American Journal of Engineering and Applied Sciences, 2012, 5 (3), 251-260
ISSN: 1941-7020

doi:10.3844/ajeassp.2012.251.260 Published Online 5 (3) 2012 (http://www.thescipub.com/ajeas.toc)

Ontological Framework for Object-Oriented Analysis and Design

Prabodha Tilakaratna and Jayantha Rajapakse

School of Information Technology, Faculty of Information Technology,
Monash University Sunway Campus, Jalan Lagoon Selatan,
Bandar Sunway, 46150, Selangor Darul Ehsan, Malaysia

Received 2012-03-12, Revised 2012-07-31; Accepted 2012-09-14

ABSTRACT

Regardless of the large number of Object-Oriented (OO) modeling languages currently being used in the Information Systems (IS) modeling process, unavailability of an OO modeling language that can be used in both the analysis and design phases disintegrates the two phases. The problem is, such disintegration can lead to a high level of missing information in the real world system from the analysis phase to the design phase. The approach of this study is to propose a framework to produce design phase models from analysis phase models using ontology based Unified Modeling Language (UML), thereby integrating the two phases. The results obtained from the porposed framework involve: A consructed language which can be used in generating the analysis phase scripts; and the development of script files based on the UML constructs at the analysis and design phases to automatically generate the UML scripts for those two phases. Since this study is a part of an ongoing research study, it can be concluded that, at the end of this study (1) both analysis and design phases would be able to integrate using a common OO modeling language (2) the manual work involved in the current analysis and design modeling would be reduced (3) the complexities and difficulties faced by the modelers (By modelers we mean the analysts and designers who are doing the analysis and design phase modeling) in using UML modeling tools would be reduced.

Keywords: Conceptual Model, System Model, Ontology, Constructed Language, XMI Format, Unified Modeling Language (UML), Object-Oriented (OO), Information Systems (IS)

1. INTRODUCTION

According to Wand and Weber (1988), IS are not just representations of real world systems. They represent how the human beings perceive the real world systems. Human perceptions regarding the real world characteristics are identified, abstracted and modeled as conceptual and system models during the analysis and design phases. Primary objective of these two phases comprise making all the captured information readily available for the subsequent IS development activities with no missing information (Kim *et al*., 2008; Mishra and Lohani, 2007). Any transformation with missing information between either human perception of real world system and conceptual model, or conceptual model and system model will result an inaccurate final outcome. Hence, the final IS will not be an accurate representation of the real world system. Thus, faultless modeling plays a significant role in IS development. Nevertheless, IS projects do not use proper modeling during the analysis and design phases due to various reasons and most trivial of them are stated below:

- No common OO modeling language exist for both analysis and design phases modeling (Evermann and Wand, 2009). To be used in both the phases in a disciplined way, an OO language should be able to model both real world characteristics (conceptual modeling) as well as the IS characteristics (system modeling) seamlessly. Many OO modeling languages do not have both these capabilities together. Thus,

Corresponding Author: Prabodha Tilakaratna, School of Information Technology, Monash University Sunway Campus, Jalan Lagoon Selatan, Bandar Sunway, 46150, Selangor Darul Ehsan, Malaysia Tel: (+603) 5514 6071

different modeling languages need to be used for each phase thereby disintegrating these phases
- So far, considerable portion of analysis and design modeling processes are largely manual (Overmyer *et al*., 2001). Primary tools used for identification, abstraction and modeling the conceptual and system models are pencil and paper, with the results being transferred to a modeling tool after the modeling is largely completed
- Various modeling tools are available to make the IS modeling process easier. But most of them are complex and less user friendly to be used. Moreover they do not provide adequate helping facilities for the modelers regarding the functionalities of those tools. Kuhrmann (2011) declares that complexity and less user friendliness associated with the modeling tools are major problems in conceptual and system modeling

Availability of a proper framework or tool which mitigates the above problems may encourage the modelers to do their job well. Currently up to our knowledge, there is no such framework or tool available. Having observed the above problems, this study proposes an ontological framework as the solution, that uses UML as the OO modeling language.

The constructs (By constructs we mean concepts and core guidelines that are used to form a language or a domain) of the existing OO modelling languages primarily developed for system modeling and are not capable of modeling the characteristics of real world systems seamlessly (Evermann and Wand, 2005a). Since the analysis phase more concern on real world systems, Evermann and Wand (2005b) suggested adding the real world system characteristics for the constructs of generic UML using ontological approach. Consequently this will create a new version of UML with new ontological UML constructs, which can be used for conceptual modeling. Ultimately it will help the use of UML for both analysis and design phases in a disciplined way.

Use of a common OO modeling language (with two language versions) will preserve the real world characteristics during the transformation process from the human perception of real world system to conceptual model. Besides, a set of UML based transformation rules from-analysis-to-design can be defined when transferring the conceptual model to the system model. This research project expects to define such transformation rules to generate system models from the corresponding conceptual models with no human involvement thereby reducing the involved manual work. This is an ongoing research study. Thus, during this study the entire work to be covered by the research study will be presented in brief with necessary real life examples. Further experiments and empirical suited regarding this ongoing study will be covered in the future research activities.

