International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 02 Issue: 05 | Aug-2015 www.irjet.net p-ISSN: 2395-0072

An Effective Language for Object Oriented Design-UML(Unified Modeling Language)

M.L.V.Roopa Vani[1], M.Chandrika Kumari[2], M.Hari Priya[3], N.Harika[4]

[1 2 3 4] *B.Tech Final Year, CSE, Dadi Institute of Engineering & Technology,Andhra Pradesh,India*

Abstract – ***Any organization has the possibility to win in the competition only when it forms the strategic alliance with the upstream and downstream enterprise. This paper articulates a way of using unified modeling language (UML) to develop business value chain activities for any enterprise by introducing basic fundamentals to develop dynamic, adhoc and agile business model. The results show that the UML is useful in the development of information systems and is independent of any programming language. The heart and soul of effective object models is its strategies and patterns, not the shape of an icon or the number of adornments. Unified Modeling Language is a standard modeling and programming language for writing software blueprints. It is very expressive language, addressing all the views needed to develop and then deploy such systems. Modeling is a central part of all the activities that lead up to the deployment of good software. Modeling is a proven and well accepted engineering technique.***

Key Words: *Unified Modeling Language, Conceptual Model, Adornments, Extensibility.*

1. INTRODUCTION

The importance of computer is increasing day by day. Computer has become popular in different fields such as industry, medicine, commerce, education and even agriculture. Now a days, every organization is dependent on computer in their works as a result of computer technology [2]. Computer is considered a time saving device and its progress helps in processing complex, long, repeated processes in a very short time with a high speed. In addition to using computer for work, people use it for fun and entertainment [1]. Today, software takes on a dual role. It is a product and, at the same time, the vehicle for delivering a product.

Most of the people consider the program and software [6, 8] to be same. But software not only consists of programs but also the supporting documents [8]. Software lifecycle models are used as tools for planning and monitoring software projects [6][7].

Different models have been proposed. Each model's effectiveness varies depending on project level. It is widely acknowledged that no single model is effective in all situations [2]. Because of this, an effective model must be selected for every project.

A software process model[7][8] is an abstract representation of a process. It presents a description of a process from some particular perspective. When any project starts it has to follow a framework with following activities [2]:

Requirement Gathering: To develop software, first we have to gather all the requirements from customer or stakeholder either through interviews or direct interaction.

Planning: Basing on the requirements, budget, time has to be adjusted to develop the software.

Analysis & Design: Complete Analysation is done on the requirements and planning. After complete analysation designing is started. For the design, different tools are used. one of the tool for designing is UML(Unified Modeling Language).

Implementation: For the above design, code is generated with proper programming language.

Testing and Maintenance: After completion of developing software, testing is performed. If the product is well tested then it will go for alpha and beta tests.

The above activities are called frame work activities. In any software life cycle models or software process models these five activities are performed and modified [5][9].

2. EVOLUTION OF UML

UML efforts started officially in October,1994 by James Rumbaugh and Grady Booch [3].
They established three goals for their work,
1) To model systems, from concept to executable artifact, using object oriented (OO) techniques.

2) To address the issues of scale inherent in complex, mission-critical systems.
3) To create a modeling language usable by both humans and machines.

- Initial version of the UML is 0.8 was released in the year 1995.
- The UML 1.0, a modeling that was well-defined, expressive, powerful and applicable to wide spectrum.
- The UML 1.1,was adopted by the OMG(Object Management Group).
- Later released versions are 1.3, 1.4, 1.5 and 2.0.

3. CONCEPTUAL MODEL OF THE UML

The UML is a language for visualizing, specifying, constructing, documenting [3] the artifacts of a software-intensive system [1].

Visualizing: It clearly describes about the semantics of the UML notations.
Specifying: It means building models that are precise unambitious and complete.
Constructing: Modeling of the structure and behavior are defined.
Documenting: It includes all sorts of artifacts like requirements, architecture,
design, source code, project plans, tests, prototypes and releases.
A good diagram can often help communicate ideas about a design, particularly when you want to avoid a lot of details .

To understand UML, we require three major elements.

- Basic building blocks
- Rules
- Common mechanisms

3. 1. BASIC BUILDING BLOCKS OF UML

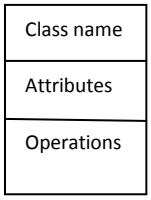
The vocabulary of the UML encompasses three kinds of building blocks [3][4].

