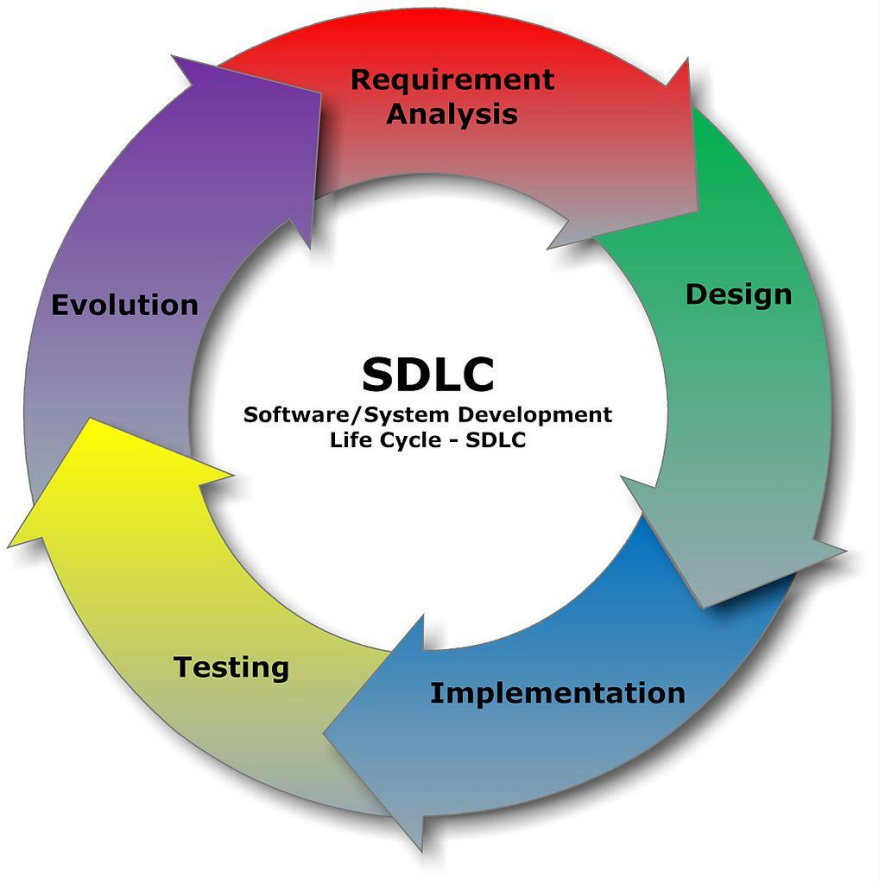
Understanding UML Diagrams

Presented by - Tahmina Khatoon

Objective

- Use Case
- Process Flow Diagram
- Data Flow Diagram
- Activity Diagram

Software/System Development Life Cycle (SDLC)



Requirement Analysis

1. Feasibility Study
2. Requirement Gathering
3. Software Requirement Specification
    a. Functional Requirement
    b. Nonfunctional Requirement
4. Software Requirement Validation

Use Case

Use Case

- Use case
- Use case diagram
- Use case table

Use Case

A use case depicts **a set of activities** performed to produce some output result. Each use case describes how an external user triggers an event to which the system must respond.

Elements of a Use Case

A use case diagram depicts two types of elements:

- **Actor** - representing the business roles
- **Use Case** - representing the business processes
- **System Boundary -** defines the scope of the system

Actor

An actor portrays any entity (or entities) that perform certain roles in a given system. The different roles the actor represents are actual business roles of the users in a system.

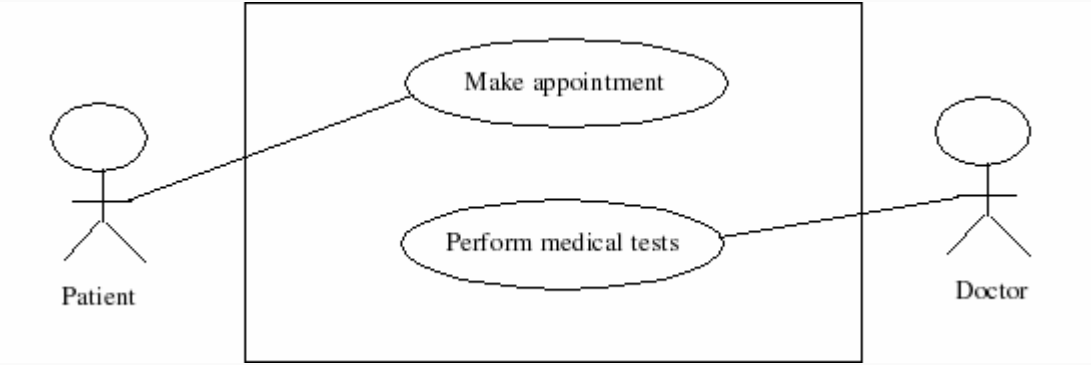
Use case

A use case in a use case diagram is a visual representation of a distinct business functionality in a system.

- To choose a business process as a likely candidate for modeling as use case, ensure that the business process is discrete in nature.
- As the first step in identifying use case, list the discrete business functions in requirements document.
- Each of these business function can be classified as a potential use case.

System Boundary

A system boundary defines the scope of what a system will be. A system can not have infinite functionality. Thus, a use case need to have definite limits. A system boundary of a use case diagram defines the limits of the system. The system boundary is shown as a rectangle spanning all use cases in the system.



Relationships in Use Case Diagram

- Communicates
- Extends
- Include or uses
- Generalization

Communicates

The participation of an actor in a use case is shown by connecting the actor symbol to use case symbol by a solid path. The actor is said to 'communicates' with the use case. This is only relation between an actor and use cases.

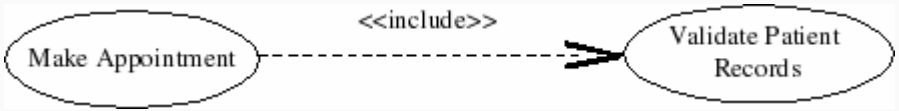
Extends

An extends shows the relationships between use cases. Relationship between use case A and use case B indicates that an instance of use case B may include (subject to specified in the extension) the behavior specified by A. An 'extends' relationship between use cases is depicted with a directed arrow having a dotted shaft. The tip of arrowhead points to the parent use case and the child use case is connected at the base of the arrow.

