An Attempt Towards a Formalizing UML Class Diagram Semantics

Bouchaib Falah[1], Mohammed Akour[2], Issar Arab[3], and Yassine M'hanna[4]
[1, 3, 4] School of Science and Engineering
[1, 3, 4] Al Akhawayn University, Ifrane, Morocco
[2]Computer Information Systems, Yarmouk University
[1] b.falah@aui.ma, [2]mohammed.akour@yu.edu.jo, [3] i.arab@aui.ma, [4] y.mhanna@aui.ma

ABSTRACT

The Unified Modeling Langue (UML) is one of the backbones of the software design since it has been widely used in constructing models of systems. Therefore, a clear understanding of this concept is highly recommended for software designers to achieve a common communication with stakeholders. However, UML is known to have some popular drawbacks that were addressed by many pioneers of the software design field. These issues that hinder UML for being widely used are lack of clear semantics, excessive size and complexity, and limited customizability. We will mainly focus on the lack of clear semantics in this paper because of its importance to perform a mapping between diagrammatic models and mathematical logic. Therefore, we will go deeply in establishing a solid relationship between UML and one of the well-known mathematical theories which is set theory.

KEY WORDS: Unified Modeling Language, UML semantics, Class diagram, Set theory

1. INTRODUCTION

Software design and analysis is the second pole of the software engineering life cycle. Many researchers argue that Object-Oriented design is the best approach to follow in modern system development. There are several techniques and approaches that are used to model and design the structure of a software. The most popular one is the UML language.

UML stands for Unified Modeling Language. The Unified Modeling Language (UML) has established itself as an industry standard for describing and designing software systems [10, 4]. In January 1997, the language has been submitted to OMG as a proposal for a standard notation of object-oriented analysis and design techniques [1]. The language was implemented, in 1994, by G. Booch, J. Rumbaugh, and I Jacabson by combining the concepts of OOA/OOD and Object modeling techniques.

The language is defined as a set of description techniques suited for specifying, visualizing, and documenting object-oriented systems [1].
Similar to alternative software engineering methods, the UML language offers a collection of graphical and textual description techniques theoretically supposed to be understood by all software developers. However, when it comes to model an application, the meaning of those description techniques is not clearly defined. This later makes the widely adopted standard notation of UML not exempt from problems, and it is mainly due to the lack of precision in the definition of its semantics [9]. Consequently, each one, when interacting with UML diagrams, can have a different interpretation of the described model.

Researching a consistent and concise formalization of UML semantics has been tackled in multiple research papers in order to finalize the mapping between what is graphical and what is logical. This means that enforcing UML semantics can only be achieved by using mathematical theories. There are various attempts to clearly define semantics for UML but they are not standardized. Therefore, they keep on being as theoretical ideas that are not widely accepted by the software engineering community.

The purpose of the paper is to provide a link between UML and the set theory. The intention is to map logical concepts of set theory with UML diagrams. We think that our approach is appropriate for this study, first because of the heavy and well-defined concepts of the set theory. Second, in object-orientation, a system is composed of a collection of objects interacting between each other. For that reason, set theory is a very appropriate choice.

The rest of the paper is laid as follows: section 2 presents some related works, Section 3 gives overview of UML, section 4 briefly described UML semantic, section 5 demonstrates Set Theory, a proposal towards a semantic groundwork of uml is described inn section 6, Section 7 combined UML semantic and Set Theory and the

mapping between these two concepts, finally Section 8 conclude the paper.