Things
Relationships
Diagrams

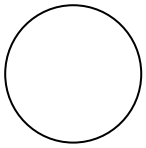
Things: These are the Object Oriented buildings blocks of UML. There exists four types.

1) Structural things: Structural things are the nouns of UML models. There are seven kinds of structural things.

a) Class: It is a description of a set of objects that share the same attributes, operations, relationships and semantics.

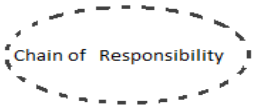


b) Interface: It is a collection of operations that specify a service of a class or component.



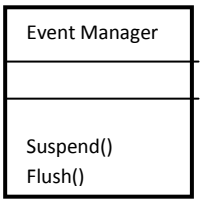
ISpelling

c)Collaboration: It defines an interaction and it have structural and behavioral dimensions.

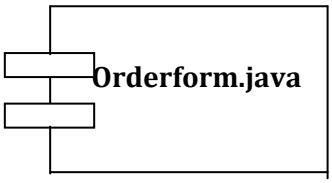


d) Usecases: It is a description of set of sequence of actions that a system performs.

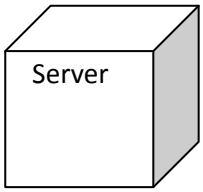
e) Active class: It is a class whose objects own more processes or threats and therefore can initiate a control activity.



f) Component: It is a physical and replaceable part of a system that confirms to and provides the realization of a set of interfaces.

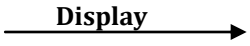


g) Nodes: It is a physical element that exists at runtime and represents a computational resource.

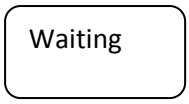


2) Behavioral things: Behavioral things are the dynamic parts of the UML models. These are the verbs of a model.

a) Interaction: It is a behavior that comprises a set of messages exchanged among the set of objects.

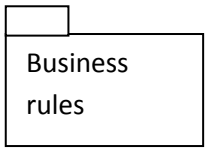


b) State machines: It is a behavior that specifies sequence of states and object or an interaction goes through during its lifetime in response to events.



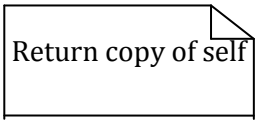
3) Grouping things: These are the organizational parts of the UML model. These are the boxes into which a model can be decomposed.
There is a kind of grouping things i.e PACKAGE.

Package: It is a general-purpose mechanism for organizing elements into groups. All the above mentioned things are placed in a package.



4) Annotational things: These are the explanatory parts of UML model. There is one primary kind of annotational thing, called a NOTE.

Note: It is simply a symbol for rendering constraints and comments attached to an element or a collection of elements.

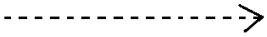


Relationships: There are four kinds of relationships in the UML.

- Dependency
- Association
- Generalization/Specialization
- Realization

1) Dependency:

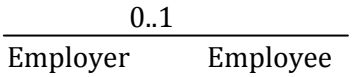
It is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing.



2) Association:

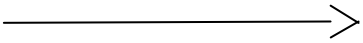
It is a structural relationship that describes a set of links, a link being a connection among objects.

.



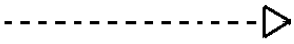
3) Generalization:

It is a relationship in which objects of the specialized elements are substitutable for objects of the generalized element.



4) Realization:

It is a semantic relationship between classifiers, wherein one classifier specifies a contract that another classifier guarantees to carry out.



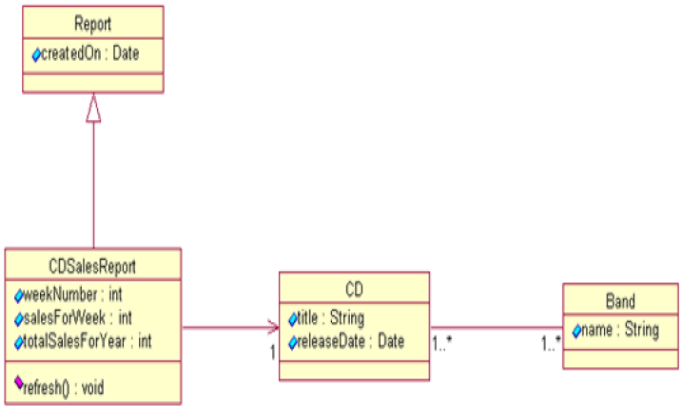
Diagrams:

It is a graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things), arcs(relationships) [4].We draw diagrams to visualize a system from different perspectives, so a diagram is a projection into a system. In theory, a diagram [10] may contain any combination of things and relationships. The UML includes nine diagrams.

