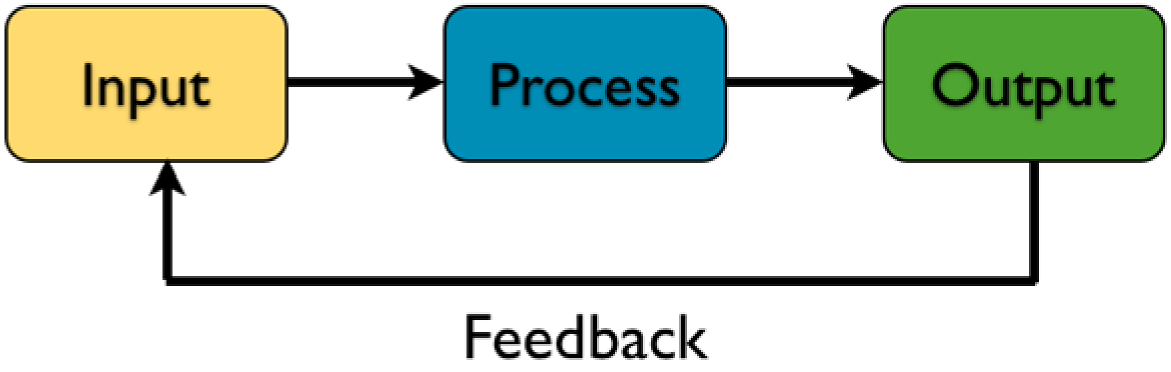
  Ex. Solar system

  The systems which are made by human are called manmade systems.

  Ex. Transportation System of a Country

What is an Information system?

Information system is an integrated set of components for Collecting, Storing, Processing and Communicating Information.



Types of Information System

- **Transaction Processing Systems (TPS)**

  Transaction Processing Systems are used to perform and record the daily routine transactions necessary to conduct a business and serve the operational level users of an organization. TPS allow managers to monitor status of operations and relations with external environment and to make Structured decisions.

  Ex. Retail point of sale system

- **Management Information Systems (MIS)**

  Management Information Systems serve the functions of planning, controlling, and decision making by extracting, processing and summarizing data from Transaction Processing Systems in order to make Structured decisions for the middle management level users of an organization. MIS Provides routine reports on firm’s current performance based on data from TPS.

  Ex. Inventory control System

- **Decision Support Systems (DSS)**

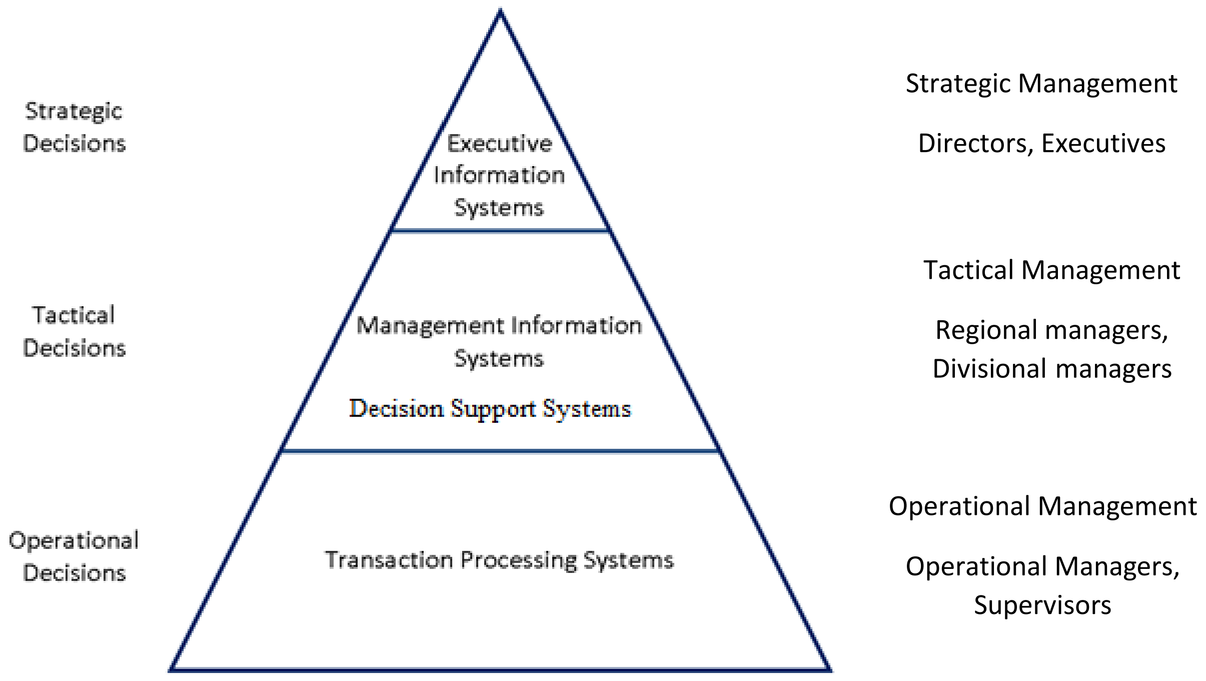
  Decision Support Systems combine data and analytical models/ data analysis tools to support Semi- structured and Unstructured decision making for the middle management level users of an organization. A decision support system use data from internal (TPS and MIS) or external (Ex. Stock prices) sources.

  Ex. Market analysis systems

- **Executive Support Systems (ESS)**

  Executive Support Systems address Unstructured decision making through advanced graphics and communications for the strategic level users of an organization. ESS incorporates data about external events (ex. new tax laws or competitors) as well as summarized information from internal MIS and DSS.

  Ex. Financial Planning Systems



Levels of Decision Making

Structure of Decisions	**Examples**
Structured decisions • Repetitive and routine decisions with predefined procedures to follow	• Restock Inventory • Determine special offers to customers
Semi-structured decisions • Have some agreement on the data, process, and/or evaluation to be used with some level of human judgment	• Designing a marketing plan • Allocate resources to managers
Unstructured decisions • Non-routine. Require insight based on many sources of information and personal experiences.	• Decide entrance or exit from a market • Decide long term objectives

➢ **Office Automation Systems (OAS)**

Office Automation Systems are designed to increase the productivity of employees in an organization through automating information gathering, communication, and presentation processes.

Ex. Work scheduling systems

➢ **Geographic Information Systems (GIS)**

Geographic Information Systems connect data with geography. GIS allow to map, model and analyze large quantities of data within a single database according to their location. It facilitates to create maps, integrate information, visualize scenarios and develop effective solutions.

Ex. Web based map services, Urban Planning and Transportation Planning applications

➢ **Knowledge Management Systems (KMS)**

Knowledge Management Systems comprise a range of practices used in an organization for acquiring, creating, storing, distributing, applying, integrating knowledge. KMS collect internal knowledge and experiences either embodied in individual or in organizational processes and practices and make it available to employees.

Ex. Knowledge work systems (KWS), Customer Feedback systems

- **Content Management Systems (CMS)**

  Content Management Systems support the creation and modification of digital content. CMS support to multiple users working in a collaborative environment. CMS features include web-based publishing, business collaboration and Digital asset management etc.

  Ex. WordPress, Joomla

- **Enterprise Resource Planning (ERP)**

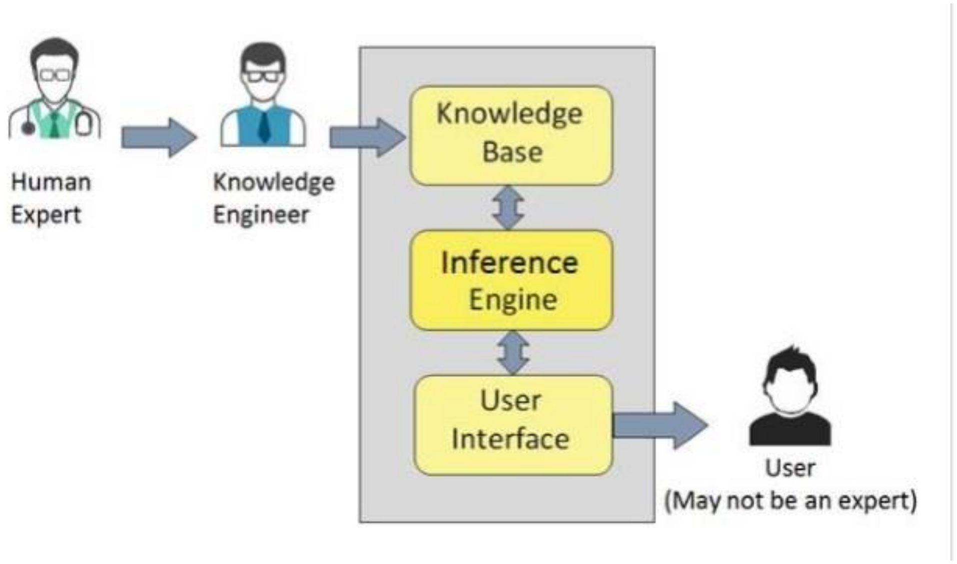
  Enterprise Resource Planning Systems are business process management systems that allow organizations to use integrated applications to manage the core business processes. An ERP system automates many back office functions related to technology, services and human resources. It integrates information flows from a variety of sources and serves a wide variety of groups and purposes in the firm.

  Ex. SAP ERP

- **Expert Systems**

  Expert Systems emulate the decision-making ability of a human expert. To design an expert system, one needs a knowledge engineer, an individual who studies how human experts make decisions and translates the rules into terms that a computer can understand. Expert systems are one of the research domains of artificial intelligence.

  Ex. MYCIN Medical expert system



- **Smart Systems**

  Smart systems incorporate functions of sensing, actuation, and control in order to describe and analyze a situation and make decisions based on the available data. Smart systems can be attributed to autonomous operation based on machine learning, energy efficiency, and networking capabilities.

  Ex. Anti-lock Braking System (ABS)

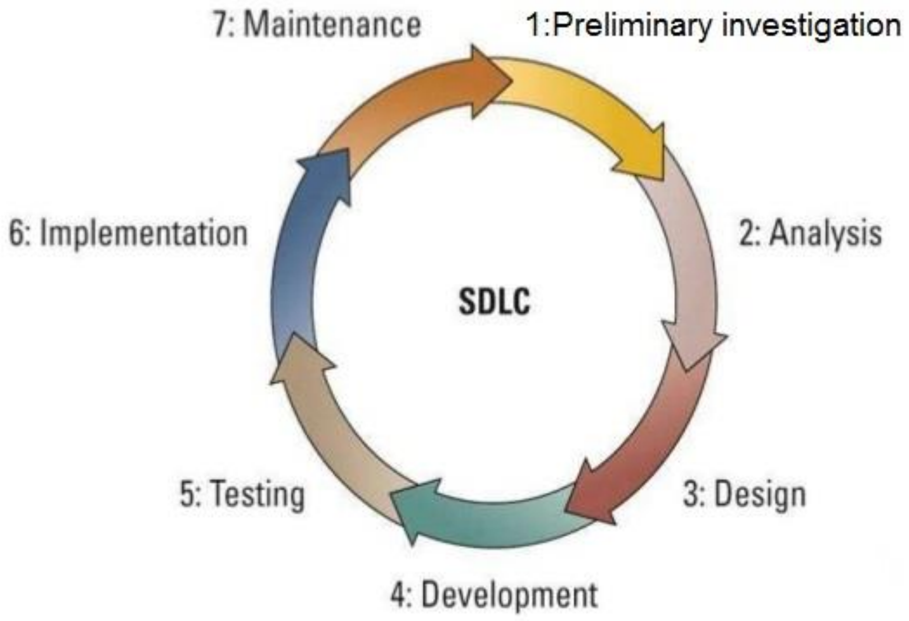
Information system development models and methods

System Development Life Cycle (SDLC)

System Development Life Cycle is the overall process of developing information systems through a multi-step process from investigation of initial requirements through Analysis, Design, Implementation and Maintenance.

System Development Life Cycle includes following activities.

1. Preliminary investigation
2. System analysis
3. System design
4. System development
5. System testing
6. System implementation
7. System Maintenance



System Development Lifecycle models

Waterfall model

The waterfall model is a sequential process model used in software development. In Waterfall approach the whole process of software development is divided into a set of separate phases.

1. Feasibility Study
2. Requirement analysis
3. System design
4. Implementation
5. Testing
6. Deployment
7. Maintenance

Feasibility Study

Feasibility study is carried out to assess the practicality of a proposed plan or method.

Requirement analysis

All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

System design

Gathered requirements in the first phase are studied and the system design is prepared.

Implementation

With inputs from the system design phase, the system is developed into a working software.

Testing

In testing phase, Developed solution is tested for any faults or failures.

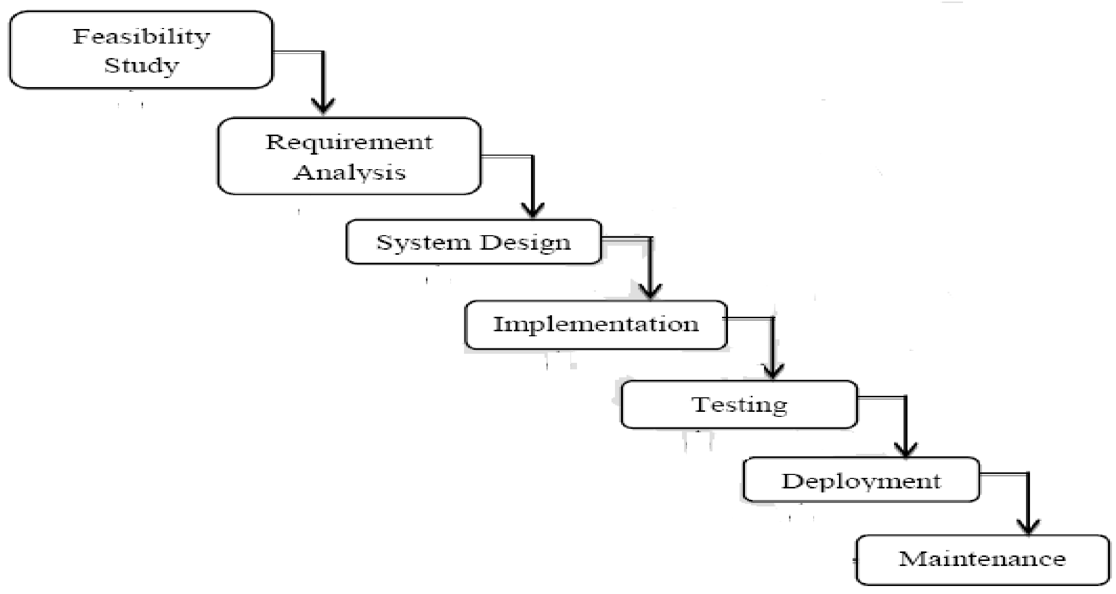
Deployment

Once the testing is completed, the product is deployed in the customer environment or released into the market.

