System Modelling

Definitions:

System Modelling is the process of developing abstract models of a system with model presenting a different view or perspective of that system.
or
A System Model represent aspects of a system and its environment.
or
System Modelling is a mean of representing a world view a detailed view of the system using same kind of Graphical Notation.

Features of a System Model :

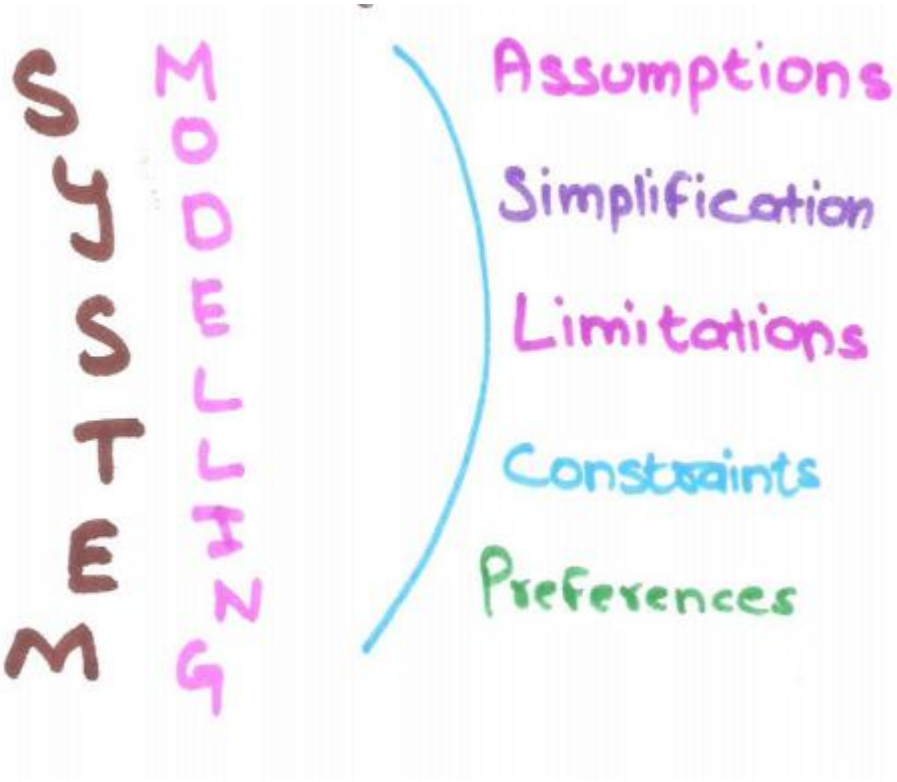
- **define the processes that serve the needs of the view under consideration.**

- **represent the behaviour of the processes and the assumptions on which the behaviour is based.**

- **explicitly define both a exogenous and endogenous input to the model.**

- **represent all linkages(input/output) that will enable engineer to better understand the view.**

To construct a model, the engineers should consider a number of restraining factors- assumptions- simplifications - limitations - constraints - preferences



- **Assumptions:**

It enables a model to reflect the problem in a reasonable manner by reducing the number of possible **permutations and variations.**

Example: Representation of 3D human forms

In this input domain maybe that the system engineer makes certain assumptions about the range of allowable human movements. **(leg cannot be wrapped around torso) so that range of Input and processing can be Limited**.

- **Simplifications:**

That enables the model to be created in a timely manner. Example: A System Engineer is modelling the needs of the service organisational and is working to understand the flow of information that spawns a service order. **Although a service order can be derived from many origins, the engineer categorises only two sources.**

Internal Demand and External Request

This enables a simplified partitioning of input that is required to generate the service order.

- **Limitations: That help to bound the system.**

Example: An aircraft avionics system is being modelled for future aircraft. Is the aircraft will be a two-engine design, the monitoring domain for propulsion will be modelled to accommodate a maximum of two engines and associative redundant system.

- **Constraints:** That will guide the manner in which the model is created and the approach taken when the model is implemented.

Example: Suppose a system for the 3D-rendering describes previously is a singleG4 based processor so computational complexity of problems must be constrained to fit within processing bounds imposed by the processor.

- **Preferences:** It indicates the preferred architecture for all data, functions and technology.

Software Simulation

Problem in software engineering was not going through proper planning. We build systems like the Wright Brothers build "aircrafts" build the whole thing, push it off a cliff, let it crash, and start over again.

What is Simulation?

Simulation is the imitation of the operation of a real-world process or system overtime.

or

System Simulation is a set of technique that uses computers to imitate the operations of the various real-world task or processes through simulation.

How System Simulation Works?

- **1) Simulating first requires that model be developed.**
- **2) Model posses characteristics, behaviors of selected abstract system.**
- **3) Computer describe a display Complex interaction among multiple variable within a system.**

Simulation is used in many context such as:

1- simulation of Technology

2- Safety Engineering

3- Testing

4- training

5- education

6- video game

7- scientific modelling

8- human system modelling

9- clinical Healthcare Simulator

Today, software tools for system modelling and simulation are being used to help to eliminate surprises reactive, computer based system are built. These tools are applied during system engineering process.