Include or uses

When a use case is depicted as using functionality of another functionality of another use case, this relationship between the use cases is named as an include or uses relationship.

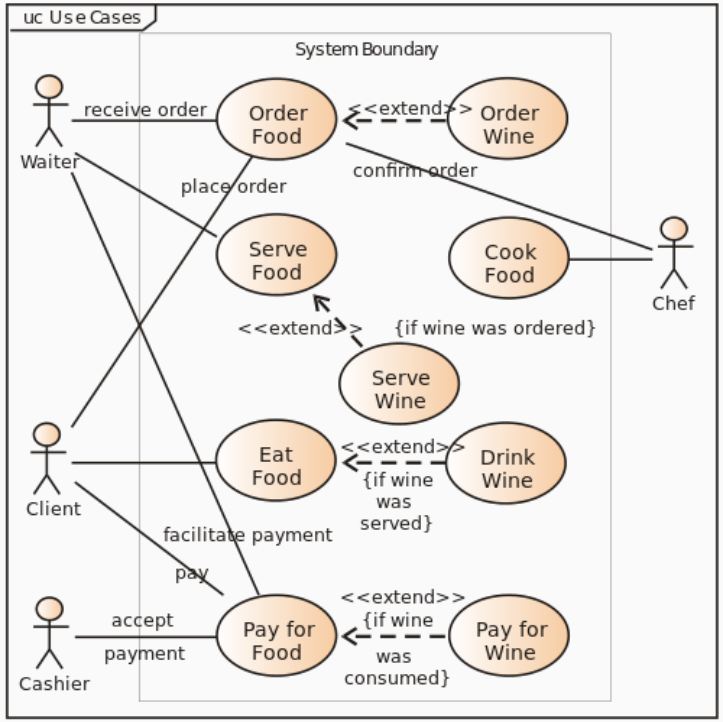
In other words, in an include relationship, a use case includes the functionality described in the another use case as a part of its business process flow.



Generalization

A generalization relationship is also a parent-child relationship between use cases. The child use case in the generalization relationship has the underlying business process meaning, but is an enhancement of the parent use case.

Example of use case diagram

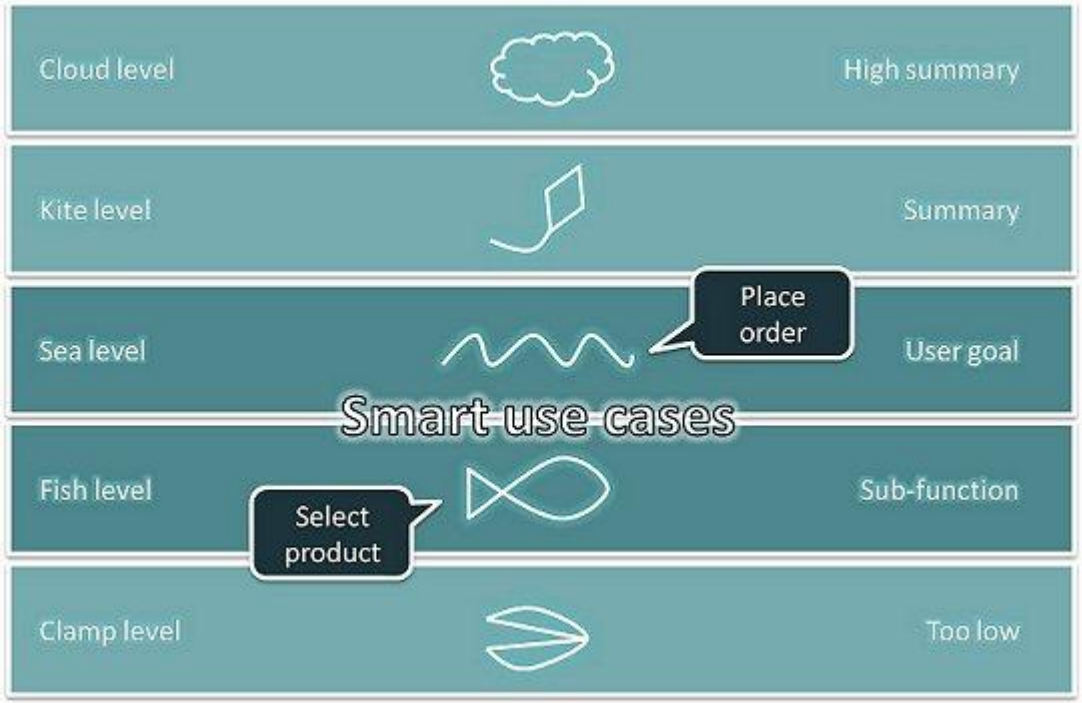


Types of Use Cases

There are three types of use cases:

- Essential
- Concrete
- Abstract

Different levels of use cases



Use case table

Use Case ID:	Enter a unique numeric identifier for the Use Case. e.g. UC-1.2.1		
Use Case Name:	Enter a short name for the Use Case using an active verb phrase. e.g. Withdraw Cash		
Use Case Level	Enter use case level name		
Created By:		**Last Updated By:**	
Date Created:		**Last Revision Date:**	
Actors:	[An actor is a person or other entity external to the software system being specified who interacts with the system and performs use cases to accomplish tasks. Different actors often correspond to different user classes, or roles, identified from the customer community that will use the product. Name the actor that will be initiating this use case (primary) and any other actors who will participate in completing the use case (secondary).]		

Use case table (continue)

Use Case ID:	Enter a unique numeric identifier for the Use Case. e.g. UC-1.2.1		
Use Case Name:	Enter a short name for the Use Case using an active verb phrase. e.g. Withdraw Cash		
Use Case Level	Enter use case level name		
Created By:		**Last Updated By:**	
Date Created:		**Last Revision Date:**	
Actors:	[An actor is a person or other entity external to the software system being specified who interacts with the system and performs use cases to accomplish tasks. Different actors often correspond to different user classes, or roles, identified from the customer community that will use the product. Name the actor that will be initiating this use case (primary) and any other actors who will participate in completing the use case (secondary).]		