Maintenance

Maintenance is done to deliver enhanced product versions or fix issues in the customer environment.

All the above phases are cascaded to each other. I.e. each phase must be completed fully before the next phase can begin.



Advantages

- ➢ Easy to manage and control due to the rigidity of the model.
- ➢ Every phase has a defined start and end point.
- ➢ Phases are processed and completed one at a time.
- ➢ Works well for smaller projects where requirements are very well understood and stable.

Disadvantages

- ➢ Inflexible partitioning of the project into distinct phases makes it difficult to respond to changing customer requirements.
- ➢ No working software is produced until late during the life cycle.
- ➢ High amounts of risk and uncertainty is involved.
- ➢ Poor model for complex and ongoing projects.

- Limited customer interaction is involved during the development of the product.

Applicability of the Waterfall model

- When the requirements are very well known, clear and fixed.
- When the project is short and Product definition is stable.

Spiral model

Spiral model is a combination of iterative development and sequential linear development processes. It is a type of Incremental model with more emphasis placed on risk analysis. The spiral model contains four phases.

1. Planning
2. Risk Analysis
3. Engineering
4. Evaluation

Planning phase

Requirements are identified and gathered during the Planning phase.

Risk Analysis phase

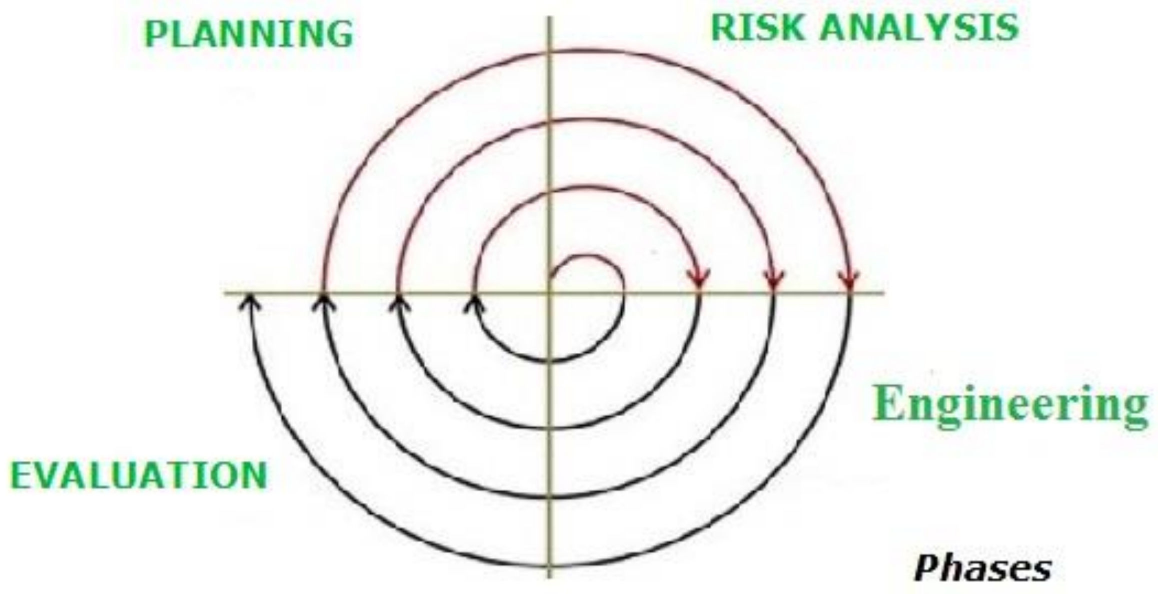
Risk Analysis is done to identify, estimate and monitor the technical and management risks. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

Engineering Phase

In the Engineering phase the software solution is developed, along with testing at the end of the phase.

Evaluation

Evaluation phase allows the customer to evaluate the output and provide feedback before the project continues to the next spiral.



Advantages

- ➢ Changing requirements can be accommodated.
- ➢ High amount of risk analysis hence, avoidance of Risk is enhanced.
- ➢ Suitable for mission-critical projects.
- ➢ Allows extensive use of prototypes.
- ➢ Strong approval and documentation control.

Disadvantages

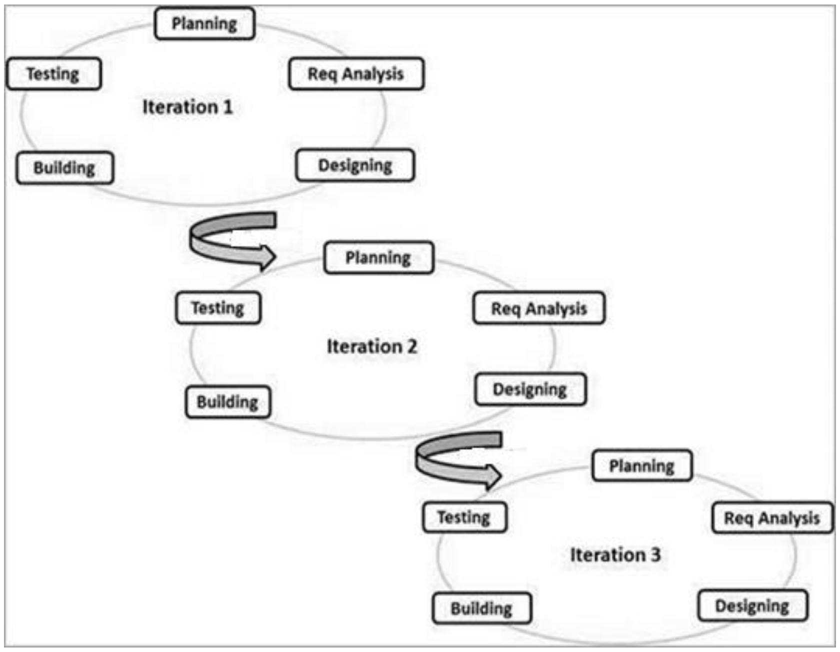
- ➢ Management of the development is more complex.
- ➢ Risk analysis requires highly specific expertise.
- ➢ Can be a costly model to use.
- ➢ Spiral may go on indefinitely.
- ➢ End of the project may not be known early.

Applicability of the Spiral model

- ➢ When evaluation risk involved is critical.
- ➢ When customers are unsure of requirements or Requirements are complex.
- ➢ When significant changes are expected during the development.

Agile model

Agile model focuses on developing a Software product in incremental and rapid cycles. The development is divided into time slices i.e.time-boxed, to deliver specific features for a release. A software build is delivered after each iteration. Each build is an increment and the final product contains all the functionalities that the user requested.



Advantages

- Provides higher customer satisfaction by rapid and continuous delivery of useful software.
- High customer collaboration.
- Resource requirements are minimum.
- Regular adaptation to changing circumstances.
- Easy to manage.

Disadvantages

- Depends heavily on customer interaction, so if customer is not clear about the requirements, team can be driven in the wrong direction.
- Lack of emphasis on necessary designing and documentation.

Applicability of the Agile model

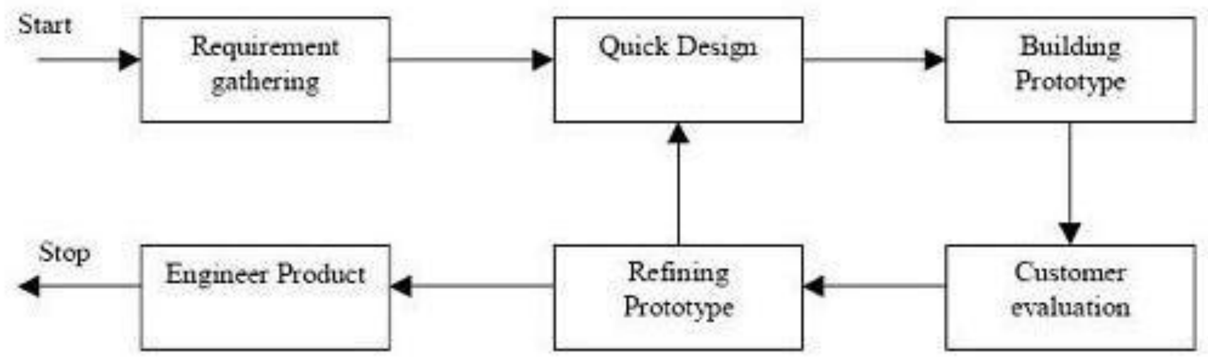
- When new changes are needed to be implemented.
- When there is a need to start the project right away (Limited planning is required to get started with the project).

Software Prototype Model

In Prototyping model, Software application prototypes are built, tested, and then reworked as necessary until an acceptable prototype is achieved and from which the final system can be developed.

Prototyping allows the end-users to evaluate developer proposals and try them out before implementation. For developers, it enables to understand the customer requirements in the early stages of software development as the prototype is demonstrated to the users to get their feedback.

- A prototype is an initial version of a system used to demonstrate concepts and try out design options. It is used to display some functionalities of a product under development but not the fully functionality.
  Ex. Reliability and security



Advantages

- End-users are actively involved in the development.
- Since a working model of the system is displayed earlier, the users get a better understanding of the system.
- Quicker user feedback leads to better solutions.

- Missing functionality can be identified easily.
- Confusing or difficult functions can be identified.

Disadvantages

- Risk of insufficient requirement analysis owing to too much dependency on the prototype.
- Scope of the system may expand beyond original plans.
- Users may get confused with the prototypes and actual systems.

Applicability of Prototyping model

- When the desired system needs to have a lot of interaction with the end users.

Rapid Application Development (RAD)

In Rapid Application Development model, the functional modules of a product are developed in parallel as prototypes and the final product is made by integrating them enabling a rapid delivery. The RAD model is based on prototyping and iterative development with no specific planning involved.

Phases of Rapid Application Development

Business modeling

The information flow is identified between various business functions. A complete business analysis is performed to find the vital information for business.

Data modeling

Information gathered from business modeling is used to define data objects that are needed for the business.

Process modeling

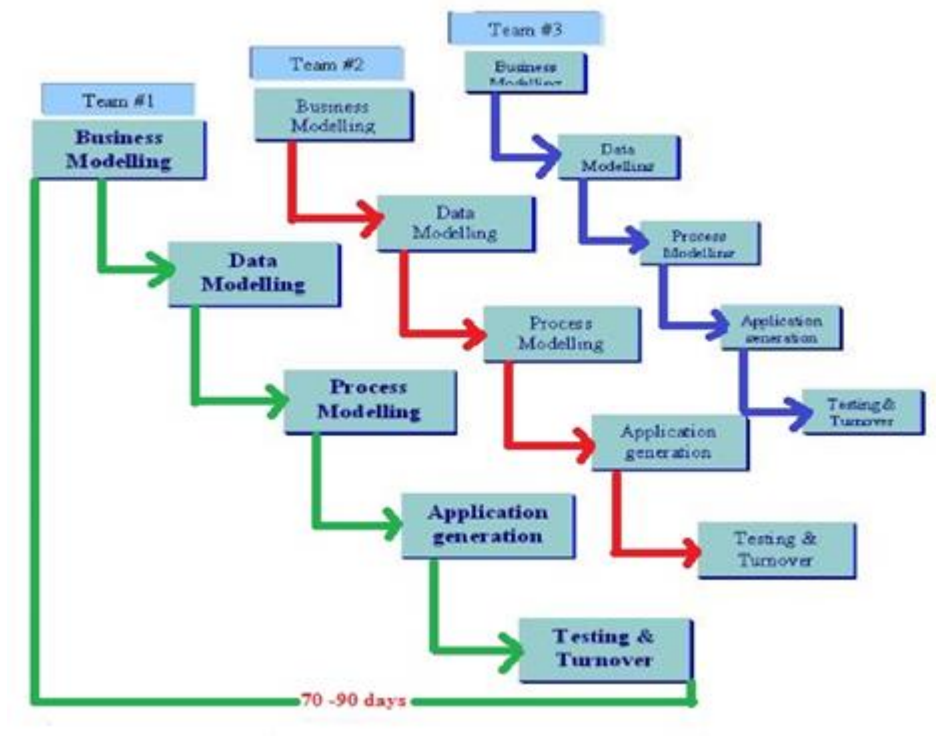
Data objects defined in data modeling are converted to achieve the business information flow with specific business objective. Process descriptions for adding, deleting, retrieving or modifying a data object are given.

Application generation

The actual system is built and coding is done by using automation tools to convert process and data models into actual prototypes.

Testing and turnover

Test built components and interfaces. The overall testing time is reduced in the RAD model as the prototypes are independently tested during every iteration.



Advantages

- Reduced development time.
- Encourages customer feedback.
- Changing requirements can be accommodated.
- Integration from very beginning solves a lot of integration issues.

Disadvantages

- Only systems that can be modularized, can be built using RAD.
- Requires highly skilled developers/designers for modeling.
- High dependency on modeling skills.

- Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

Applicability of RAD model

- When a system can be modularized to be delivered in an incremental manner.
- When there is a need to produce the system in a short span of time.
- When the budget permits use of automated code generating tools.

System development methodologies

Methodology

Methodology is a specific, step by step strategy for completing one or more phases of the System development Life Cycle. There are two types of System Development methodologies.

1. Structured Methodology
2. Object-Oriented methodology

Structured Methodology

Structured Methodology provides a framework (structure) with a set of well-defined guidelines through steps of tasks. Structured methodologies support Process-oriented design of a software product. Process-oriented methodologies build models of systems based on studying the flow of inputs to processes to outputs.

Object-Oriented methodology

Object-Oriented methodology focuses on capturing the structure and behavior of information systems into small modules that combines both data and process. OO methods model a system as a collection of objects that work interactively. OO is a Data-oriented Methodology.

Features of Object-Oriented methodology

Classes and Objects

A class is a template definition of the behavior and attributes of a particular kind of object. An object is a specific instance of a class.

Encapsulation

Encapsulation combines processes and data into a single entity. Encapsulation hides Data of an object from the rest of the system and available only through the services of the class.

Inheritance

It is the process by which object of one class acquires the properties or features of another class. The concept of inheritance provide the idea of reusability.

Abstraction

Abstraction is the process of taking or selecting only the necessary methods and attributes to specify the object.

Polymorphism

Polymorphism is the ability to take multiple forms. It allows to perform a single action in different ways.

Structured System Analysis & Design Methodology (SSADM)

One of the structured methods for System Analysis & Design approach introduced in early 1980s by the Central Computer and Telecommunications Agency (CCTA) in UK. SSADM involves study the present system and sketches a blueprint to develop a new system or to modify the existing system.

Modules of SSADM

Feasibility Study

The business area is analyzed to determine whether the system development is practical and beneficial.

Requirements Analysis

The requirements of the system to be developed are identified and the current business environment is modeled in terms of the processes carried out and the data stored.

Requirements Specification

Detailed functional and non-functional requirements are defined and new techniques are introduced to define the required processing and data storage.

