Object Oriented Analysis and Design Notes

System Engineering

System engineering is the need to model some complex system. We need to understand each and everything about the system and make a useful system model, based on the user.

It overlaps with Software Engineering in many areas. Software Engineering is about developing software based solutions while System Engineering is about problem solving.

We divide the system **(WORLD VIEW)** into specific domains of interest **(VIEWS)** by using a divide-and-conquer approach.

Example: We can divide the
Pakistan Railway organization → Railway System, Management System, Security System

- Railway System → Information System, Maintenance System
    - Information System → Weather, Tracking, Track Progress
- Management System → Booking System, Staff, Managers
- Security System → Surveillance System, Checking System

etc.

Hierarchy

World View [WV] → **Domain View** [Di] → **Element View** [Ej] → **Detailed View** [DVk]

Relations can be made using the symbols and making a set.
WV = {D1, D2, D3... Dn}
Di = {E1, E2, E3... En}
Ej = Components {C1, C2, C3... Cn}

System Specification

It is the foundation for Software, Hardware, Database and Human Engineering.

Computer Based Systems
A set or arrangement of elements that are organized to accomplish some pre-defined goal by processing information

System Elements

- Software
- Hardware
- Database
- People
- Documentation
- Procedures

Information Engineering

Goal

Define most effective Architecture to use and Create a plan to implement it.

Hierarchy

WV = **Enterprise** (Information and Strategy Planning)
Di = **Business Area** (Analysis)
Ej = **Information System** (Design)
DVk = **Elements** (Construction and Integration)

Product Engineering

Goal

Make a product, based on the user's wishes, with the specified capabilities.

Hierarchy

WV = **Product** (System Analysis)
Di = **Hardware/Software** (Component Engineering)
Ej = **Data/Functions/Behaviours** (Analysis and Design)
DVk = **Components** (Construction and Integration)

Requirements Engineering

It is the process by which **services** are established and the **constraints** under which it operates/develops, based on the **customer's wishes** from the system.

Software Requirements

It is a feature/function/capability/property etc. that the product must have.

Requirements vs Design

What a system must do → **Requirements** (Client needs)
How a system does it → **Design** (Solutions)

Process of Requirements Derivation

1) Feasibility Study

Is it possible with available resources?

Purpose
To find out if the project is possible with the given constraints and the cost.

Activities are the **Estimation** of resources needed for the project. The **Documents** produced after the feasibility study vary a lot based on different factors. The major **Problem** is the of Lack of Objectivity/Biased-ness.

2) Requirements Analysis

What the clients want from the system?

It is the first technical step. It involves **Problem Recognition** which is studying the plan, communicating and finding the problem that user sees. **Problem Evaluation** is the focus on what.

The users, managers and engineers that are involved are called **Stakeholders.**

Problems

- Stakeholders don't know what they want.
- Stakeholders give unclear requirements.
- Stakeholders can give requirements opposite to other Stakeholders.
- Requirements can change.

3) Requirements Definition

Define the requirements in easy to understand form.

It is a statement in natural language (+diagrams) of the system (services and constraints).

4) Requirements Specification

Define the requirements in detail.

It is the structured document which has the detailed description of the system.

Problems

The requirements phase is hard. Some reasons for that:

- **Requirements Elicitation** - Gathering requirements.
  It is hard because:
  - Unclear who the user is.
  - Different customers, Different needs.
  - Customers themselves are unclear and confused.
  - Engineers and Customers don't understand each other.
  - It is a big, boring, tiresome and money-investing job.
- **Requirements Volatility** - Change in requirements.
  - If they change → Productivity decreases, Customer is happy.
  - If they stay the same → Productivity increases, Customer is unhappy.
- **Language** - Natural vs Formal
  - Natural Language → Imprecise but Expressive for Customers.
  - Formal Language → Precise but Hard to understand for Customers.
- **Requirements Traceablity**

Software Requirements Specification

The SRS is a document which contains every single specified requirement.

Role of SRS

- It drives the rest of the cycle.
- It forces good problem analysis.
- It serves as an agreement between customers and developers about the product.

Communication Techniques

- Starting the process by having the **First Interview or Meeting** → Focus on Problem, Solution and Effectiveness of the Meeting.

- Asking **Context-Free** Questions → Focus on Customers, Goals and Benefits.

- After this, a meeting format which is focused on Problem Solving, Negotiation and Specification is used and there are two popular ones.

1- FAST (**Facilitated Application Specification Techniques**)

Make a joint team of customers and developers which identify the problem and give solutions.

FAST Basic Guidelines

- Customers and Developers meet.
- Preparation and Participation Rules are made.
- The Agenda should be balanced between formal and informal.
- A facilitator controls the meeting and can be the Customer, Developer or an Outsider.
- A definition mechanism is used on the Wall Board, Stickers or Chart.

2- QFD (**Quality Function Deployment**)

Turn the customer needs into software requirements.

Steps

1. Function Deployment (value)
2. Information Deployment (input/output)
3. Task Deployment (behavior)
4. Value Analysis (priority)
5. Customer Voice Table (requirements)

Diagrams

DFD (**Data Flow Diagram**)

It is a way of showing flow of data in a system.

Logical DFD focuses on the **WHAT**.
Physical DFD focuses on the **HOW**.

The Context Diagram: highest level DFD, that shows relationship between the system and external entities.

Level 0: a basic DFD, shows the entire system as a single bubble.

Level 1: goes into detail where the main process is broken into sub processes. These sub products/processes are shown by smaller circles.

Components of a DFD

- **Process**
  Takes in incoming data flow and Gives out outgoing data flow.

- **Data Flow**
  Lines through which data flows. The arrows are labeled by the activity name.

- **Data Store/Warehouse**
  Storage of Data, also called Files.

- **External Entity**
  Sources(Inputs) and **Sinks/Destinations**(Outputs) of the system.

Notation

Yourdon and Coad

Symbols

1. Process → Circle.
2. Data Flow → Arrows.
3. Data Store/Warehouse → Open Rectangle.
4. External Entity → Square or Rectangle.

Gane and Sarson

Symbols

1. Process → Rounded Rectangle (level is mentioned in the upper portion).
2. Data Flow → Arrows.
3. Data Store/Warehouse → Rectangle (with a letter).
4. External Entity → Rectangular Pill

Use Case Diagram

A diagram that relates the Actors to System with relationships.

System

The thing which is being worked on. The Boundary of the System is shown by a Rectangle.

Actors
People/Entities, with a role, operating on the system. Shown using Stickfigures. The primary actor is on left side and the secondary actors are on the right side.

Use Cases
How the system interacts with Actors. They are shown using Ovals.

Relationships between Actors and Usecases → Simple Line.
Relationships between Usecases → Arrows labeled <<uses>> or <<extends>>.
<<uses>> shows dependence
<<extends>> shows alternatives

Level 0 DFD - Context Diagram
Level 1 DFD
Level 2 DFD