Role of hardware and software, database and people is being specified. These tools enable a system engineer to "test drive" a specification of the system.

Analysis Principles

Over the past two decades, a large number of analysis modelling methods have been developed.

By **Analyzing Problems And Their Causes,** investigator have developed a variety of

- **- notations and**

•- corresponding sets of heuristic overcome them.

Each analysis method has a unique point of view, however all analysis methods are related by a set of operational principles:

- **1) The information domain of a problem must be represented and understood.**
- **2) The functions that the software is to perform must be defined.**
- **3) the behavior of the software must be represented.**
- **4) The Models that depict information and behavior must be partitioned**

in a manner that uncovers detail in a layered (or Hierarchical) fashion.

- **5) the analysis process should move from essential information towards implementation detail.**

By applying this principles, an analyst approaches a problem systematically.

A Set of guiding principles fro Requirement Engineering:

- **Understand** the problem before you begin to create the analysis model.
- **Develop Prototype** that enable a user to understand how human / machine interaction will occur.
- **Record** the origin of and reason for every requirement.
- **Use Multiple use of requirements**- building data, functional and behavioral models provide the software engineering with different 3 views.
- **Rank Requirements:** Tight deadline may preclude the implementation of every software requirement. If an incremental process model is applied, those requirements to be delivered in the first increment must be identified.
- **Work to Eliminate Ambiguity:** As most requirements are described in natural language, the opportunity for everybody abounds. The use of formal technical review is one way to uncover and eliminate ambiguity.

Analysis Principle- Modelling

Software Development Team first check the type of entity for which model has to make is either physical entity of software entity.

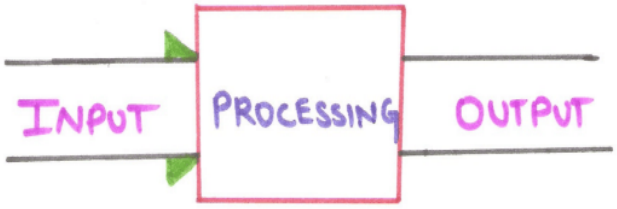
Physical Entity: Physical thing like a building, a plane, a machine, engineers at make a functional model that is identical in form and shape but smaller in scale.

Software Entityecure Entity: In this model must be capable of representing the information that **software transforms, the functions (and sub functions)** that enable the transformation to occur, and the behavior of the system as the transformation taking place.

The second and third operational analysis principles requires that we build models of functions and behavior.

Functional Models:

Software transforms information, and it needs to perform three generic functions.



The functional model begins with a single context level model(i.e name of the software to be built).s over a series of iterations, more and more functional details is provided, until a thorough delineation of all system functioning is represented.

Behavioral Models:

Most software responds to events from outside world. This stimulus response characteristics from the basis of the behavioral model. A computer program always exists in some state - An Externally observable mode of behavior(example computing, printing, polling, waiting) that is changed only when some event occurs.

For Example:

Software will remain in the wait state until all these happens:

- **An internal clock indicates that sometime interval has passed.**
- **An external event (example Mouse movement) cause an interrupt.**
- **An external system signals the software to act the same manner.**

A behavioral model creates are representation of the states of the software and the events that causes a software to change state.

Analysis Principle: Partitioning

Problem is partitioned - divide into parts that can be easily understood an established interfaces between the parts.

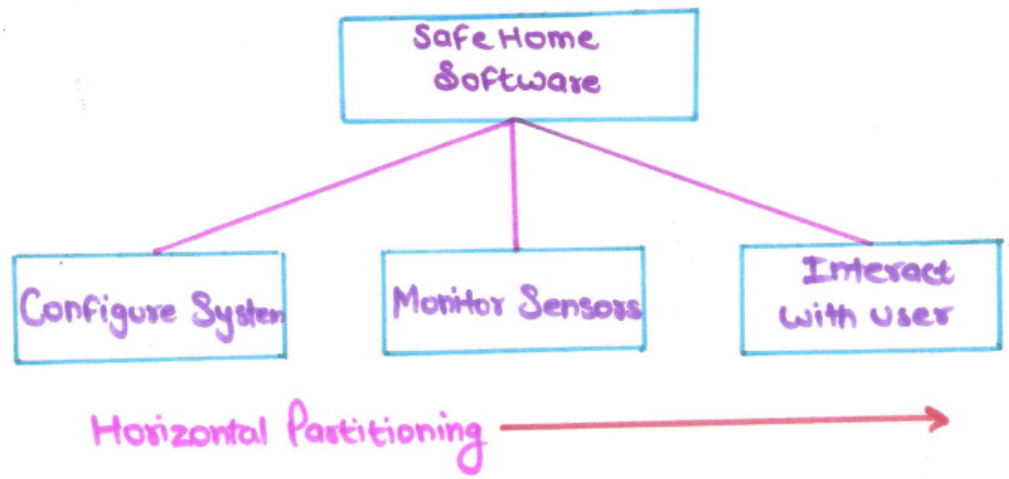
Form the fourth operational analysis principles suggests that the **Information, Functional and Behavioral** domains of the software can be partitioned.