Logical System Specification

System to be developed is specified logically without taking technical constraints into consideration.

Physical Design

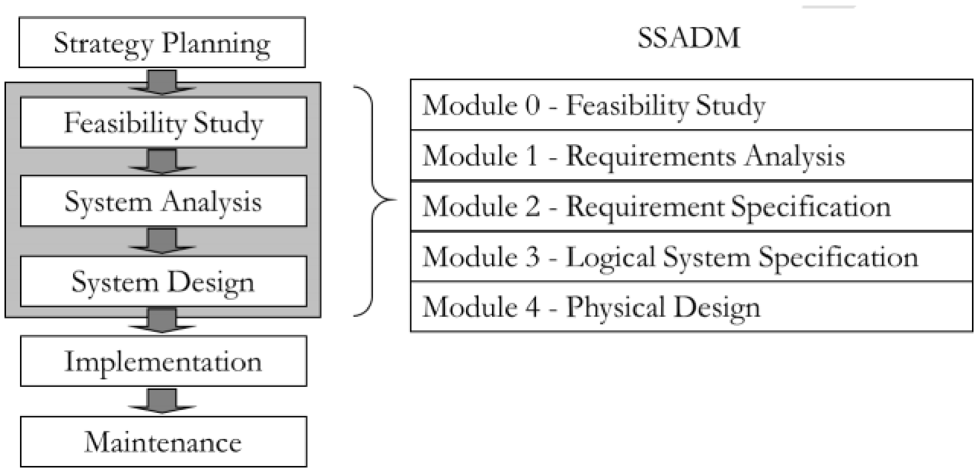
Logical design is transformed into a physical design taking technical constraints into consideration.

Benefits

- More focus on analysis and design
- Better quality system specification and documentation
- Effective communication and user involvement
- Flexible continuity when staff changes
- Improved management control

Stages of the SDLC covered by SSADM

SSADM covers most of the System Development Life Cycle (SDLC) from Feasibility Study to System Design with more focus on analysis and design.


Strategy Planning
Feasibility Study
System Analysis
System Design
Implementation
Maintenance
SSADM
Module 0 - Feasibility Study
Module 1 - Requirements Analysis
Module 2 - Requirement Specification
Module 3 - Logical System Specification
Module 4 - Physical Design

Preliminary Investigation

Preliminary Investigation is the first stage of system development life cycle. Preliminary Investigation focuses on recognizing the need for a new system and reaching a clear initial picture of what the physical system actually is.

Preliminary investigation is done in two phases.

1. Problem definition
2. Feasibility study

Problem definition

A preliminary survey of current system is carried out to identify the problems in the current system. It identifies the priorities of the problems to be solved and suggests alternative solutions. It identifies the scope of the system.

System scope

The scope is the set of boundaries that define the extent of a software project**.** It includes major deliverables, key stakeholders and constraints.

Feasibility study

Feasibility study evaluates the solution strategies for its feasibility. It is conducted to find out whether the proposed system is possible, affordable and acceptable for the organization. Feasibility is evaluated from both developers' and users' point of view. The feasibility of the system is evaluated mainly on technical, economical, operational and organizational aspects.

Technical feasibility

Technical feasibility evaluates whether the developers have ability to construct the proposed system. The technical assessment answers questions such as whether the technology needed for the system exists, how difficult will the system to be developed, or whether the developers have enough experiences to use required technology.

Economic feasibility

Economic feasibility studies cost and benefits to evaluate whether the benefits justify the investments in the system development. An important outcome of the economic feasibility study is the cost-benefit analysis.

Operational feasibility

Operational feasibility assesses the willingness and ability of the users to support and use the proposed system. Will the system be useful when it is developed and installed? Will there be any resistance from users to the system development? How will the work environment be affected? And how much will it change?

Organization feasibility

Organization feasibility determines the extent to which the proposed system supports the objectives of the organization’s strategy. In here, the system is taken as a subset of the whole organization.

The result of the feasibility study is a formal document known as “Feasibility Report”, a statement for the top management detailing the nature and scope of the proposed solution.

Once the feasibility study is done, the project is approved or disapproved based on the results of the study. If the project seems feasible and desirable, then the project is finally approved otherwise no further work is carried out.

Requirement analysis

Requirement analysis is the process of studying and analyzing the user needs to arrive at a definition of the problem domain and system requirements. The Aim is to document the customer requirements i.e. the Problem.

This phase involves two activities,

1. Requirements gathering and analysis
2. Requirements specification

Requirements gathering and analysis

Information is gathered via meetings, interviews and discussions. The main objective of requirement analysis is to discover the boundaries of the new system and how system must interact within the problem domain. Requirement analysis helps to detect and resolve conflicts between user requirements.

IEEE standard for requirement definition

- Essential requirements are defined with “**Shall**”
- “Nice to have” requirements are defined with “**Should**”
- An Actor

  Ex. The system Shall be able to keep product details
  The user shall be able to register a new account and log into the system

Different Types of System Requirements

- **Functional requirements**
- **Non Functional Requirements**

Functional requirements

Functional requirements describe the activities that the system should carry out. They describe a software system as a set of functions.

Ex. **Shall** be able to generate payment receipts

Should display Total no of sales in Red

Non Functional Requirements

Non Functional Requirements are characteristics/qualities of the system that cannot be expressed as functions. They describe how well or with in what limits requirement should be satisfied.

Ex. Security, Reliability, Usability, Maintainability and Portability

Shall facilitates adequate protection for user accounts

Requirements Specification

A detailed and precise description of the system requirements is set out to act as a basis for a contract between client and software developer.

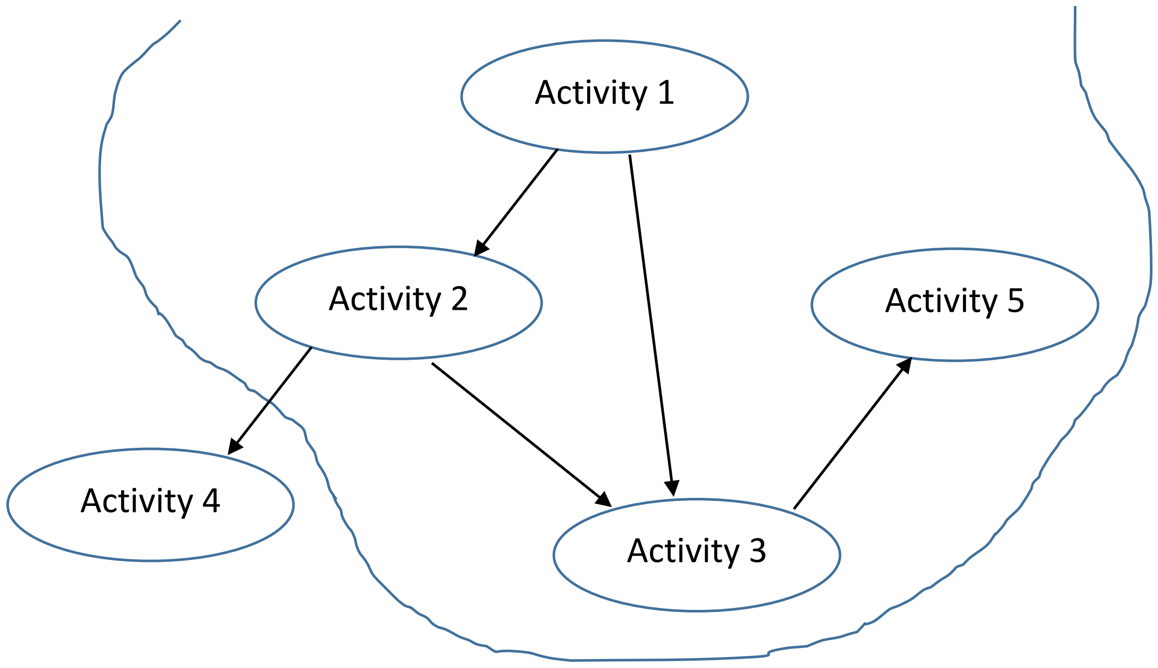
Requirements Analysis Tools

Business Activity Modeling

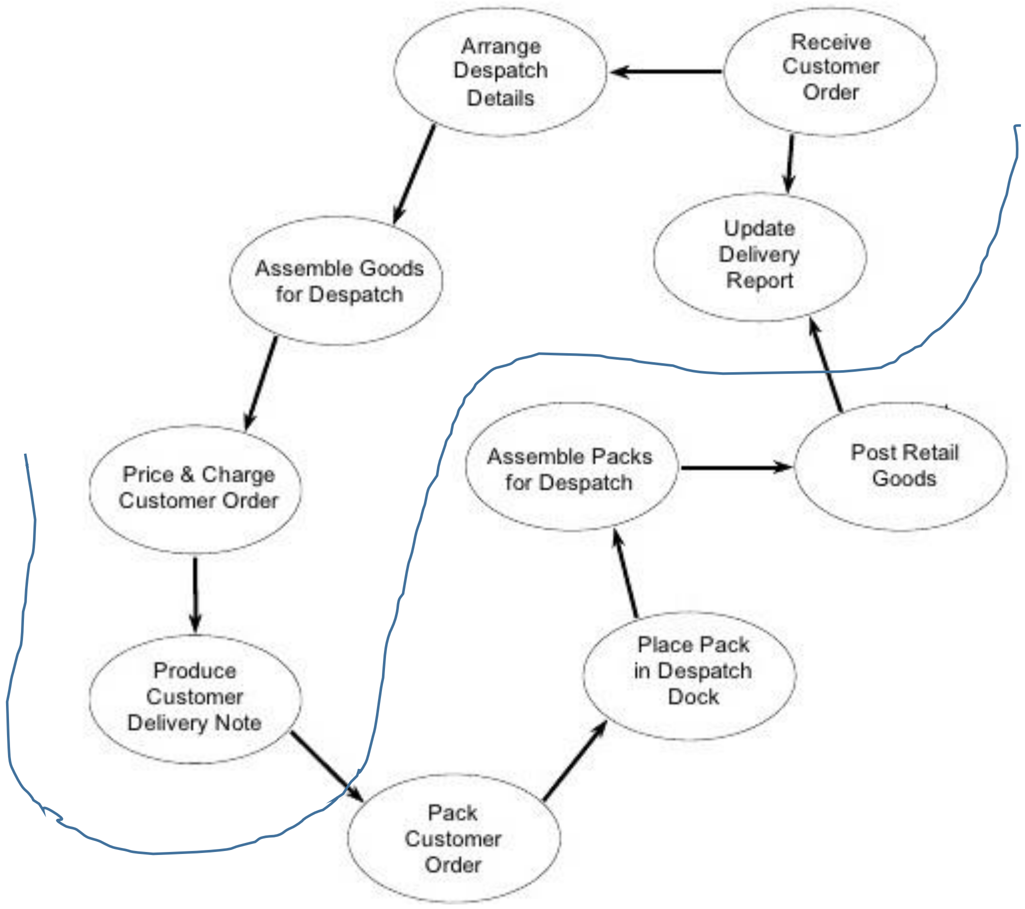
Business Activity Modelling is a technique to show high-level activities and logical dependencies as a conceptual model, based on a stakeholder's perspective. It is used to show the business activities that the actors in the system's environment do and their associations. It is a start-off technique for understanding "what's going on" in the system under investigation.

Business Activity Modeling Steps

1. Identify business activities carried out in the system environment
2. Graphically represent the activities
3. Link activities in the order they occur
4. Define the System's boundary by grouping data dependent activities



Ex. E-commerce transaction


Arrange Despatch Details
Receive Customer Order
Assemble Goods for Despatch
Update Delivery Report
Price & Charge Customer Order
Assemble Packs for Despatch
Post Retail Goods
Produce Customer Delivery Note
Place Pack in Despatch Dock
Pack Customer Order

Data Flow Modeling (DFM)

Data Flow Modeling is used to model data processing in the system. It is used to define partitions into sub systems. DFM consists of a set of Data Flow Diagrams (DFD) and associated textual descriptions.

Data Flow Diagrams (DFD)

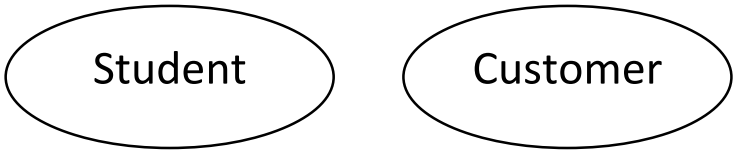
DFD illustrates the way in which the data is passed around the system, how data is processed within the system and where data is stored in the system.

Components of a DFD

External Entities

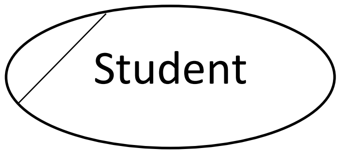
- Represents people, organizations or other systems external to the system under investigation
- Acts as a source or a recipient of data
- Name should refer to a generic type, not to an instance of that type

Ex.



- It is common practice to have duplicates of external entities in order to avoid crossing lines, or just to make a diagram more readable.

Ex.

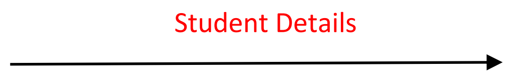


A duplicate external entity

Data Flows

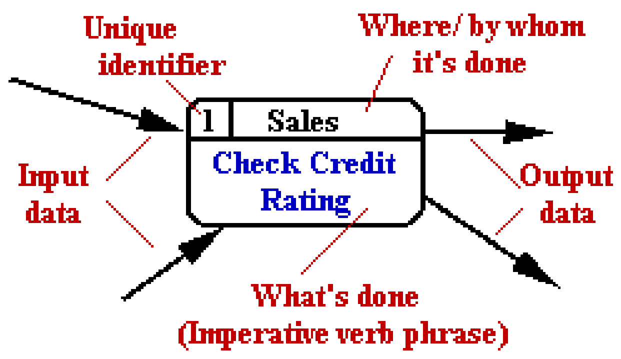
- Show flows of data to, from and within the system
- indicate the direction of the flow of data
- Link other components in a DFD
- Each data flow should be named

- Represented with solid arrows, however data flows between two external entities are shown by **dashed arrows**
- Intersections should be avoided



Processes

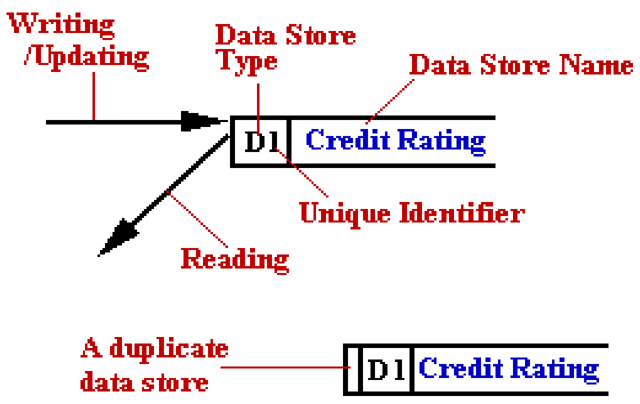
- Represents business activities carried out in the system
- Each process has three properties: Id, Name and Location
- Processes that don't need any further decomposition is called elementary processes



Data Stores

- Used to hold data within the system
- Four types,
    - Computerized (D)
    - Temporary data (T) – Ex. temporary program file
    - Manual (M) – Ex. filing cabinet
    - Manual Temporary T(M) – Ex.in-tray, mail box

- Each data store has three properties: Id, Type and Name



Data Flow Modelling Rules

- All data flows going into the system must be received by a process
- All data flows going out of the system must be generated by a process
- Data can flow directly between,
  - Two external entities
  - An external entity and a process
  - Two processes
  - A Process and a data store
- A direct data flow cannot exist between,
  - An external entity and a data store
  - Two data stores
- Each process should have at least one input and an output.
- Each data store should have at least one data flow in and one data flow out.
- Inputs to Data Stores come only from Processes
- Outputs from Data Stores go only to Processes
- Processes and Data stores must not originate data

DFD Naming Guidelines

External Entity	→ Noun
Data Flow	→ Names of data
Process	→ Verb phrase
Data Store	→ Noun

Context Diagram

Context Diagram is an overall, simplified, view of the target system, which contains only one process box and the primary inputs and outputs. Context Diagram is the highest level of abstraction of the system. It represents the entire system as a single process and shows how the system interacts with its external entities.

