01 - Explores the systems concept and uses systems analysis and design methodology in developing information system

Explores the systems concept and uses systems analysis and design methodology in developing information system

- What is a System?

```
A collection of inter-related components working together to achieve a common goal
```

There are 3 requirements to be satisfied in order to be a system

1. There should be an input
2. There should be a process
3. There should be an output

If any of these requirements aren't met, it's not considered as a system

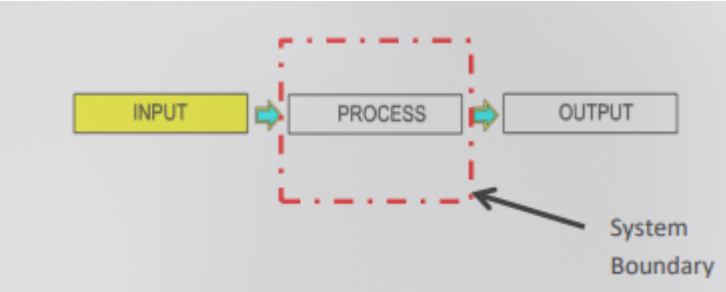
Classification of Systems

There are 3 main groups of systems

- Open and closed systems
- Natural and man-made systems
- Living and non-living systems

Open System

```
A system that interacts freely with its environment, taking inputs and returning output.
```



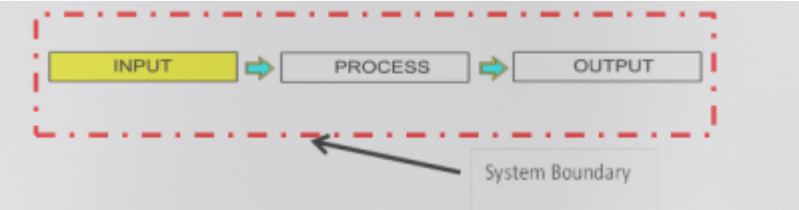
> Living organisms are considered open systems because they take in substances from their environment

Examples

- living beings
- A pond

Closed System

```
Closed system refer to systems having relatively little interaction with other environment or the outside environment.
```



> No system in the real world is ever perfectly closed

Examples

- Human nervous system
- Blood circulatory system

Natural systems

```
Systems which are in the environment made by nature are called natural system
```

Examples

- Animals digestive system

Man-made systems

```
System which is made by man is called man made system
```

Examples

- Transport system
- Education system

Living systems

```
Living systems are open self-organizing living things that interact with their environment
```

Examples

- Human digestive system#

Non-living systems

```
The systems consist of nonliving things are categorized into physical systems
```

Examples

- Solar system

> Information system is a man-made, non-living, open system

02 - Compares and contrasts different types of man-made systems in terms of their objectives and functionality

Compares and contrasts different types of man-made systems in terms of their objectives and functionality

- What is an Information System?

> Information systems are interrelated components working together to collect, process, store, and disseminate information to support decision making, coordination, control, analysis, and visualization in an organization.

A typical organization has 3 sections as,

- Operational - Operational management (L3)
- Middle - Tactical management (L2)
- Upper level - Strategic Management (L1)

Operational level

- The operational level is concerned with **performing day to day business transactions** of the organisation
- Users at this level **make structured decisions**
- Cashiers at a point of sale, bank tellers, nurses in a hospital, customer care staff

Tactical Management Level

- This organisation level is dominated by middle-level managers, heads of departments, supervisors
- oversee the activities of the users at the operational management level.
- Tactical users make **semi-structured decisions**

Strategic Management Level

- This is the most senior level in an organisation
- The users at this level make unstructured decisions
- They use information from tactical managers and external data to guide them when making unstructured decisions.

Different Types of Manmade Information Systems

- Transaction Processing Systems (TPS) - L3
- Management Information Systems (MIS) - L2
- Decisions Support Systems (DSS) - L1
- Office Automation Systems (OAS) - All
- Executive Support Systems (ESS) - L1
- Geographical Information Systems (GIS) - All
- Knowledge Management Systems (KMS) - All
- Content Management Systems (CMS) -
- Enterprise Resource Planning Systems (ERPS)
- Expert Systems
- Smart Systems

Transaction Processing Systems (TPS)

```
- Used to record day to day business transactions of the organization
- to answer routine questions
```

Examples

- Point of Sale Systems
- Payroll system
- Airline booking systems

Management Information Systems (MIS)

```
- used by tactical managers to monitor the organization's current performance
status
- analyzes the input with routine algorithms
        - i.e. aggregate, compare and summarizes the results to produced reports
that tactical managers use to monitor, control and predict future performance.
        - i.e. predict how much of goods or inventory should be ordered for the
second quarter based on the sales of the first quarter.
```

Examples

- Sales management systems

- Budget systems
- HR

Decisions Support Systems (DSS)

```
- used by senior management to make nonroutine decisions
- helps to predict the impact towars a cerain descision
- provide solutions to problems that are unique and change frequently
- uses information from TPS and MIS
- use sophisticated mathematical models, and statistical techniques
        - i.e What would be the impact of employees' performance if we double the production lot at the factory?
```

Examples

- Financial planning systems
- Bank loan management systems

Office Automation Systems (OAS)

```
- execute a variety of office operations, such as word processing, electronic spreadsheet, e-mail, and video conferencing.
- allows to get many tasks accomplished faster
- lets eleminate the need of a large staff
- less storage is needed to store data.
```

Executive Support Systems (ESS)

```
- reporting tool (software) that allows you to turn your organization's data into useful summarized reports
- reports are generally used by executive level managers
```

Geographic Information Systems (GIS)

```
- allow to map, model, query and analyze large quantities of data within a single database according to their location (GIS facilities)
```

Knowledge Management Systems (KMS)

```
- used in an organization to identify, create, represent, distribute and enable adoption of insight and experiences
- can help with staff training and orientation, support better sales
```

Content Management Systems (CMS)

```
- computer applications that support the creation and modification of digital content
```

Enterprise Resource Planning (ERP)

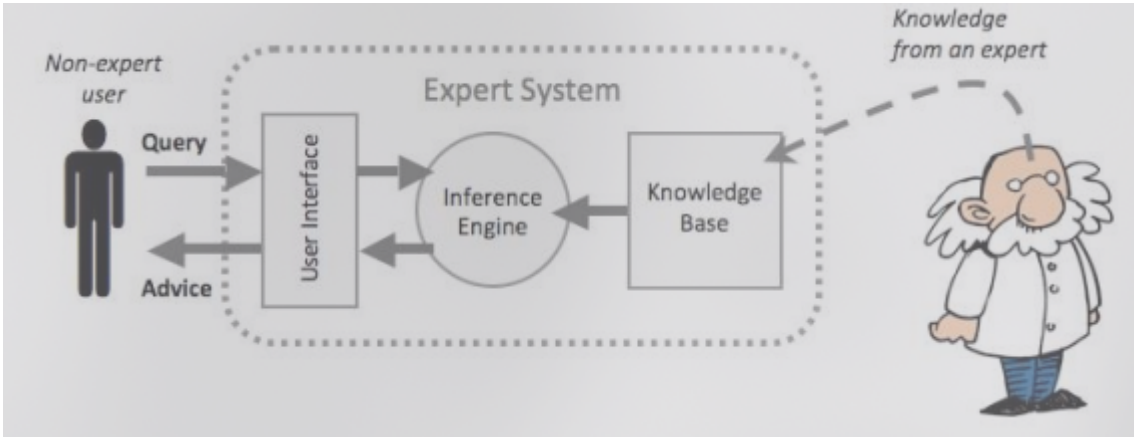
```
- business process management systems that allow organizations to use integrated applications to manage the businesses
- integrates product planning, development, manufacturing, sales and marketing
```

Expert Systems

```
- computer applications that use artificial intelligence
- attempts to act like a human expert on a particular subject area
```

Such a system is made up of three parts:

- A user interface - This is the system that allows a non-expert user to query
- A knowledge base - This is a collection of facts and rules
- A inference engine - This acts rather like a search engine, examining the knowledge base for information



Uses of Expert systems

- Medical diagnosis
- Providing financial advice
- Helping to identify items
- Helping to discover locations to drill for water

Problems related to expert systems

- Can't easily adapt to new circumstances and environment
  - If we need to add something new, the engineer should ask the expert (knowledge base) and add it to the system
- Can be difficult to use (mostly by the non-expert user)

Smart Systems

```
Smart systems are the ones which makes human life more and more easier & convenient.
```

Examples

- Home Automation systems

03 - Explores different information system development models and methods

Explores different information system development models and methods

- What is software engineering?

```
Software engineering is a branch of computer science, defined as a process of
analyzing user requirements and then designing, building, and testing software
application which will satisfy those requirements.
```

- What are the 2 main functions of a software engineer?

```
1. Development of new applications
2. Maintainance
```

- Good Software Engineering Practices Create

```
Successful projects
Business value
Lower stress levels for developers
Happy customers
```

There are 3 main Stakeholders in a software project.

1. **User** (Most important stakeholder) - Users are the ones who use the system after it has been developed to perform their day to day tasks.
2. **Project Sponsor** - this category of the stakeholders is responsible for the financial aspect of the project and ensuring that the project is completed.
3. **Developer** - this category is usually made up of systems analysts and programmers. The system analysts are responsible for collecting the user requirements and writing system requirements.