1.1. Related Work

The accuracy of the final IS to be developed depends on how well it is modeled during the analysis and design phases. Erroneous transformation of information in either of the two phases will result an erroneous representation of the real world system at the implementation of the IS (Kim *et al*., 2008). Thus, the modeling plays a trivial role in IS development.

Nevertheless, the OO modeling process is largely manual and difficult. Normally modelers start modeling by identifying the characteristics of real world systems as perceived by human beings. Those characteristics will be transformed into conceptual models and, during this stage usually pencil and paper are used. Same manual procedure is being repeated at the design phase. Doing the modeling manually is not an easy task. The reason is that, OO modeling languages such as UML have grown quite large and currently covers about 250 modeling classes that are highly interrelated (Silingas and Butleris, 2009). Favre (2003) have evaluated UML using a quality framework and identified that UML is one of the most complex modeling approach.

Manual work and the complexity involve in current modeling practices are said to be reduced by modeling tools. But the user friendliness and documents support regarding the functionalities provided by many OO modeling tools are not adequate enough. Nguyen and Chun (2006) conducted a research study with six modeling tools and identified that those tools currently provide little assistance in managing the associations with the models and hence provide little support for modeling.

As aforementioned, although modelers tend to use OO modeling languages in both analysis and design phases, those languages are developed to be used only in system modeling. UML is one such standard OO modeling languages, which can be used in IS modeling (Koppe, 2010; Paige *et al*., 2003; Rumbaugh *et al*., 1999). Nevertheless, UML constructs are developed to describe and design the functionalities and characteristics of ISs. Thus, it provides less support in modeling the characteristics of real world systems and hence cannot be used effectively in conceptual modeling.

As a result of the above reasons, even if the OO modeling is important in IS development, it is not carried out in a proper and a disciplined way. Many IS development teams build OO models on whiteboards only during user group meetings to help communicate their understandings of the real world systems (Fowler, 2003). Hence it is important to find solutions to make the OO modeling in a more accurate and disciplined way.

Having observed the aforementioned requirements, Evermann and Wand (2005b) modified the UML to be used with conceptual modeling by defining new UML language constructs from Bunge's ontological concepts

(Bunge, 1977; 1979). A mapping is created between the ontology based real world characteristics and the UML constructs thereby making this new ontology based UML version to be used in conceptual modeling. Thus, UML can be used as a common OO modeling language in both analysis and design phases to integrate those two phases with no missing information. Core objective of the framework proposed in this study comprises defining proper UML based transformation rules to convert conceptual models into system models seamlessly.

Besides, the proposed framework converts the human perception of real world systems into language statements of a specific constructed language defined for this framework. Next, those constructed language statements will be transformed into conceptual models. Constructed language is a language that has been built by a person or a group of people, rather than naturally evolving over time (Gopsill, 1989; McGuigan and Foster, 2011). These languages are being built for various reasons; to ease human communication (e.g. Esperanto-an international auxiliary language), to develop computer programs (e.g., Python-a programming language), to do linguistic experiments. (Oostendorp, 2000). But up to our knowledge, currently no constructed language is built to satisfy the following three requirements together; (1) represents real world characteristics (2) uses normal English language words to represent them in constructed language statements (3) can be mapped with the constructs of UML. Hence the proper definition of a new constructed language for the framework rules will ultimately transforms the real world characteristics into conceptual models in a disciplined way, thereby integrating real world system, conceptual model and system model seamlessly.

The interface of the framework should be designed in a way to take inputs (real world system characteristics) in Natural Language (NL) format and produce constructed language statements. Since NL is easily used by anyone, many researchers try to develop NL based approaches for UML modeling. Tichy and Koerner (2010) use NL processing and semantic technologies to generate UML models from NL inputs. They directly convert the NL input documents into a constructed language which later will be transformed into UML models. Although the approach is similar to ours, the constructed language built by them has a graphical notation which cannot be incorporate with the UML constructs used in our framework. Deeptimahanti and Babar (2009) propose a domain independent UML tool which generates UML models from NL requirements using efficient Natural Language Processing (NLP) tools. They directly convert the NL inputs into XML Metadata Interchange (XMI) files of conceptual models using generic UML. Since the constructs of generic UML does not support conceptual modeling, this framework is also not suitable for our purpose. The interface proposed by Overmyer *et al.* (2001) in their software tool Linguistic Assistance for Domain Analysis (LIDA), is the most applicable interface to be incorporated with our framework. LIDA tool provides an interface for the analysts to feed NL documents, from which the UML models are being generated. Only the required NL words will be captured form the input documents in generating the diagrams. The basic structure of this interface needs to be modified to be suited with the requirements of our framework where it should; (1) allow the analysts to input real world characteristics except the concepts captured form the input document (2) convert all the captured NL words into constructed language statements using the rules specified for that language. With these modifications, LIDA tool interface can be used with our framework.

The study proceeds as follows. Materials and methods of the framework descibes next with the proposed modeling framework. Subsequently, the results obtained from the practical use of the proposed framework is explained. Since this is an ongoing research study, final part of the paper concludes the research study along with a discussion of the future research.