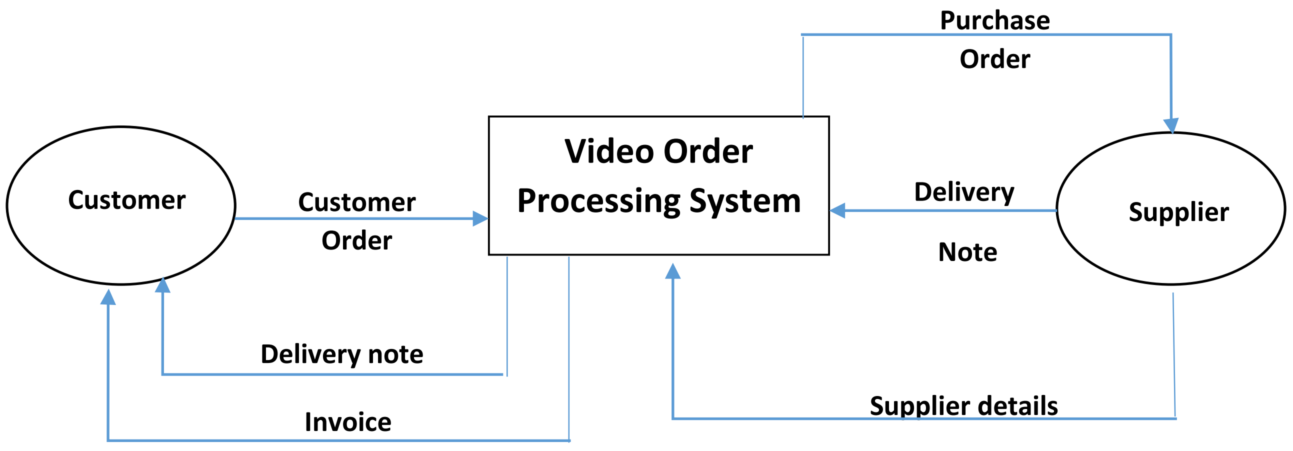
Steps to draw a Context Diagram

- Identify all the external entities interact with the system
- Identify all the data flows from external entities to the system and to external entities from the system
- Represent the system as a single process

- Represent external entities
- Add data flows between the system and external entities

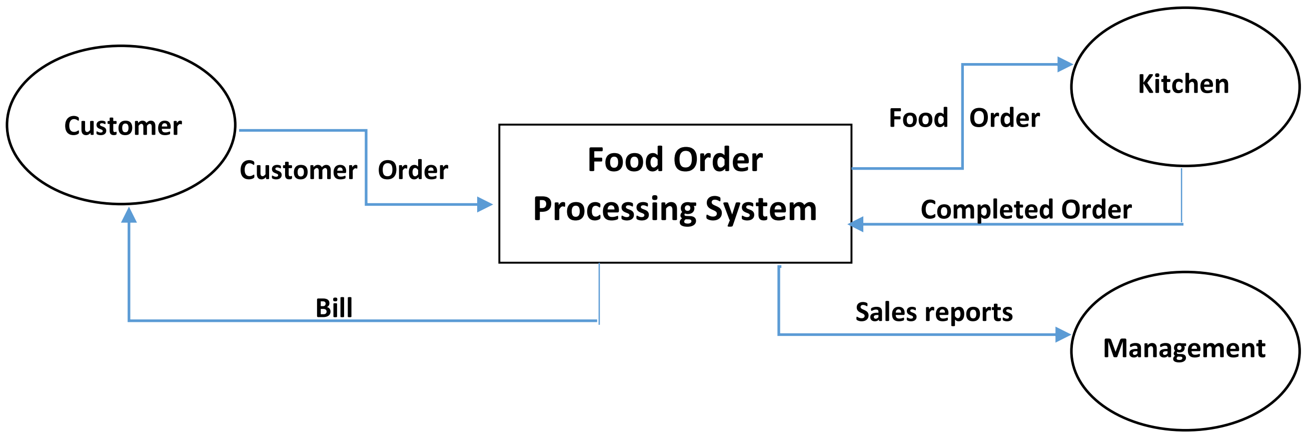
Ex. A Video shop

When a customer orders a video tape the Video shop will deliver the video with a delivery note and an invoice. A purchase order will be sent to the supplier if the item is not available. And the supplier will send a delivery note after delivering the requested videos. The Video shop maintains a supplier list.



Ex. Burger Queen Restaurant

When a customer makes an order, the restaurant records the customer order. It then generates a food order and send it to the kitchen. After delivering the order restaurant bills the customer. Sales reports are generated and send to the management daily.



Case Study

A Camp Reservation System for Adventure Camping

As its name suggests, the primary business of Adventure Camping is in selling camping holiday packages to the general public. A typical camping holiday package consists of one or more tents and few essential items for a reserved period of time, maintained by the company in their own camping sites.

Typically, customers make their enquiries over the phone on forthcoming holiday packages from the sales desk at Adventure Camping. The main duties at sales desk are to refer the package details and provide them to the customer, go through the pending and confirmed reservations to check the availability of the interested package and negotiate with the customer about the dates and offer other packages if the required dates have already been reserved. If an interested package is available, the customer is asked to provide his/her personal details and the number of tents needed and then a pending reservation is made.

To confirm the pending reservation, the customer is requested to settle the full payment to the cashier at least 7 days prior to the reservation and collect the payment receipt. Otherwise the pending reservation will be expired. The cashier uses reservation details provided by the customer to go through the pending reservations in order to accept payment and issue receipt. Once paid, the cashier stamps the corresponding pending reservation as “Paid”. At the end of the day, she uses payment details to prepare a daily-collection summary and sends it to the management.

The reservations clerk goes through the pending reservations in every morning to extract any stamped and/or expired reservations. For each stamped pending reservation, she makes a confirmed reservation for the corresponding camping holiday package with customer details in the pending reservation. Then the both stamped and expired pending reservations are discarded. For each confirmed reservation, she prepares a document known as booking note, which consists of customer details, number of tents needed and any other special remarks.

The booking-note is then sent to the site supervisor at the relevant camping site to arrange the site to suite to the customer’s requirements. The customer can obtain the above mentioned essential items provided by Adventure Camping by producing the payment receipt to the site supervisor at the time of check-in to the site.

Occasionally, the reservations clerk receives new site information from the management. Using such information, she prepares new camping holiday packages and makes them available for reservation. She also uses revised rates sent by the management to update the rates of the existing packages.

At present, the company carries out its day-to-day operations in a completely manual manner and the amount of paper work makes the system rather error-prone and inconvenient. Adventure Camping therefore wants to computerize their reservation system not only to cover the present operations but also to meet their future demand. As a part of their high level strategic review, the management has identified the following major business problems experienced in the current system.

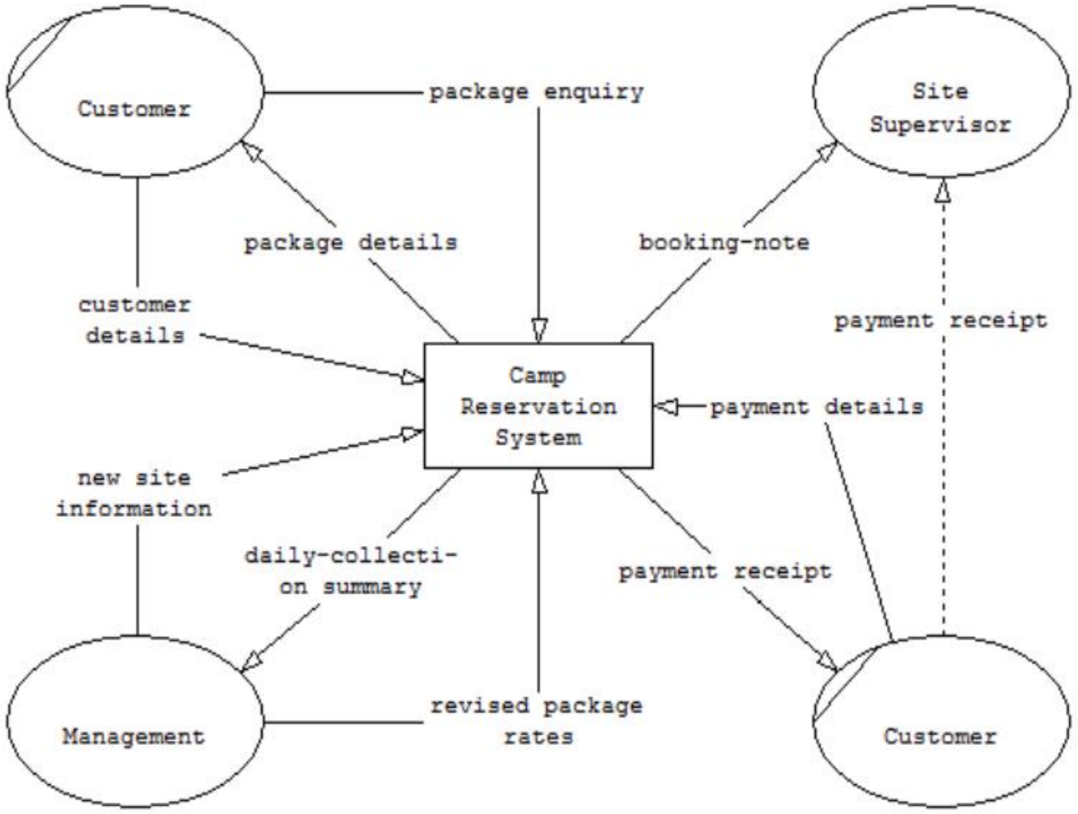
- It is inconvenient to go through the customer history to provide discounts for the regular customers.
- When the reservation clerk is going through the confirmed or pending reservations, the sales desk may have to hold all the incoming customer enquires till such information is available for her reference.
- When the pending reservations are in reference at the sales desk or at the reservation clerk, the cashier may have to ask the customers to wait at the counter till it is available for her reference.

Clearly many more business issues and requirements will be uncovered during the new computerization project, but it is highly expected to address the above problems with any proposed system. It is also nice to have the assistance of the new system for the management decision-making process with an array of data analysis tools. Moreover, to address a worldwide audience, Adventure Camping is looking for new means of advertising, reservation and payment.

The office staff of Adventure Camping is familiar with the software such as Microsoft Office and the company already has few IBM compatible PCs running on Windows XP. Therefore, the management is highly keen in making use of these existing hardware and software wherever possible in the new computerization project.

If you are the system analyst of the consulting software firm hired by Adventure Camping for the development work, analyze and then design a computerized information system to solve their business problem using SSADM.

Context Diagram

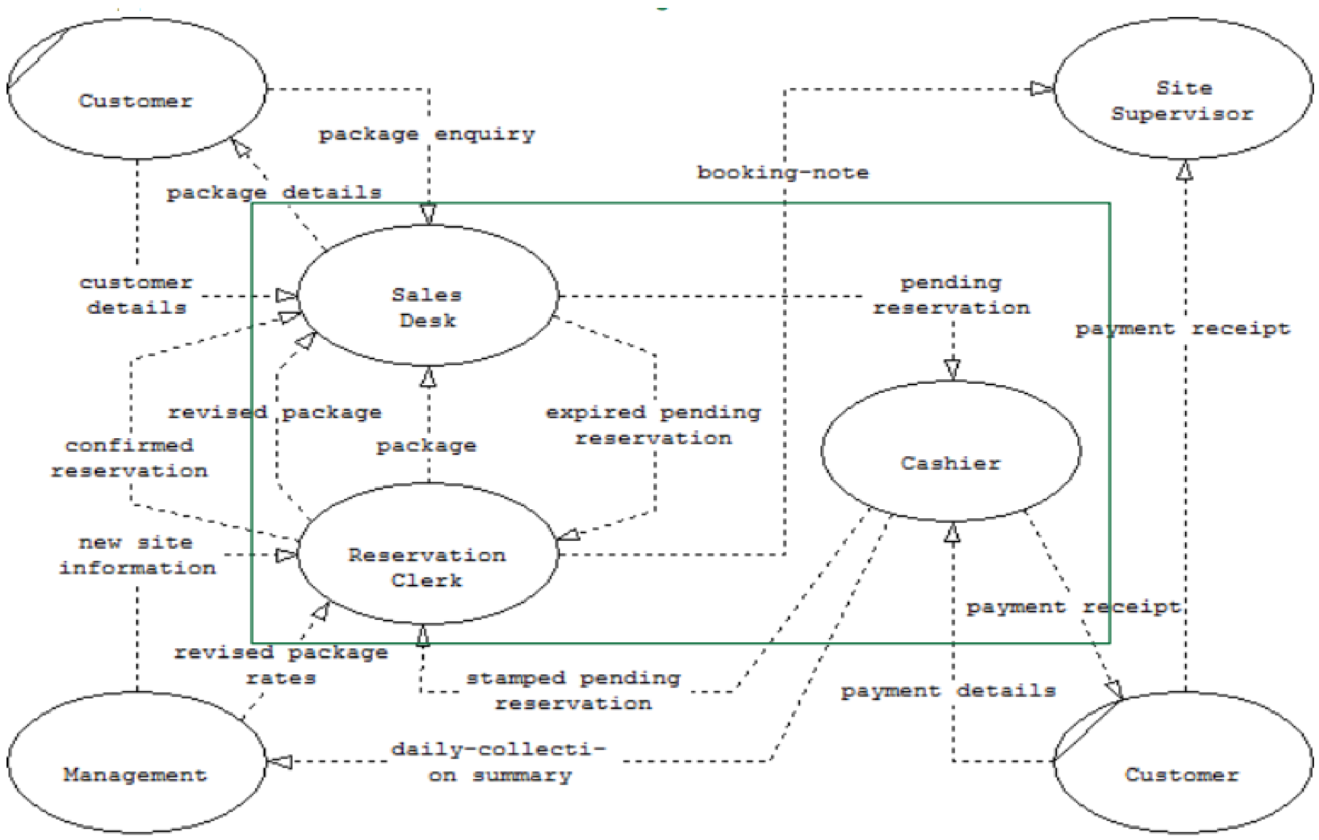


Document Flow Diagram

Document Flow Diagram acts as a bridge between the Context Diagram and Level 1 DFD. It illustrates how documents (papers, conversations, data passed among computers) passed in the system.