Systems Development Life Cycle (SDLC)

```
The system development life cycle refers to the processing of Strategy Planning,
Feasibility Study, System Analysis, System Design, Implementation and Maintenance
an information system.
```

Steps in the SDLC

1. Strategy planning

2. Feasibility study
3. System analysis
4. System design
5. Implementation
6. Manitainance

The main objective of system development life cycle

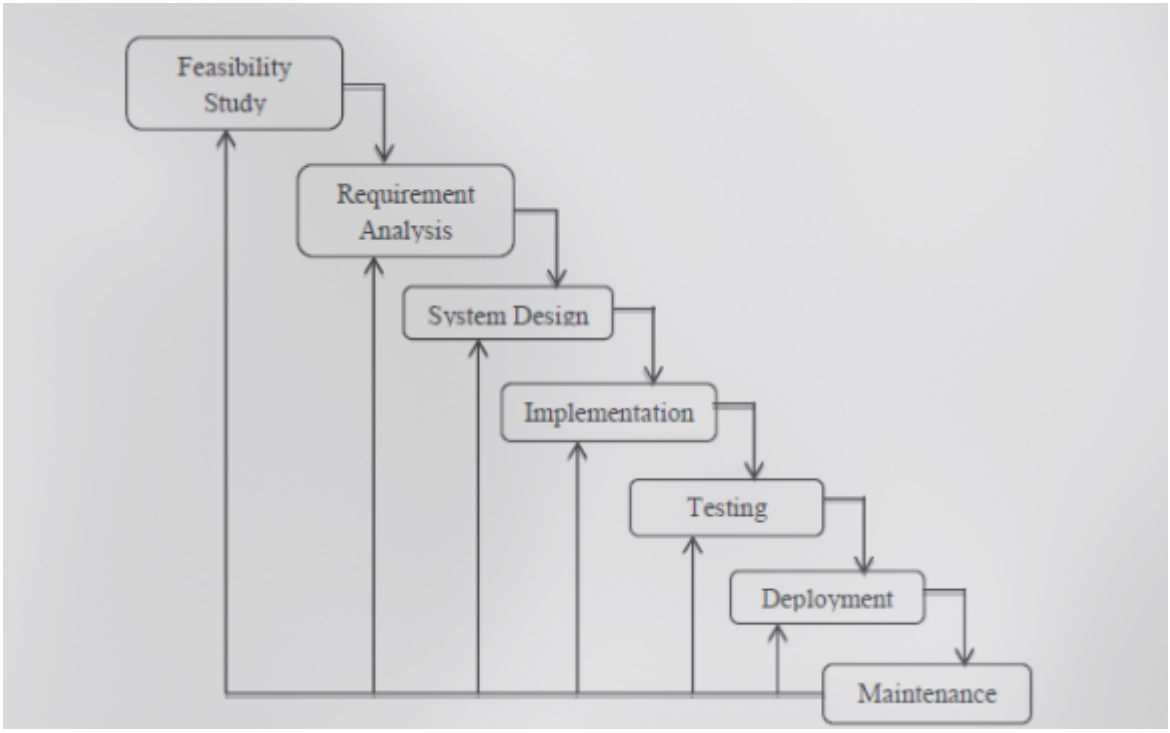
- to produce high quality information systems that meet or exceed the expectations of the users within the agreed budget and time frame.

SDLC uses a number of development models to achieve this objective.

1. Waterfall modal
2. Spiral model
3. Agile model
4. Prototyping model
5. Rapid Application Development (RAD) model

Waterfall Model

- sequential design model
- next stage starts only after the completion of the previous stage
- first stage is usually drawn on the top and the subsequent stages below and to the left bottom



- Feasibility Study
  - determines whether the solution considered to accomplish the requirements is practical and workable in the software. Information such as resource availability, cost estimation for software development, benefits of the software to the organization after it is developed and cost to be incurred on its maintenance are considered during the feasibility study.
- Requirement Gathering and analysis

    - All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- System Design
    - the requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in **specifying hardware and system requirements** and helps in defining the overall system architecture.
- Implementation
    - with inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- Integration and Testing
    - All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- Deployment of system
    - Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- Maintenance
    - There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

> The biggest challenge of the waterfall model is **adoption to change**. It is **not easy to incorporate new user requirements.**

Advantages of Waterfall Model

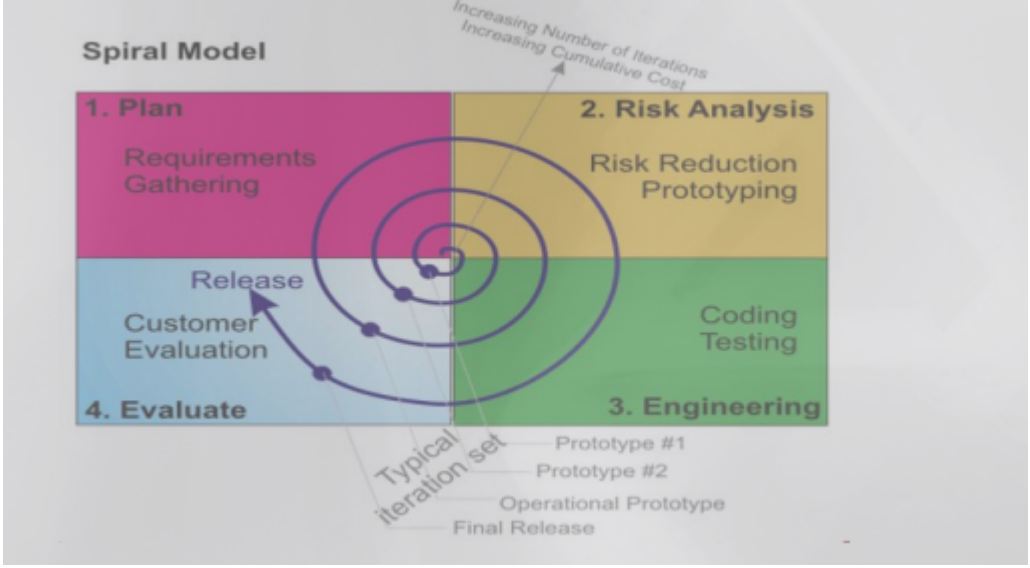
- Simple, easy to use
- Easy to manage
- Clear documentation
- Better to use when there are fixed requirements.

Disadvantages of Waterfall Model

- Difficult to adapt to change
- High amount of risk
- Not good for complex projects
- No working products till the end of SDLC

Spiral Model

- combination of a waterfall model and iterative model
- Each phase **begins with a design goal** and **ends with the client reviewing the progress**.
- first mentioned by **Barry Boehm**
- starts with a small set of requirements
- goes through each development phase for those set of requirements



Advantages of Spiral Model

- High amount risk analysis
- Additional functionalities can be added later

Disadvantages of Spiral Model

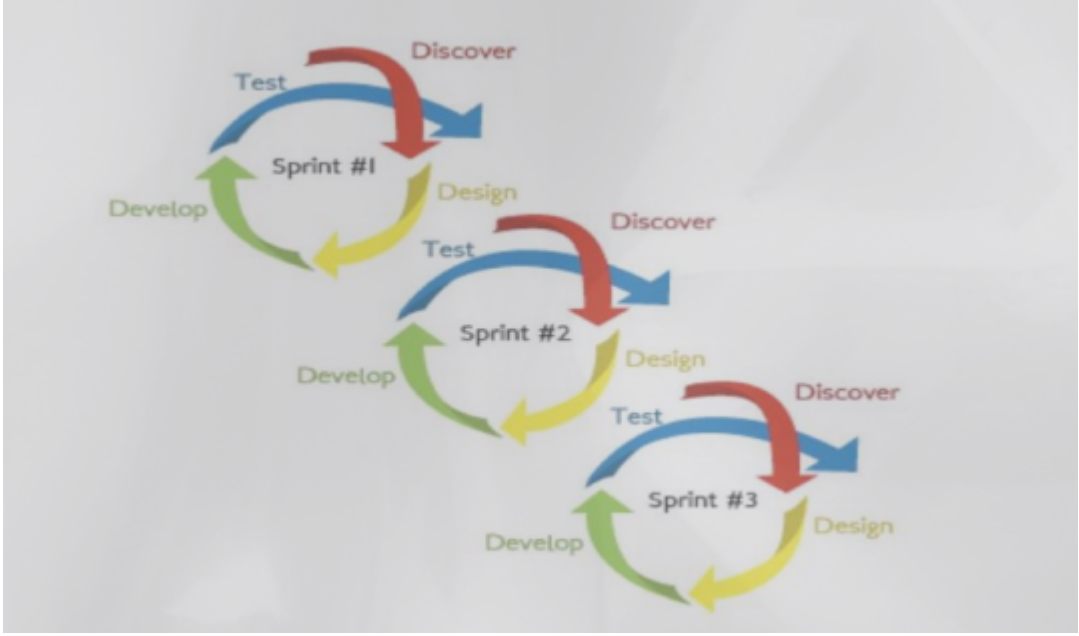
- Can be a costly model
- Risk analysis requires high specific expertise

When to use Spiral Methodology?

- In large projects
- when releases are frequent
- when risk and costs evaluation is important
- for medium to high risk projects
- when requirements are unclear or complex
- when changes may require at any time

Agile Model

- A sprint in agile terms is a well-defined task to be accomplished within a given time
- All stakeholders must meet in person to get the feedback on the sprint before they can move on to the next sprint if any.



Advantages of Agile Model

- Customer satisfaction with rapid delivery
- Continuous attention to technical excellence and good design
- Rapid and flexible responses to change.
- Easy and **fast** addition of requirements and releases.

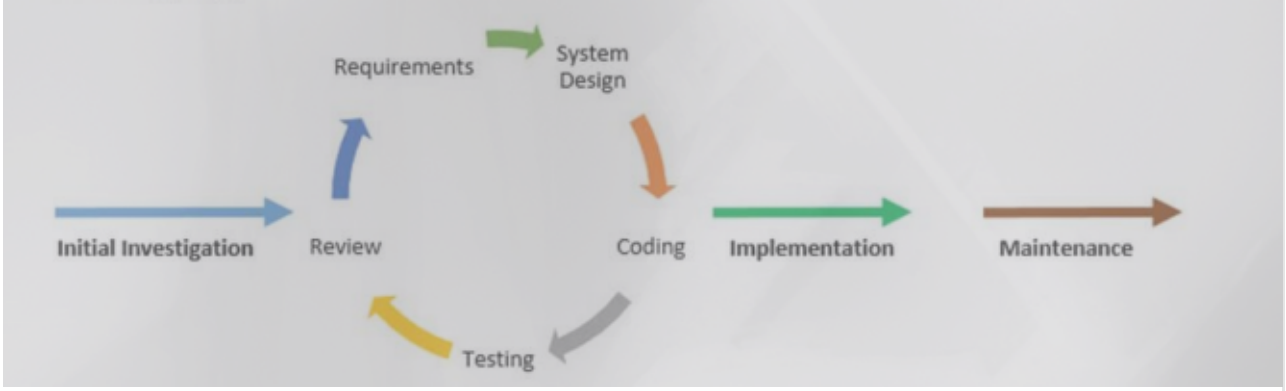
Disadvantages of Agile Model

- Lack of emphasis and documentation
- The projects can easily get take off track as the customer is not clear with what the final outcome they want
- Difficult to assess the effort required at the beginning for some large project.