Use case table (continue)

Description:	[Provide a brief description of the reason for and outcome of this use case.]
Trigger:	[Identify the event that initiates the use case. This could be an external business event or system event that causes the use case to begin, or it could be the first step in the normal flow.]
Preconditions:	[List any activities that must take place, or any conditions that must be true, before the use case can be started. Number each pre-condition. e.g. 1. Customer has active deposit account with ATM privileges 2. Customer has an activated ATM card.]
Postconditions:	[Describe the state of the system at the conclusion of the use case execution. Should include both *minimal guarantees* (what must happen even if the actor's goal is not achieved) and the *success guarantees* (what happens when the actor's goal is achieved. Number each post-condition. e.g. 1. Customer receives cash 2. Customer account balance is reduced by the amount of the withdrawal and transaction fees]

Use case table (continue)

Normal Flow:	[Provide a detailed description of the user actions and system responses that will take place during execution of the use case under **normal, expected** conditions. This dialog sequence will ultimately lead to accomplishing the goal stated in the use case name and description. 1. Customer inserts ATM card 2. Customer enters PIN 3. System prompts customer to enter language performance English or Spanish 4. System validates if customer is in the bank network 5. System prompts user to select transaction type 6. Customer selects Withdrawal From Checking 7. System prompts user to enter withdrawal amount 8. … 9. System ejects ATM card]

Use case table (continue)

Alternative Flows: **[Alternative Flow 1 – Not in Network]**	[Document **legitimate** branches from the main flow to handle special conditions (also known as extensions). For each alternative flow reference the branching step number of the normal flow and the condition which must be true in order for this extension to be executed. e.g. Alternative flows in the *Withdraw Cash* transaction: 4a. In step 4 of the normal flow, if the customer is not in the bank network 1. System will prompt customer to accept network fee 2. Customer accepts 3. Use Case resumes on step 5 4b. In step 4 of the normal flow, if the customer is not in the bank network 1. System will prompt customer to accept network fee 2. Customer declines 3. Transaction is terminated 4. Use Case resumes on step 9 of normal flow Note: Insert a new row for each distinctive alternative flow.]

Use case table (continue)

Exceptions:	[Describe any anticipated **error conditions** that could occur during execution of the use case, and define how the system is to respond to those conditions. e.g. Exceptions to the Withdraw Case transaction 2a. In step 2 of the normal flow, if the customer enters and invalid PIN 1. Transaction is disapproved 2. Message to customer to re-enter PIN 3. Customer enters correct PIN 4. Use Case resumes on step 3 of normal flow]
Includes:	[List any other use cases that are included (“called”) by this use case. Common functionality that appears in multiple use cases can be split out into a separate use case that is included by the ones that need that common functionality. e.g. steps 1-4 in the normal flow would be required for all types of ATM transactions- a Use Case could be written for these steps and “included” in all ATM Use Cases.]
Frequency of Use:	[How often will this Use Case be executed. This information is primarily useful for designers. e.g. enter values such as 50 per hour, 200 per day, once a week, once a year, on demand etc.]

Use case table (continue)

Special Requirements:	[Identify any additional requirements, such as nonfunctional requirements, for the use case that may need to be addressed during design or implementation. These may include performance requirements or other quality attributes.]
Assumptions:	[List any assumptions that were made in the analysis that led to accepting this use case into the product description and writing the use case description. e.g. For the *Withdraw Cash* Use Case, an assumption could be: The Bank Customer understands either English or Spanish language.]
Notes and Issues:	[List any additional comments about this use case or any remaining open issues or TBDs (To Be Determined) that must be resolved. e.g. 1. What is the maximum size of the PIN that a use can have?]

Tips For Writing Good Use Case Names

- Good Use Case Names **Reflect User Goals**
- Good Use Case Names are **As Short As Possible**
- Good Use Case Names Use **Meaningful Verbs**
- Good Use Case Names Use **An Active Voice**
- Good Use Case Names Use **The Present Tense**
- Good Use Case Names **Don't Identify The Actor**
- Good Use Case Names Are **Consistent**

http://tynerblain.com/blog/2007/01/22/how-to-write-good-use-case-names/

Process Flow Diagram

Process Flow Diagram (PFD)

Process flow diagram

or

System flow diagram

or

Process flow chart

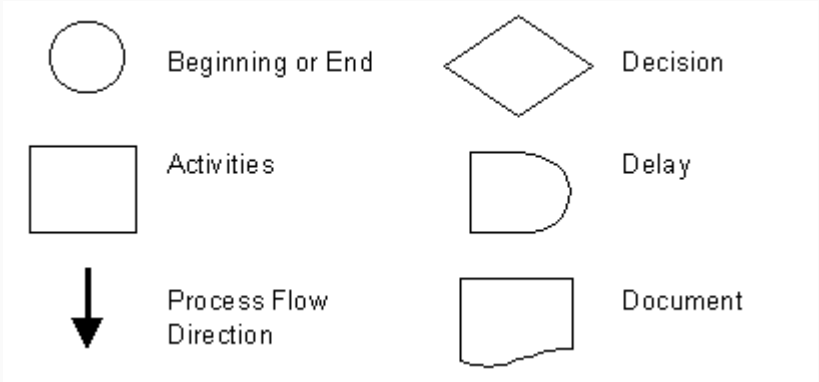
Process Flow Diagram

The Process Flow chart provides a visual representation of the steps in a process.