- Steps to draw a Document Flow Diagram
    - Identify Sources and Recipients of documents
    - Identify the documents which connect them
    - Represent both Sources and Recipients in external entity symbols
    - Add data flow arrows to represent the documents connecting the Sources and Recipients
    - Add System boundary to exclude the external entities identified in the Context diagram

Source	**Document**	**Recipient**
Management	New site information	Reservation Clerk
Reservation Clerk	Package	Sales Desk
Customer	Package enquiry	Sales Desk
Sales Desk	Package details	Customer
Customer	Customer details	Sales Desk
Sales Desk	Pending reservation	Cashier
Customer	Payment details	Cashier
Cashier	Payment receipt	Customer
Cashier	Daily-collection summary	Management
Cashier	Stamped pending reservation	Reservation Clerk
Sales Desk	Expired pending reservation	Reservation Clerk
Reservation Clerk	Confirmed reservation	Sales Desk
Reservation Clerk	Booking-note	Site Supervisor
Customer	Payment receipt	Site Supervisor
Management	Revised package rates	Reservation Clerk
Reservation Clerk	Revised package	Sales Desk



Level 1 Data Flow Diagram

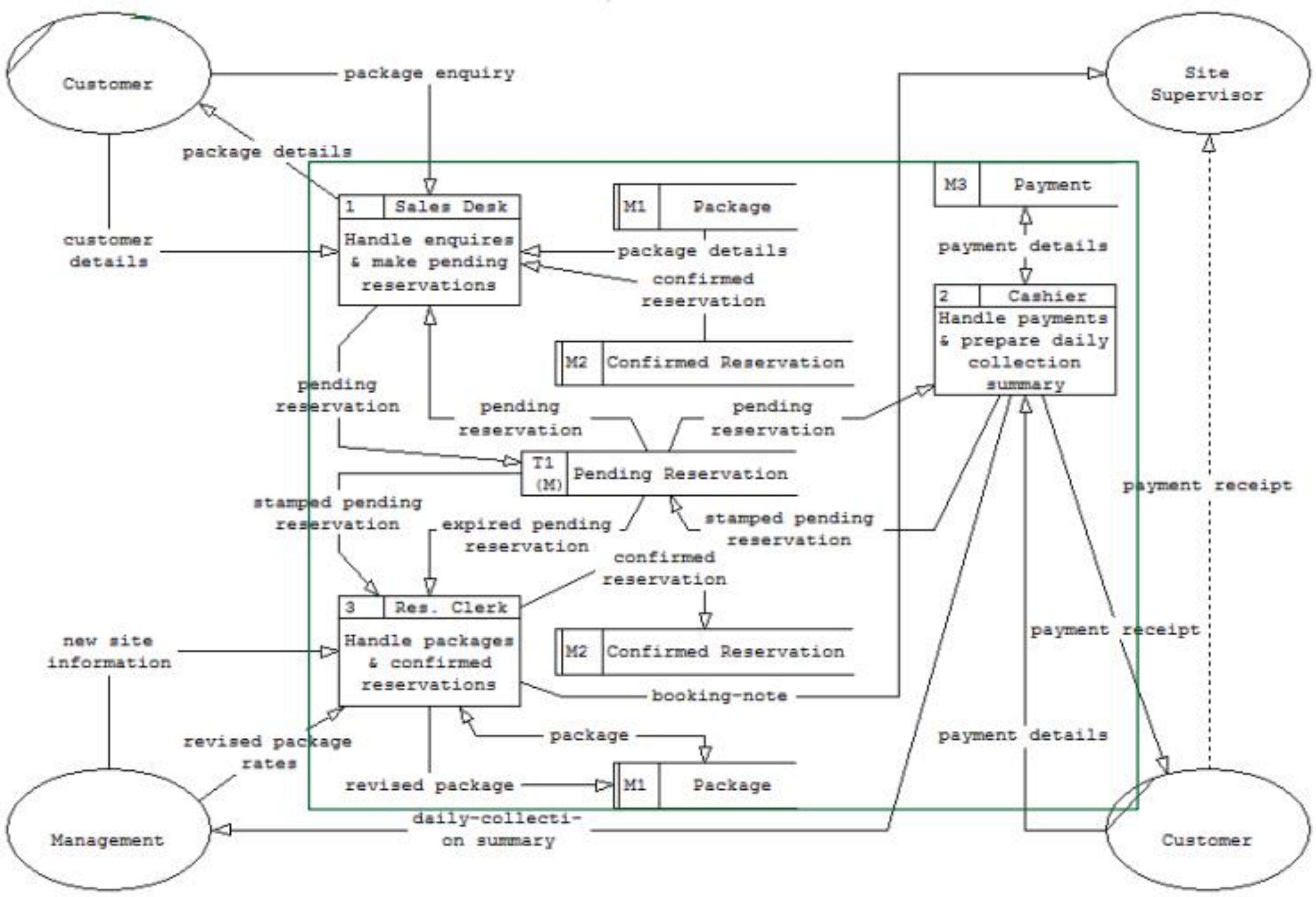
Level 1 DFD provides a higher level overview of the system's data processing. It shows data movements among the major components of the system. Level 1 DFD must be consistent with the Context Diagram.

Balancing DFDs

When decomposing a DFD, should conserve inputs to and outputs from a process at the next level of decomposition.

Transforming to Level 1 DFD

➢ For each document flow,
  - Identify the process that, generates the document or receives the document
  - Identify whether the document flow is stored by the process or created by the process by stored data.
  - Identify the required data stores



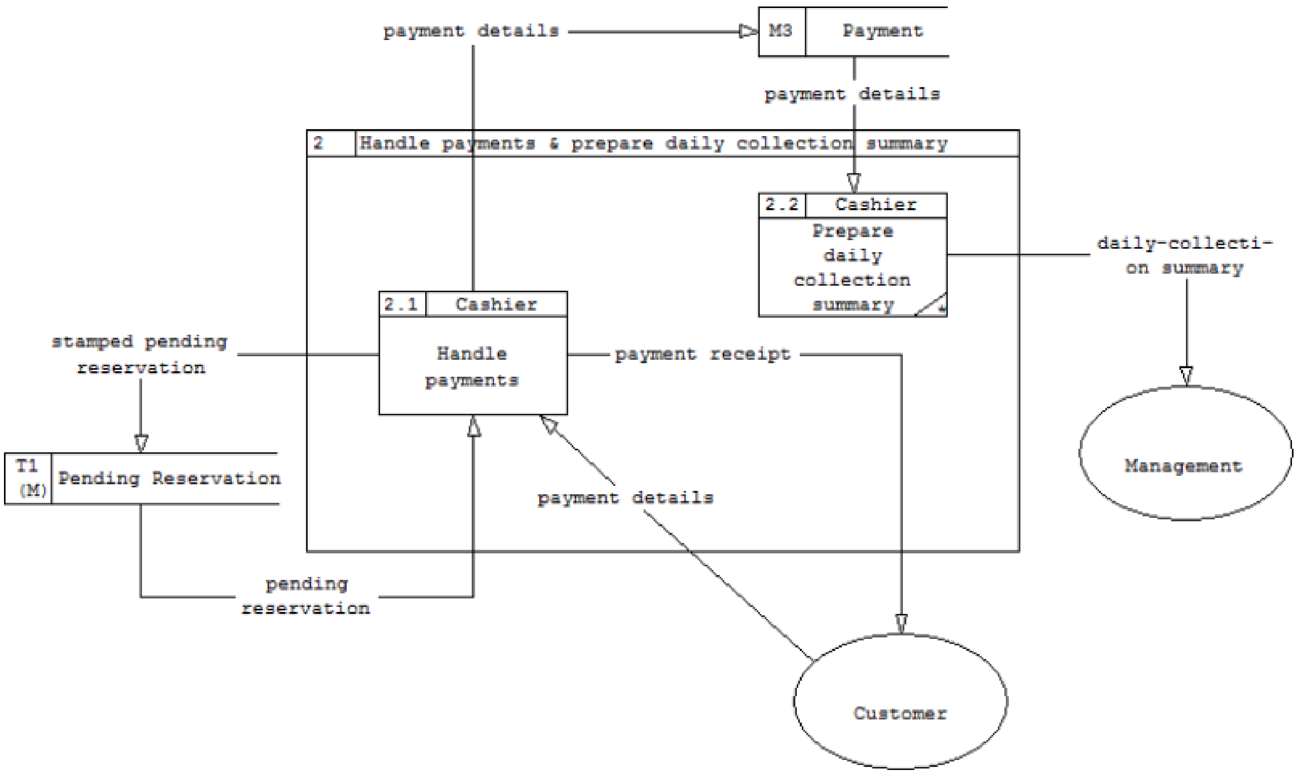
Lower level DFDs

- Level 2, 3,..
- Provide a way to go into more details successively
- Enable the top down approach
- Must be consistent with the parent DFD
- Typically limited to 4 - 6 processes

Rules on Decomposition

- Id and Name of the higher level process should become the Id and Location of the lower level DFD
- Data Sources and Recipients of the higher level process should remain outside the boundary of the lower level DFD
- If higher level process N is decomposed, Its lower level processes are identified as N.1, N.2, …

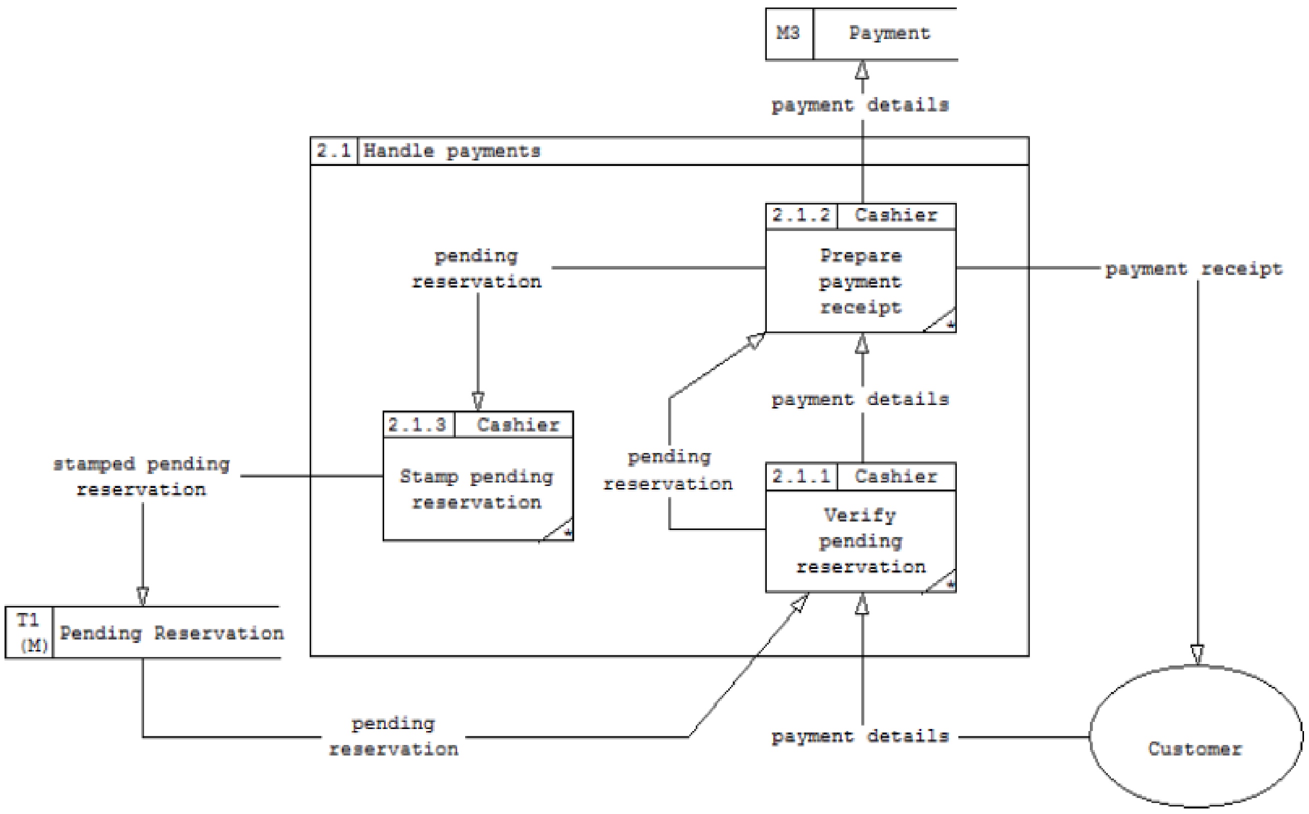
Ex. L2 DFD of Process 2



Elementary Process

- A process that doesn't need any further decomposition
- Associates a textual description known as Elementary Process Description (EPD)