Prototyping Model

- semi-functional simulation model of the actual system to be developed
- make use of prototypes.
- both developers and users to get feedback early
- makes it easy for users to specify their requirements
- makes it easy for developers understanding the requirements of the users
- 2 main types
  - Evolutionary prototyping - started with clear requirements
  - Throw-away prototyping - started with partial requirements, purpose is to make the requirements clear

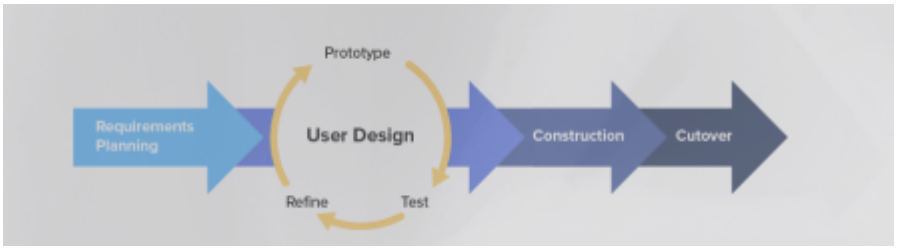
Rapid Application Development (RAD) Model



- an adoption of the waterfall model
- targets at **developing software in a short span of time**
- follow the **iterative**
- emphasises on delivering projects in small pieces
- larger projects are divided into a series of smaller projects
- main features of RAD model are that **it focuses on the reuse of templates, tools, processes, and code.**

When to use RAD Methodology?

- when a project needs to be produced in a short time span
- when the requirements are known
- when the user will be involved all through the life cycle
- when technical risk is less
- when a budget is high enough to afford designers for modelling



Advantages of RAD Model

- Minimal planning
- Fast prototyping
- Developing instead of planning
- Encourage customer feedback
- Quick initial review

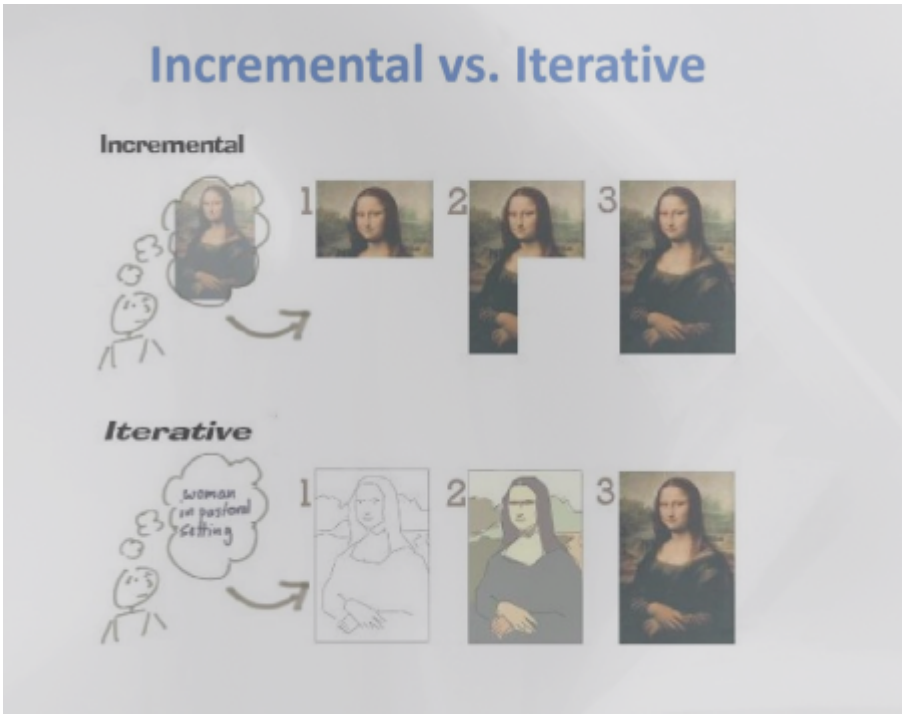
Disadvantages of RAD Model

- Need a big budget to afford expertise in different fields
- High dependency on modelling skills

Model	Advantages	Disadvantages
Waterfall	Easy to use	Difficult to adapt to change
	Better to use with known requirements	High amount of risk

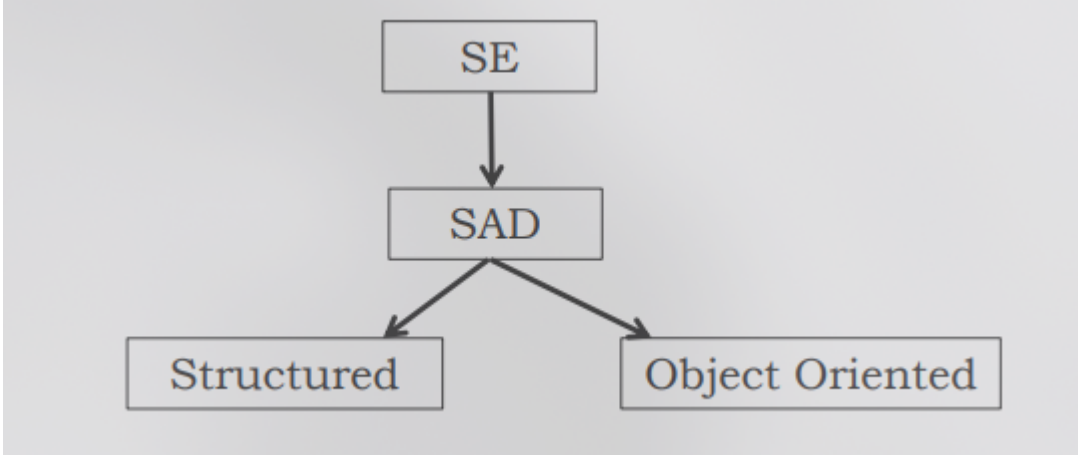
Model	Advantages	Disadvantages
	Clear documentation	No working products till the end of SDLC
Spiral	High amount risk analysis	Risk analysis requires high specific expertise
	Additional functionalities can be added easily	Can be a costly model
Agile	Customer satisfaction with rapid delivery	Lack of emphasis and documentation
	Easy, fast addition of requirements and releases.	Difficult to asses the effort required at the beginning
Prototyping	makes it easy for users to specify their requirements	Takes a longer time to finish
	makes it easy for developers understanding the requirements of the users	User requirements may change often
RAD	Minimal planning and fast prototyping	High dependency on modeling skills
	Encourage customer feedback	Need a big budget to afford expertise in different fields
	Developing the software in a short period of time	

Incremental vs Iterative



04 - Examines the Structured System Analysis and Design Methodology (SSADM)

Examines the Structured System Analysis and Design Methodology (SSADM)



There are many structured methods of SAD.

- SADT
- Structured Design
- Yourdon Structured Method

Why do we need a standard method for SAD?

> Different organizations used these different methods to analysis the system and design the system. Hence the Documentations were different. As a result new developer couldn't understand the pictures and symbols that used in the system. Hence UK government wanted to have one standard method for government development and government asks particular company to develop one standard method.
>
> – SSADM – UK standard!

What is SSADM?

> Structured System Analysis and Design Method (SSADM) is a standard for system analysis and application design.

Properties of SSADM

- Used for systems with stable and known requirements which doesn't change often
- It was developed in early 1980s in the U.K
- Development stage is not a very important stage.
- Requires to complete one stage before starting the next
- Have to do feasibility study, requirement analysis and design again and again (suitable for waterfall model)
- As a result SSADM can apply for the system that system requirements are relatively stable and clear. (i.e Library System)

All the Information Systems has two aspects such as **data** and **process**

Object-oriented methodology (Object-oriented System Analysis and Design -OOSAD)

OOSAD used to design a system as a collection of objects that work interactively

In mid-90's company call **Rational** recruited above 3 people to develop one method. Then Rational released that as **UML** (Unified Modeling Language) and Rational Rose is a tool for that.

- **Grady Booch** proposed Booch Method for OOSAD.
- **James Rambo** proposed Object Modeling Techniques
- **Ivan Jakobsan** proposed OOSE (Object Oriented Software Engineering)

Key difference between SSADM and OOSADM

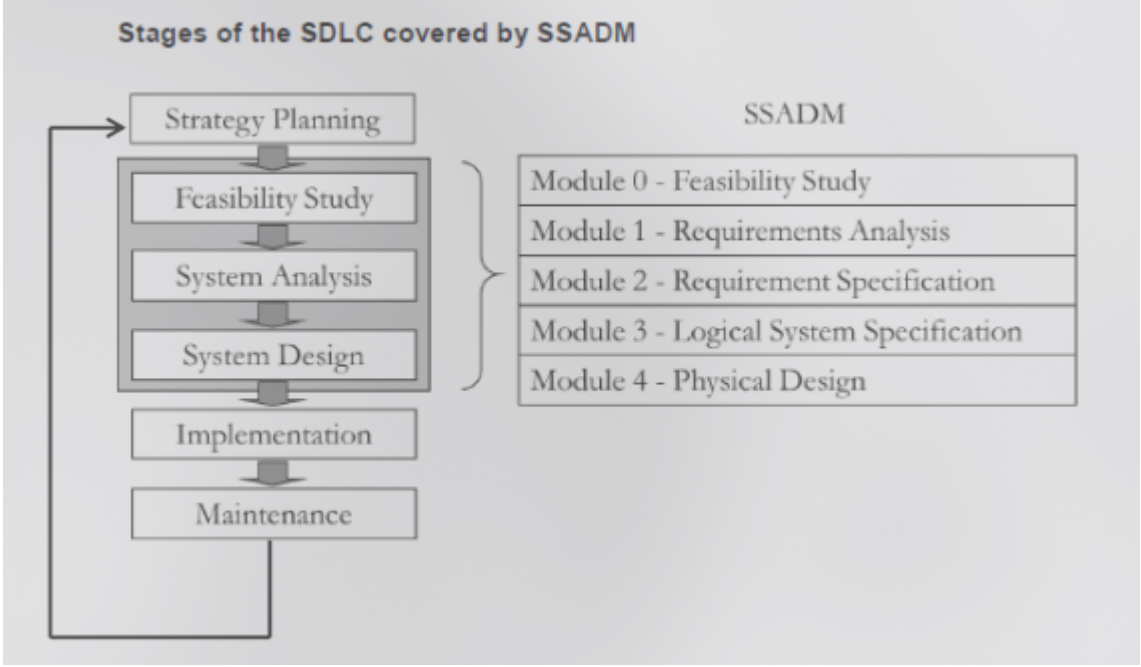
	Structured	Object-Oriented
Methodology	SDLC	Iterative/Incremental
Focus	Processs	Objects
Risk	High	Low
Reuse	Low	High
Maturity	Mature and widespread	Emerging (1997)
Suitable for	Well-defined projects with stable user requirements	Risky large projects with changing user requirements

Properties of OOSADM

- Best for systems with requirements that changes often
- Uses visual modeling to improve communication

What is Software Development Life Cycle (SDLC)?

