2013

Representing Interactional and External Environmental Semantics using Unified Modelling Language (UML)

Prabodha Tilakaratna
Monash University, prabodha@monash.edu

Jayantha Rajapakse
Monash University, jayantha.rajapakse@monash.edu

Information Systems: Transforming the Future

24th Australasian Conference on Information Systems, 4-6 December 2013, Melbourne

Proudly sponsored by

ACIS 2013 Principal Sponsor

Representing Interactional and External Environmental Semantics using Unified Modelling Language (UML)

Prabodha Tilakaratna
School of Information Technology
Monash University Malaysia
Email: prabodha@monash.edu

Jayantha Rajapakse
School of Information Technology
Monash University Malaysia
Email: jayantha.rajapakse@monash.edu

Abstract

Object-oriented methodology is widely used in the information system development field. Nonetheless, recent research studies have discovered that the modelling grammars that use object-oriented methodology lack necessary constructs to represent certain real-world semantics. Therefore, the use of such grammars with their shortcomings can produce defective conceptual models, thereby producing defective information systems. Evermann and Wand (2005, 2009) studied this issue and proposed a set of rules for object-oriented grammatical constructs to represent static and behavioural semantics of a real-world phenomenon. This paper extends its work by proposing object-oriented grammatical rules for the interactional and external environmental semantics of a real-world phenomenon. This representation is exemplified using an object-oriented modelling grammar namely Unified Modelling Language (UML). Subsequently, a set of new rules has been validated using a case study. This extended UML facilitates seamless integration between the conceptual model and its system model.

Keywords

Interactional semantics, External environmental semantics, Object-oriented modelling grammars, Use case diagram, Unified modelling language.

INTRODUCTION

Information System (IS) development commences with *conceptual modelling,* which is carried out to model the semantics of a given real-world scenario (Milton et al., 2010; Weber, 2003). An accurate representation of the real-world semantics with a conceptual model is essential for the success of the final IS. Modellers use variety of modelling grammars for conceptual modelling (Milton et al., 2012; Rajapakse & Orlowska, 1995). However, with the introduction of Object-Oriented (OO) methodology for IS development, rapid growth could be seen in the use of OO modelling grammars for conceptual modelling. Hence, when comparing with non-OO modelling grammars, OO modelling grammars provide more powerful IS modelling capabilities by combining both the data and process aspects of a given real-world scenario (Anglim et al., 2009; Lee et al., 1994).

Nevertheless, the fundamental concept of OO methodology: the *object,* and the associated concepts such as *inheritance*, *encapsulation*, *abstraction*, and *polymorphism* focus on *system modelling* (Khoo, 2006; Tilakaratna & Rajapakse, 2012a). Therefore, OO modelling grammars lack necessary constructs to represent real-world semantics. For example, as discussed by Evermann and Wand (2009), although OO grammatical constructs such as *method* and *operation* have clear meanings in system modelling, it is not clear what they represent in the real-world phenomena. Only a few researchers studied the shortcomings of OO modelling grammars in terms of conceptual modelling (Evermann & Wand, 2005, 2009). They have observed that such shortcomings create deficiencies during the representation of real-world semantics with OO grammatical constructs (e.g. unavailability or having excess amount of OO grammatical constructs to be mapped with a single real-world semantic). Nevertheless, no OO modelling grammar appears to exist that fully supports conceptual modelling.

Hence, IS development projects that use OO modelling grammars for system modelling must either: (1) Choose to employ the same OO modelling grammar for conceptual modelling in spite of its shortcomings. Such shortcomings may result in defective conceptual models. (2) Choose to employ a non-OO modelling grammar for conceptual modelling. The use of two different modelling grammars for the two modelling phases require more time, money, and effort to be spent in the transformation of conceptual models into system models. One systematic way to resolve these shortcomings is to use an extended OO modelling grammar which is capable of representing any real-world semantic. Nevertheless, real-world possesses an infinite number of semantics. Thus, it is impossible for the constructs of a single modelling grammar to represent them all. Wand and Weber (1993)

proposed a solution for this problem by adapting and extending an ontology (Bunge, 1977), which can better cover real-world phenomena that are encountered frequently during IS development (Wand & Weber, 1993).

Evermann and Wand (2005, 2009) started modifying OO modelling grammars for conceptual modelling, with the use of the set of ontological concepts proposed by Wand and Weber (1993). They have used Unified Modelling Language (UML) as the OO modelling grammar. UML has been chosen because, it is a widely used OO modelling grammar developed for IS modelling (Evermann & Wand, 2005). In addition, Evermann and Wand (2005, 2009) and Tilakaratna and Rajapakse (2012a, 2012b) provide some examples of the unclear and incomplete situations which represent the lack of the conceptual modelling capabilities in the UML, which could ultimately have an impact negatively on the development of ISs. However, Evermann and Wand (2005, 2009) only studied the representation of static and behavioural semantics of real-world phenomena. Nevertheless, real-world semantics that are frequently encountered during IS development can be categorised into four main categories: static, behavioural, interactional and external environmental semantics. Hence the focus of this paper is to extend Evermann and Wand's work by proposing a set of rules for UML constructs to represent *interactional* and *external environmental* semantics of a real-world phenomenon.