2. RELATED WORK

Since the first release of UML, researches on the field have done many attempts in order to formalize and define clear semantics of the UML notations. The first trial was done back by the developers of UML, when they tried to present a meta-model for UML concepts within the language documentation. The meta-model itself was given in UML notation by a class diagram and annotations in prose [2]. The second attempt was released by R. Breu et al. in their paper "*Towards a Formalization of the Unified Modeling Language*" [2]. They precise that their approach to the formal foundation of UML was based on the well-studied and established mathematical theory of streams and stream processing functions [3]. A third approach researched in the same focus was to introduce a general framework for formalizing a subset of UML diagrams in terms of different formal languages based on a homomorphic mapping between metamodels describing UML and the formal language [7]. This framework enabled the authors of the paper to construct of a consistent set of rules for transforming UML diagrams into specifications in the formal language. Those specifications will enable either execution through simulation or analysis through model checking, using existing tools. However, it had some limitations since it was focused on analyzing models for embedded systems. Other researches chose to base their formalization on a different methodology. The process followed by those professionals in the field consisted of formalizing a subset of the UML Metamodel and showing that the class of models generated by the translators based on Computer-Aided Software Engineering (CASE) development methods is within the class of models of the UML Metamodel [8]. Akour et al. [16] suggests the use of natural language processing techniques in order to extract information from the use case specifications and translate these information into a class diagram.

3. UNIFIED MODELING LANGUAGE OVERVIEW

Recently UML is being widely accepted in both industry and academia as a language for Architecture Description [14]. Unified Modeling language was intended to put object oriented design and analysis in one umbrella. Different model languages were developed before UML but the main drawback was their limited coverage of the whole concept of software design and analysis. These languages focused only on analyzing the software but they lack a method for implementing a well formed design of the software. Other languages on the hand, focused on designing rather than analyzing the system under examination. To better understand a system, software engineers needed a method or a model to investigate both the design and analysis of a system. This understanding is better achieved by generating graphical representations of a system. In order to illustrate this aspect, we will give an example that shows the need for a diagrammatic language. We have tried to provide a textual description of a software to various software engineering students and asked them to give their overview of the system. After analyzing the answers, we have found that several interpretations were retrieved from the same textual description. This latter example shows clearly why we need to present the system using diagrams. UML was adopted as a standard to eliminate the misunderstanding and have a common interpretation over the system being developed. In general, there are 13 main diagrams specified by UML. They are divided by two categories: Structure and Behavior Diagram. Structure diagram consists of class diagram, composite structure diagram, component diagram, deployment diagram, object diagram and package diagram. While behavior diagram consists of activity diagram, sequence diagram, communication diagram, interaction overview diagram, timing diagram, use case diagram and state machine diagram [5].

A system model does not need to have the whole collection of UML diagrams. The most popular UML diagrams used in industry are use case diagram, sequence diagram, class diagram and statechart diagram [5]. We have discussed earlier that the most well-known issue for UML is its lack of clear semantics. However, we need to give what does semantics mean in the context of UML in general. Before that, we will give a detailed explanation of the term semantics.

4. SEMANTICS

Semantics is the study of the relation between form and meaning. There is a big difference concerning the definition of semantics from software engineering and other scientific fields like Mathematics or logic. A software engineer or developer uses semantics when tackling the behavior of the system being developed. Therefore, they map semantics with behavior which is a slight deviation of the real definition of semantics. In Mathematics and logic, semantics refers to the meaning of a notation and this regardless of whether this notation deals with structure or behavior [6].

Since UML was a new language that models systems, it needed to follow three steps to define a meaning for it. In general, a meaning is defined for a new language by following three main steps [6]:

- Define precisely the syntax of the new language, which characterizes all the possible expressions of the language [6].
- Identify a well understood language, herein called the semantics language [6].
- Define a mapping from expressions in the syntax of the new language to the semantics language [6].

What is special about UML semantics? Defining semantics is very interesting in terms of UML since the latter has interesting characteristics which will be discussed briefly in this section. The first characteristic of UML is its visual representation of a system being developed. Second, UML is not for execution, but for modeling, thus incorporating abstraction and under-specification techniques [6]. Furthermore, UML is combined of a set of partially overlapping sub notations [6].

UML is essential to capture paramount properties of a system by disregarding features that have less impact on the implementation. As a result, UML can lead to have more than one possible interpretation hence more than one possible implementation.

Therefore, semantics should be carefully defined to reflect explicit specifications of the model. Architecture must be clearly understood if others are to build systems from it, analyze it, maintain it, and learn from it [15]. The next section will discuss the theory being used to formalize the UML semantics.