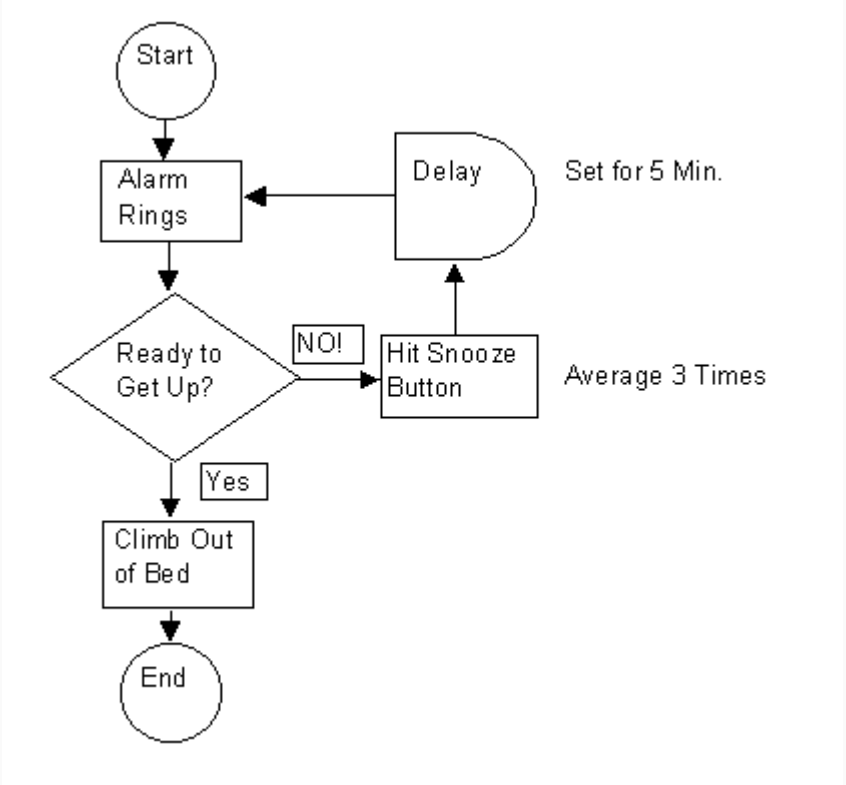
Why should we use PFD

- Gives everyone a clear understanding of the process
- Helps to identify non-value-added operations
- Facilitates teamwork and communication
- Keeps everyone on the same page

PFD notations



Example of PFD

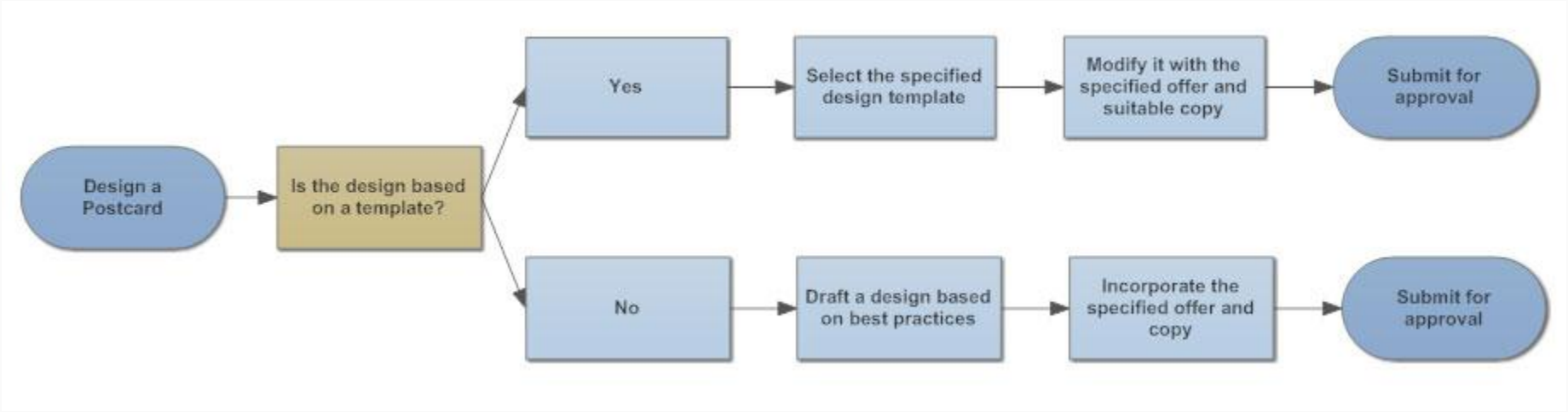


Tips of better PFD

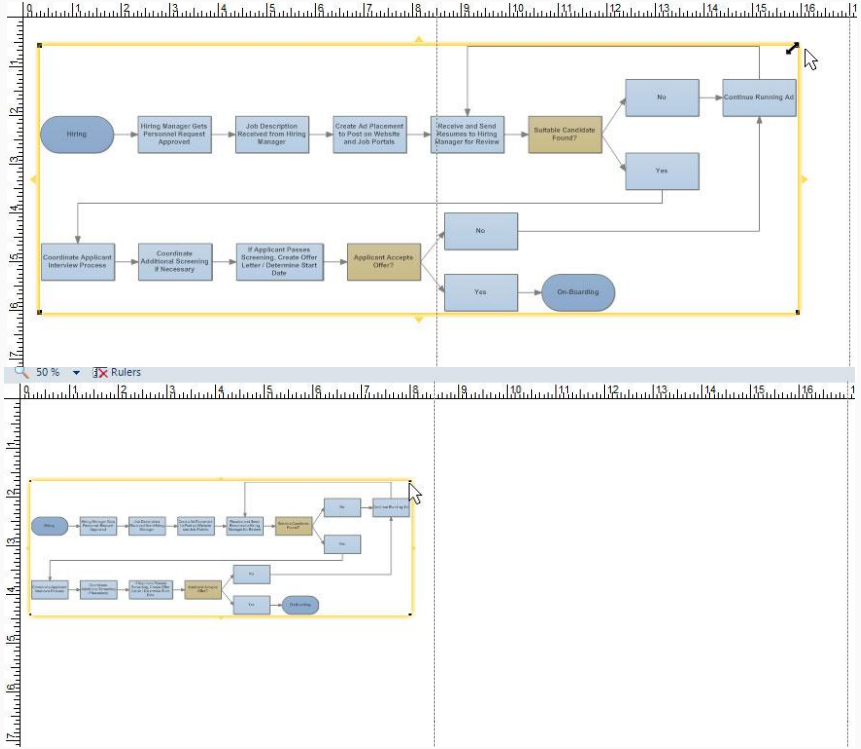
- Use Consistent Design Elements
- Keep Everything on One Page
- Flow Data from Left to Right
- Place Return Lines Under the Flow Diagram