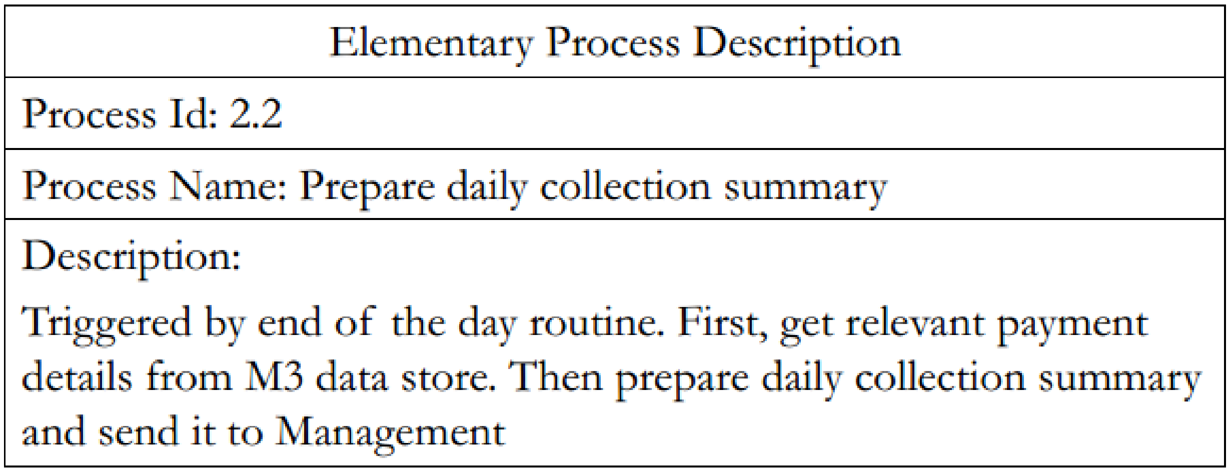
Ex. L3 DFD of Process 2.1



Elementary Process Description

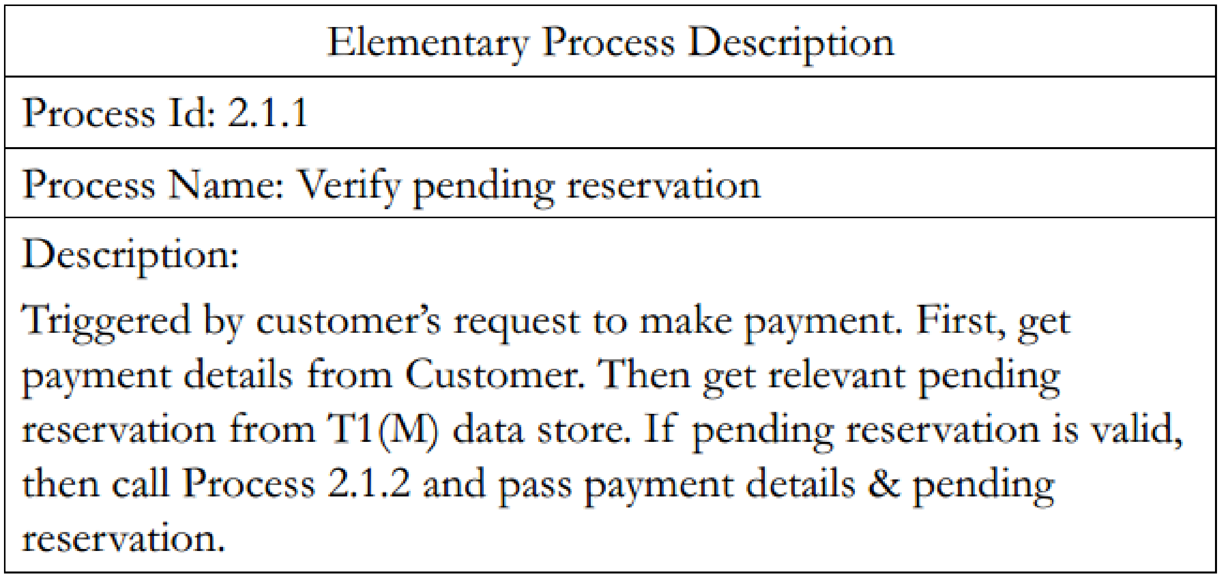
- Contains enough details for program specification
- Written in plain English or pseudo code

Ex. Elementary Process Description of Process 2.2



Elementary Process Description
Process Id: 2.2
Process Name: Prepare daily collection summary
Description: Triggered by end of the day routine. First, get relevant payment details from M3 data store. Then prepare daily collection summary and send it to Management

Ex. Elementary Process Description of Process 2.2.1



Elementary Process Description
Process Id: 2.1.1
Process Name: Verify pending reservation
Description: Triggered by customer's request to make payment. First, get payment details from Customer. Then get relevant pending reservation from T1(M) data store. If pending reservation is valid, then call Process 2.1.2 and pass payment details & pending reservation.

Logical Data Modelling (LDM)

Logical Data Modelling logically models the nature, structure and the applicable business rules of data stored in the system. LDM Consists of,

- Logical Data Structure (LDS) diagram
- A set of associated textual descriptions

Logical Data Structure (LDS)

LDS is a logical model of system's data processed by the processes identified in DFM. LDS,

- Illustrates how data interrelates to each other
- Shows how business rules apply on system's data
- LDS has 2 Components,
    1. Entities
    2. Relationships

Entity

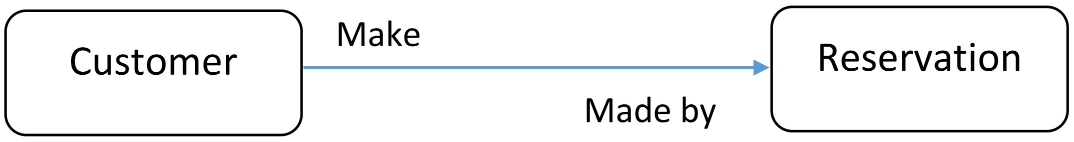
- A collection of logically associated and uniquely identifiable objects or concepts.
- Property of an entity is called an attribute
- Name should reflect the type of the entity, not an instance of that entity

Ex.

Relationship

- Shows how entities relate to each other

Ex.



Cardinality of a Relationship

Shows number of occurrences of each entity participating in a given relationship.

Three types of Cardinality

- One to one
- One to many
- Many to many

Optionality of a Relationship

Shows whether the relationship exists for all occurrences of participating entities or not.

Steps to draw LSD

- Derive entities from the physical data stores identified in DFM
    - Discover new entities
    - Merge existing entities
    - Discard, if already identified

- Identify relationships exist among the entities
    - Construct Entity matrix
    - Determine the cardinality and the optionality of the relationship

Ex. Camp Reservation System

- Deriving entities

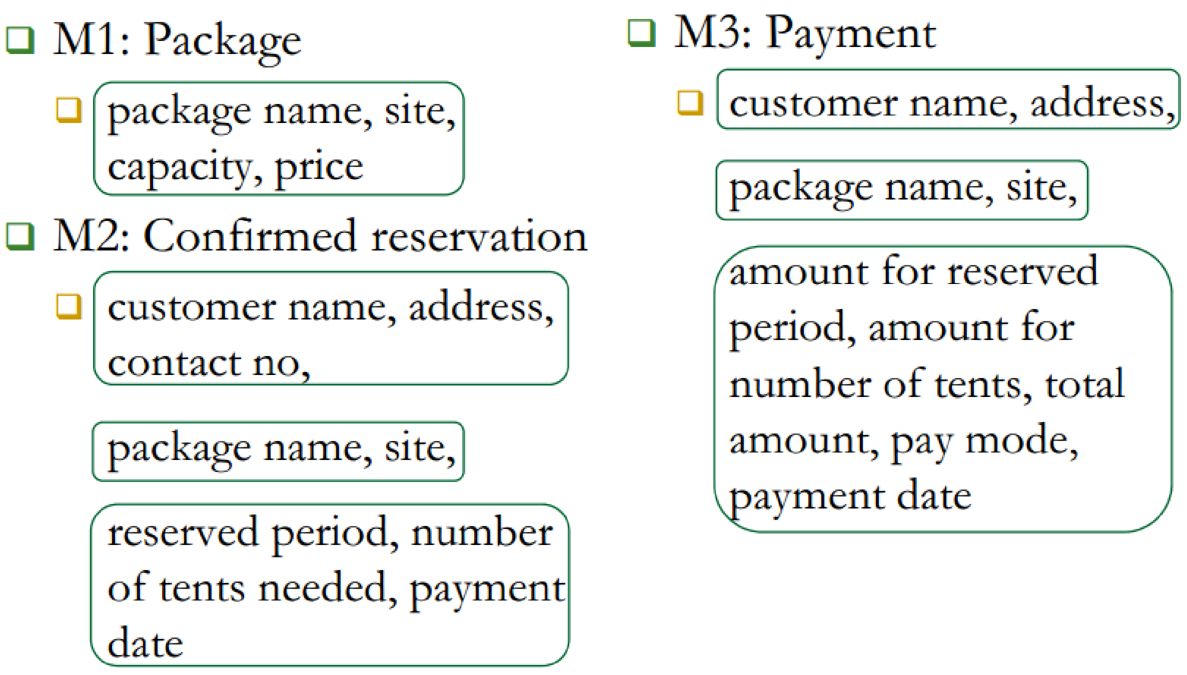
  Ex.

  Entity: Package

  Attributes: package name, site, capacity, price

  Entity: Confirmed reservation

  Attributes: customer name, address, contact no**,** package name, site, reserved period, number of tents needed, payment date



- M1: Package
  - package name, site, capacity, price
- M2: Confirmed reservation
  - customer name, address, contact no,

    package name, site,

    reserved period, number of tents needed, payment date
- M3: Payment
  - customer name, address,

    package name, site,

    amount for reserved period, amount for number of tents, total amount, pay mode, payment date

Entity: Package

Attributes: package name, site, capacity, price

Entity: Customer

Attributes: customer name, address, contact no

Entity: Reservation

Attributes: reserved period, number of tents needed, payment date

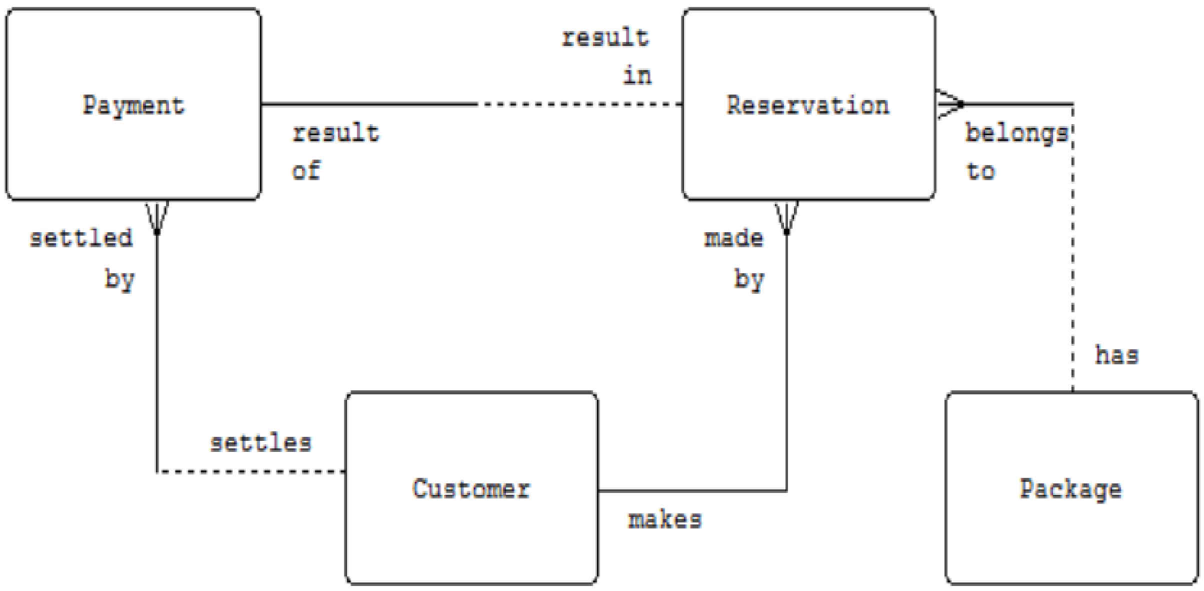
Entity: Payment

Attributes: amount for reserved period, amount for number of tents, total amount, pay mode, payment date

- Constructing Entity Matrix
    - Helps to identify the relationships among entities
    - Provides all possible pairings of entities to identify relationships among them in the interested system
    - Associations between each pair of entities are checked for once

	Customer	Package	Reservation	Payment
Customer			×	×
Package	■		×	
Reservation	■	■		×
Payment	■	■	■	

Logical Data Structure



Business System Options (BSO)

BSOs are possible solutions for the business problem. BSOs help to select the most appropriate option that satisfies the business requirements.

- May involve dropping some of less important requirements if they can not be cost justified
- Probably be a hybrid option

A BSO consists of,

- A functional description
- A high-level technical description
- Major benefits to the business
- Approximate cost estimate
- Development time scale
- Impact on organization and other existing systems

Ex. BSOs for Camp Reservation system

- BSO 1: Multi-user MIS
  Satisfies all the essential requirements of the business
- BSO 2: Web-based MIS
  Satisfies all the essential requirements plus online advertising, reservations & payments
- BSO 3: Multi-user DSS
  Satisfies all the essential requirements plus decision making support

BSO Vs. Requirements

ID	Requirement	BSO 1	BSO 2	BSO 3
1	Shall be able to keep package details	✕	✕	✕
2	Shall facilitate to get reservation details	✕	✕	✕
3	Shall facilitate to generate payment receipts	✕	✕	✕
4	Shall provide a GUI	✕	✕	✕
5	Should be able to provide decision-making support			✕
6	Should facilitate to advertise on web		✕	

Designing the Proposed system

Physical Vs. Logical Data Flow Modelling

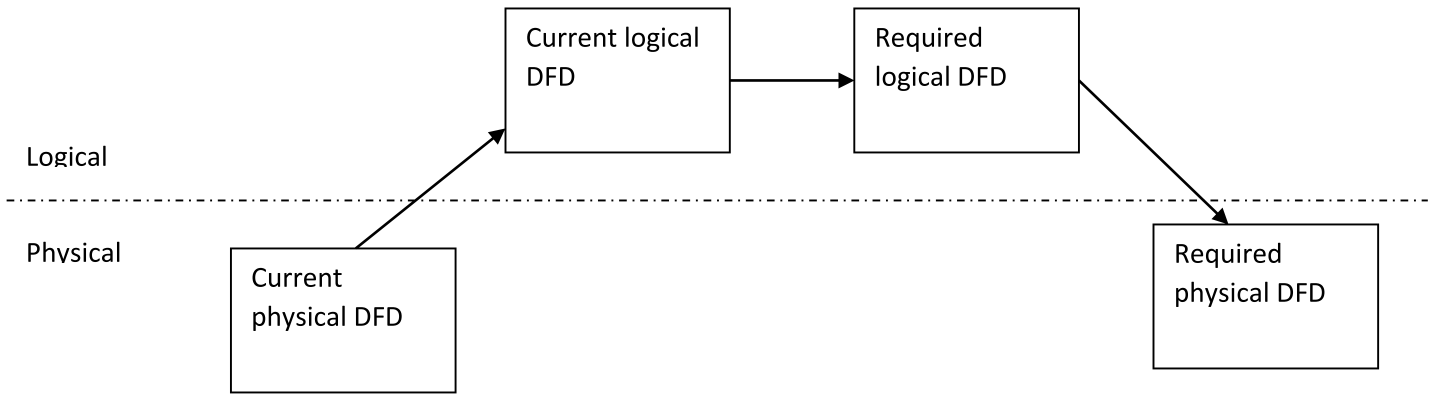
Physical DFM

- Shows how data is actually processed and where data is actually stored in the current system

Logical DFM

- Shows how data should be processed and where data should be stored in the proposed system

The progression of DFDs



Logical Data Flow Modelling

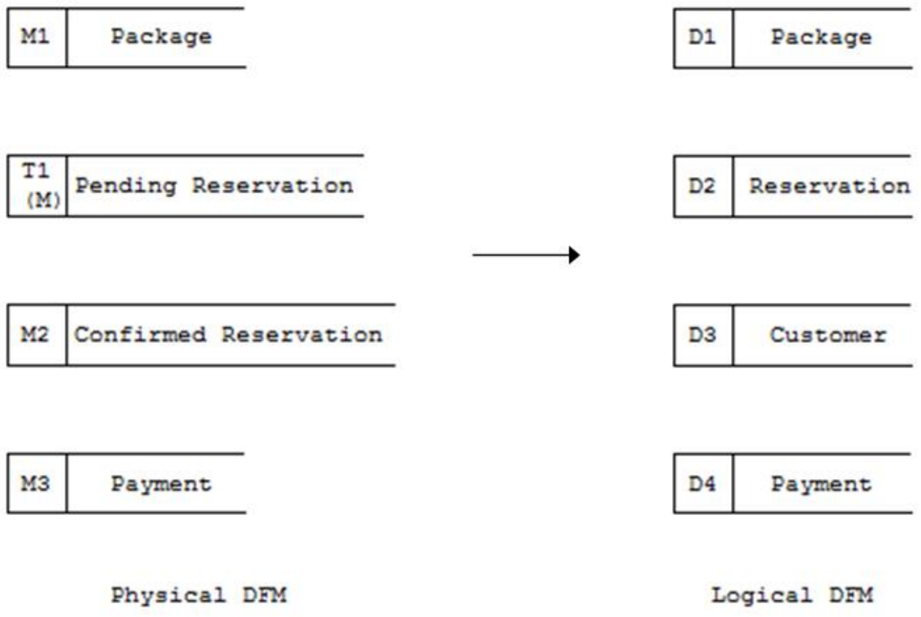
A bottom-up approach that removes physical elements of elementary processes and lower level DFDs and then reconstructs the hierarchy by re-grouping them. Consists of a set of DFDs and EPDs.

