Formalization of UML class diagram using description logics

Article · June 2010

DOI: 10.1109/ITSIM.2010.5561621

3 authors, including:

Wan Mohd Nasir Wan Kadir
Universiti Teknologi Malaysia
83 PUBLICATIONS **444** CITATIONS
SEE PROFILE

Radziah Mohamad
Universiti Teknologi Malaysia
84 PUBLICATIONS **207** CITATIONS
SEE PROFILE

Formalization of UML Class Diagram Using Description Logics

Lusiana Efrizoni
Teknik Informatika
STMIK-AMIK Riau – Pekanbaru
Jl. Purwodadi Indah Km.10 Panam
Riau, Indonesia
e-mail: lusi_dl@yahoo.co.id

Wan M.N. Wan-Kadir[1] and Radziah Mohamad[2]
Software Engineering Department
Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia
81300 UTM Skudai, Johor, Malaysia
e-mail: wnasir@utm.my[1], radziahm@utm.my[2]

Abstract—**Unified Modelling Language (UML) is as a standard object-oriented modelling notation that is widely accepted and used in software development industry. In general, the UML notation is informally defined in term of natural language description (English) and Object Constraint Language (OCL) which makes difficult to formally analyzed and error-prone. In this paper, we elucidate the preliminary result on an approach to formally define UML class diagram using logic-based representation formalism. We represent how to define the UML class diagram using Description Logics (DLs).**

Keywords-component; formalization; UML class diagram; logic-based; description logic

I. Introduction and Motivation

The emergence and development of new methodology has a positive impact to software engineering community as an attempt to reduce the complexity and diversity of software development process. Currently, object-oriented (OO) approach is more popular used in software specification due to relative easy to use and understand that is graphically presented. Most of OO approach is informally defined and it has any restrictiveness such as not support the rigorous analysis and its notation is limited in expression. One example of the OO language is Unified Modelling Language, UML in shortly. UML is as a standard OO modelling notation that is widely accepted and used in industry application.

UML is a graphical modelling language with the various type models can be used in specification, visualization and documentation phase in software development process. The UML specification is defined by metamodelling approach might capture the abstract syntax, concrete syntax and semantics of UML. The current work, the abstract syntax of UML is described by UML class diagram meanwhile the concrete syntax and semantic of UML notations define a model's meaning in a combination of the Natural Language (English), and Object Constraint Language (OCL). Either these languages are not sufficiently to state the semantic of UML notation or support to analyze the UML models. These languages always need to be combined with others to state the complex constraints [1]. In additional, most of UML notation is informally defined so it makes error-prone and difficult to analyze formally. This work is to provide the precise semantic i.e. UML class diagram formally.

This paper shows the preliminary result based on an approach to formally define UML class diagram. UML class diagrams are a diagram type that well-established in UML model and play an important role in analysis and complexity of designed system. UML class diagram is used to specify the static structure of the software system which is under study. In order to analyze the UML class diagram, a formal and sound foundation is required. We are interested to explore logic-based representation formalism (i.e Description Logics) to carry out the formalization of UML class diagram. A description Logics (DLs) [2] can support the specification phase in software development. Based on the literature perspective, DLs is well-suited to express the design models completed with a declarative nature. This paper focuses on how to define the UML class diagram using DLs.

The structure of the rest of this paper is as follows. Section 2 discusses the related work. In section 3 of this paper presents a brief review of the description logics. Section 4 descriptions the preliminary result which depicts the basic conceptual framework for formalization and presents the UML class diagram formalization result. Finally, section 5 discusses conclusion and future work.

II. Related Work

UML is de-facto standard for OO modelling [3-7] in industry application to develop a software system. UML is based on graphical notation that is supported by visual syntax and expressiveness which is easy to understand. However, it lacks of precise to define the semantics and hard to convince that the end design is consistent, unambiguous and complete. As a result, several works have been done to remedy this problem. For example, Object Constraint Language (OCL) tried to reduce them [4, 8, 9] but OCL is not enough to do it because OCL constraint cannot be executed. Further more it is difficult to be checked and detected so it gives problem in development and maintenance phase [8]. In addition, the description of UML's syntax can be described using OCL. Unfortunately, it has no complete and precise semantic specification which gives confusion and complicated problems during a model analysis.