2. MATERIALS AND METHODS OF THE FRAMEWORK

Current modeling process in IS development is depicted in **Fig. 1**. It distinctly shows that both analysis and design phases build models using two different modeling languages. Besides, large portion of analysis and design phases modeling is manual. The UML based ontological framework proposed in this study mitigates above problems thereby enhancing the current IS modeling process. Primary objective of this proposed framework involve integrating both analysis and design phases using a common OO modeling language, UML. Thus, UML must be capable to be used in both analysis and design phases in a disciplined way. The ontology based framework that comprises a new UML version for the analysis phase proposed by Evermann and Wand (2005b) is used in our framework for the creation of conceptual models. Evermann and Wand's framework is specifically chosen because:

- They have selected UML as the OO modeling language for their framework, which is the same used in our framework
- The concept behind their framework well suits in solving the problem of using the OO modeling languages in conceptual modeling in a disciplined way

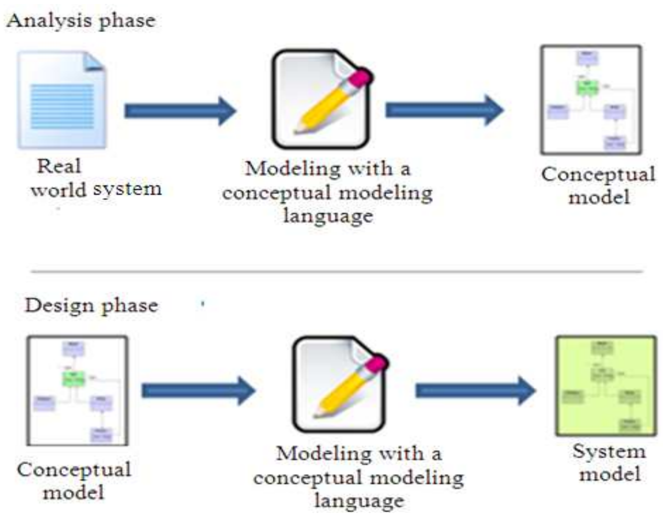

Analysis phase
Real world system
Modeling with a conceptual modeling language
Conceptual model
Design phase
Conceptual model
Modeling with a conceptual modeling language
System model

Fig. 1. Existing modeling process in the current practice. (Before introducing the framework)

- They have initiated incorporating the real world characteristics to UML, which can be used as the starting point of our framework

Having a common OO modeling language for both conceptual and system modeling only, will not precisely integrate analysis and design phases. Along with a common modeling language, the availability of proper transformation rules to transform the conceptual models into system models with no missing information will ensure a proper and accurate integration between analysis and design phases. A precise comparison between the UML constructs defined for each phase will help to build new transformation rules. Nevertheless, analysis phase UML constructs are defined based on Bunge's ontology (Evermann and Wand, 2005b). Thus prior to the comparison, design phase UML constructs also need to be defined using the same ontology.

Currently up to our knowledge, no research study has been carried out to define ontology based UML constructs for the design phase. To define ontology based UML constructs, IS characteristics need to be applied for UML using ontology. Once the UML constructs of both analysis and design phases are defined using Bunge's ontology, definition of the transformation rules can be initiated. Next, those rules need to be mapped to the UML version used in system modeling (i.e., generic UML). Subsequently, if the transformation rules are defined seamlessly, two important conclusions can be made out of that:

- A model created using analysis phase UML version can precisely be transformed into a model of the design phase UML version seamlessly. Which means conceptual models can be transformed into system models without losing any information captured from the real world system
- A precise integration can be created between analysis and design phases using UML. The first conclussion is used as the basis of the proposed framework. Besides, the framework is further enhanced as depicted in **Fig. 2**. Enhanced framework is capable of taking the human perception regarding real world systems into the statements of a constructed language (a language specifically built for this framework) at the analysis phase and to output the conceptual models from them

The framework interface is developed according to the interface of the LIDA tool (Overmyer *et al*., 2001), as specified in the Related work section. The LIDA interface will be modified based on the language rules of a constructed language, thereby converting all the inputs fed to the interface into constructed language statements. Interface development will be done during the implementation of the software solution of the framework, which is out of the scope of this research study. Even though NL is more convenient for the modelers, a constructed language is devised to be used inside the framework except NL. Because, NL words and sentences are too complicated to be mapped with UML constructs. The constructed language is also similar to NL but more specific and abstract than that and easy to be mapped with UML constructs. Further details about this constructed language are given under the Framework in practice section with real world examples.