LDM of the proposed system

Rationalizing Data Stores

- Add new entities required to support any new functionality of the proposed system
- Use logical data stores (entities) identified in the LDS

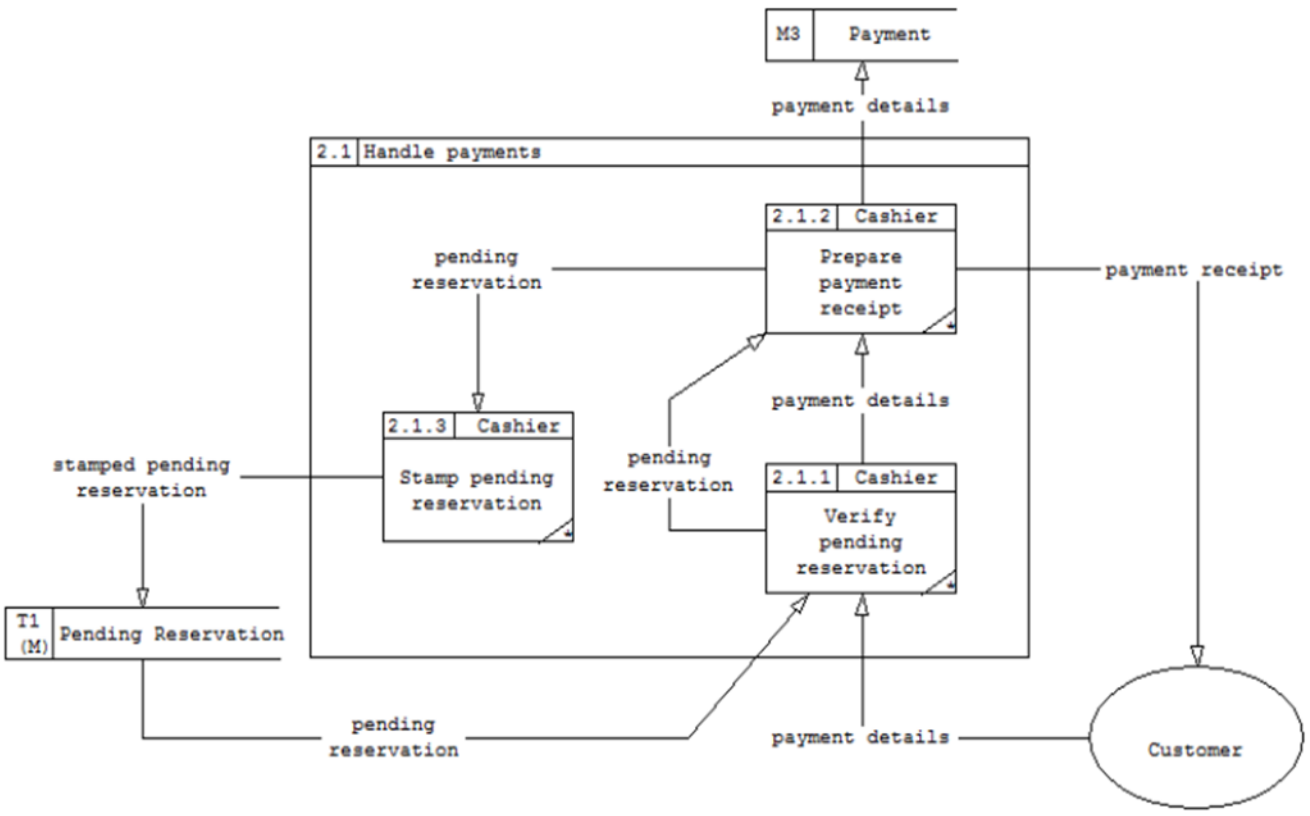
- Remove any temporary data store that simply halts data temporarily
- Transform all data stores into digitized data stores



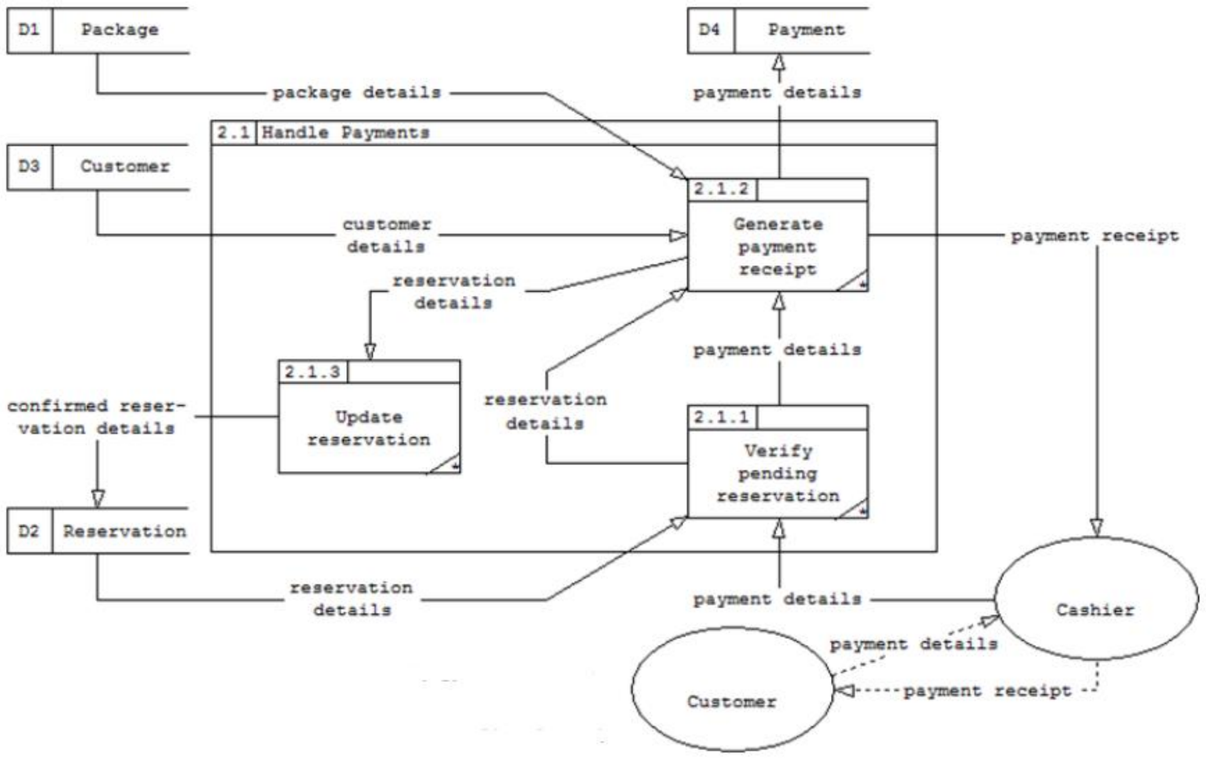
Rationalizing Processes

- Start from elementary processes
- As the location of a process only indicates a physical constraint, remove it from all processes
- As the person (location) who actually did a process in physical DFM now feeds inputs to the process in logical DFM, transform location into an external entity
- Remove any purely human activity and replace it with an external entity
- Add new processes required to support any new functionality of the proposed system
- Reconstructs the hierarchy by re-grouping logical processes based on their functionality

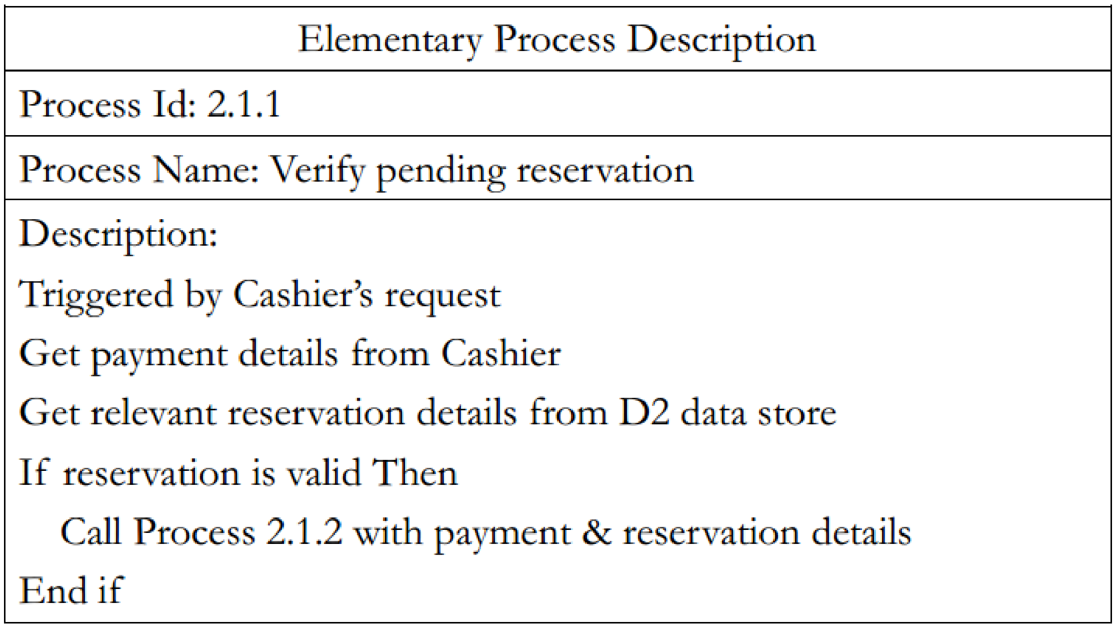
Ex. L3 DFD of Process 2.1



Logical DFM of Process 2.1



Elementary Process Description of Logical Process 2.1.1



Elementary Process Description
Process Id: 2.1.1
Process Name: Verify pending reservation

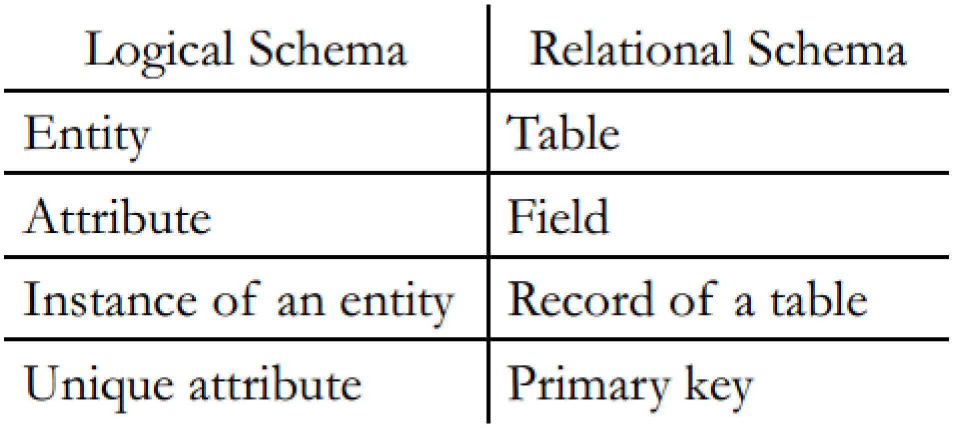
Description:

```
Triggered by Cashier’s request
Get payment details from Cashier
Get relevant reservation details from D2 data store
If reservation is valid Then
    Call Process 2.1.2 with payment & reservation details
End if
```

Physical Design of Database

Steps to construct the Physical Design

- Map logical schema to relational schema



Logical Schema	Relational Schema
Entity	Table
Attribute	Field
Instance of an entity	Record of a table
Unique attribute	Primary key

- Normalize all the relations in relational schema to 3NF
- Tabulate a Table Specification and a Record Specification for each relation in the normalized relational schema
- Construct the physical database