Currently, researchers frequently worked to obtain the high expressiveness of UML by formalization. According to Zhihong and Mingtian [10], there are two objective why formalization is needed in UML construction. Firstly, the precise formalization can be use to formally explain the semantics of UML. This objective, can give the similar perception about UML. Secondly, the formalization can produce an automatic reasoning procedure which helps us in checking UML. To reach the first objective, many formalizing techniques have been proposed. Formal Method (FM) may be considered to formalize UML relating to both UML's syntax and UML's semantics or individually. FM is an appropriate technique and a precise way to produce software.

The current FM that was applied in this work such as Z notation [1, 11-13], B notation[14, 15], VDM++ [9], PVS [3, 16], FOL [17] and DLs [10, 18, 19]. However, most of these works is not sufficient to provide a formal basic for the abstract syntax and semantics of notation. The literature perspective mentioned that FM is an approach that difficult to understand and expensive to use [14, 15, 20], but if FM is integrated with OO, it can reduce difficulty in the usage. Using OO modelling allows reducing the semantics gap between the certain problem domain and the evolving structural models. Unfortunately, the existing tools and current work is still limited and ineffective [9]. The various type of UML diagrams used in formalization is class, interaction, state chart, use case, object, and collaboration as well as sequence diagram. In this paper only considers using a class diagram and DLs notations to formalize the UML class diagram.

UML class diagram is choice based on the following reasons:

1) Class Diagrams (CDs) play a prominent role in analysis and design of a complex system. During analysis, classes refer to people, place, event, and things which the system will be about to capture information later. While, implementation specific artefacts like windows, forms, and other object are used to build the system during the design and implementation.

2) CDs are the most important structure diagram and well-established. It is able to describe the information on the interest domain in term of object organized by classes and relationship among them. CD is also a formal way of representing objects that is used and created by business roles.

3) A CD is core of modelling element which is used in behavioural diagram and the abstract of UML syntax is explained by CDs.

Whilst, DLs is considered based on the following reasons:

1) It is suited to provide the knowledge for the static structure of software application.

2) It is based on formal well-studied and understood semantics i.e. descriptive semantics.

3) There are still wide developments in formalization of UML class diagram in which many elements are not defined by DLs yet.

4) Its community is well organized and still at infancy stage.

III. A Brief Review of the Description Logics

Description Logics (DLs) [2] is an approach that based on the mathematical logic which can be used to formalize the knowledge representation. DLs have objectives to characterize and motivation the fields in term of application knowledge to represent a structured into well-understood way. DLs are also able to represent the structural knowledge of an application domain through a knowledge base including a terminology and a world description. The main domain of DLs is interested on concept (unary predicate), roles (binary predicate), and individuals (constant) respectively [2].

The basic formalism of a DL system consists of three components [21]:

1) *Constructors* representing concept and role,

2) *Knowledge base* (KB) consisting of the TBox (*terminology*) and the Abox (*world description*). The Tbox presents the vocabulary of an application domain, while the Abox includes *assertions* about named individuals in terms of this vocabulary,

3) *Inferences* as reasoning mechanisms of Tbox and Abox.

A. Notational Conventions

The basic formalisms of DLs are formed based on the following components such as: the formalism for describing concepts or the description language, define both terminological (TBox) and assertion (ABox) formalism, and the reasoning formalism. Elementary descriptions of description language are atomic concepts and atomic roles. Concept constructors can be applied to build the complex descriptions. In abstract notation, A and B are used to define *atomic concept;* C and D for concept description; R and S are assumed as roles and for functional roles (feature attribute) the letter f, g.

In defining the concept name by DLs, it is started with an uppercase letter and then followed by the lowercase letter (e.g., Human, Male). Roles name (also functional ones) starts with a lowercase letter (e.g., hasChild, marriedTo) and individual names are all uppercase (e.g., CHARLES, MARY). One example of description language is $\mathcal{AL}$ ($Attribute\ Language$) and others are extension of $\mathcal{AL}$. the concept descriptions of $\mathcal{AL}$ are formed based on the following syntax rules:

$$
\begin{aligned}
C, D \rightarrow\ & A \mid \text{; Atomic concept} \\
& \top \mid \text{; Universal concept} \\
& \bot \mid \text{; Bottom concept} \\
& \neg A \mid \text{; Atomic negation} \\
& C \sqcap D \mid \text{; Intersection} \\
& \forall R.C \mid \text{; Value restriction} \\
& \exists R.\top \text{; Limited existential quantification}
\end{aligned}
$$