Use Consistent Design Elements

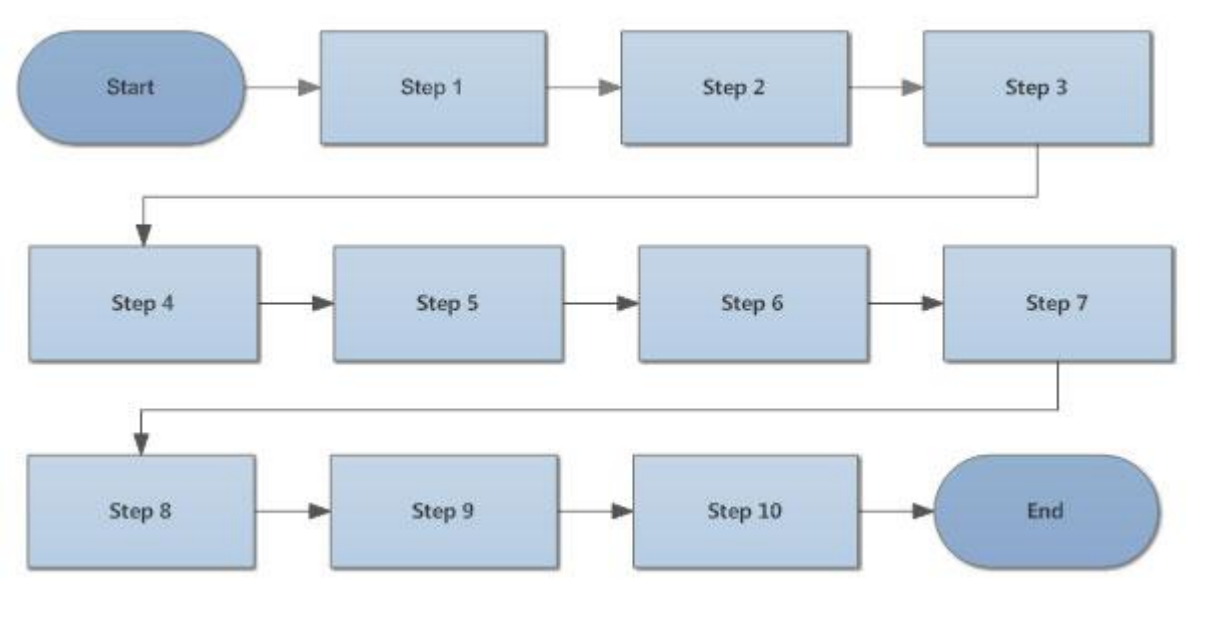
Shapes, lines and texts within a flowchart diagram should be consistent.