5. SET THEORY OVERVIEW

In our methodology to tackle the issue of semantics for the UML, we will base our approach on the well-studied field of discrete math, precisely the set theory [4]. Set theory represents the field of mathematical logic that deals with sets and their properties. A set is informally defined as a collection of objects that can be represented and grouped into sets. However, not all sets can implement the power of set theory since it is mostly applied to objects relevant to mathematics. Hence, all UML concepts will be mapped to discrete mathematical notations. In the next subsections, we are going to give a brief overview of the set theory and its components.

5.1 Basic Notations

A set is defined as a blend of well-distinguished items taken from our visual or mental experience into one substance. The items are known as the elements of the set [12]. Let S be a set, and x an item. We say that x is an element of S and we denote it by the expression $x \in S$.

- $S = \{A, B, C\}$ denotes the set whose elements are A, B and C.
- A is an element of the set S is denoted by the expression $A \in S$.
- D is *not* an element of set S is denoted by the expression $D \notin S$.
- The set of all elements from S such that the function F is true is denoted by the expression $\{x \in S \mid F(x)\}$.

E.g., $T=\{x \in N \mid 5<x<15\}$

Empty set

- The empty set represents the unique set that contains no elements and it is denoted by $\varnothing$.

- Let S be a set of elements and $\varnothing$ be an empty set.

For all sets S,

1. $\varnothing \subseteq S$
2. $S \cup \varnothing = S$
3. $S \cap \varnothing = \varnothing$
4. $S \cap S^c = \varnothing$

Disjoint Set

- Let A and B be two sets. We say that A and B are disjoint if and only if $A \cap B = \varnothing$.
- mutually disjoint Sets are all the sets $S_1, S_2, \ldots, S_n$ such that:

for all i,j = 1,2,…, n

$S_i \cap S_j = \varnothing$ whenever $i \neq j$.

Partitions

- A partition is a collection of nonempty sets $\{S_1, S_2, \ldots, S_n\}$ of a set S

If and only if

1. $S = S_1 \cup S_2 \cup \ldots \cup S_n$
2. $S_1, S_2, \ldots, S_n$ are mutually disjoint.

Power set

- Let S be a set, the power set of S is the set of all subsets of S and it is denoted by P(S).

E.g.,: Let S={a,b}

P (S) = {∅, {a}, {b}, {a,b}}

5.2 Relations Between Sets

Let A and B be two sets.

- We say that A is a subset of B and denote it by the expression $A \subseteq B$ if and only if every element of A is also an element of B.

Symbolically,

- $A \subseteq B \Leftrightarrow \forall x$, if $x \in A$ then $x \in B$.
- $A \not\subset B \Leftrightarrow \exists x$ such that $x \in A$ and $x \notin B$.

- We say that A is A equals B and denote it by the expression A = B if and only if every element of A is in B and every element of B is in A.

Symbolically,

- $A=B \Leftrightarrow A \subseteq B$ and $B \subseteq A$.

5.3 Operations on Sets

Let's assume two sets A and B that are subsets of a bigger set S [11]:

.

- The union of A and B is denoted by the expression $A \cup B$ and it can be expressed as:
  $\{x \in S \mid x \in A \text{ or } x \in B\}$
- The intersection of A and B is denoted by the expression $A \cap B$ and it can be expressed as:
  $\{x \in S \mid x \in A \text{ and } x \in B\}$
- The difference of B minus A is denoted by the expression B−A and it can be expressed as:
  $\{x \in S \mid x \in B \text{ and } x \notin A\}$
- The complement of A is denoted by the expression A^c and it can be expressed as:
  $\{x \in S \mid x \notin A\}$

5.4 Set Properties

The set theory incorporates many operations and apply properties on sets. There exist 6 properties related to sets [11]:

- Commutative Laws:
  $(a)\ A \cap B = B \cap A$
  $(b)\ A \cup B = B \cup A$
- Associative Laws:
  $(a)\ (A \cap B) \cap C = A \cap (B \cap C)$
  $(b)\ (A \cup B) \cup C = A \cup (B \cup C)$
- Distributive Laws:
  $(a)\ A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
  $(b)\ A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
- Double Complement Law:
  $(A^c)^c = A$
- De Morgan's Laws:
  $(a)\ (A \cap B)^c = A^c \cup B^c$
  $(b)\ (A \cup B)^c = A^c \cap B^c$
- Absorption Laws:
  $(a)\ A \cup (A \cap B) = A$
  $(b)\ A \cap (A \cup B) = A$

6. A PROPOSAL TOWARDS A SEMANTIC GROUNDWORK OF UML

In this section we present a proposal towards a semantic basis of UML. We will describe our approach to a formalization by using the power of a mathematical logical system model and mapping the UML concepts to that mathematical model.

6.1 Formalization Roadmap

We have stated in the introduction why a clear semantics of UML description techniques is needed. It is useful for both developers and designers to have a precise and concise semantics in order to reach the best product by the end of the software lifecycle.

Each UML description technique has its main components combined to produce a UML diagram. As an example, the class diagram is composed of classes, whereas the state diagram is composed of objects, states and events, the sequence diagram is composed of objects and messages, and the collaboration diagram is composed of objects and messages as it is shown in figure 1.

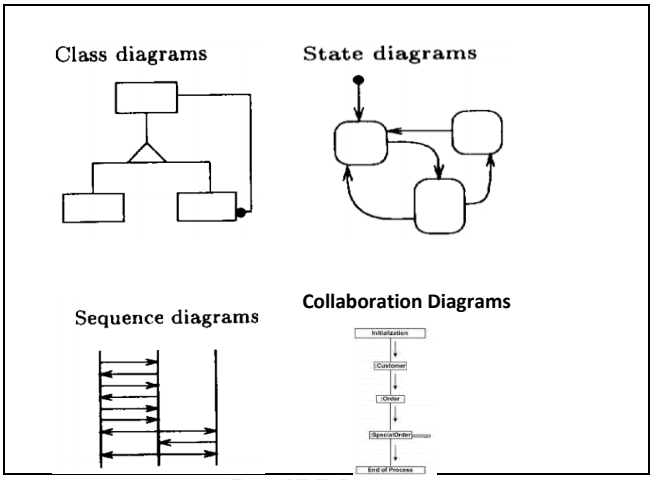

Fig.1: UML Diagrams

Since the formalization of UML is complex enough, the solution to reduce this complexity is to create a middle layer between the diagrams and the set theory. This new layer will be the Universal Set. A set that comprises all the elements of the diagrams such as classes, objects, and messages.

6.2 Universal Set

The Universal Set described below is adapted precisely for the purpose of a formalization of the UML. This set includes all relevant aspects of UML such as objects, classes, messages, events, etc. Once a universal set is constructed, we can derive from it the power set which will include all the subsets of the universal set. Hence, we will have the set of classes, objects, messages … as an element of the power set, see figure 2.

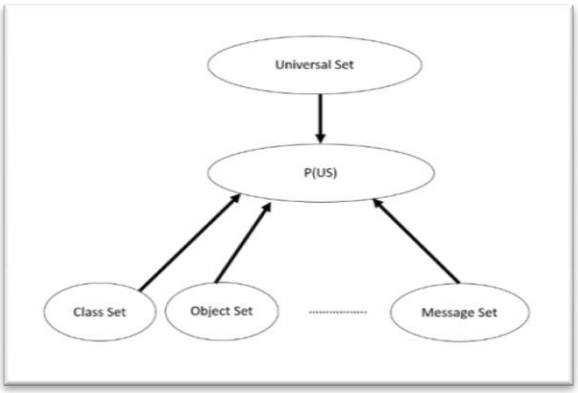

Fig. 2: Layered approach toward Formalization

7. UML AND SET THEORY

In our approach we refer to the Universal Set of a system as US. In our model, we refer to the set of objects by the notion OBJ which is a subset of the Universal Set and it is denoted by the expression:

$$\text{OBJ} \subseteq \text{US}$$

We refer to an object by the notation *obj* with a unique identifier *id*:

$$obj_{id} \in OBJ$$

A class is a set of objects. Also, a class is defined by a set of attributes and methods. In our approach we represent a class by c and the set of all classes by C:

$$\text{C} \subseteq \text{US}$$
$$c_{id} \in C$$

We refer to the set of attributes as ATR where each attribute has a unique Cid (class id) and a name *atr*:

$$\text{ATR} \subseteq \text{US}$$
$$atr_{Cid} \in ATR$$

We refer to the set of attributes as METH where each attribute has a unique Cid (class id) and a name *meth()*:

$$\text{METH} \subseteq \text{US}$$

$$\text{meth}()_{Cid} \in METH$$

Let's suppose that we have a class named Calculator with an id Cid=1. The class has 3 attribute: operand_1, operand_2, and operator. Also, the class has 4 methods add(), subtract(), multiply(), divide(). Then, we can define the class Calculator as follows:

$$\text{Calculator} = \{operand\_1_1,\ operand\_2_1,\ and\ operator_1,\ add()_1,\ subtract()_1,\ multiply()_1,\ divide()_1\}$$

We refer to the set of relationships as RELATIONSIP where each state has a unique id:

$$\text{RELATIONSIP} \subseteq \text{US}$$

There are three types of relationships, inheritance which will be referred to by INH, dependency DEP, and Association that contains Aggregation (AGG) and Composition (COMP).

$$INH \in \text{RELATIONSIP}$$
$$DEP \in \text{RELATIONSIP}$$
$$AGG \in \text{RELATIONSIP}$$
$$COMP \in \text{RELATIONSIP}$$

7.1 Class Diagram in Set Theory

Now that we know how to represent a set of classes using sets we can write a diagram using sets.

Let's assume the US with the following classes:

c_1: Animal
c_2: Mammal
c_3: Oviparous
c_4: Biped
c_5: Quadruped
c_6: Reptile
c_7: Bird
INH: inheritance

$$\text{US} = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, INH\}$$

Let's name the diagram of mammals as A, where A is the set of classes that form the diagram, and the diagram of oviparous as B, where B is the set of classes that form the diagram.

Hence A and B can be written as the following:

$$\text{A}= \{c_1, c_2, \{c_4, INH, c_2\}, \{c_5, INH, c_2\}, \{c_2, INH, c_1\}\}$$

$$\text{B}= \{c_1, c_3, \{c_6, INH, c_3\}, \{c_7, INH, c_3\}, \{c_3, INH, c_1\}\}$$

Figure 3 and 4 bellow represent the class diagrams of the set A and B, respectively.