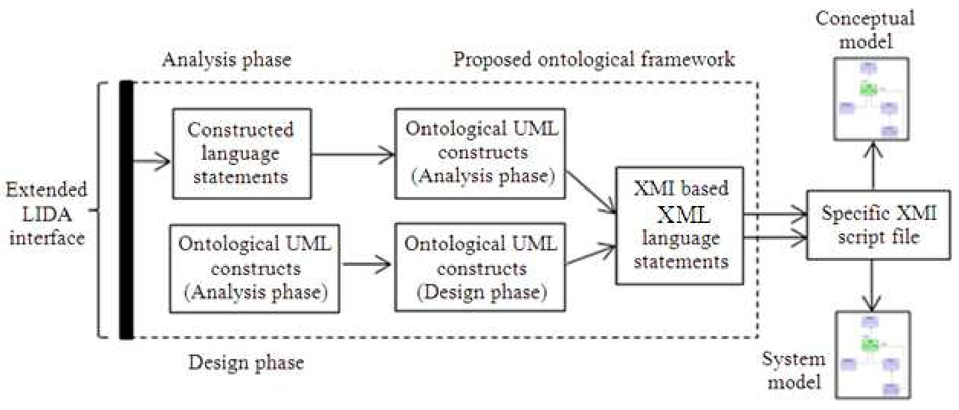

Conceptual model
Analysis phase
Proposed ontological framework
Extended LIDA interface
Constructed language statements
Ontological UML constructs (Analysis phase)
XMI based XML language statements
Specific XMI script file
Ontological UML constructs (Analysis phase)
Ontological UML constructs (Design phase)
Design phase
System model

Fig. 2. Proposed ontological framework

Once the constructed language statements are provided by the interface, those will be converted into ontological UML constructs of conceptual model, using an ontological approach. The ontological UML constructs generated for the conceptual model are used as the input at the design phase, thereby transforming them into ontological UML constructs of system model.

Next, the created ontological UML constructs for both analysis and design phases will be converted into the language statements of another language, Extensible Markup Language (XML). The ontological UML constructs will not directly be converted into UML models. Because, our purpose is to make the final conceptual and system model to be accessed using most of the currently available UML modeling tools. In order to do so, the information of the two models (the UML constructs) needs to be saved using a file format which is compatible with most of the standard UML modeling tools in the current practice. Hence, XMI is selected as the file format which is compatible with most of the standard UML modeling tools. XML is the programming language which is used to create XMI files. Thus, the ontological UML constructs of both analysis and design phases need to be transformed into XML language statements. Later, those XMI statements will create the XMI script files for both constructs sets.

This XMI script files contains transformation information about how to convert the UML language constructs into UML symbols. Since XMI format is compatible with most UML modeling tools, once a script file is imported to a UML tool, it has the ability to transform the symbols into the corresponding UML diagram as specified in the script file. Ultimately conceptual model for the analysis phase and system model for the design phase can be generated with the help of these XMI script files.

3. RESULTS FROM THE PRACTICAL USE OF THE FRAMEWORK

UML possesses fourteen different types of diagrams which can be used in OO modeling. Out of them, only class diagram is considered to be used with the proposed framework. Class diagram is the main diagram type in UML which represent the main objects and their interactions of an IS to be developed. Its purpose is to graphically depict the relationships holding among objects manipulated by a system (Evans, 1998). Specifically this diagram type is used because, UML class diagrams are already enhanced to be used in both conceptual and system modeling.

Using UML class diagrams, some parts of the framework are being investigated. Those parts are described below with a real world scenario. An IT institute wants to develop a system to assign their students to lecturers, based on the subjects they have selected. Properties of a student include student ID, name and address and for a lecturer those are lecturer ID, name and subjects they are teaching. Registration of the student and lecturer records needs to be performed. Furthermore modification and deletion of student records and subject assignment for the lecturers need be done via the system. Each student should be assigned to one lecturer and each lecture can have zero to many students. This scenario is used in the following two sub sections to describe the activities of the framework.

3.1. Building the Constructed Language

This sub section presents information about the use of constructed language for the proposed framework. The modified LIDA interface (Overmyer *et al*., 2001) will be capable of converting the inputs fed into to the interface constructed language statements. Although modelers can feed inputs to the LIDA interface in NL, a specific constructed language is used within the framework except NL, because of the complexity in using NL. NL inputs can contain thousands of words and different complex sentences, which may be difficult to map with the ontological UML constructs (analysis phase). Hence, a specific constructed language will be built for our framework.

As depicted in **Fig. 2**, constructed language statements generated from the modified LIDA interface are converted into the ontological UML constructs of analysis phase. This means, the scope of the constructed language comprises representing the UML constructs defined for the analysis phase with no missing information. Hence, this language uses the UML constructs defined for the analysis phase as the basis in building the language. Thus, the constructed language uses normal English words and sentences but builds in a way to represent the large English vocabulary with a limited but sufficient number of words. Once the constructed language is built to represent all the UML constructs defined for the analysis phase precisely, it can be claimed that the limit of the constructed language is reached up to the required level. Besides, this constructed language organizes the long and complex English sentences in a clearer manner with simple statements. Thus, all the aforementioned factors ultimately help to make the constructed language statements more precise and clearer than the NL inputs (McGuigan and Foster, 2011).

Some important constructed language rules are specified below. Fundamental ontological UML constructs defined for the analysis phase are considered in building those rules.

3.2. Rule 1

A UML class must be declared with a single word and within square brackets.