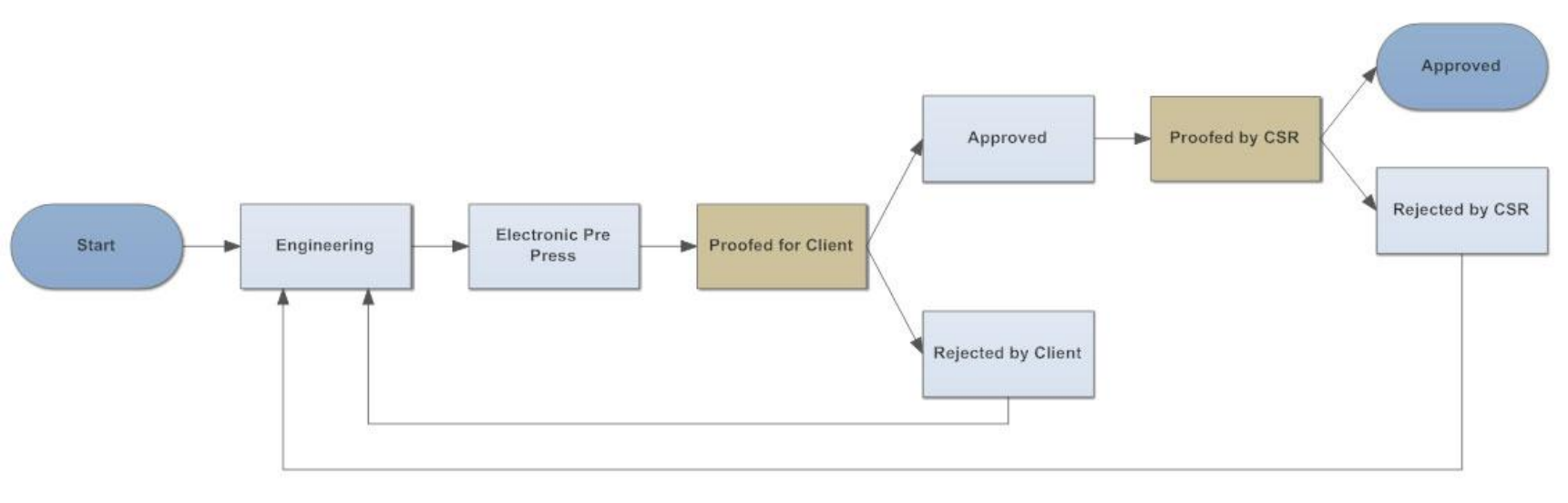
Keep Everything on One Page

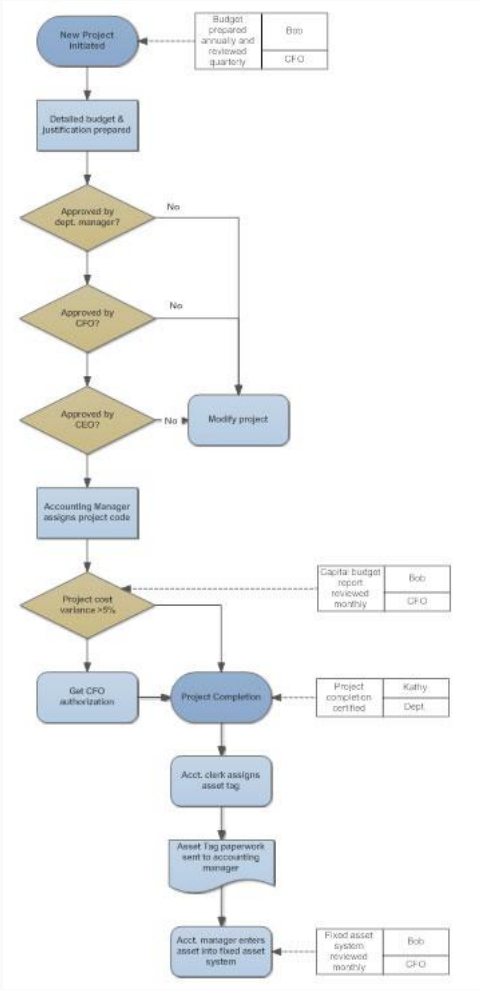


Flow Data from Left to Right



Place Return Lines Under the Flow Diagram




New Project Initiated
Budget prepared annually and reviewed quarterly
Bob
CFO
Detailed budget & justification prepared
Approved by dept. manager?
No
Approved by CFO?
No
Approved by CEO?
No
Modify project
Accounting Manager assigns project code
Project cost variance >5%
Capital budget report reviewed monthly
Bob
CFO
Get CFO authorization
Project Completion
Project completion certified
Kathy
Dept.
Acct. clerk assigns asset tag
Asset Tag paperwork sent to accounting manager
Acct. manager enters asset into fixed asset system
Fixed asset system reviewed monthly
Bob
CFO

Data Flow Diagram (DFD)

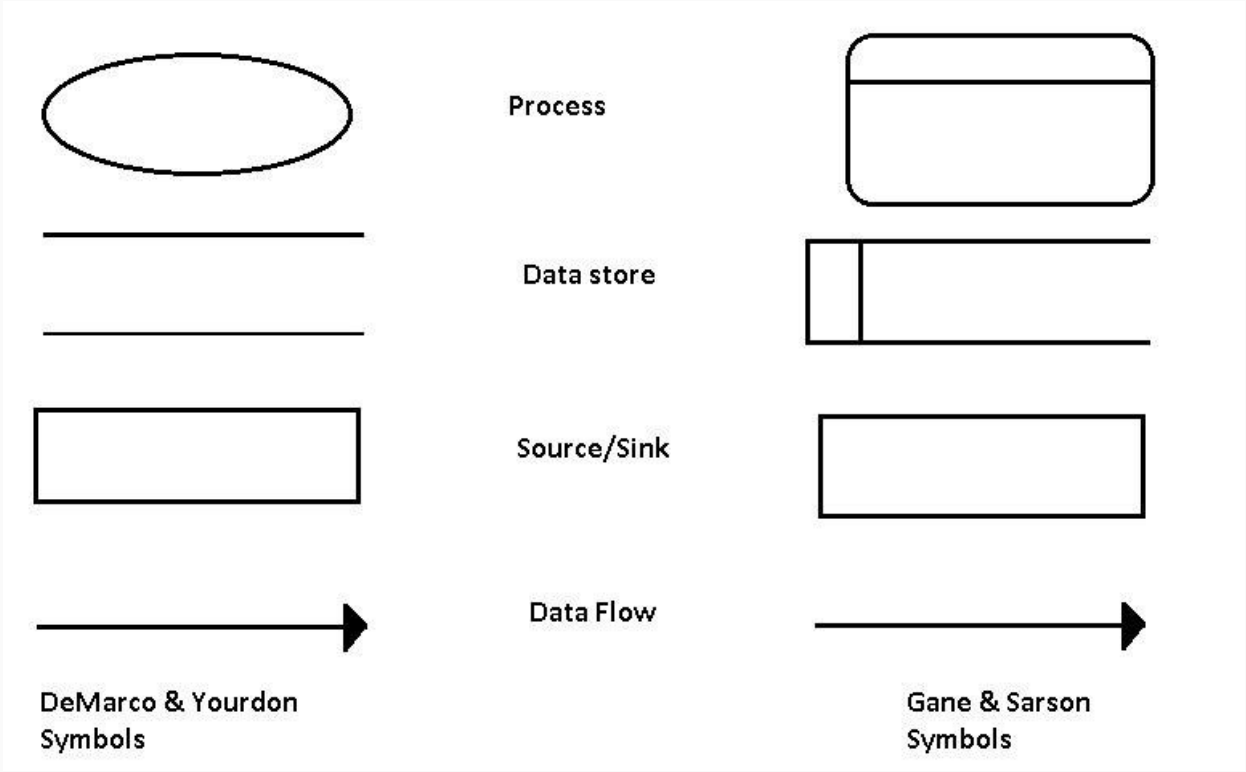
Data Flow Diagram (DFD)

DFDs show the flow of data from external entities into the system, showed how the data moved from one process to another, as well as its logical storage.

DFD notations

- **Rounded rectangles or Circle** - representing processes, which take data as input, do something to it, and output it.
- **Open-ended rectangles -** representing data stores, including electronic stores such as databases or XML files and physical stores such as or filing cabinets or stacks of paper.
- **Squares -** representing external entities, which are sources or destinations of data.
- **Arrows -** representing the data flows, which can either be electronic data or physical items.