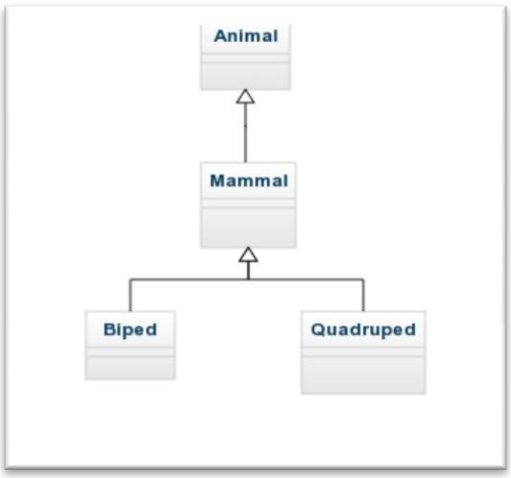

Fig. 3: Class Diagram of the set A

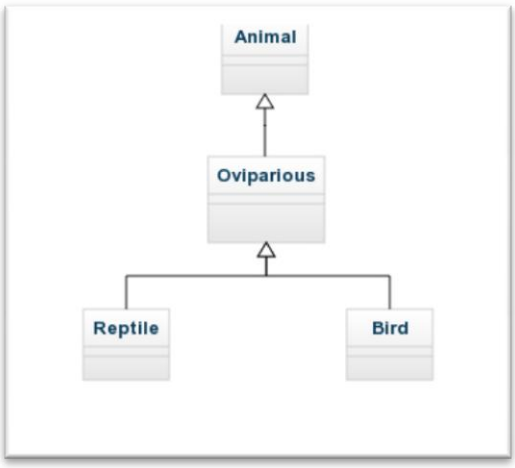

Fig. 4: Class Diagram of the set B

7.2 Operations on Class Sets

Operation1: A ∪ B diagram contains every single diagram component from description technique A and B. Similar components are blended into one and their meta association elements are linked to the blended diagram components. Figure 5 is a representation of the A ∪ B class diagram.

$$\{c \in S \mid c \in A \text{ or } c \in B\}$$

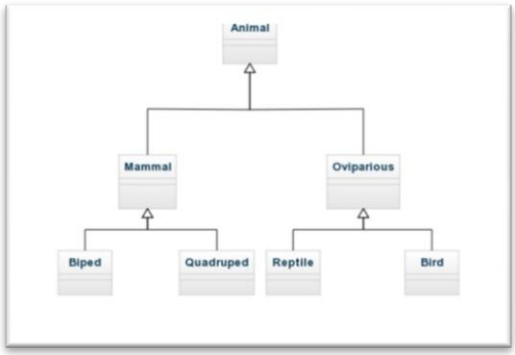

Fig.5: Class Diagram of the set A ∪ B

Operation2: A ∩ B diagram contains only similar diagram components from description technique A and B. Figure 6 is a representation of the A ∩ B class diagram.

$$\{c \in S \mid c \in A \text{ and } c \in B\}$$

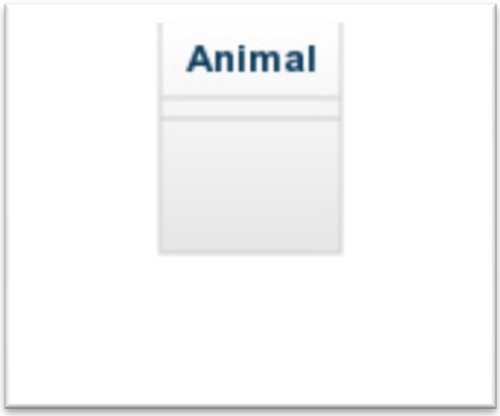

Fig. 6: Class Diagram of the set A ∩ B

Operation3: A - B diagram contains all diagram components from description technique A but not the common ones between A and B. Figure 7 is a representation of the A - B class diagram.

$$\{c \in S \mid c \in A \text{ and } c \notin B\}$$

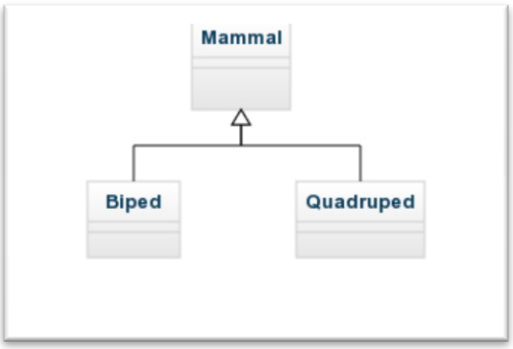

Fig. 7: Class Diagram of the set A - B

Operation4: A^c diagram contains all components that are not in the description technique A. In other words, it is similar to US – A. Figure 8 is a representation of the A^c class diagram.