Evermann and Wand (2005b) defined UML rules for the analysis phase using Bunge (1977; 1979) ontology. According to their definition and according to Bunge's ontology, only physical things in the world are modeled as objects (Evermann and Wand, 2005b). They have found alternative constructs in UML for conceptual items such as 'lecture', 'order' and described those in their research paper in detail. Having observed Evermann and Wand's UML rules for the analysis phase, we have made some amendments to the aforementioned rule as corollary one.

3.3. Corollary 1

In OO conceptual modeling, every class name specified inside a square bracket must represent a physical thing in the real world.

UML classes and attributes need to be defined in constructed language statements as follows.

3.4. Definition 1

Once a class is identified, that can be declared as; class: [class name]. Initially 'class' key word should be declared along with the class name, which is separated by a colon.

3.5. Definition 2

Once an attribute is identified, that can be declared as; attribute: Attribute name [respective class name]. Initially 'attribute' key word should be declared along with the attribute name which is separated by a colon. Attribute name should follow the respective class name.

The above language rules defined for the constructed language can be illustrated using the real world scenario as follows. Two main classes are identified from the scenario, student and lecturer:

Class	:	[Student]; [Lecturer]
Attribute	:	student-ID, name, address, [Student]; lecturer-ID, name, subjects, [Lecturer]

According to the analysis phase UML rules, UML attributes are divided into two parts as attributes of ordinary classes and of association classes. Attributes of an ordinary classes means the attributes possess by the class itself (e.g., colour). Attributes of association classes means the attributes that exist between two or more classes (e.g. employed by) (Evermann and Wand, 2005b). According to this UML rule, following constructed language rules can be defined.

3.6. Rule 2

Attributes of ordinary classes should be declared with 'attributeA' key word and attributes of association classes should be declared with 'arrtibuteO' keyword.

3.7. Corollary 2

Attributes of ordinary classes can only have one corresponding class and attributes of association classes must have more than one class.

The given examples for the two attribute types; colour and employed by, are not given in the real world scenario. But we can add them to the student and lecturer classes as the below given way:

Attribute:	colour [Student]
Attribute:	employedBy [Student],[Lecturer]

During the design phase of OO modeling, methods that each class is responsible of are modeled. Nevertheless, at the analysis phase not the methods but the messages sent and received by each class are modeled. For the ease of transforming the conceptual model into system model, the word method will be used as the constructed language symbol.

3.8. Definition 3

Once a method (message to or from) of a particular class is identified, that can be declared as; method:

method name [respective class name]. Initially 'method' key word should be declared along with method name which is separated by a colon. Method name should follow the respective class name. For example, method (message to or from) can be added to the above defined classes as follows:

method: getRegistered(), makeModify() [Student]; addCourse() [Lecturer]

UML class diagrams possess three different relationship types; association, whole/part relationships and generalization/specialization. Out of those three relationship types, an association relationship can be represented in constructed language as follows. Tour

3.9. Definition 4

Association relationship-Two class names should be written as declared earlier and should be written in between the 'association' key word.

Multiplicity also can be declared with the association relationship declaration. Multiplicity means the term used to describe constraints on the number of participating classes (Bennett *et al*., 2001).

3.10. Corollary 3

Multiplicity of a class should be defined just after the name of that class and it should be declared within brackets.

The two classes identified from the scenario can be related with each other through association relationship. This association relationship can be represented using constructed language statements along with their multiplicity, as follows:

Association: [Student] (0..*) association [Lecturer] (1)

Above constructed language rules are defined for classes, attributes, operations and association relationships. Those rules are defined in a way to map the constructed language statements with the analysis phase UML constructs. But the analysis phase ontology based UML contains hundreds of different constructs and more complex UML rules also remain. Hence the constructed language needs to be built in a way to be correctly mapped with as much as currently available analysis phase ontological UML constructs and UML rules.

3.11. XMI Script Files

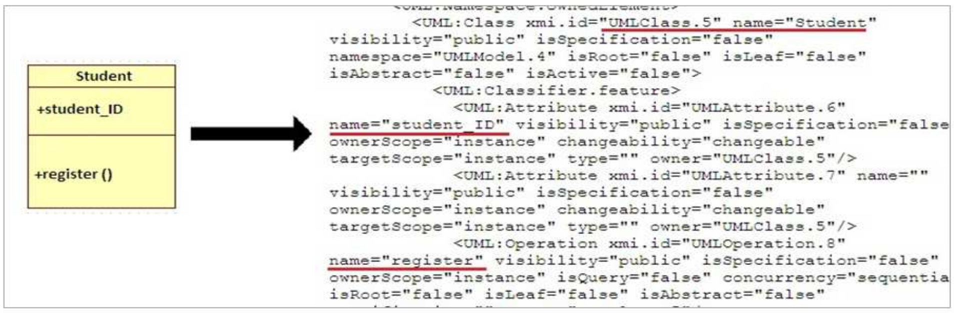
The proposed ontological framework does not directly convert the ontological UML constructs generated for conceptual and system models into UML diagrams. But it converts these ontological UML constructs into a file which stores information about the UML diagram to be generated. The reason behind this is; if the UML constructs directly convert into conceptual and system models, a new file format (e.g., *.doc is the file format for MS Word 2003) which is specific to this framework needs to be defined to save the outputs. Then the output results can only be modified using this framework. Nevertheless, the aim is to develop the framework in a way, where the conceptual and system models output by the framework can be modified using almost all UML modeling tools. Thus, the modelers can either use this framework or their preferred modeling tool to modify the models output from our framework.