For Example, if we suppose that $Female$ and $Mother$ are atomic concepts and $hasChild$ is atomic role. Using the universal concept ($\top$), we can form the concept $Female \sqcap \exists hasChild.\top$ and $Mother \sqcap \forall hasChild.Female$. These concepts describe that Female has a child and mother all of whose children are female. Using the bottom concept ($\bot$), we can form the concept a mother without a child by $Mother \sqcap \forall hasChild.\bot$.

Beside description language, terminology is also as basic formalism of DLs. In most general, terminological axioms have the form: $C \sqsubseteq D$, $(R \sqsubseteq S)$ are called inclusions or $C \equiv D$, $(R \equiv S)$ are called equalities, where C, D are concepts (R, S are roles). The semantics of axioms is defined as an interpretation I satisfies and inclusion $C \sqsubseteq D$ if $C^{I} \subseteq D^{I}$ and it satisfies an quality $C \equiv D$ if $C^{I} = D^{I}$. DLs are able to perform specific kinds of reasoning. The different kinds of reasoning performed by DLs system are defined as logical inferences.

B. Syntax and Semantics of DLs

In this section, we briefly describe the syntax and semantic of the DLs. The concept descriptions are formed according to the following syntax:

TABLE I
SOME SYNTAX OF CONCEPT CONSTRUCTOR

Constructor	Concrete Syntax	Abstract Syntax
Top	TOP	$\top$
Bottom	BOTTOM	$\bot$
Intersection	(and $C_1 \ldots C_n$)	$C_1 \sqcap \ldots \sqcap C_n$
Union	(or $C_1 \ldots C_n$)	$C_1 \sqcup \ldots \sqcup C_n$
Negation	(not C)	$\neg C$
Value restriction	(all R C)	$\forall R.C$
Limited existential quantification	(some R)	$\exists R.\top$
Existential quantification	(some R C)	$\exists R.C$
At-least number restriction	(at-least n R)	$\geq n\,R$
At-most number restriction	(at-most n R)	$\leq n\,R$
Exact number restriction	(exactly n R)	$= n\,R$
Qualified at-least restriction	(at-least n R C)	$\geq n\,R\,C$
Qualified at-most restriction	(at-most n R C)	$\leq n\,R\,C$
Qualified exact restriction	(exactly n R C)	$= n\,R\,C$
Same-as, agreement	(same-as u_1 u_2)	$u_1 = u_2$
Role-value-map	(subset R_1 R_2)	$R_1 \sqsubseteq R_2$
Role fillers	(fillers R $I_1 \ldots I_n$)	$\exists R.I_1 \sqcap \ldots \sqcap \exists R.I_n$
One-of	(one-of $I_1 .. I_n$)	$I_1 \sqcup \ldots \sqcup I_n$