DFD notations



Data flow rules

Data flow can take place between -

- A data store to a process
- A process to a data store
- An external entity to a process
- From a process to an external entity

Data flow can not take place -

- An external entity to an external entity
- A data storage to a data storage

Describing a System with DFD

1. Context Level DFD
2. Level 1 DFD

Context Level DFD

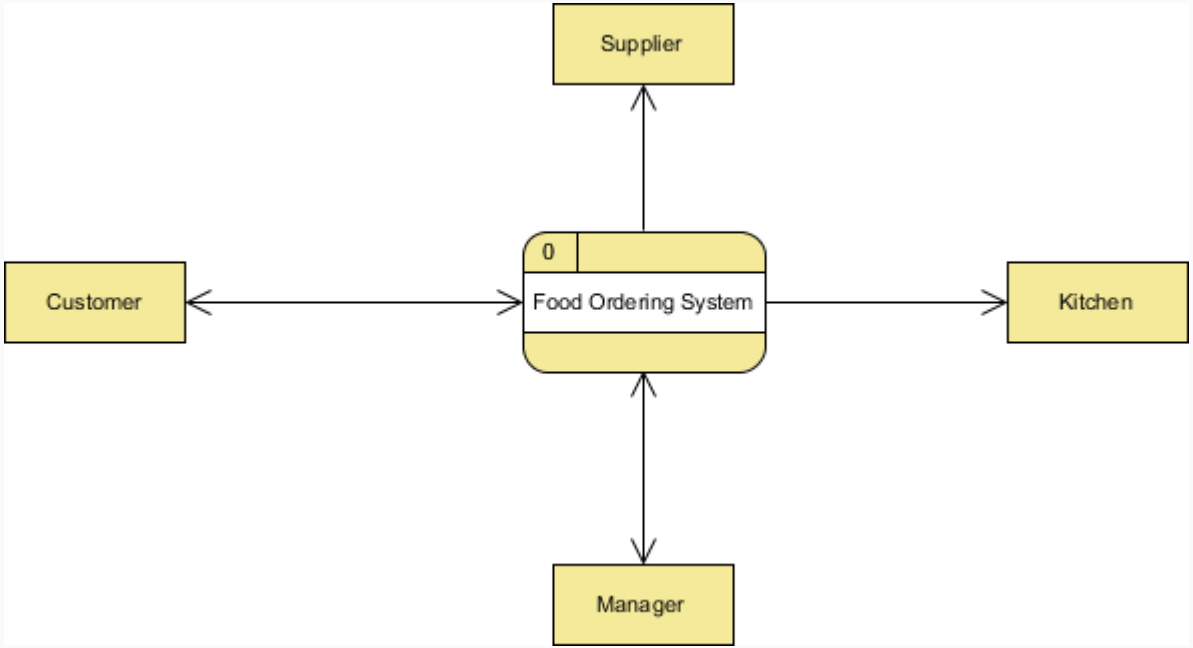
- A context level DFD is the most basic form of DFD.
- It aims to show how the entire system works at a glance.
- There is **only one process** in the system and all the data flows either into or out of this process.
- Context level DFD’s demonstrates the interactions between the process and external entities.
- They do not contain Data Stores.

How to design context level DFD

1. First identify the process, all the external entities and all the data flows.
2. State any assumptions which is necessary to make about the system.
3. Draw the process in the middle of the page.
4. Draw external entities in the corners
5. Finally connect entities to our process with the data flows.

Context Level DFD

Food Ordering System Example



Level 1 DFD

Level 1 DFD’s aim to give an overview of the full system. They look at the sytem in more detail. Major processes are broken down into sub-processes. Level 1 DFD’s also indentifies data stores that are used by the major processes.

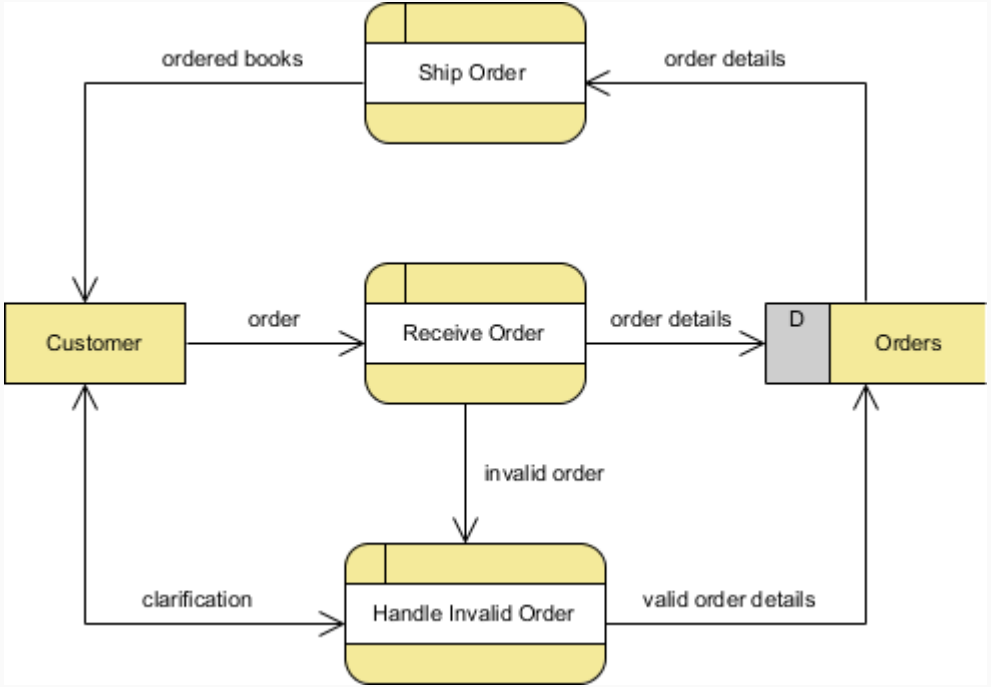
How to design level 1 DFD

1. Must start by examining the Context Level DFD.
2. Must break up the single process into its sub-processes.
3. Must pick out the data stores from the text that are given and include them in DFD.
4. Like the Context Level DFD’s, all entities, data stores and processes must be labelled.
5. Must also state any assumptions made from the text.

https://eternalsunshineoftheismind.wordpress.com/2013/03/05/context-level-dfds-level-1-dfds/comment-page-1/

Level 1 DFD Example

Food Ordering System Example



Activity Diagram

Activity Diagram

- Activity diagram represents **data and activity flows** in an application.
- Activity diagrams is used to **visualize the workflow** of a business use case.