In order to achieve this, ontological UML constructs need to be converted into a file format which is compatible with most of the standard UML modeling tools. This compatibility requirement triggered the use of XMI in our framework. XMI is a file format which mostly used to interchange the data between different UML modeling tools. Most of the standard UML modeling tools can import XMI files as well as export the UML diagrams developed using those tools as XMI files. What XMI files do is, they store the details of UML diagrams given by the ontological UML constructs using XML language. XML is the markup language which is used to script XMI files. For example, student class can be taken from the aforementioned real world scenario with student-ID attribute and register operation. As depicted in **Fig. 3**, the details of class student are stored in a XMI script file using XML language statements.

As illustrated, XMI file only contains some XML tags (XML statements written within < > are known as XML tags) and XMI file is only a text document. Hence XMI script file itself does not show the conceptual or system model directly; instead that file needs to be open with an XMI compatible UML modeling tool thereby the tool converts the XML tags specified in the XMI script file into corresponding UML symbols.

Definition or modifications of the existing UML constructs will not be performed during this research study. Instead, mapping ontological concepts to the existing UML constructs and transformation rules definition for the design phase will be performed. Hence, defining new XML tags for the existing UML constructs is not required. Because that, if an existing UML construct or a set of constructs is drawn using a XMI compatible modeling tool, that tool is capable of exporting the corresponding UML diagram as an XMI file. This means for each existing UML construct, predefined XML tags are available in each UML modeling tool.

Thus, we only have to ensure that the ontological UML constructs generated by the framework for each conceptual and system model, are transformed into XML tags with no missing information. In order to do this, necessary XML tag(s) need to be assigned for each ontological construct for both UML versions. So mapping the XML tags for the ontological UML constructs can be achieved as depicted in **Table 1**.



```
        <UML:Class xmi.id="UMLClass.5" name="Student"
visibility="public" isSpecification="false"
namespace="UMLModel.4" isRoot="false" isLeaf="false"
isAbstract="false" isActive="false">
          <UML:Classifier.feature>
            <UML:Attribute xmi.id="UMLAttribute.6"
name="student_ID" visibility="public" isSpecification="false
ownerScope="instance" changeability="changeable"
targetScope="instance" type="" owner="UMLClass.5"/>
            <UML:Attribute xmi.id="UMLAttribute.7" name=""
visibility="public" isSpecification="false"
ownerScope="instance" changeability="changeable"
targetScope="instance" type="" owner="UMLClass.5"/>
            <UML:Operation xmi.id="UMLOperation.8"
name="register" visibility="public" isSpecification="false"
ownerScope="instance" isQuery="false" concurrency="sequentia
isRoot="false" isLeaf="false" isAbstract="false"
```

Fig. 3. Student UML class with its' XMI script file information

```
<XPD:OBJ name = "NameCompartment" type = "UMLNameCompartmentView" guid =
"10C3RPr3LEGMqSNgNhncRgAA">
<XPD:OBJ name = "NameLabel" type = "LabelView" guid = "vo06Y8KMC0K/FeGHw0/69AAA">
<XPD:ATTR name = "FontStyle" type = "integer">4</XPD:ATTR>
<XPD:ATTR name = "Text" type = "string">Student</XPD:ATTR>
</XPD:OBJ>
<XPD:OBJ name = "StereotypeLabel" type = "LabelView" guid = "A9CAqIpSAUiiTl86XL07WQAA">
<XPD:ATTR name = "Visible" type = "boolean">False</XPD:ATTR>
</XPD:OBJ>
```

Fig. 4. Complete set of XML tags used to define an UML object

Table 1. Basic XML tags assignment to analysis phase UML constructs and ontological concepts

Analysis phase UML construct	Ontological concept	Analysis phase XML definition (Basic XML tag)
Object	Thing	<XPD:ATTR name = Name type = string> student-Object </XPD:ATTR>
Attribute	Property	<XPD:ATTR name = Name type = string> student-ID </XPD:ATTR>
Class	Functional schema	<XPD:ATTR name = Name type = string> Student </XPD:ATTR>

The first two columns of the table are taken from Evermann and Wand (2005b) research study and in the third column necessary XML tags are mentioned. The table only illustrates the primary XML tag which includes the UML construct name. Except this tag, some more XML tags are required to describe a UML construct precisely. For example, the XML tags set depicted in **Fig. 4**, are required to describe an UML object correctly.

The XML language used in **Fig. 4** is known as XML Processing Description Language (XPDL), hence all the XML tags start from that word. Each object represented in this language contains an arbitrary mixture of ATTR (attribute) and OBJ (sub-object) elements and using them all the objects will be described in the XMI script file. Correspondingly, the above depicted set of XML tags includes all the details of the UML object; Student.