TABLE 2
SOME DLS CONCEPT AND ROLE CONSTRUCTOR

Constructor	Syntax	Semantics
Top	$\top$	Δ^I
Bottom	$\bot$	$\emptyset$
Intersection	$C \sqcap D$	$C^I \cap D^I$
Union	$C \sqcup D$	$C^I \cup D^I$
Negation	$\neg C$	Δ^I / C^I
Value restriction	$\forall R.C$	$\{a \in \Delta^I \mid \forall b.\,(a,b) \in R^I \rightarrow b \in C^I\}$
Existential quant.	$\exists R.C$	$\{a \in \Delta^I \mid \exists b.\,(a,b) \in R^I \wedge b \in C^I\}$
Unqualified number restriction	$\geq n\,R$ $\leq n\,R$ $= n\,R$	$\{a \in \Delta^I \mid b \in \Delta^I \mid (a,b) \in R^I\} \mid\geq n\}$ $\{a \in \Delta^I \mid b \in \Delta^I \mid (a,b) \in R^I\} \mid\leq n\}$ $\{a \in \Delta^I \mid b \in \Delta^I \mid (a,b) \in R^I\} \mid= n\}$
Qualified number restriction	$\geq n\,R\,C$ $\leq n\,R\,C$ $= n\,R\,C$	$\{a \in \Delta^I \mid b \in \Delta^I \mid (a,b) \in R^I \wedge b \in C^I\} \mid\geq n\}$ $\{a \in \Delta^I \mid b \in \Delta^I \mid (a,b) \in R^I \wedge b \in C^I\} \mid\leq n\}$ $\{a \in \Delta^I \mid b \in \Delta^I \mid (a,b) \in R^I \wedge b \in C^I\} \mid= n\}$
Role-value-map	$R \subseteq S$ $R = S$	$\{a \in \Delta^I \mid \forall b.\,(a,b) \in R^I \rightarrow (a,b) \in S^I\}$ $\{a \in \Delta^I \mid \forall b.\,(a,b) \in R^I \leftrightarrow (a,b) \in S^I\}$
Agreement and disagreement	$u_1 = u_2$ $u_1 \neq u_2$	$\{a \in \Delta^I \mid \exists b \in \Delta^I.\,u_1^I(a) = b = u_1^I(a)\}$ $\{a \in \Delta^I \mid \exists b_1.b_2 \in \Delta^I.\,u_1^I(a) = b_1 \neq b_2 = u_1^I(a)\}$
Role name	R	R^I
Inverse	R^-	$\{(b,a) \in \Delta^I \times \Delta^I \mid (a,b) \in R^I\}$
Composition	$R \circ S$	$\{(a,c) \mid \exists b.\,(a,b) \in R^I \wedge (b,c) \in S^I\}$
Role Conj.	$R_1 \sqcap R_2$	$R_1^I \cap R_2^I$
Role Hierch.	$R_1 \sqsubseteq R_2$	$R_1^I \subseteq R_2^I$

IV. PRELIMINARIES

In this section, the proposed approach is presented. Section A presents the formalization framework as foundation how to formalization must be done. Section B is brief introduction and about UML Class diagram. Element of UML class diagram will be shown in section C. Finally, the application of the proposed approach will be represented by a simple example.

A. The Formalization Framework

The proposed approach consists of a conceptual framework and its supported theoretical foundation. In general speaking, a framework is defined as a basic foundation of conceptual structure that provides a guideline in constructing of something that develops the defined structure into something useful.

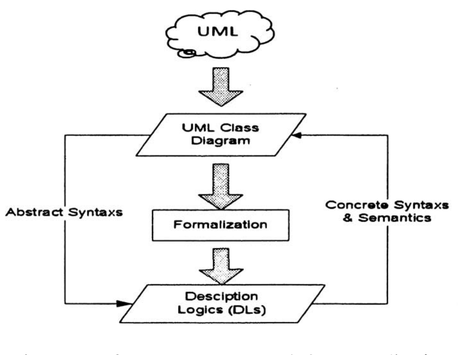

Figure 1. A Conceptual Framework for Formalization

Figure 1 shows a conceptual framework formalization that become as foundation in this work how to formalize the UML class diagram using one of family of logic-based approach i.e. Description Logics (DLs). DLs will be used to define the syntax and semantic of elements of UML class diagram which refers to table 1 and 2 trough the formalization process. The formalization is expand based on the previous work [7, 19] and its result helps to make the same perception e.g. consistence understanding of elements of UML class diagram.

B. UML Class Diagram

This work concentrates on UML class diagram for the conceptual perspective. The formalization of UML class diagram is based on formal approach taking advantages from both a conceptual modelling diagram (UML) and DLs. Logic in formalization is basically considered as a set of reason and consist of: a given precise semantics, formal verification, and virtually unlimited expressiveness.

Class Diagram is a static diagram or model that shows the classes and relationship among classes that remain constant in the system over time. A class diagram depicts classes of the objects under study is completed with the common feature and the relationship among them. The common feature of class diagram is comprise the structural feature (i.e. attribute and association ends) and the behavioural feature (i.e. operation). The following sections will present the elements of UML class diagram as shown in figure 2.