SDLC is a process used by the software industry to design, develop and test high quality software



Feasibility Study

This is done to study whether the project is feasible or not. This phase can be divided into 4 parts.

- Business Feasibility
- Financial Feasibility - checking whether I have the money to build a project
- Technical Feasibility - checking whether I have the technical knowledge to build a project
- Political and Organizational Feasibility

Can identify the problems at the beginning by doing feasibility study.

Requirements Analysis

We have to identify that what are the requirements should be satisfied for the system to be developed

There are 2 main types of requirements.

- **Functional requirements** - main task that we want our system to do (A teller machine giving us info about money)
- **Non-functional requirement**. - other requirements that support the main functions/tasks (The teller machine having a touch screen)

Requirements Specification

- Once you identify requirements, and then specify them in detail. It is call **requirement specification**
- This is a legal document which should be written in formal English (with shall and all that)

Logical System Specification

- In the sense we design the system to be developed **without considering**

the technical constraints

- Logical design is **independent of technology**

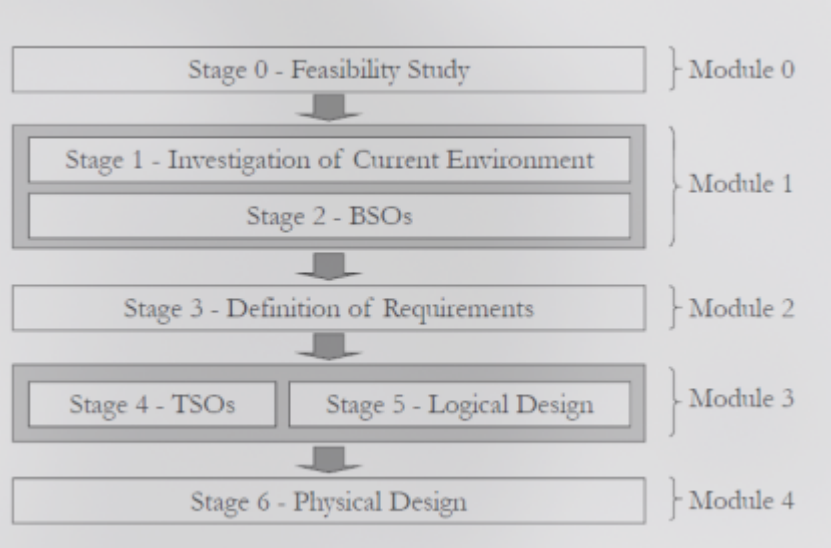
Physical Design

- Once you have the logical design, then we can transform it into a physical design.
- The physical design is **dependent on the technology**.

Framework

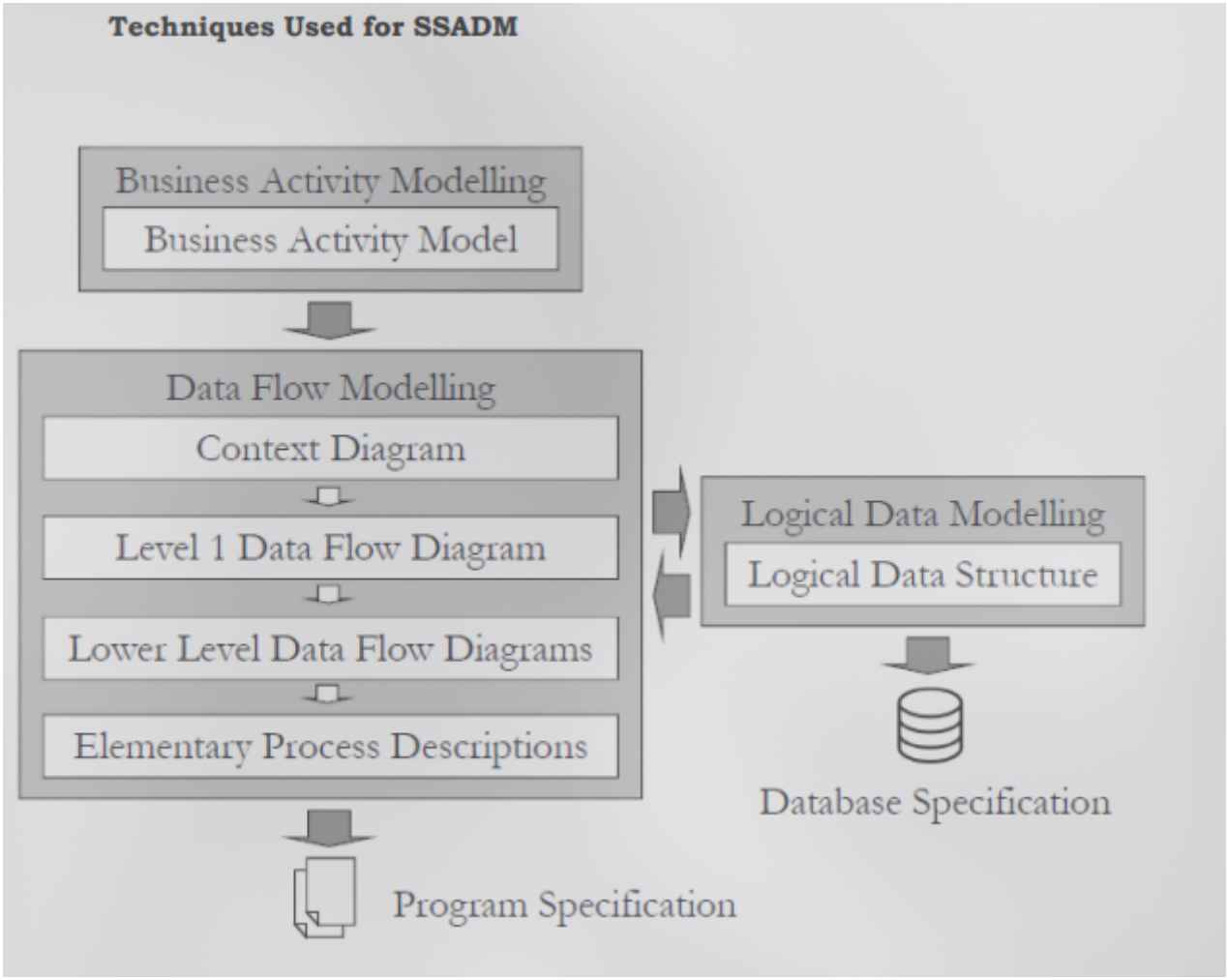
SSADM has a framework/skeleton. We have to do all the analysis and design things within this frame.

In this framework, 5 modules in SSADM are broken into 7 stages.



- **Stage 0 (Feasibility study)**
  - Check whether the project is feasible or not
- **Stage 1 (Investigation of current environment)**
  - Uncover/identify requirements /problems of the current system (manual/computerize)
- **Stage 2 (BSOs - Business System Options)**
  - There are alternative solutions to satisfy those requirements. All solutions satisfy mandatory requirements. **Client has to select one solution** out of those alternative solutions and Analyst can explain the advantages & disadvantages.
- **Stage 3 (Definition of Requirements)**
  - Defining the requirements for the selected solution
- **Stage 4 (TSOS)**
  - Evaluate the technical system options. Find out what are the alternative technical options to transform to the physical design
- **Stage 5 (Logical Design)**
  - Design the future system logically without considering the technical constraints.
- **Stage 6 (Physical Design)**
  - Transform the logical design into the physical design according to the selected technical option.

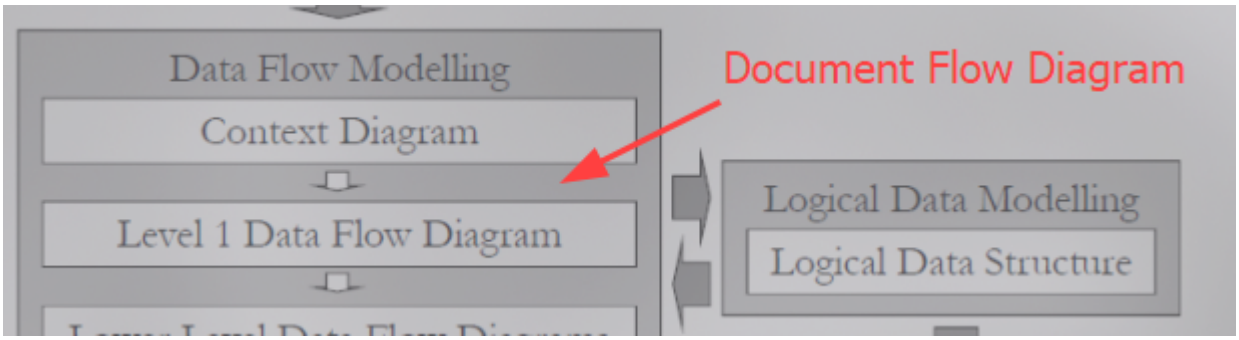
Diagram map



Document flow diagram

Here, there is another diagram called `Document flow digram` between the Context Diagram and the Level 1 DFD.