Thus, all the XML tags used for each ontological UML construct need to be identified clearly, to have a perfect transformation. Sometimes, merging the XML tags of two different UML constructs may need to be done in order to map the XML tags seamlessly with the ontological UML constructs and their corresponding ontological concepts. Correspondingly this process need to be continued for the ontological UML constructs of both analysis and design phases. Accurate XML tag assignment ensures a faultless transformation form conceptual and system models to XMI script files thereby representing the real world system accurately in both analysis and design phases.

4. CONCLUSION

This study proposes an ontology based UML framework for OO conceptual and system modeling. It is a part of an ongoing research project and certain parts of the framework are illustrated using real world examples. The future work will cover the following aspects of the framework:

- As explained in the Constructed language sub section; to represent the characteristics of real world systems precisely at the analysis phase, constructed language rules need to be developed to map the ontological UML constructs defined for the analysis phase with no missing information
- As mentioned in this study, a faultless model transformation can be achieved by defining proper transformation rules between analysis and design phases by matching the two sets of ontological language constructs defined for each phase. Since ontological UML constructs are already defined for analysis phase, we hope to work in defining

ontological UML constructs for the design phase. This can be done by assigning system domain characteristics into generic UML, using ontological concepts. Mario Bunge (1977; 1979) ontological approach is specifically used in our research study because

- It is rooted in ontological work done over a long period of time and it is in line with the old practices as well as current practices in the world (Bunge, 1977)
- It has been successfully adopted to IS modeling by several researchers (Evermann and Wand, 2005b; 2009; Wand and Weber, 1988; 1989)
- Evermann and Wand is using this ontology in developing UML constructs for analysis phase, which we use in our framework (Evermann and Wand, 2005b)

Once the system domain characteristics are assigned with necessary ontological concepts, those need to be mapped with generic UML to define UML constructs for the design phase. According to Bunge's ontology, thing is the fundamental concept and all the other ontological concepts built under this thing. The world consist of things and only of things and moreover those substantial things physically exist in the world. Only the constructs of UML class diagram are considered for the mapping with ontological concepts because, in this study we only use UML class diagrams to explain the framework. Since all the concepts of Bunge's ontology are built under thing, logically it should be mapped with the fundamental UML construct, object. But, OO system modeling defines a UML object as; Something that is or is capable of being seen, touched or otherwise sensed and about which users store data and associated behavior (Whitten and Bentley, 2005). Hence objects can or cannot be physical (substantial) in OO system modeling thereby mapping Bunge thing to UML object is infeasible.

Since Bunge's fundamental ontological concept (thing) is difficult to map with a design phase UML construct, mapping the other ontological concepts with relevant UML constructs may not be an easy task. Because, until a suitable UML construct will be found for Bunge-thing, it will not be possible to proceed with the remaining ontological concepts. Hence, this part of the framework is supposed to be carried out as a future research work:

- Subsequently, the framework will convert the two sets of ontological UML constructs defined for analysis and design phases into XML language statements. Using these statements, XMI scrip files will be generated. As mentioned in the Framework in practice section, no new XML tag or set of tags need to be defined to be suited with ontological UML constructs. What need to be done is, ensuring that all the ontological UML constructs used in conceptual and system modleing can be precisely mapped using the predefined XML tags with no missing information. At this level this activity is continuing and to be completed as a future work.
- Once the activities comprises in the proposed framework are being completed precisely, an empirical study will be carried out to evaluate the validity and the usefullness of the framework in the practical scenario
- Finally we hope to do an investigation to find the UML tools in current practice, which can be made compatible with our framework. Because, XMI cannot be used with every UML tool. Even if XMI is used, different tools may have different unique ways in representing the XMI script file. After a thorough analysis of these conflicts and gaps, we expect to find a standard format for the XMI script file, which may capable to be used with all the UML tools that uses XMI. This is the final future activity we expect to carry out with our research study

With all the above mentioned activities, our final goal is to create an UML based ontological framework that can be used to (1) integrate both analysis and design phases using a common OO modeling language (2) reduce the manual work involved in the current OOAD process (3) minimize the complexities and difficulties faced by the modelers in using UML modeling tools.

6. ACKNOWLEDGEMENT

We are indebted to Prof. Ron Weber (Faculty of Information Technology-Monash University, Australia) and Mr. Kapila Dias (University of Colombo School of Computing, Sri Lanka) for the helpful comments and reference suggestions given on this study.