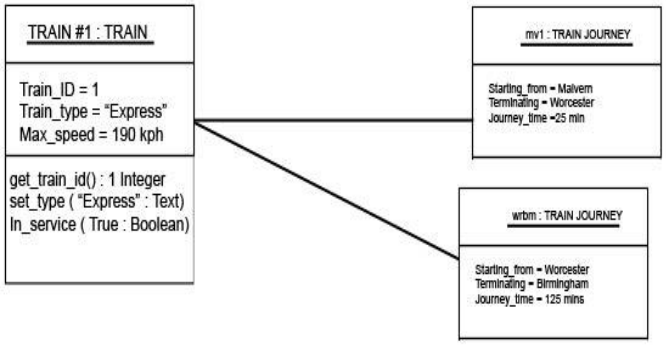
1. Class diagram
2. Object diagram
3. Use case diagram
4. Sequence diagram
5. Collaboration diagram
6. Statechart diagram
7. Activity diagram
8. Component diagram

9. Deployment diagram

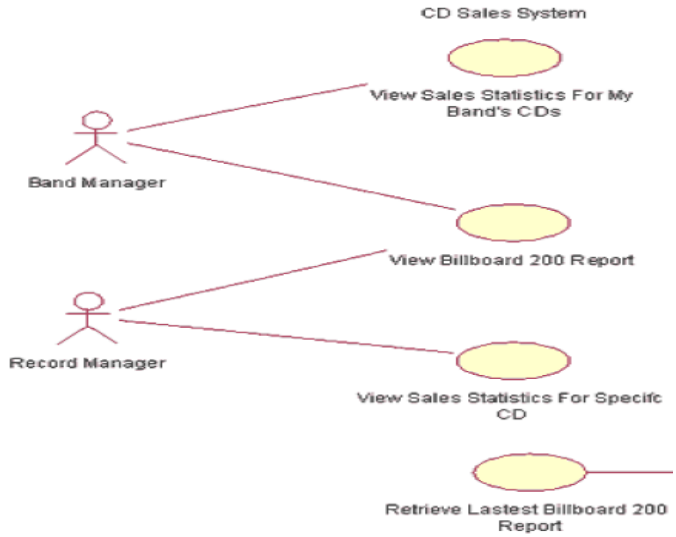
1) Class diagram: The class diagram shows a set of classes and collaborations. These address the static design view of a system.



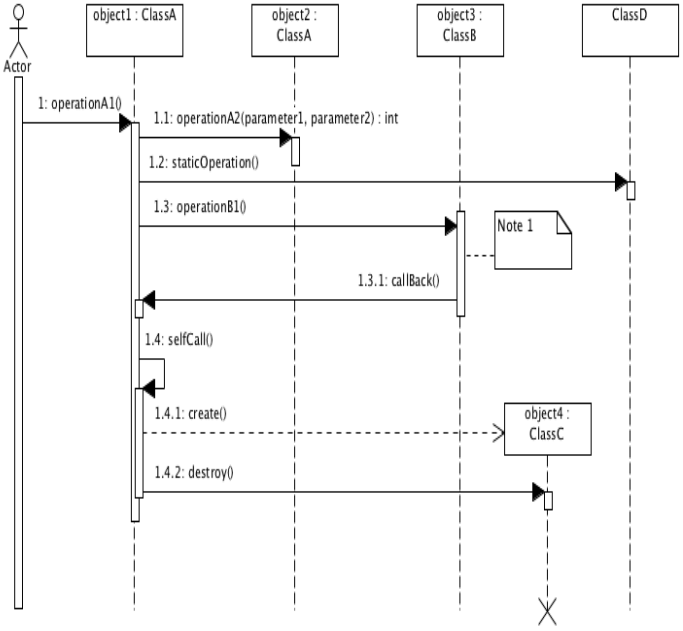
2) Object diagram: An object diagram shows a set of objects and their relationships. These represents static snapshots of instances of things found in class diagrams.