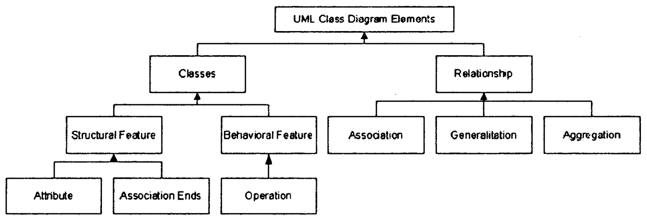

Figure 2. UML Class Diagram metamodel

C. Element of UML Class Diagram Formalization

UML class diagram depicts a structural aspect of the model of a system in graphic. This graphic shows what conceptual 'things' exist in a system and what relationships exist among them. In this section, the Class Diagram (CD) formalization for the structure syntax as below:

1) Classes

Classes in UML model represent a set of objects or its instance in which the objects belong to. A class called instancesis the class forming instantiation or extension of the class, with the common feature (see Figure 2). Properties in UML 2.0 contain two distinct notations: attributes and association ends. A property owned by a class is attribute, whilst a property owned by an association is known association ends.

A class is a main block of UML class diagram which is used to store and manage information. Each class is graphically depicted using three compartments with class' name at the top, attributes in the middle, and operation at the bottom (see figure 3). An UML class C is represented by atomic concept C. The above description describes that a class is generally composed a set of attributes and operations denotes simplify as $C = (C_n, a_i, a_j, O_i,)$, where:

- C_n denotes class name, the class name has to be unique in the whole diagram.
- a_i is a set of attribute name of the class, for i=1..n.
- a_j is a set of attribute type of the class, for j=1...n.
- O_i is a set of operations of a class, for i=1..n

A class, we state in DLs assertion as:

$$\forall x, y. C(x) \sqsubseteq \forall a(x). T(y) \sqcap P(x)$$

Where: $C(x)$: name x of a class; $a(x)$: name x of attribute, $T(y)$: type of attribute, and $P(x)$: name x of operation of a class.

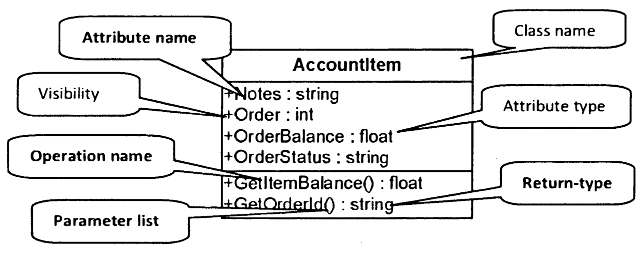

Figure 3. Common Properties ofa Class in UML Class Diagram

2) Attribute

Attributes are presented in the UML 2.0 metamodel by property 'structural feature' that describe the state of an object. Let a be an attribute of a class C with type T, an optional multiplicity $i..j$. This means that an attribute is denoted by a name, possibly followed by a multiplicity and the associated type of the attribute as follows:

Visibility name: type multiplicity

- Visibility indicates whether an attribute is public or private.
- Name denotes the name of the attribute.
- The type of the attribute specifies which kinds of values or object an attribute can contain.
- The multiplicity of an attribute indicates how kinds of values or object may fill the attribute.

The attributes of a class denote by DLs assertion as (without the definition of visibility):

$$\forall x, y. C(x) \sqsubseteq \forall a(x). T(y)$$

It has attributes name a_x and type T_y. Attribute name a_x must be unique only in the class it belongs to, possibly multiplicity (i.e. implicit or explicit). Implicit multiplicity is assumed to be 1...1 (i.e. the attribute is mandatory and single value), and an explicit multiplicity with a minimal and maximal number of value e.g. 1...*.

Multiplicity is a constraint on the number of instances of one class that may be connected to the other class instance. An assertion for multiplicity $[i..j]$ states that for a associated to each instances of C, at least i and most j instances C'. To state the multiplicity $[i \ldots j]$of a that is:

$$\forall x, y. C(x) \sqsubseteq \forall a(x). T(y) \sqcap \exists a(x) \sqcap (i \leq \{a(x)\} \leq j)$$

Note that, if j is * then the second conjunction can be omitted and if the multiplicity is [0...*] then the whole assertion can be ignoring but if multiplicity is [1...1], the first conjunction can be omitted. For example by refer to Figure 3, it is possible to write:

$$AccountItem \sqsubseteq \forall Notes.string \sqcap \exists Notes \sqcap (\leq 1 Notes)$$

3) Operation

Operation is actions or functions from the object of class to which the operation is associated. The full UML syntax for operation is:

Visibility name (parameter-list): return-type

The name denotes the name of operation and is a string. The parameter-list is the list of parameter of the operation. The return-type is a comma-separated list of return types. If an operation has no parameter(s), the parentheses are still shown but are empty. In generally, operations have names and parameters P ($\forall P$ has a name and a type). An operation of a class can be stated by mean of DLs such:

$$\forall x, y. C(x) \sqsubseteq \forall P_{f(x).y}. R_i(y) \sqcap (i \leq P_{f(x).y} \geq j)$$

for $i, j \in 1 \ldots n$

Where $C(x)$ denotes the name of a class, R is return values or type of result belonging $R_1 \dots R_n$, f_x is the operation's name P_y is y parameter belonging to the classes $P_1 .. P_n$.

For example (based on Figure 3):

$$AccountItem \sqsubseteq \forall P_{GetOrderId()}.string \sqcap (\leq 1 P_{GetOrderId()})$$

4) Assocation and Agregation

A relationship between two or more classes (or a class and itself) in UML class diagram is represented by an association. An association is labelled using a verb phrase A or a role name r that described properties of the association (e.g. attribute, operation and others). In UML, an association is a relation between instances of two or more classes as shown in figure 4(a). An association between two classes is a property of both classes.

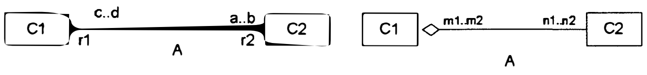

Figure 4. Binary association (a) and aggregation (b) in UML

Figure 5 presents multiplicities of binary association, in DLs will be stated:

$$A \sqsubseteq \exists r_1.C_1 \sqcap \exists r_2.C_2$$
$$\top \sqsubseteq \forall A.C_2 \sqcap \forall A^{-}.C_1$$

To state that each instance of C_1 is connected by A to at least a and at most b with r_1 and each instance of C_2 is connected by A^{-} to at least a and at most b

$$\forall x_1.C_1(x_1) \sqsubseteq (a \leq \{r_1^{-}.A\} \leq b)$$
$$\forall x_2.C_2(x_2) \sqsubseteq (c \leq \{r_2^{-}A^{-}\} \leq d)$$

Example of association:

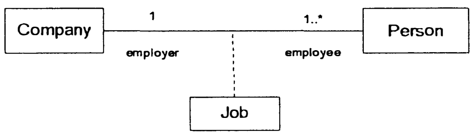

Figure 5. An example of binary association

In DLs can be written as follow:

$$Job \sqsubseteq \exists employer.Company \sqcap \exists employee.Person$$
$$\top \sqsubseteq \forall Job.Person \sqcap \forall Job^{-}.Company$$
$$Person \sqsubseteq (1 \leq employee^{-}.Job^{-} \leq 1)$$
$$Company \sqsubseteq (\geq 1 employer^{-}.Job)$$

n-ary association A among classes $C_1 \dots C_n$ is a n-ary predicate in FOL. We need to define the component of the predicate must belong to correct classes as:

$$\forall x_1, \dots, x_n.A(x_1, \dots, x_n) \sqsubseteq C_1(x_1) \sqcap, \dots, \sqcap C_n(x_n).$$

An aggregation A bases on the figure 4 (b) without multiplicities, we formalize using DL as follow:

$$\forall x_1.C_1(x_1) \sqsubseteq (\geq 1A)$$
$$\forall x_2.C_2(x_2) \sqsubseteq (1 \leq A^{-} \leq 1)$$

5) Generalization and Inheritance

In UML, generalization depicts that one class is identified as the super-class and the others as subclasses of it. The properties and operation of the super class are also valid for objects of the subclass. Note that every instance of each subclass is also instance of the super class. If an UML class C_2 generalizes a classC_1, we can express this in DL assertion: $C_1 \sqsubset C_2$ or $C_1 \sqsubseteq C_2$ means every properties of C_2 are also the properties of C_1 but $C_1 \quad C_2$. Generalization in UML can be grouped into a class hierarchy, as shown in figure 6.

