Software Modeling_An Introduction

A **model** is a representation of things in the real world concentrating on the essential items. It is a means of understanding problems involved in building something. It can be taken as a means of communication between those who are involved and the users as part of some deliverable. A model is a graphical representation of a system

Categories of models

Static models: These describe a set of elements and any relationships that exist between them (class diagrams, object, component and deployment diagrams)

Dynamic models: These describe the behavior of one or more elements over time. (usecase, sequence, collaboration, state chart, activity diagrams)

In modeling we look at the essential elements that enable us to communicate.

Modeling language: A language allows consistence in communication and is based on;

A syntax which is a set of rules that define what symbols can be used on a diagram

Semantics which establish what diagrams and symbols mean

5 criteria of considering a language

- It must be sufficiently expressive…must be able to represent the message
- It must be easy to use for expression of the message
- It must be unambiguous..it shouldn't have more than one interpretation
- It must be supported by suitable tools so that you are able to apply them
- It must be widely used

The unified modeling language: It is represented by many diagrams. It is an object oriented language for specifying, visualizing, constructing, and documenting the artifacts of software systems as well as for business modeling. It is a visual modeling language not a visual programming language. It is used for graphically depicting OOA and design models. It doesn't only allow specifying the requirement of a system and capturing decisions but also promotes communication among key persons involved in the development process

Four principles of modeling

- The choice of the model is important ie the diagrams to be used
- The level of precision may differ in details
- Models are connected to reality…must support reality
- No single model is sufficient

Use case; This illustrates a unit of functionality provided by the system. Its main aim is to help the development teams visualize the functional requirements of the system including the actors to essential processes as well as the relationships among the different use cases

An actor is a person or system that will interact with the system. A use case is typically used to communicate the high level functions of the system and system scope.

Class diagram: This shows how the different entities (people, things, data) relate to each other. In other words it shows the static nature of the system. It can be used to show the logical classes which are typically the things business people talk about in an organization eg interest rates, home morgages etc

Advantages of UML

- Its proprietary
- It supports OO approach
- Supports both static and dynamic modeling
- It is independent of any software development process
- It is independent of implementation technologies

Before we model, we need a **Conceptual model**: it is the initial understanding of the model and for that case, it is the main essence of the model.

There are four steps involved in conceptual modeling

- Identification of real world entities described in the requirements document which are concerned with the core functionality of the system leaving out consideration of the user interface and this will be known as the conceptual classes of the model
- Identification of the properties of the entities and the relationships between them eg students and lecturers
- Identify the constraints/conditions that will be imposed on a model
- Represent the model using diagrams in associated text

Concepts in developing models

OO Approach: Here we are talking about classes. It is data centric ie objects are instances of classes. A class describes everything about an object. Objects have a state, behavior and property

A conceptual model is a model which is made up concepts and their relationships. This helps us to understand the entities in the real world and how they relate with each other. The conceptual model of UML can be mastered by learning the following 3 major elements;

- A model is visual
- A class is a descriptor of a set of objects with its attributes and operations whereas a class is divided into 3 components ie the name, attributes and operations. Attributes are properties of a class that describe a class
- A method is a procedure that implements the operation

Technical advantages of OOL

- Abstraction
- Encapsulation
- Re-use
- Message passing
- Inheritance
- Polymorphism
- Greater reuse of data and code
- Increased programmers productivity
- Improved software quality, greater standability of the software
- Shorter development times

Computer Aided software

Case tools: These are automation of step by step methodologies for software and system development to reduce the amount of repetitive work the developer has to do.

Advantages of case tools;

- They enforce the standard design methodology and design discipline
- They organize and coronate design components and provide rapid access to them via a design repository
- They automate tedious and error prone portions of analysis and design
- They automate code generation, testing and control roll out
- They improve the productivity of workers

Categories of case tools

Front end case tools; They focus on capturing analysis and designing information of the other stages of the system development process

Back end case tools; These address the coding and maintenance activities or convert the automatic specification into program code.

Objects send messages (senders) as others receive messages (receivers). A message interface defines what an object does

Two things before we start modeling

- OO Analysis: we identify objects and their relationships
- OOD: we collaborate the objects and the relationships into a design

3 major elements of UML

- UML building blocks
- The rules that connect these building blocks
- Common mechanism of UML…..architecture

The building blocks include things, relationships and UML diagrams.

The UML things have the things that are structural and these describe the static part of UML models. The behavioral represent the dynamic parts of UML models. Grouping puts together both the structural and the behavioral to form a packet. It is the organization part of UML

Annatotional give the explanatory part of UML.

Structural things: these define the static part of the model and they represent both physical and conceptual elements in UML they include;

Class which represents a set of objects having similar responsibilities or process

Interface which defines a set of operations which specify the responsibility of the class ie a collection of external visible operations

Collaboration which defines the interaction between elements

Use case which represents a set of actions performed by the system for a single goal

Component which describe the physical part of the system

Node that describe the physical element of the system that exist at run time ie the computational resource at run time, processing power, web server, memory etc

Behavioral things: these show the dynamic nature of the system by describing the interaction in the system. They include:

Interaction: it is a behavior that consists of a group of messages exchanged among element to accomplish a specific task

State machine: this defines the sequence of states an object goes through in response to events.

Relationships: these show how the elements are associated with each other and this association describes the functionality of the application. There are four kinds of relationships;

- *Realization*: two elements are connected where one element describes some responsibility and the other implements it. This relationship exists in case of classes, use cases, interfaces and collaboration
- *Generalisation*: connects a specialized element (the child) with a generalized element (the parent)
- *Dependence*: relates two things in which a change in one (independent) affects the other element(dependent)
- *Association*: basically a set of links that connect elements of a UML model. Can be described into two ie composition (…is part of a whole eg engine and a car) and aggregation (has a eg…customer and address)