Why should we use activity diagram

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.
- Analyze a use case by describing and timing the necessary actions
- Illustrate a complex sequential algorithm

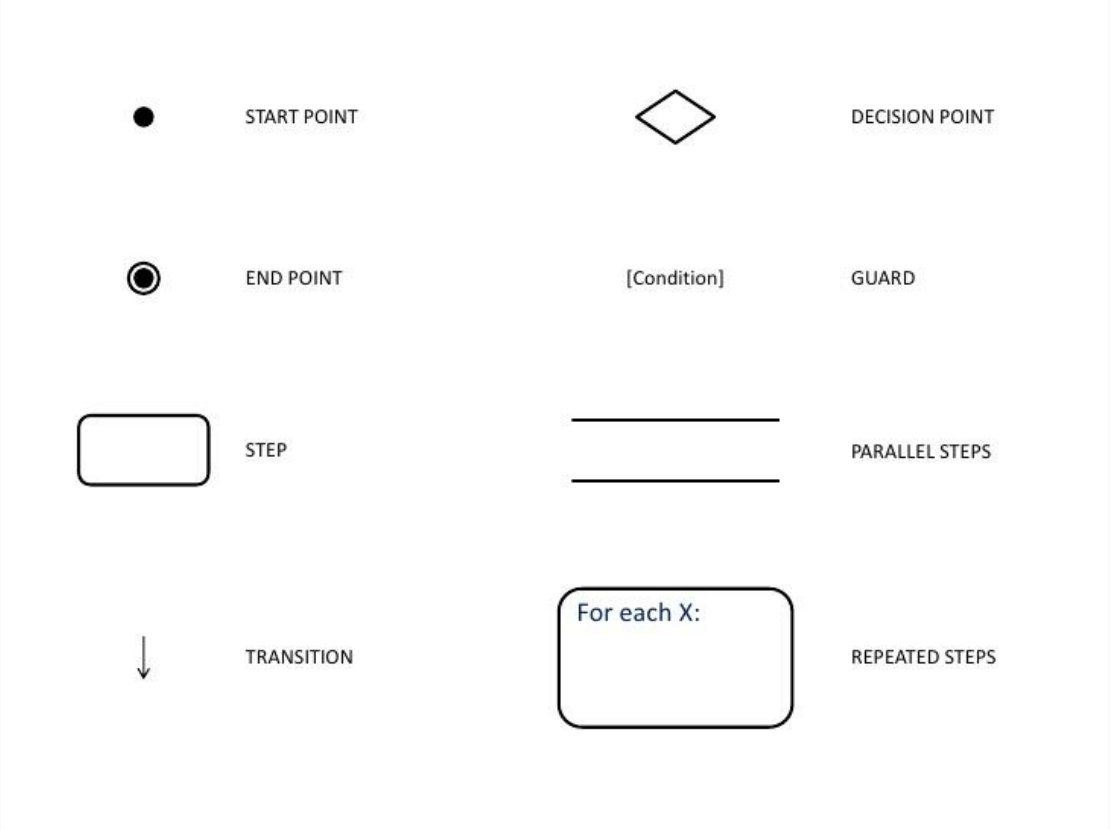
How activity diagram and flowchart are different

Activity diagrams are mainly used as a flow chart consists of activities performed by the system. But activity diagram are not exactly a flow chart as they have some additional capabilities. These additional capabilities include **branching**, **parallel flow** etc.

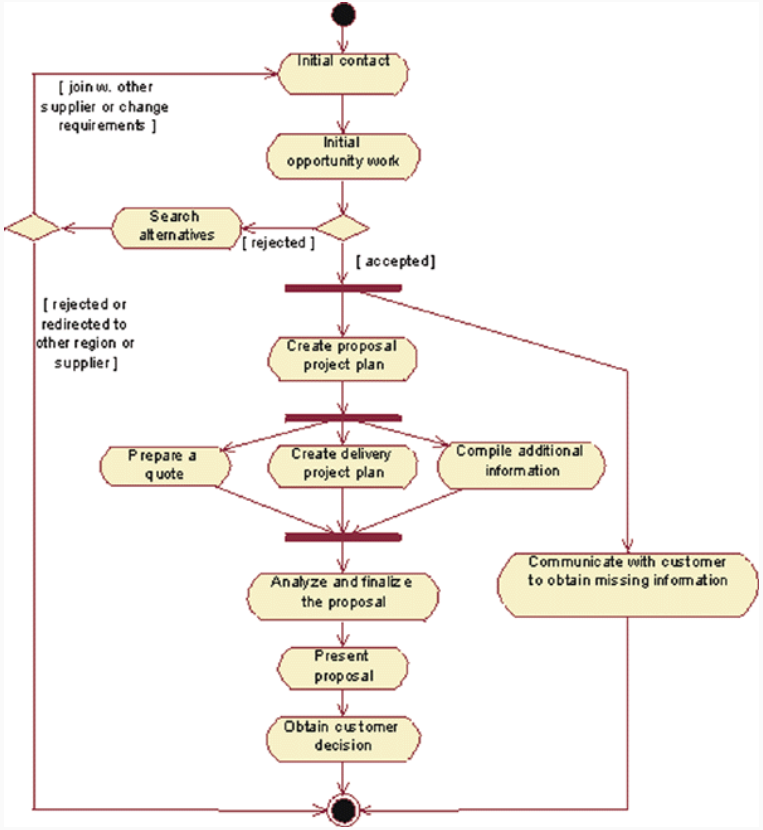
Activity Diagram notations

1. **Activity** - represents an action or a set of actions to be taken.
2. **Control flow -** shows the sequence of execution.
3. **Initial start node** – the beginning of the set of actions (the start basically)
4. **Final node** - stops all flow in an activity diagram.
5. **Decision node -** represents a test condition – much like an IF statement.
6. **Merge Node** - same as decision node in shape but it merges subflow
7. **Synchronization bars -** represents parallel subflows

Activity Diagram Notation



Activity diagram’s example



Sequence Diagram

Sequence Diagram

A sequence diagram illustrates the objects that **participate in a use case** and the **messages that pass between them over time for one use case**. A sequence diagram is a dynamic model that supports a dynamic view of the evolving systems. It shows the explicit sequence of messages that are passed between objects in a defined interaction.

Since sequence diagrams emphasize the time-based ordering of the activity that takes place among a set of objects, they are very helpful for understanding real-time specifications and complex use cases.

http://www.smartdraw.com/sequence-diagram/

Thank you

for your time and patience.