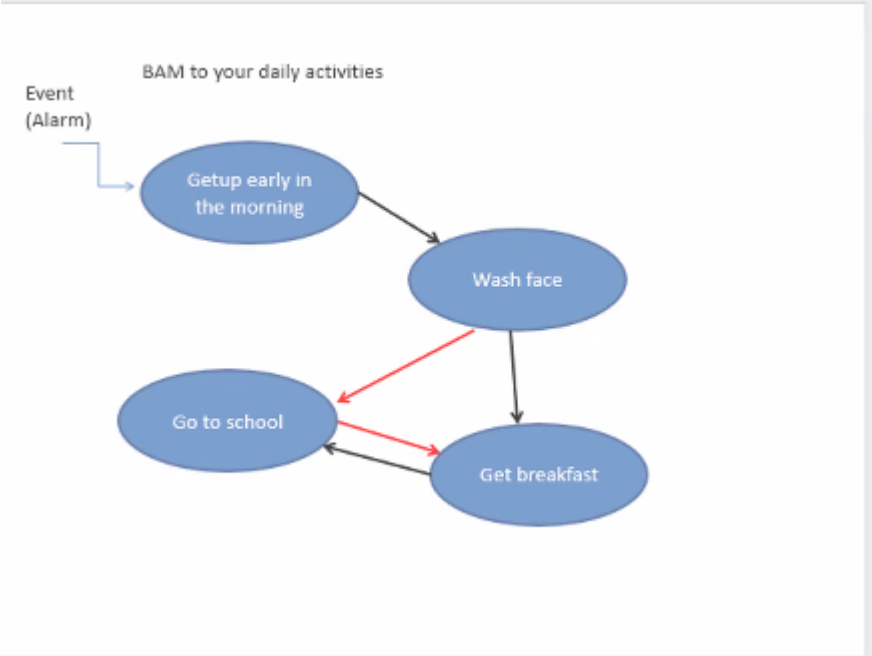
- This show what documents(physical/soft copy) being passed between the system
- This acts as a bright between the context diagram and the level 1 DFD



Drawing Business Activity Models (BAM)

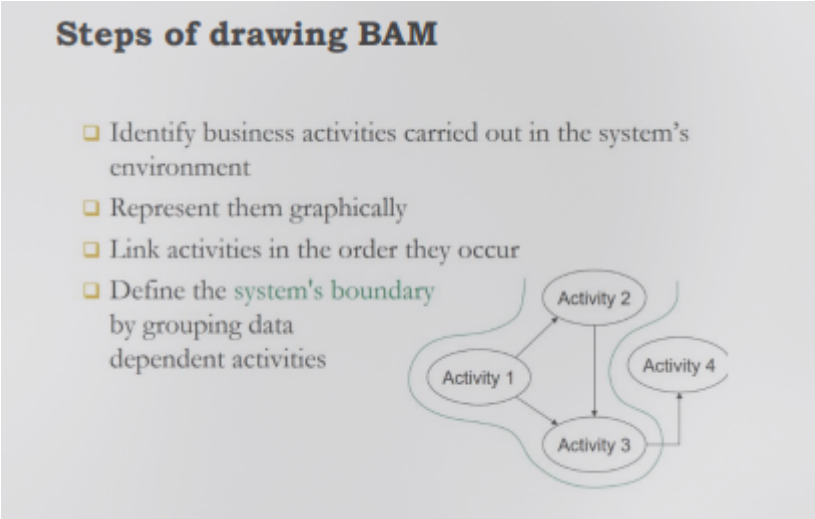
> Diagrmas
>
> - There are diagrams drawn in different levels, like mentioned in 04 - Examines the Structured System Analysis and Design Methodology (SSADM) > Diagram map Therefore, first we are going to take a look at `Business Activity Model` diagrams.

Business Activity Model Diagrams



Important points

- An event should be triggered to start an event
- When drawing the diagrams, actors with direct interactions with the database are included, other actors and their activities aren't included in the diagram.



Example

Bookland is a book store that specializes in selling rare books, which are usually not available from other sources. As a practice, Bookland maintains just a one copy of any book at a time due to the higher price of rare books.

Typically, a customer makes a **book enquiry** over the phone from the **Sales Assistant** at Bookland. She then goes through **book details** in the **Inventory** file to check whether such a book is in their stock. If the details of the book is found in the Inventory file, the Sales Assistant then refers to the **Hold-on Requests** tray to make sure that the **book status** is 'available' and makes a **reply to the enquiry**. If the book status is 'available' and the **customer** wishes to reserve the book, she then takes customer's **personal details** and places a **hold-on request** against that book in the Hold-on Requests tray making the book no longer available.

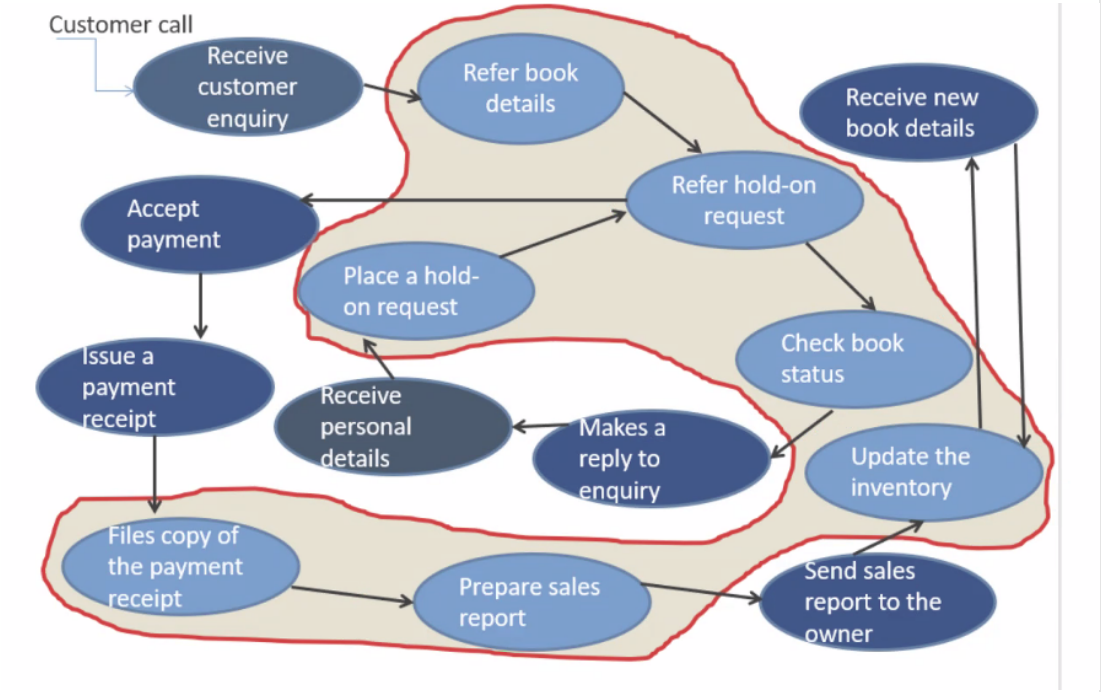
When the customer comes to make the payment and collect the book, the **Cashier** at Bookland refers to the Hold-on Requests tray and finds the relevant hold-on request made by the customer. If there is a valid hold-on request, she then accepts the **payment** from the customer, issues a **payment receipt** to the customer and finalizes the sale. The customer is then allowed to take away the book. The Cashier also files the **copy of the payment receipt** in the **Sales** file, uses it at the end of the day to compile a **sales report** and sends it to the **Owner** of the Bookland. After every sale, the Cashier updates the book details in the Inventory file and keeps the stock up to date. When the Owner supplies books to Bookland, he sends **details of books** to Cashier and the Cashier adds them one by one to the Inventory file.

- Actors and their activities

Actors	Activities
Customer	makes a book enquiry
	makes the payment
	collects the book
Sales Assistant	goes through book details
	Checks for the availability of the book in stock
	refers to hold-on requests
	checks if the book status is available
	makes a reply to enquiry
	takes customers personal details
	places a hold-on request
Cashier	refers to the hold-on request
	finds the relevant hold-on request
	accepts the payment if found a relevant request
	issues a payment receipt
	finalizes the sale
	allows the customer to take the book
	takes a copy of the payment receipt
	compiles a sales report using the payment receipt
	sends the sales receipt to the owner
	updates the book details in the Inventory
	adds the book details to the inventory file
Owner	supplies books to Bookland
	Sends details of books to Cashier

From these activities that these actors do, only the activities that affects the database system is needed to be taken in to consideration when designing the system.

- Cashier making a hold-on request
- Cashier referring to the book inventory
  Because considering other actions would be pointless as they don't affect the system.
- Customer reserving a book (even though this indirectly interacts with the database this is not taken, only direct links are taken in to consideration)
- Owner sending the book details to the Cashier



> Actions in the encirecled part are the only actions which have a direct link with the database. So those are the only actions which should be included in the finalized system.

Like said in 7.4 there are 2 types of functions as functional and non-functional

- **Functional requirements** - main task that we want our system to do (A teller machine giving us info about money)
- **Non-functional requirement**. - other requirements that support the main functions/tasks (The teller machine having a touch screen)

These both types can also be divided as Essential requirements and Nice-to-have requirements.

ATM Machine Requirement Catalog

	Functional Requirements	
ID	Description	
1	~~User~~ shall be able to withdraw money	Essential Requirements
2	~~User~~ shall be able to view balance	
3	~~User~~ should be able to deposit money -	Nice to have Requirement
1	~~System~~ shall be able to dispense money	Essential Requirements
2	~~System~~ shall be able to display balance	
3	~~System~~ should be able to accept money -	Nice to have Requirement

	Non - Functional Requirements
ID	Description
1	~~System~~ should provide a touch sensitive GUI – nice to have
2	~~System~~ shall provide accuracy of 98% on withdrawal
3	~~System~~ shall use 256 bits encryption for communication (security)

Essential Requirements

Not strongly associated with teller functionality
(non- functional requirements)

- Usability
- Reliability
- Efficiency
- Security
- User-friendliness
- Performance

Data Flow Modeling

- Used to model processing of data in the system
- DFM consists of
  - A set of Data Flow Diagrams (DFDs) and
  - Associated textual descriptions
- Defines partitions into sub systems
- Presents different parts of the system at appropriate levels of details for different stakeholders of the project
  - Client
  - End-user
  - Developer

DFD

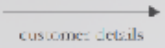
- Illustrates
  - How data is processed within the system
  - How data is passed around the system
  - Where data is stored in the system
- Components
  - External Entities
  - Data Flows
  - Processes
  - Data Stores

External Entity

- 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
- Example

Data Flows

- Show flows of data to, from and within the system
- Link other components in a DFD
- Could be unidirectional or bidirectional
- Represented with solid arrows
- Between two external components are shown by dashed arrows
- Intersections should be avoided
- Example



Context Diagram (Level 0 DFD)

- This is the first most level DFD diagram with the highest level of abstraction
- This **shows how the system interacts with the external entities**
- Provides an over-view of the whole system in a single process.