7. REFERENCES

1. Bennett, S., S. McRobb and R. Farmer, 2001. Object-Oriented Systems Analysis and Design using UML. 2nd Edn., McGraw-Hill Education. ISBN-10: 0077098641, pp: 632.
2. Bunge, M., 1977. Treatise on Basic Philosophy: Ontology I: The Furniture of the World. 1st Edn., Springer, Dordrecht, ISBN-10: 9027707804, pp: 370.
3. Bunge, M., 1979. A World of Systems. 1st Edn., Springer, Dordrecht, ISBN-10: 9027709459, pp: 314.
4. Deeptimahanti, D.K. and M.A. Babar, 2009. An automated tool for generating UML models from natural language requirements. Proceedings of the 24th IEEE/ACM International Conference on Automated Software Engineering, Nov. 16-20, IEEE Xplore Press, Auckland, pp: 680-682. DOI: 10.1109/ASE.2009.48

5. Evans, A.S., 1998. Reasoning with UML class diagrams. Proceedings of the 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques, Oct. 21-23, IEEE Xplore Press, Boca Raton, FL., pp: 102-113. DOI: 10.1109/WIFT.1998.766304
6. Evermann, J. and Y. Wand, 2005a. Toward formalizing domain modeling semantics in language syntax. IEEE Trans. Software Eng., 31: 21-37. DOI: 10.1109/TSE.2005.15
7. Evermann, J. and Y. Wand, 2005b. Ontology based object-oriented domain modelling: Fundamental concepts. Requirements Eng., 10: 146-160. DOI: 10.1007/s00766-004-0208-2
8. Evermann, J. and Y. Wand, 2009. Ontology based object-oriented domain modeling: Representing behavior. J. Database Manage., 20: 48-77. http://www.journalogy.net/Publication/4398062/ontology-based-object-oriented-domain-modeling-representing-behavior
9. Favre, L., 2003. UML and the Unified Process. 1st Edn., Idea Group Inc., Hershey, ISBN-10: 1931777446, pp: 314.
10. Fowler, M., 2003. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd Edn., Addison-Wesley Professional, Essex, UK., ISBN-10: 0321193687, pp: 208.
11. Gopsill, F.P., 1989. International languages: A matter for Interlingua. 1st Edn., British Interlingua Society, ISBN-10: 0951169564, pp: 282.
12. Kim, J., S. Park and V. Sugumaran, 2008. DRAMA: A framework for domain requirements analysis and modeling architectures in software product lines. J. Syst. Software, 81: 37-55. DOI: 10.1016/j.jss.2007.04.011
13. Koppe, C., 2010. Teaching Software Process with Openup. University of Amsterdam. http://koeppe.nl/publications/teachingsoftwareprocesswithopenup_v2.pdf
14. Kuhrmann, M., 2011. User assistance during domain-specific language design. Proceedings of the ICSE 2011 Workshop on Flexible Modeling Tools (Flexi Tools), May 22-22, Waikiki, Honolulu, HI, USA., pp: 1-5.
15. McGuigan, B. and N. Foster, 2011. What is a Constructed language? Conjecture Corporation. http://www.wisegeek.com/what-is-a-constructed-language.htm
16. Mishra, R.K. and B. Lohani, 2007. An object-oriented software development approach to design simulator for airborne altimetric lidar. Proceedings of the National Conference on Emerging Trends in Information Technology, (ETIT' 07), SGITS Indore, India.
17. Nguyen, P. and R. Chun, 2006. Model driven development with interactive use cases and UML models. Proceedings of the International Conference on Software Engineering Research and Practice Programming Languages and Compilers, (SERPPLC' 06), pp: 534-540. http://arnetminer.org/publication/model-driven-development-with-interactive-use-cases-and-uml-models-585429.html;jsessionid=3BB7F318364B42100786B7E3385142CD.tt
18. Oostendorp, M.V., 2000. Constructed language and linguistic theory. Association Belgique de Linguistique. http://www.vanoostendorp.nl/pdf/cllt.pdf
19. Overmyer, S.P., L. Benoit and R. Owen, 2001. Conceptual modeling through linguistic analysis using LIDA. Proceedings of the 23rd International Conference on Software Engineering, May 12-19, IEEE Xplore Press, pp: 401-410. DOI: 10.1109/ICSE.2001.919113
20. Paige, R., P. Agarwal and P. Brooke, 2003. Combining agile practices with UML and EJB: A case study in agile development. Proceedings of the 4th International Conference on Extreme Programming and Agile Processes in Software Engineering, (EPAPSE' 03) Springer-Verlag Berlin, pp: 351-353. http://dl.acm.org/citation.cfm?id=1763935
21. Rumbaugh, J., I. Jacobson and G. Booch, 1999. The Unified Modeling Language Reference Manual. 1st Edn., Addison-Wesley Professional, Reading, Mass., ISBN-10: 020130998X, pp: 550.
22. Silingas, D. and R. Butleris, 2009. Towards customizing UML tools for enterprise architecture modeling. IADIS.
23. Tichy, W.F. and S.J. Koerner, 2010. Text to software: Developing tools to close the gaps in software engineering. Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, Nov. 7-11, ACM Press, New York, USA., pp: 379-383. DOI: 10.1145/1882362.1882439
24. Wand, Y. and R. Weber, 1988. An ontological analysis of some fundamental information systems concepts. Proceedings of the 9th International Conference on Information Systems, (IS' 88), pp: 213-225.
25. Wand, Y. and R. Weber, 1989. An ontological evaluation of systems analysis and design methods. Inform. Syst. Concepts.
26. Whitten, J. and L. Bentley, 2005. Systems Analysis and Design Methods. 7th Edn., McGraw-Hill, Irwin ISBN-10: 0073052337, pp: 768.