$$\{c \in S \mid c \notin A\}$$

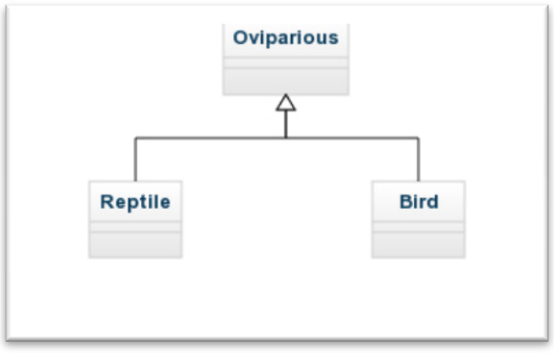

Fig. 8: Class Diagram of the set A^c

7.3 Extending the Model

The principals described in this paper are direct implementation of set theory on set of components of a diagram. The approach can be expanded further for other description techniques. Moreover, the approach can be used to draw mathematical inferences and build a standardized modeling of all description techniques. More than that, the mathematical approach can be used to precise clear semantic based on the set theory knowledge.

8. CONCLUSION AND FUTURE WORKS

UML is the de facto modeling language used by software developers during the initial stages of software development such as requirements engineering, architectural and detailed design [13]. The UML language offers a collection of graphical and textual description techniques theoretically supposed to be understood by all software developers. However, when it comes to model an application, the meaning of those description techniques is not clearly defined. Therefore, each one can have a different interpretation of the described model. As a solution to deal with the issue, we suggest a formal semantics defined based on the mathematical set theory.

Applying more notations of set theory will be our next effort towards establishing a well-formed UML semantics. We are believing that an automated-based too will be very effective in analyzing the different subsets of the universal sets. For example, the tool will take produced subsets of the power set and generate various diagrams which will be further analyzed. This will lead to a very consistent and concise graphical representation of the system being developed.

9. REFERENCES

[1] G. Booch, J. Rumbaugh, and I. Jacobson. *"The Unified Modeling Language for Object-Oriented Development"*, Version 1.0, 1996.

[2] R. Breu, U. Hinkel, C. Hofmann, C. Klein, B. Paech, B. Rumpe, and V. Thurner. *"Towards a Formalization of the Unified Modeling Language."*

[3] M.Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T. F. Gritzner, and R. Weber. "*The Design of Distributed Systems"* – An Introduction to FOCUS – revised version – SFB-Bericht 342/2-2/92 A, Technische Universitat Munchen, January 1993.

[4] Object Management Group: OMG Unified Modeling Language Specification Version 1.5 (2003). On-line at http://www.omg.org/uml

[5] N Ibrahim,Rosziati Ibrahim. *"Semantic Rules of UML Specification"*

[6] Stuart Ken, Andy Evans and Bernhard Rumpe, "*UML Semantics*"

[7] W E. McUmber,B H.C. Cheng et al., *"A General Framework for Formalizing UML with Formal Languages"*

[8] J.Smith, M.Kokar, *"UML Formalization and Transformation"*s

[9] J.M. Bruel, B.Henderson-Sellers et al., *"Improving the UML Metamodel to Rigorously Specify Aggregation and Composition*", Internal Research Report R2I-01-02, June 11th 2001.

[10] P. Selonen, *"Set Operations for the Unified Modeling Language*", Tampere University of Technology, Institute of Software Systems, Tampere, Finland.

[11] W. A. R. Weiss, *"AN INTRODUCTION TO SET THEORY*", October 2ed, 2008.

[12] P. H. Schmitt, *"UML and its Meaning*", 2003.

[13] H. O. Salami, M. A. Ahmed, "*UML Artifacts Reuse: State of the Art*", the International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue, March 1-2, 2013.

[14] *"Software Architecture Description & UML Workshop"* the 7th International Conference on UML Modeling Languages and Applications, October 11-15, 2004, Lisbon, Portugal

[15] J. Ivers, P. Clements, D. Garlan, R. Nord, B. Schmerl, and J. R. O. Silva, *"Documenting Architectural Connectors with UML 2",* the 7th International Conference on UML Modeling Languages and Applications, October 11-15, 2004, Lisbon, Portugal

[16] Mohammed Akour, Bouchaib Falah, Bassim Madi, and Nacir Bouali ,An Effective Approach for Transforming Use Cases Specifications into Class Diagram Adv. Sci. Lett. 22, 2972–2976 (2016)