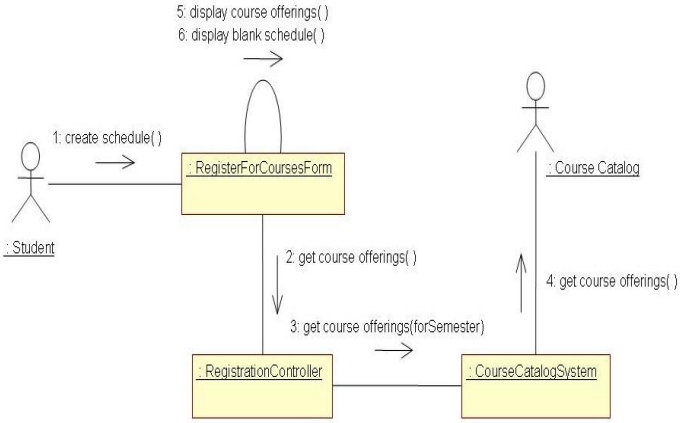
3) Usecase diagram: It shows a set of use cases and actors (a special kind of class) and their relationships[10]. These addresses the static use case view of a system.



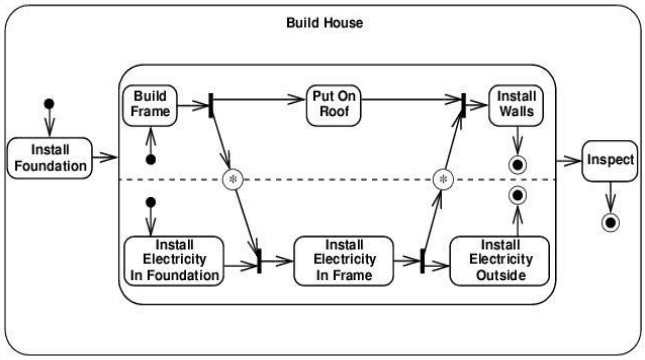
4) Sequence diagram: A sequence diagram is an interaction diagram that emphasizes the time ordering of messages.



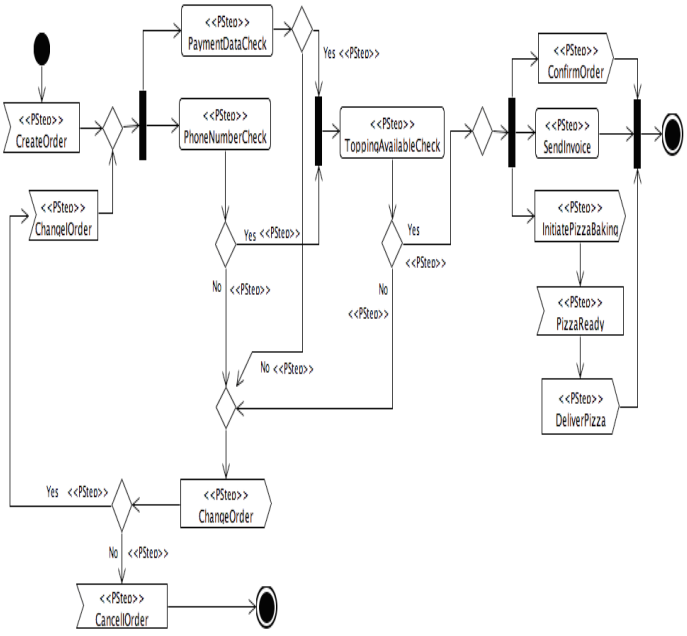
5) Collaboration diagram: It is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages.



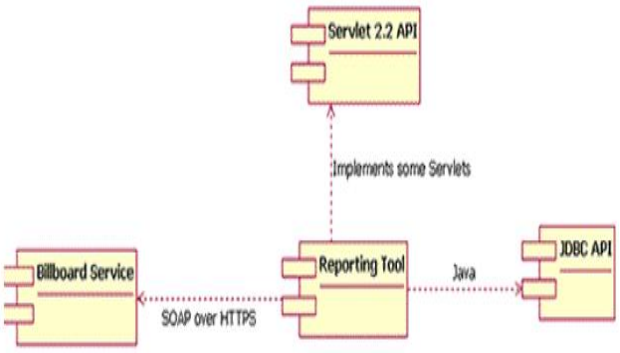
6) Statechart diagram: It shows the state machine consisting of states transitions, events and activities. It addresses the dynamic view of a system.



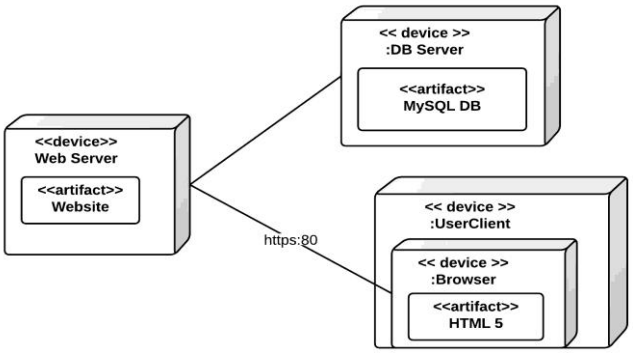
7) Activity diagram: It is a special kind of a state chart diagram that shows the flow from activity to activity with in a system. It addresses the dynamic view of a system.