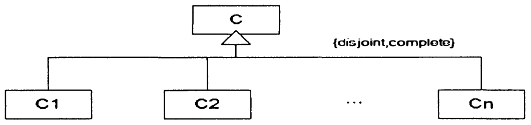

Figure 6. A class hierarchy in UML

A Class hierarchy in UML can be expressed in FOL assertion:

$$\forall x.C_i(x) \sqsubseteq C(x) \quad for\ i \in \{1..n\}$$
$$C \sqsubseteq C_1 \sqcup C_2 \sqcup \dots \sqcup C_n$$

6) Constraint

Constraint can be used to add information (i.e. disjoint and complete) in UML' class hierarchy as additional properties of a class, between each child classes and parent class. Disjointness is a condition where the different subclasses cannot have common instance which is stated in DLs assertion as follows:

$$\forall x.C_i(x) \supset \neg C_j(x)\ for\ i,j \in 1 \dots n\ where\ i \neq j$$

Completeness is a condition where each instance of superclass is also instance of at least one of the subclasses. In DLs is formalized as:

$$\forall x.C(x) \sqsubseteq C_1 \sqcup C_2 \sqcup \dots \sqcup C_n$$

7) Composition

Composition is an instance of a class that become a part of instance of another class. It means that sometimes an object is made up of other object. Composition is depicted with a fulfil diamond. In DLs assertion is stated as follow:

$$\forall x.C_2(x) \in C(x) \sqcup \forall x.\neg C(x) \in C_2(x)$$

D. A Simple Example

In this section, a simple example (see Figure 7) is shown to illustrate the whole formalization. In Class Diagram, there are six classes (Student, Undergrade, Graduate, Class, Lecturer, and Course) and one generalization which can be stated in three assertions as follow:

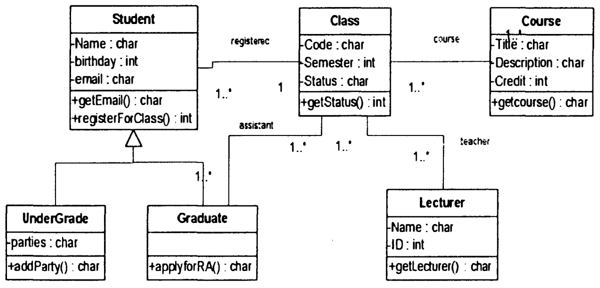

Figure 7. A simple cxample of UML Class Diagram

Definition of properties of Student's class capture attributes and operation as follows:

$$\forall x, y. Student(x) \sqsubseteq \forall name(x). char(y);$$
$$\forall x, y. Student(x) \sqsubseteq \forall birthday(x). int(y);$$
$$\forall x, y. Student(x) \sqsubseteq email(x). char(y);$$
$$Student \sqsubseteq \forall P_{getEmail()}. char \sqcap \left(\leq 1 P_{getEmail()}\right);$$
$$Student \sqsubseteq \forall P_{registerforClass()}. int \sqcap (\leq 1 P_{registerforClass()})$$

The same manner defines attributes and operations for each other class. Association of both classes between student's class and hierarchy class are depicted as follow (the same manner of other classes):

$$\top \sqsubseteq \forall register. Class \sqcap \forall register^{-}. Class$$
$$Class \sqsubseteq (\geq 1 register^{-})$$
$$Student \sqsubseteq (1 \leq register \leq 1)$$
$$\forall x. Student(x) \sqsubseteq \forall Undergrade(x) \sqcup \forall Graduate(x)$$
$$\forall x. Undergraduate(x) \sqsubseteq Student(x)$$

V. CONCLUSION AND FUTURE WORK

This paper has shown the formalization of element of UML class diagram in term of specified formal logic i.e. Description Logics (DLs). Through this formalization, the deductive capabilities of DLs have been exploited as it is shown in the result. The work maps construct a class diagram elements into DLs without include in term of visibility of both attribute and operation. We realize this formalization is still not satisfied because any properties are not defined yet such as dependency relation and the formalization is still manually produced.

In the future, formalization of elements of UML class diagram will be a challenging task. We aim to extend this formalization which is able to capture others properties in UML 2.0 class diagram. The formalization will be done automatically using the current tools and analyzed by real case study in huge UML model.

VI. ACKNOWLEDGMENT

This research is partially funded by Fundamental Research Grant Scheme (FRGS) and Indonesian government's higher education department. The authors would like to convey their utmost appreciation to the anonymous reviewer and respective individuals for their involvement and invaluable feedbacks throughout conducting this research

REFERENCES

[1] S.-K. Kim and D. Carrington, "Formalizing the UML Class Diagram Using Object-Z," in *UML'99*, R. France and B. Rumpe, Eds.: Springers-Verlag Berlin Heidlberg, 1999.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, "The Description Logic Handbook: Theory, Implementation, and Applications," Cambridge University Press, 2003.