How to draw a context diagram?

- Identify all the external entities interact with the system
- Identify data flows from the external entities to the system as well as to external entities from the system
- Represent external entities in their graphical notation
- Represent the system as a single process
- Add data flows between external entities and the system

- Also, we can use `dotted lines` to specify and relation between 2 external entities

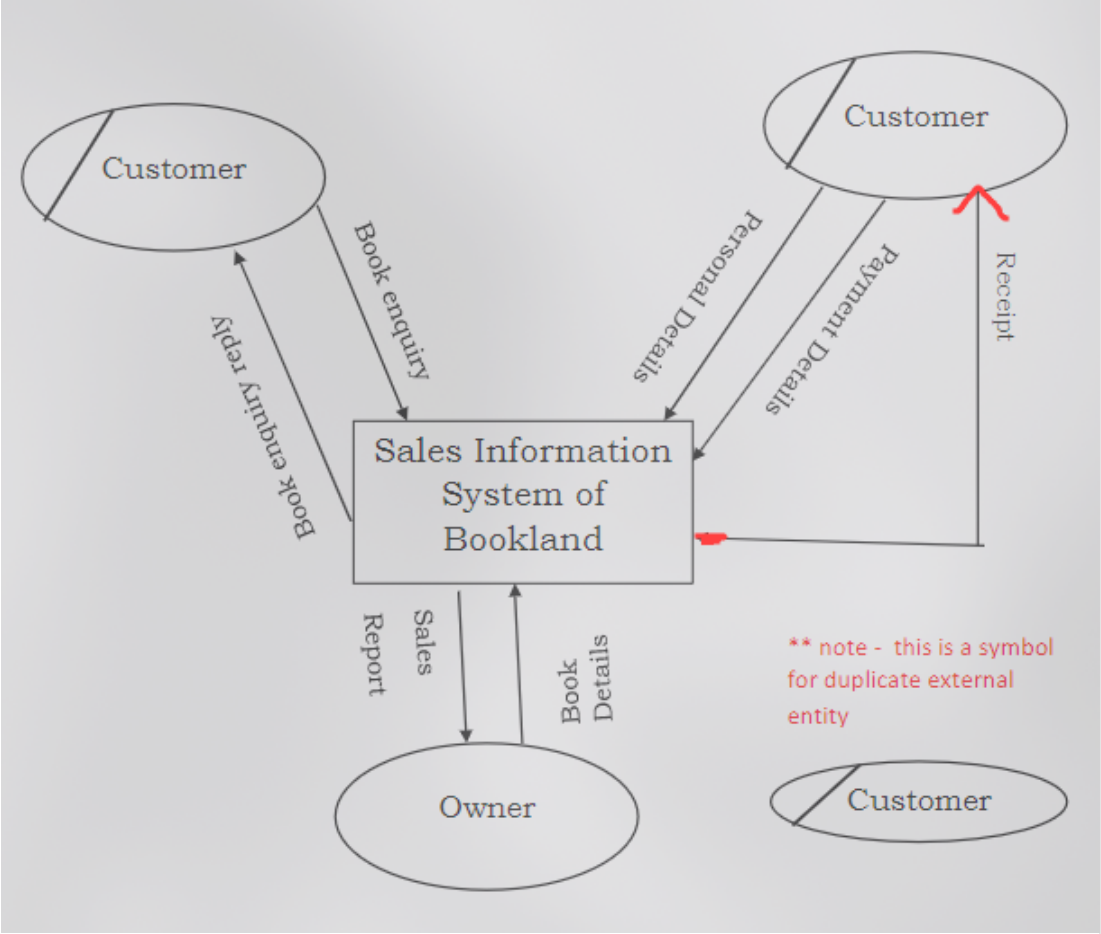
Read the **Bookland Book store** story well and create a table as shown below.

External Entity	Source (S)/ Recipient (R)	Data Flow
Customer	S	Book enquiry
Customer	R	Book enquiry reply
Customer	S	Personal details
Customer	S	Payment details
Customer	R	Receipt
Owner	R	Sales report
Owner	S	Book details

This is a table we create for the easy understanding of what should be included in the context diagram. This table represents,

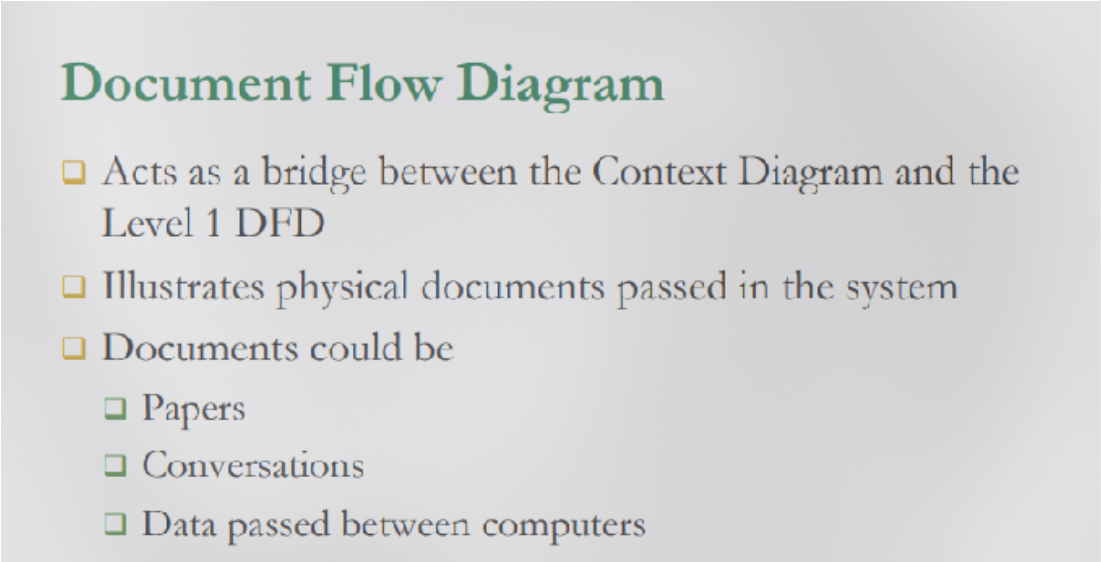
- Who the external entities are
- Who the Source and the Recipient are
- Data which passes by the sender to the source

Once you have the table made, then you can go and draw the diagram as below.



Document Flow Diagram

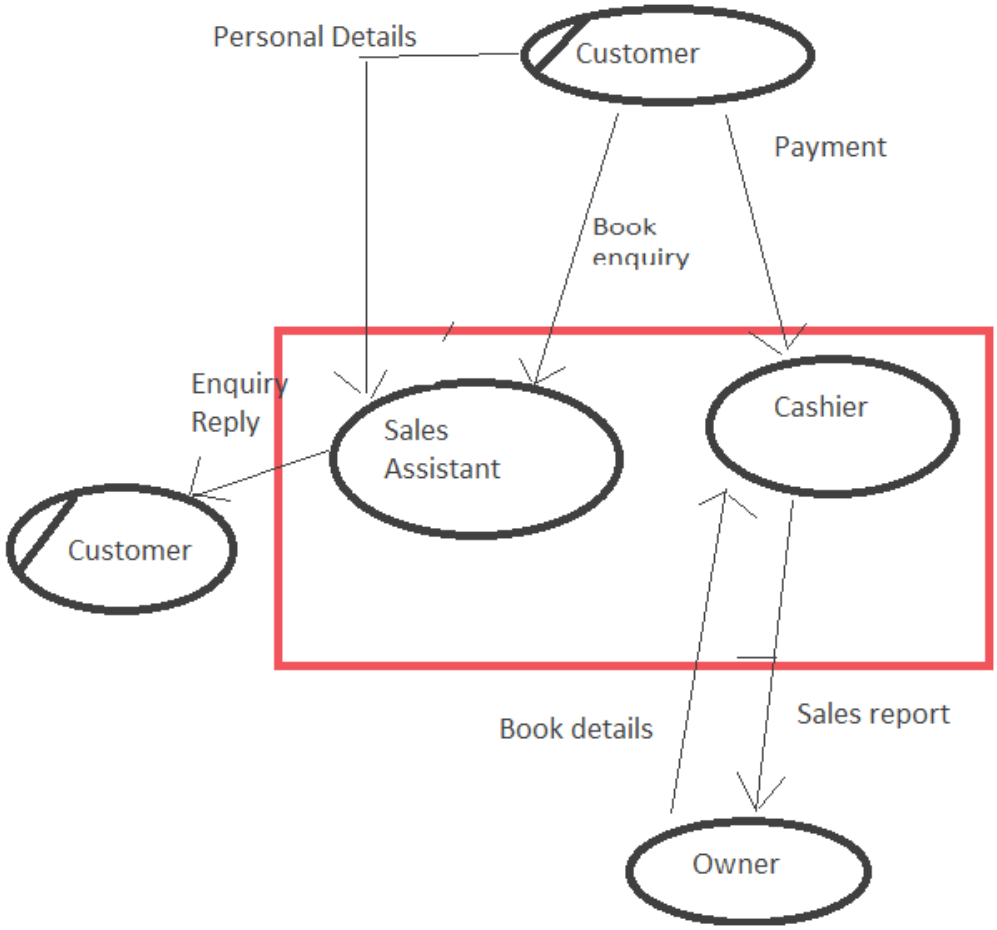
- Part 4



- Document Flow Diagram for the 2017 AL past paper question example

Sender	Document	Recipient
Customer	Call	Sales Assistant
Customer	Book enquiry	Sales Assistant
Customer	Personal details	Sales Assistant