Partitioning decomposes a problem into its constituent parts. Conceptually, establish a hierarchical representation of function or information and then partition the uppermost element by :

- **1) Exposing** increasing detail by moving vertically in the Hierarchical.
- **2) functionally** decomposing the problem by moving horizontally in the hierarchy.

Example: Safe home security system.

Home security system software enables the home owners to configure the security when it is installed, monitors all sensors connected to the security system and interact with the homeowner through a keypad and function keys and contained in the SafeHome control panel.



During installation, the SafeHome control panel is used to ' program ' and configure the system.

Each Sensor is assigned a number and type, a master password is programmed for arming and disarming the system, and telephone number are input for dialing when a sensor event occurs.

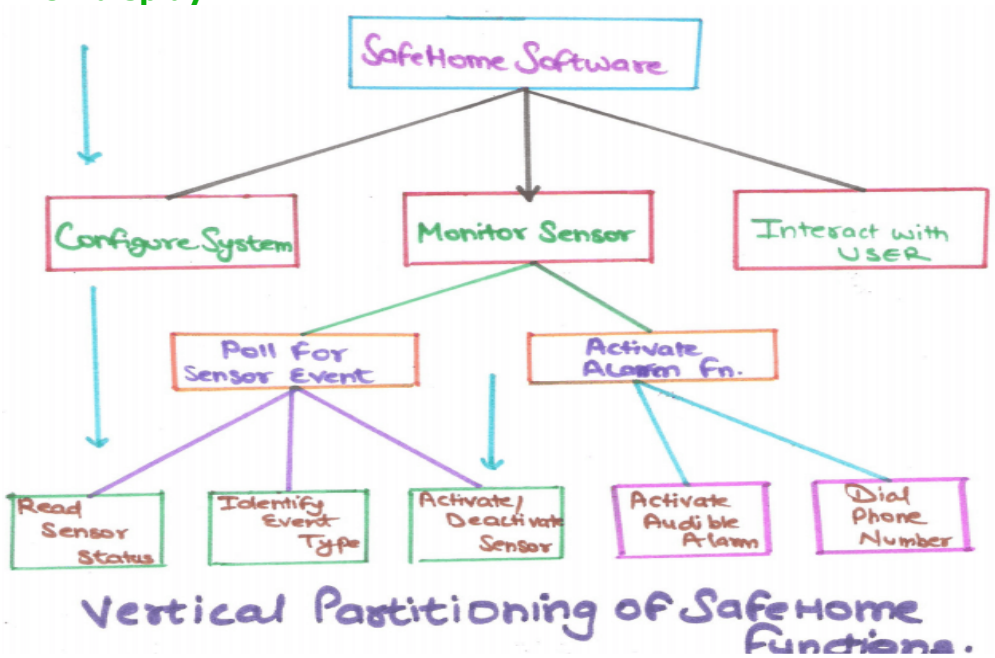
How SafeHome Works?

When sensor event is recognized, the software invokes an audible alarm attached to the system. After a delay time that is specified by the home owner during system configuration activities

The software dials a telephone number of a monitoring service, provide information about location, reporting the nature of the event that has been detected.

The telephone number is dialed every 20 seconds until telephone connection is obtained.

• All interaction with SafeHome is managed by a user-interaction subsystem that reads input provided through the keypad and function keys, **displays prompting messages on the LCD display, displays system status information on the LCD display.**



Software Prototyping

It is a process of implementing the presumed software requirements with an intention to learn more about the actual requirements or alternative design that satisfy the set of actual requirements. or **Prototyping** is the process that enables developer to create a small model of software.

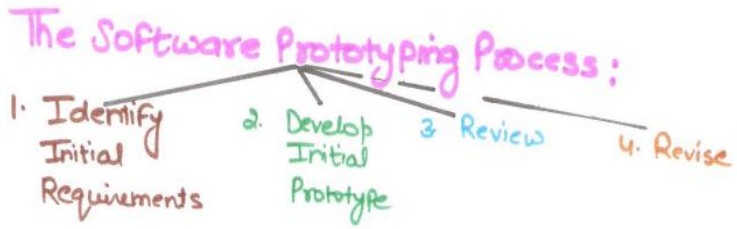
or

Prototyping gives the software publisher the opportunity to evaluate the product, insurer it's doing what's intended, and determine if improvements needs to be made.

The software prototyping process:

- **1- identify initial requirements**

- **2- Develop initial prototype**
- **3- review**
- **4- revise**



1) Identify initial requirements: Software publisher decides

- A) What software will be able to do
- B) who will be exact users
- C) user expectations from product

2) Develop initial prototype: In this developer will consider the requirements as proposed by the publisher and begin to put together a model of what the finished product might look like . Some Initial prototype may be as simple as a drawing on a whiteboard.

3) Review: Once the prototype is developed, the publisher has a chance to see what the product might look like. In more advanced prototypes, the end consumer may have an opportunity to try out the product and offer suggestions improvement. This is also called Beta Testing.

4) Revise: The final step in the process is to make revisions to the prototype based on the feedback of the publisher and/or beta testers.