[3] L.-Y. Pan, F. Liu, and F.-Y. Ma, "Meta-modeling architecture based method for transforming UML class diagram into PVS specification," *Shanghai Jiaotong Daxue Xuebao/Journal of Shanghai Jiaotong University*, vol. 38, pp. 159-163, 2004.

[4] T. Ali, M. Nauman, and M. Alam, "An accessible formal specification of the UML and OCL meta-model in Isabelle/HOL," Lahore, Pakistan, 2007.

[5] H. Zhou, Z.-Q. Huang, L. Wang, and L. Chen, "Automated metrics for industrial applications software polymorphic interactions," Singapore, Singapore, 2008, pp. 477-482.

[6] F. J. Lucas Martinez and A. Toval Alvarez, "A precise approach for the analysis of the UML models consistency," Klagenfurt, Austria, 2005, pp. 74-84.

[7] D. Berradi, D. Calvanese, and G. D. Giacomo, "Reasoning on UML Class Diagram," *Artificial Intelligence*, vol. 168, pp. 70-118, 2005.

[8] C. Avila, G. Flores, and Y. Cheon, "A library-based approach to translating OCL constraints to JML assertions for runtime checking," Las Vegas, NV, United states, 2008, pp. 403-408.

[9] E. MIT, "DEVELOPING VDM++ OPERATIONS FROM UML DIAGRAMS," in *School of Computing, Science and Engineering*: University of Salford, UK, 2008.

[10] Z. Zhihong and Z. Mingtian, "Some Considerations in Formalizing UML Class Diagrams with Description Logics," in *Proceedings of the 2003 IEEE International Conference on RoboticsJntelligent Systems*, Changsha, China, October 2003, pp. 111-115.

[11] H. Miao, L. Liu, and L. Li, "Formalizing UML Models with Object-Z*," in *ICFEM2002*, 2002, pp. 523-534.

[12] S. Napapak and W. Vatanawood, "Transformation of class diagrams into Z specifications," Las Vegas, NV, United states, 2004, pp. 387-392.

[13] S. Sengupta and S. Bhattacharya, "Formalization of UML Diagrams and Their Consistency Verification —A Z Notation Based Approach," in *ISEC'08*, Hyderabad, India., February 19-22, 2008.

[14] A. Idani, Y. Ledru, and D. Bert, "Derivation of UML class diagrams as static views of formal B developments," Manchester, United kingdom, 2005, pp. 37-51.

[15] A. Idani, Y. Ledru, and D. Bert, "A reverse-engineering approach to understanding B specifications with UML diagrams," Columbia, MD, United states, 2006, pp. 97-106.

[16] D. B. Aredo, "Formalizing UML Class Diagram in PVS," in *Proc. of Workshop on Rigorous Modeling and Analysis with the UML: Challenges and Limitations*, denver, Colorado, USA, 1999.

[17] L. Shan and H. Zhu, "A formal descriptive semantics of UML," Kitayushu-City, Japan, 2008, pp. 375-396.

[18] D. Barardi, D. Calvanese, and G. D. Giacomo, "Reasoning on UML Class Diagram Using Description Logic Based Systems," in *Proc. of the KI2001 workshop on Application of Description Logics 2001, CEUR electronic Workshop Proceedings*, 2001.

[19] A. Cali, D. Clavanese, G. D. Giacomo, and M. Lenzerini, "A Formal Framework for Reasoning on UML Class Diagram," in *Proc. of the 13th Int. Sym. on Methodologies for Intelligent Systems (IS-MIS 2002)*, 2002.

[20] R. Grnmo, F. Srensen, B. Mller-Pedersen, and S. Krogdahl, "Semantics-based weaving of UML sequence diagrams," Zurich, Switzerland, 2008, pp. 122-136.

[21] S. G. H. Tabatabaei, W. M. N. W. Kadir, and S. Ibrahim, "Automatic Discovery and Composition of Semantic Web Services Using AI Planning and Web Service Modeling Ontology," *International Journal of Web Services Practices*, vol. 4, pp. 1-10, 2009.