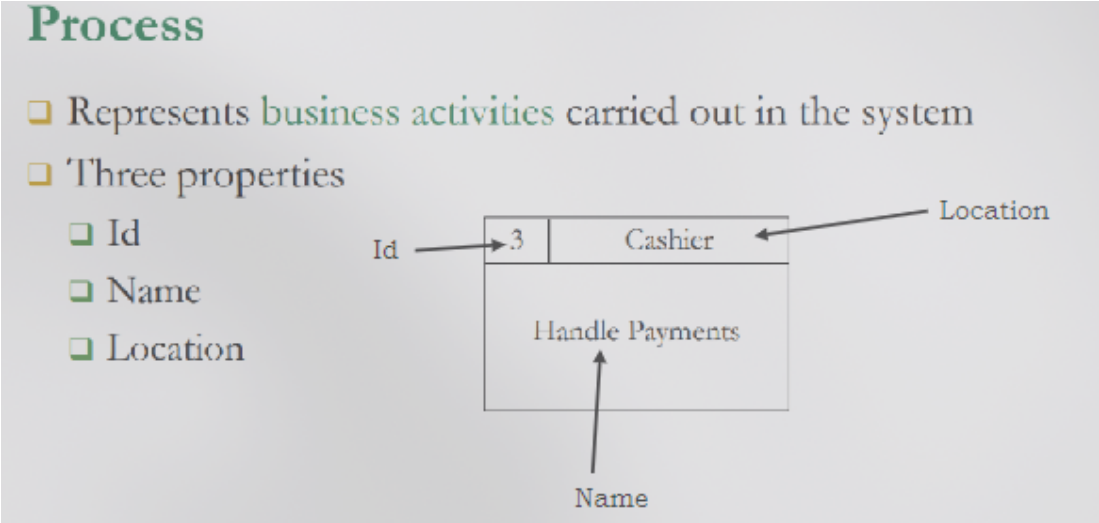
Sender	Document	Recipient
Sales Assistant	Enquiry Reply	Customer
Customer	Payment	Cashier
Cashier	Sales report	Owner
Owner	Book details	Cashier

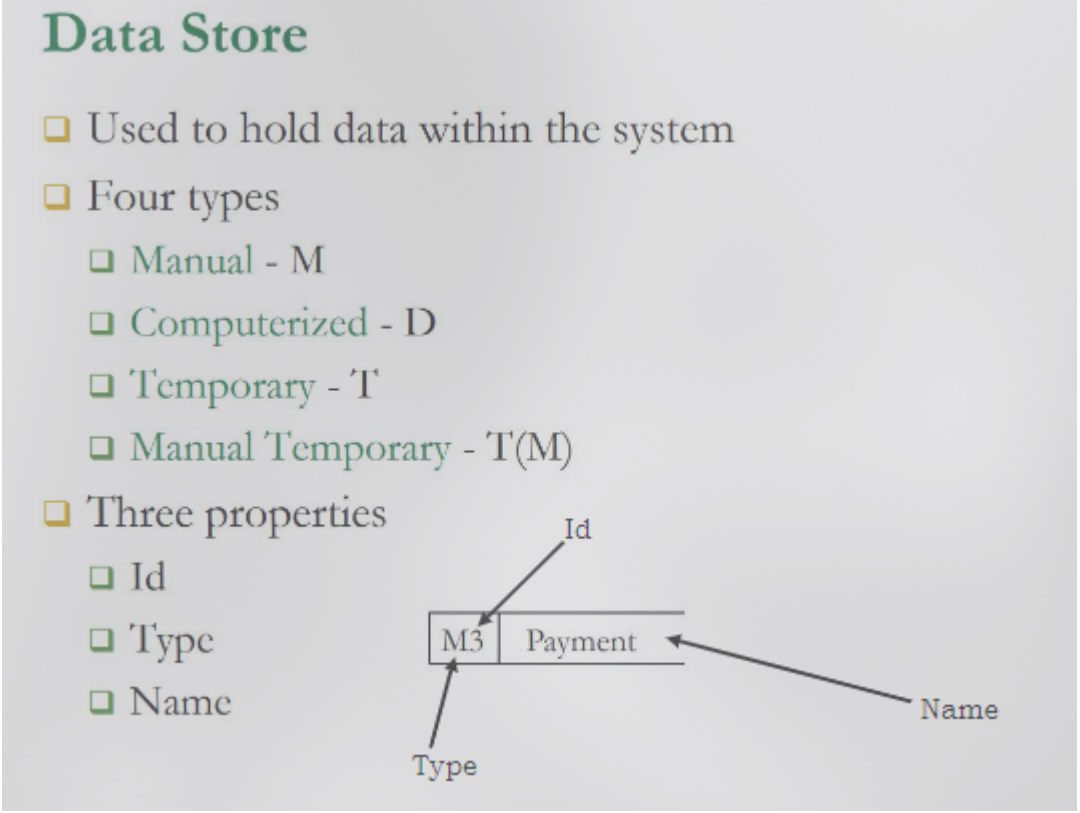


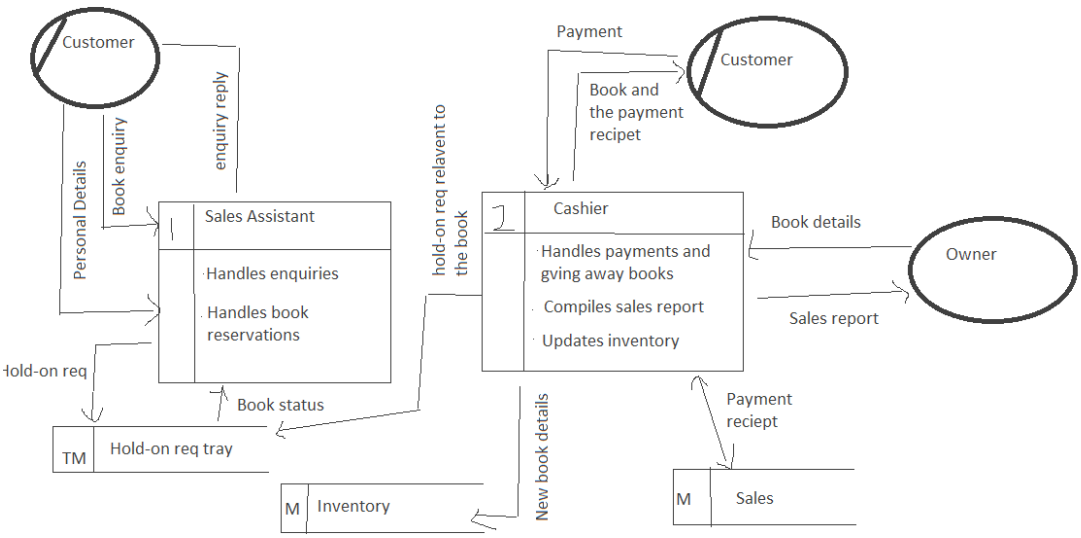
Here we dont include the call as its the triggering event and its only shown in the Business Activity Model Diagram

---

Level 1 DFD





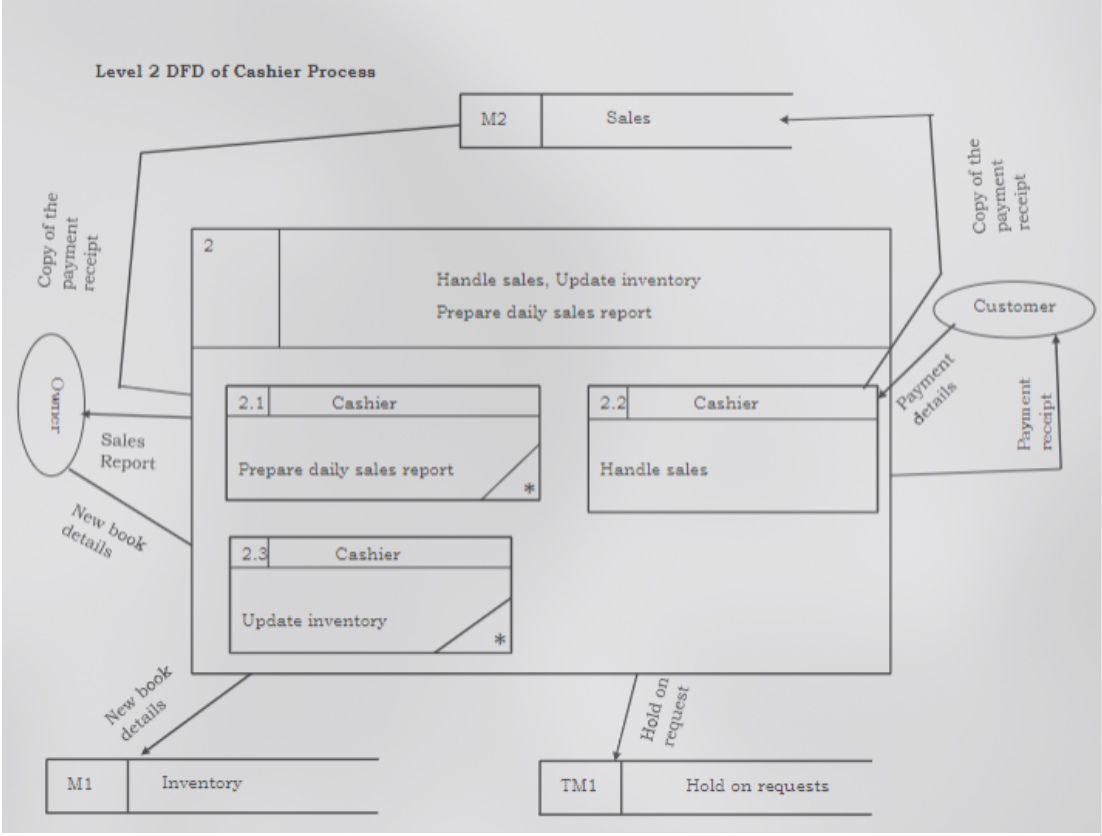


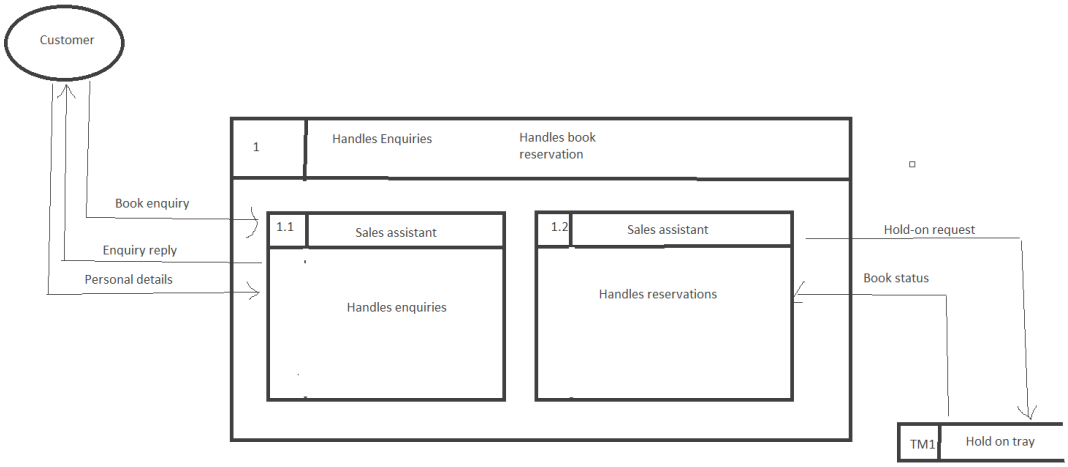
Level 2 DFD

In this level, the same diagram is expanded more. The process block which include the processes done by each and every entity is expanded and rewritten with its relation with the external entities. Some changes happen in the way the block is drawn.