Table Specification

Ex. Customer

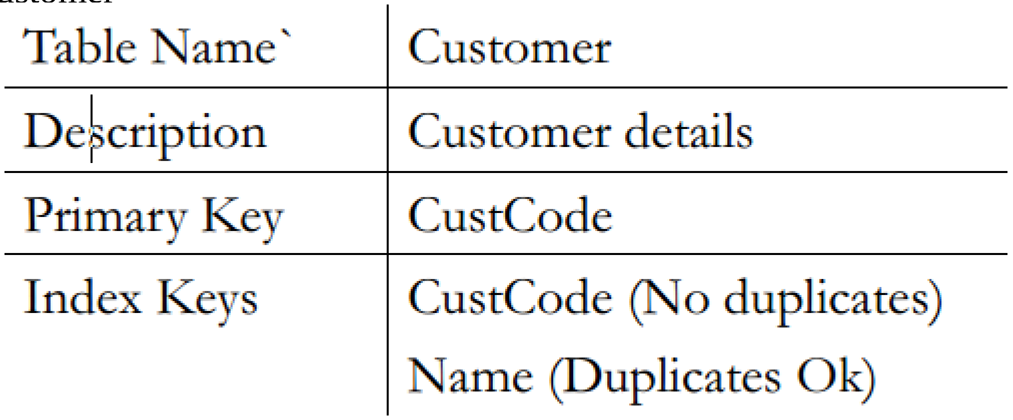
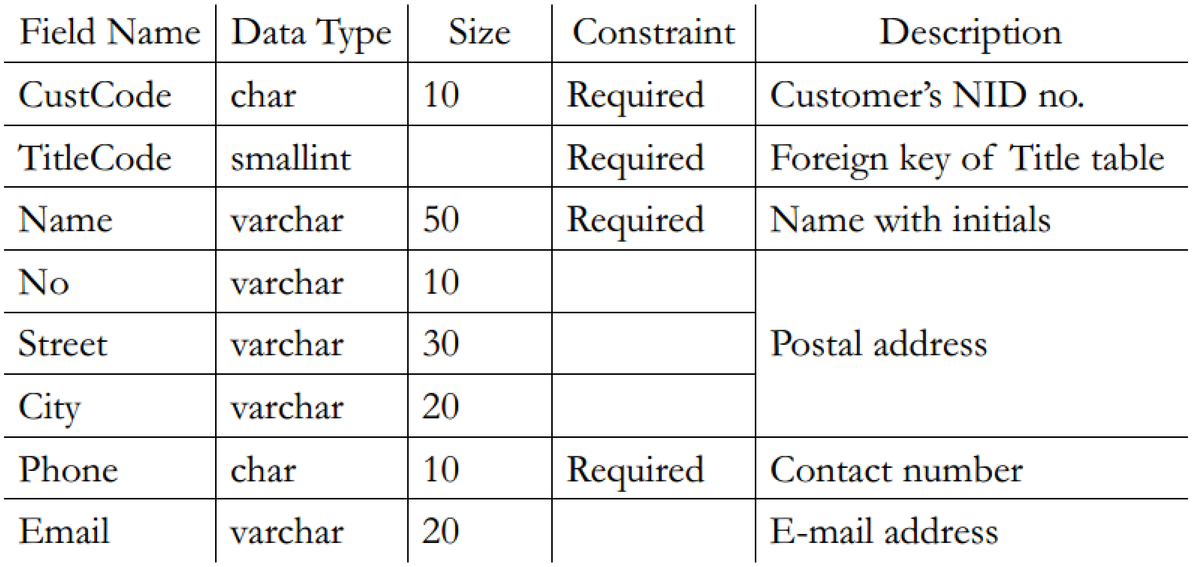


Table Name`	Customer
Description	Customer details
Primary Key	CustCode
Index Keys	CustCode (No duplicates) Name (Duplicates Ok)

Record Specification

Ex. Customer



Field Name	Data Type	Size	Constraint	Description
CustCode	char	10	Required	Customer's NID no.
TitleCode	smallint		Required	Foreign key of Title table
Name	varchar	50	Required	Name with initials
No	varchar	10		Postal address
Street	varchar	30		
City	varchar	20		
Phone	char	10	Required	Contact number
Email	varchar	20		E-mail address

Data Dictionary

- Is an integral part of database
- Holds information about the database and the data that it stores (data about data - metadata)
- Contains the actual database descriptions used by the Database Management System (DBMS)

Developing and testing the proposed system

Development of the proposed system

- Program development
- Database development

Testing the proposed system

Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

Test Cases

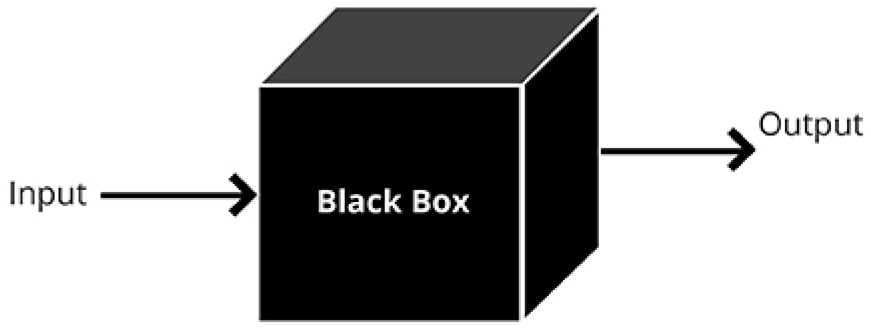
A test case is a set of actions executed to verify a particular feature or functionality of a software application. Test cases are documented by the Quality Assurance team while the software development is going on.

Software testing techniques

Black Box testing

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing.

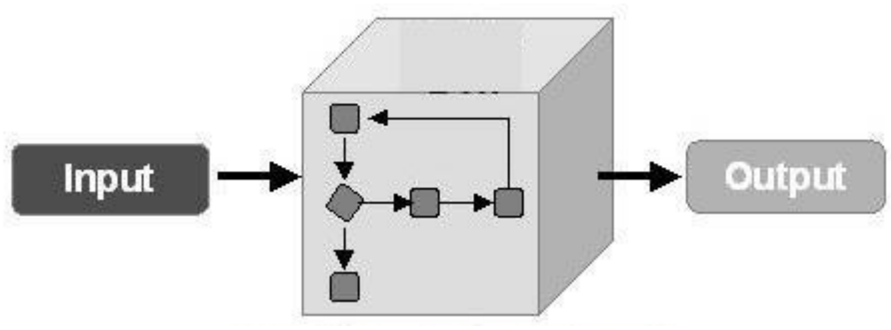
- Software to be tested is treated as a black box and its behavior is examined by studying the inputs and outputs
- Knowing the intended functionalities of a software, tests are conducted to see whether the software can deliver them
- Test cases are derived from the requirement specification of the software to be tested



White Box testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called glass-box testing or open-box testing.

- Takes internal implementation of a software to derive test cases to test the software
- Performed in the early stages of the testing process
- Usually applied for testing relatively small program units
- Analysis of the program code determines how many test cases are required to exercise all the internal components of the software (statements, branches, paths) adequately



Black Box testing Vs. White Box testing

Black Box testing	**White Box testing**
The internal workings of an application need not be known.	Tester has full knowledge of the internal workings of the application.
Performed by end-users and also by testers and developers.	Performed by testers and developers.
Limited coverage	Maximum coverage is attained
Does not help in optimizing the code	Helps in optimizing the code
Moderately skilled testers can test the application	A skilled tester is needed

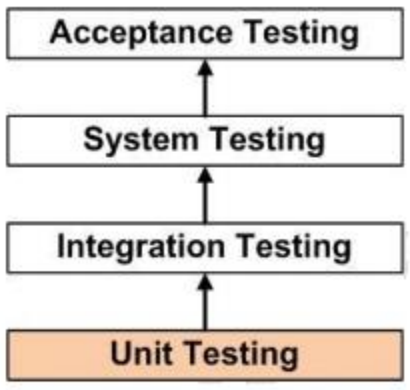
Software testing types/ levels

In software development,

- Software systems are built with sub-systems
- Sub-systems are built with individual program units such as functions or classes

Therefore, software testing,

- Starts with testing of these individual program units (**Unit Testing**)
- Continues with the testing of the integration of these units (**Integration Testing**)
- And the testing of the system's functionality as a whole (**System Testing**)
- Finally ends with testing to see whether the system is acceptable to the users (**Acceptance Testing**)



Unit Testing

Unit testing is performed by the respective developers on the individual units of source code assigned areas. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality. White box techniques are used.

Integration Testing

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Usually carried out by integration testers or test teams, can either be white box or black box.

Can be done in two ways.

- Bottom-up integration testing
  This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
- Top-down integration testing
  The highest-level modules are tested first and progressively, lower-level modules are tested thereafter.

System Testing

Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards, functional and non-functional specifications. Usually carried out by test teams independent of the programmers who developed the system. Black box techniques are used.

Acceptance Testing

The purpose of Acceptance Testing is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery**.** Usually carried out by test teams independent of the programmers and/or users who developed the system, black box techniques are used.

Deployment of the developed system

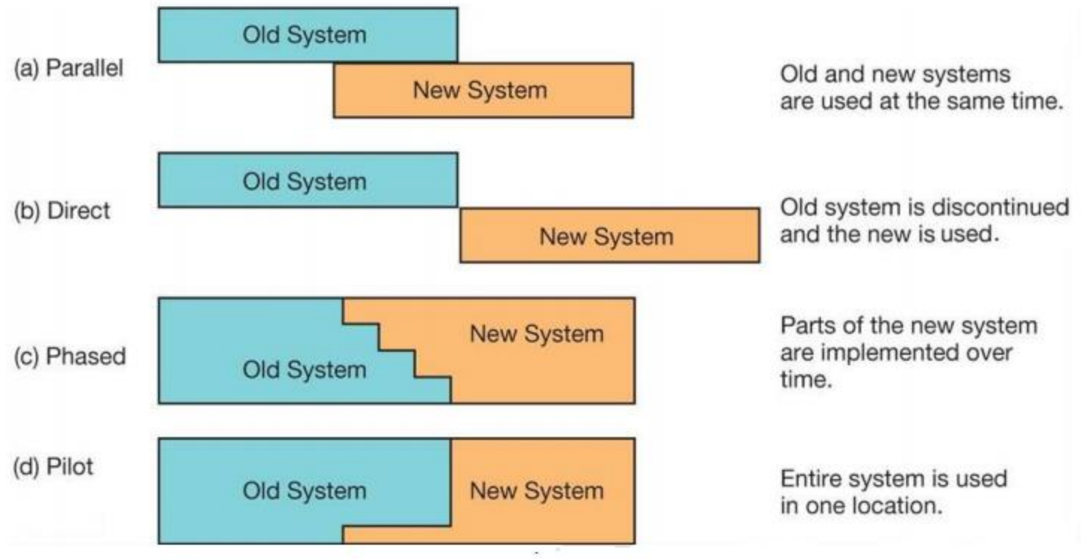
System deployment includes all the activities that make a software system available for use. The general deployment process consists of several interrelated activities with possible transitions between them.

Activities carried out in System deployment.

- Hardware/Software installation
- Data migration
- User training

These activities can occur at the developer's side or at the user's side or both.

Deployment methods


(a) Parallel
Old System
New System
Old and new systems are used at the same time.
(b) Direct
Old System
New System
Old system is discontinued and the new is used.
(c) Phased
New System
Old System
Parts of the new system are implemented over time.
(d) Pilot
Old System
New System
Entire system is used in one location.

Parallel Deployment

- Parallel deployment is a method of hardware or software migration that involves using the existing and new systems simultaneously until the implementation is judged to be completed and satisfactory.
- During the transition, users work with both systems as they gradually learn the new software.
- There is generally some duplication of effort as, for example, data must be entered in both systems. That duplication may lead to data quality issues.
- Parallel adoption is the most foolproof method.
- It can also be the most time-consuming option and is usually the most expensive.

Direct Deployment

- This is the simplest methods of deployment.
- Direct deployment involves getting rid of the existing system and transferring all users to the new system simultaneously.
- Implementation is faster as the old system is no longer available, users cannot put off learning the new system.

- The complete and simultaneous implementation avoids issues that can arise when users are working with different software or hardware.
- It can be hard on users to have to learn the new system immediately.
- The method can arise unpredictable and serious problems during implementation.
- Direct deployment may be the only viable method if the two systems are incompatible.
- This is the least expensive migration method, assuming there are no major problems.

Phased Deployment

- Phased deployment involves incremental implementation of a new system.
- As everything is not rolled out at once, the organization doesn't have to deal with all the potential implementation issues at the same time.
- Information learned from early implementation stages can be applied to guide the rest of the process, so that there are fewer issues as the implementation continues.
- A phased rollout also allows users to adjust to the new system gradually.
- It can be confusing to have groups of users working with different systems or to have users working with elements of different systems. That confusion can also lead to data quality issues.

Pilot Deployment

- A pilot deployment that involves rolling out the new system to a small group of users for testing and evaluation.
- During the pilot implementation, the users of the test group can provide valuable feedback on the system to make the eventual rollout to all users go more smoothly.
- Once the test group has approved the system, it can be rolled out across the organization.
- The testers can then help train other users for the new system.
- The test group can also determine whether the system is a viable option for the organization.

Maintenance

The maintenance phase of the SDLC occurs after the software is in full operation.

The maintenance phase involves,

- Making changes to hardware, software, and documentation to support its operational effectiveness.
- Making changes to improve a system's performance, correct errors, deal with security issues, or address new user requirements.

To ensure that the modifications do not disrupt operations or degrade a system's performance or security, organizations use change management standards and procedures.

System implementation with Commercial-Off-The-Shelf (COTS) packaged systems

Commercial-Off-The-Shelf (COTS) System

Commercial-off-the-shelf (COTS) software is a term for software products that are ready-made and are readily available for purchase in the commercial market.

COTS software systems are designed for both horizontal and vertical markets. In order to cater for the needs of many, vendors spend an enormous amount of time and effort to incorporate world-class functionality into their systems.

Advantages of COTS Packages

- Can be implemented in significantly less time than a custom developed software.
- Less resources are required in terms of human capital, office space and money.
- Have a greater chance of incorporating industry set standards.
- Tend to be far more configurable than custom developed software.

Disadvantages of COTS Packages

- Can be highly complex and usually include many features that will be never used.
- Vendors may cease their support or may go out of business.
- There may be a need for customizing the software to fully fit individual business functionalities or the business processes may need to be changed to accommodate them.
- Difficult to gain any competitive advantage from using COTS system as the competitors can also buy and use the same COTS.

Custom developed software systems

Custom developed software systems are designed specifically to the requirements and built to operate exactly as needed.

Advantages of using custom developed software systems

- Satisfy unique requirements
- Minimize changes in business procedures and policies
- Meet constraints of existing systems
- Can gain real competitive advantage with custom developed software solutions
- Can develop internal resources and capabilities

Disadvantages of using custom developed software systems

- Require a large initial investment and resources
- The development process can take a long time

Business Process

A set of activities, responsibilities, resources and data flows that interact to accomplish a business function.

Business process modeling

Business process modeling involves an in-depth analysis and then an optimization of the business process by removing inefficiencies and bottlenecks.

Business process gap analysis

As no COTS software solution has been specifically designed to meet any organization's unique requirements, there will be a gap between the business processes supported by the existing systems and those supported by the COTS software system.

It is very important to understand this gap well before the implementation begins and ensure that the organization can accept this gap without degrading its business performance.

Business process mapping

In most COTS software systems, there will be gaps between what the organization needs and what the system delivers. For each such gap organizations have to decide whether,

- To remove the requirement and use the system
- Modify the business process
- Extend the solution

If the COTS software system is extended, it is required to fully specify the requirements for those new capabilities just as done for any new product development effort.

Business process mapping is a group activity performed by teams of subject matter experts that gather to draw step-by-step diagrams to document how business is carried out.

It is mostly used by consultants and business professionals to capture the current state of business operations in preparation for business improvement initiatives.

Business process mapping can also be very beneficial in helping to increase productivity among staff, implementing or decommissioning systems, streamlining processes, and protecting knowledge capital.