8) Component diagram: It shows the organizations and dependencies among the set of components [4]. These addresses the static implementation view of a system.



9) Deployment diagram: It shows the configuration of run-time processing nodes and the components that live on them. These addresses the static deployment view of an architecture.



3.2.RULES OF THE UML:

The UML has a number of rules that specify what a well-formed model should look like. A well-formed model is one that is semantically self-consistant and in harmony with all its related models [3]. The UML has semantic rules for

Names: What you can call things, relationships and diagrams.

Scope: The content that gives specific meaning to a name.

Visibility: How those names can be seen and used by others.

Integrity: How things properly and consistently related to one another.

Execution: What it means to run or simulate a dynamic model

Elided: Certain elements are hidden to simplify the view.

Incomplete: Certain elements may be missing.

Inconsistent: The integrity of the model is not guaranteed.

3.3. COMMON MECHANISMS IN THE UML:

It is made simplier by the presence of four common mechanisms that apply consistently throughout the language [3].

Specifications
Adornments
Common divisions
Extensibility mechanisms

1) Specifications: The UML is more than just a graphical language. Rther, behind every part of its graphical noations there is a specification that provides a textual statement of the syntax and semantics of that building block.

2) Adornments: Most elements in the UML have a unique and direct graphical notation that provides a visual representation of the most important aspects of the element.

3) Common divisions: There is a division of class and object. A class is an abstraction; an object is one concentrate manifestation of that abstraction. In the UML, you can model classes as well as objects.

4) Extensibility mechanisms: The UML's extensibility mechanisms include

Stereo types: It extends the vocabulary of the UML, allowing you to create a new kinds of building blocks that are derived from existing ones but that are specific to your problem.

Tagged values: It extends the properties of a UML building block, allowing you to create new information in that elements specification.

Constraint: It extends the semantics of UML building block, allowing you to add new rules or modify existing ones.

4. CONCLUSIONS

The Software life cycle models are the tools that help to manage software life cycle. These models specify process consistency and improvement. Each model has its own strengths and weaknesses. This model design is based on unified modeling language. In this way we can conclude that these are the basics of Unified Modeling Language in brief.

The above mentioned basics are sufficient for a person to design and model his/her project perfectly. To implement designs using UML , we can use rational rose , star UML and visual paradigm software.

REFERENCES

[1] *Zave, P., The Operational Versus the Conventional Approach to Software Development, Communications of the ACM, 27, 104-118, 1984.*

[2] Roger S Pressmen, "Software Engineering - a Practitioner's Approch" 6th Eddition Mc Graw Hill international Edition, Pearson education, ISBN 007 - 124083 – 7

[3] Grady Booch, James Rumbaugh, Jacobson "Unified Modeling Language User Guide". PE,ISBN 81-7758-372-7

[4] Rational. UML1.1 Notation guide. Rational Software,1997

[5] Waman S Jawdekar, "Software Engineering Principles and Practices" Tata Mc graw hill ISBN 0 -07 - 058371 – 4

[6] Comer, E, "Alternative Software Life Cycle Models", presented at International Conference on Software Engineering 1997.

[7] JJ Kuhl, "Project Lifecycle Models: How They Differ and When to Use Them",2002 www.businessesolutions.com.

[8] Ian Sommerville, "Software Engineering", Addison Wesley, 7th edition, 2004.

[9] Steve Easterbrook, "Software Lifecycles", University of Toronto Department of Computer Science, 2001.

[10] Measuring the function points from the Points of Relationships of UML- Mr.K.Koteswara Rao, Mr.Srinivasan Nagaraj , Mr. Jitender Ahuja

BIOGRAPHIES

M.L.V.Roopa Vani is a final year student of Computer Science & Engineering branch in Dadi Institute of Engineering & Technology.

M. Chandrika Kumari is a final year student of Computer Science & Engineering branch in Dadi Institute of Engineering & Technology.

M.Hari Priya is a final year student of Computer Science & Engineering branch in Dadi Institute of Engineering & Technolo

N.Harika is a final year student of Computer Science & Engineering branch in Dadi Institute of Engineering & Technology.