1. Processes of the entity is written in the header of the block
2. The body will contain separate blocks containing the processes written inside

Other things are pretty much the same, all the other relations are shown from arrows.





Elementary Process

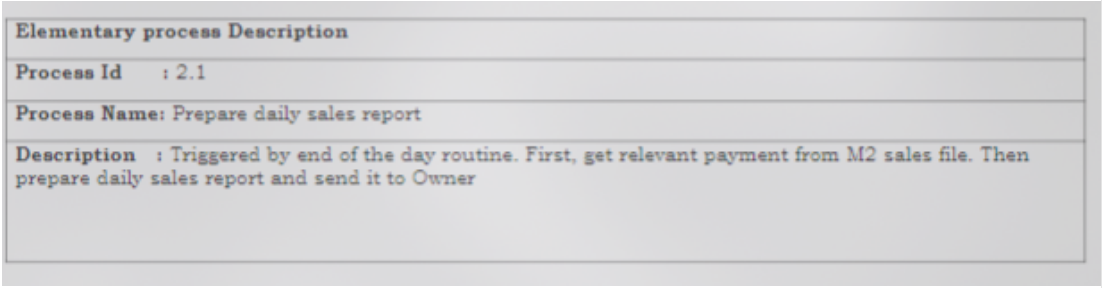
If a process is no longer decomposable, it is called an elementary process. We use the following symbol to denote that in the Level 2 DFDs

Data is required to decide whether a process is decomposable or not, so if not said, it should be kept as a normal process which can be decomposed.

We can describe the elementary process using a table as follows

1. It is written in plain English as a pseudo code.

2. This should contain enough details to write the program specification.



Elementary process Description
Process Id : 2.1
Process Name: Prepare daily sales report
Description : Triggered by end of the day routine. First, get relevant payment from M2 sales file. Then prepare daily sales report and send it to Owner

05 - Investigates the need for a new information system and its feasibility

Investigates the need for a new information system and its feasibility

Preliminary Investigation

Preliminary investigation is done in two phases namely,

1. Problem definition - a preliminary survey of the system is carried out to identify the scope of the system.
2. Feasibility study - the proposed system is evaluated for its feasibility. Feasibility of a system means whether the development of a new or improved system is practical and beneficial.

What is Feasibility Study?

> Feasibility is defined as the practical extent to which a project can be performed successfully.

The objective of the feasibility study

- To analyze whether the software will meet organizational requirements.
- To determine whether the software can be implemented using the current technology
- To determine whether the software can be integrated with other existing software

Types of Feasibility

- **Technical feasibility** - this evaluates whether the developers have ability to construct the proposed system. Check whether the developers have the enough technical knowledge to build the system
- **Economical feasibility** - this studies cost and benefits to evaluate whether the benefits justify the investments in the system development
- **Operational feasibility** - this assesses the willingness and ability of the users to support and use the proposed system
- **Organizational feasibility** - this determines the extent to which the proposed system supports the objectives of the organization's strategy.

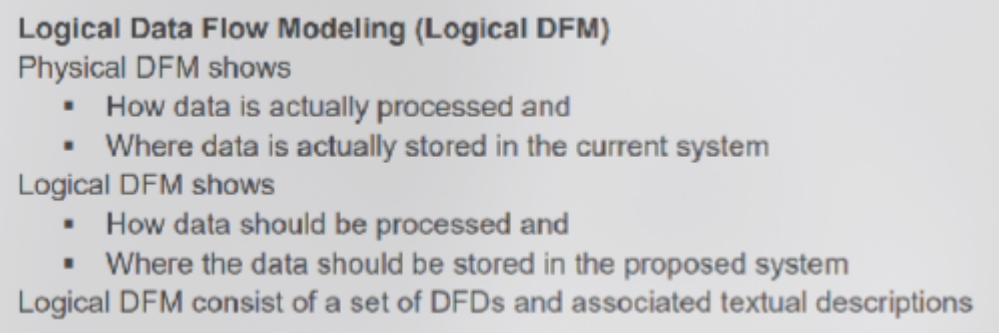
Check whether will this system help the organization as a subset.

06 - Physical and Logical Data Flow Modeling

Physical and Logical Data Flow Modeling

There are 2 main types of Data Flow Models

1. Physical DFM - The current system
2. Logical DFM - The proposed system to be made



> DFD diagrams can be draw to both of these DFMs

Testing

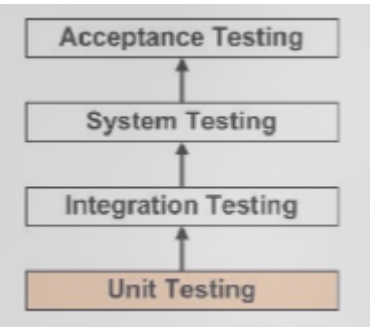
There are 2 types of main testing methods

1. White box
2. Black box

White Box	Black Box
Source code of the application is given	Source code is kept hidden
Done in early stages of testing	Uses to test whether the required functionalities are working as they should
Applied for relatively small unit	Test cases are taken from the program specification

Also, the testing phase can be divided into 4 parts

1. Unit testing
2. Integration testing
3. System testing
4. Acceptance testing

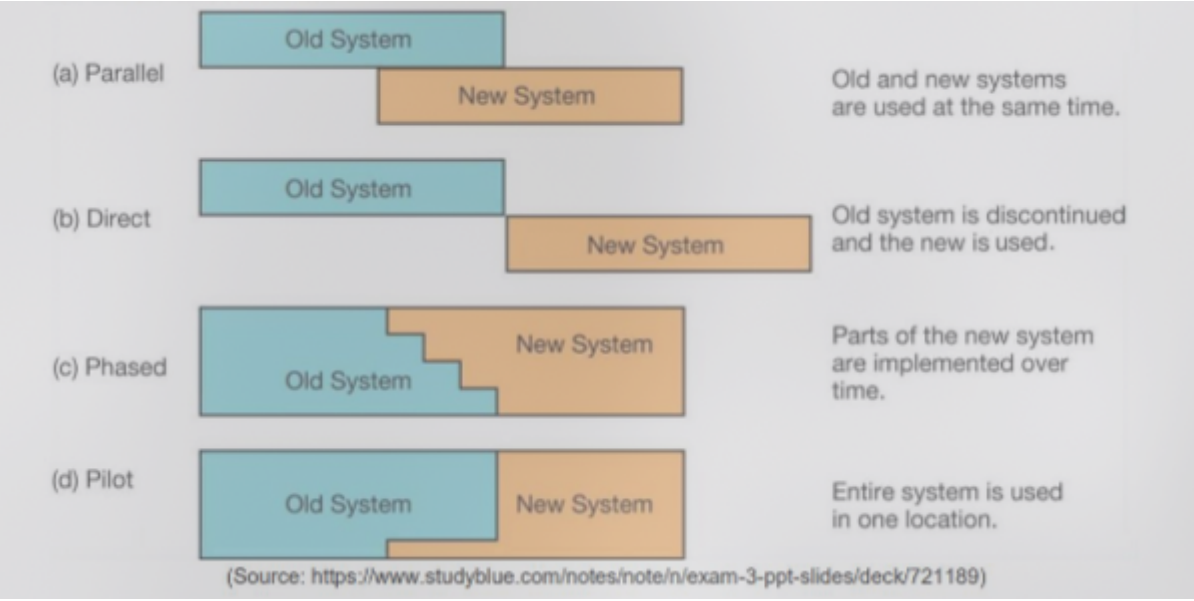


1. **Unit Testing**
    - testing of individual units of the system
    - done by **programmers**
    - **white box testing** is used
2. **Integration Testing**
    - continues to test the units when they are integrated together
    - done by **Integration testers**
    - either **white box or black box** can be used
3. **System Testing**
    - testing the whole system's functionality
    - done by programmers (not the ones who made the system)
    - **black box** testing is used
4. **Acceptance Testing**
    - testing whether the system is acceptable by the users
    - done by programmers or users who developed the system
    - **black box** testing is used

Deployment

There are 4 types of deployments

1. Parallel deployment
2. Direct deployment
3. Phased deployment
4. Pilot deployment



1. **Parallel Deployment**
    - duplication of effort can be seen (ie data must be entered in both systems)
    - time-consuming
    - most bulletproof method
2. **Direct Deployment**
    - smallest method of deployment
    - implementation is faster
    - least expensive deployment

    - can be hard on users as they have to learn a whole new system from scratch
    - if 2 systems are incompatible, this deployment should be used
3. **Pilot Deployment**
    - Can get valuable feedback on the system while using the other system
    - Easy on users
    - implementation risk is high (what if the system fails, the whole location gets shut down)
4. **Phased Deployment**
    - No implementation risk as the whole system isn't being replaced
    - Information learned from the previous stages can be used to make the other better.
    - Can be confusing for users to have some using the older system and some using the new system
    - This can lead to quality issues.