This paper is structured as follows. The next section discloses the related work. Section 3 addresses the research methodology. Section 4 provides an overview of the modelling of real-world interactional and external environment semantics with UML, followed by representation and interpretation mappings. Section 5 describes the empirical evidence to support the results. Section 6 concludes the paper.

RELATED WORK

Bunge's ontology has been used by many researchers in assessing the appropriateness of modelling grammars for conceptual modelling. Those modelling grammars belong to various categories such as: business process modelling grammars (Recker et al., 2009), process modelling grammars (Rajapakse, 1996; Rosemann et al., 2006), and data modelling grammars (Pieris & Rajapakse, 2012; Wand & Weber, 1995; Weber & Zhang, 1996). However, Evermann and Wand (2005, 2009) commenced the use of ontological concepts to assess the appropriateness of OO modelling grammars for conceptual modelling. They have exemplified their research works using two UML diagrams (UML grammar possesses fourteen diagrams) namely class diagram and state machine diagram.

Evermann and Wand have undertaken their experiments in two phases: Initially the static concepts of Bunge's ontology represented with the constructs of UML class diagram (Evermann & Wand, 2005). Subsequently, the behavioural ontological concepts were represented with the constructs of UML state machine diagram (Evermann & Wand, 2009). The results of these two mapping processes are depicted in Table 1 and Table 2. The fundamental concepts of Bunge's ontology used by Evermann and Wand (2005, 2009) are also relevant for our research study and are given below.

Thing: This is the elementary notion of Bunge's ontology. The world is made of things that are physical and substantial. *Composition*: Things can composed together to create composite things. However, null things cannot be composed together to create composite things.

Property: Things possess properties which can be considered as the attributes of the thing. *Intrinsic property*: A property which is intrinsic to a thing and is isolated from the other things in the world. For example, 'skin colour' is an intrinsic property to each and every human being. *Mutual property*: A property of a thing, which relates to one or many other things in the world. Simply, this is a property of two or more things. For example, 'child of' is a mutual property which relates to both mother and father. *Emergent property*: A property of a composite thing. For example computer is a composite thing which is composed of things such as hard disk, mother board, and processor. Accordingly, computer speed is a property of the entire computer.

Functional schema: A set of state functions which are used to represent and describe the properties of a thing. *Natural kind*: A set of things which are adhered to a same set of laws and consequently exhibit the same type of behaviour. *Kind*: A set of things that possess a set of common properties. *Law*: A restriction which is placed on the properties of a thing.

State: Values of the attributes assigned to a thing. For example, a student in school may have two states, 'passed' or 'failed'. *Transition*: A transformation or a change that happens between two states. It can be lawful or unlawful. For example, changing from 'alive' to 'dead'. *Lawful transformation*: A lawful change happens between two lawful states. For example changing from 'alive' to 'dead' is a lawful transformation whereas changing from 'dead' to 'alive' is an unlawful transformation. *Interaction*: Two or more things acting on each other. *Acts on*: If one thing affects another thing and if the history of those two things is not independent from each other, those two things are said to be act on or being bonded to each other. *Process*: Ordered set of events or a sequence of states of one thing. *Subclass*: A set of things that possess a set of common properties and additional set of properties which are not common for the set of things.

System: A set of things where (1) each thing in the set is coupled at least with one other thing (2) it is impossible to partition the set of things as the history of one partition is dependent of the history of the other partition. *System environment*: Set of things that are not in the composition of the system, which can interact on the things in the composition of the system. For example, friends are external environment of the family system. *System composition*: This means the things in the system. For example, a family system may have father, mother and children in the system composition. *System decomposition*: The subsystems owned by the system where the composition of the system equals to the union of the composition of all subsystems.

Table 1: Evermann and Wand (2005) mapping of ontological concepts and class diagram constructs.

Ontological concept	**UML construct**	**Remarks**
Thing	Object	
Property	Attribute	
Intrinsic property	Attribute of ordinary class	
Mutual property	Attribute of association class	
Emergent property	Class attribute	Attribute of an aggregate made of class instances (i.e. objects).
Functional schema	Class	Described by class.
Natural kind	Set of objects (extension of class)	
Composition	Aggregation	
N/ A	Composition and Association	Do not have an ontological equivalence.
A set of mutual properties	Association class	

Table 2: Evermann and Wand's (2009) mapping of ontological concepts and state machine diagram constructs.

Ontological concept	**UML construct**	**Remarks**
State	State	Connection with attributes of objects made.
	Sub state	Sub components of a state.
	Action state	Action states are composite states.
Transition	Transition and Operation	
Lawful transformation	Method	
Interaction	Message	Message is not a substantial thing.

Results of Evermann and Wand (2005, 2009) research studies show that the constructs of class and state machine diagrams can be used appropriately in representing the real-world static and behavioural semantics. Some empirical assessments given in their papers provide further justifications for the successful use of the ontological concepts with class and state machine diagram constructs, which improved the results of conceptual modelling.

RESEARCH METHODOLOGY

Similar to the research methodology used by Evermann and Wand (2005, 2009), we also assign real-world semantics to UML constructs by creating mappings between UML constructs and ontological concepts. The mapping process covers the following two steps: *representation mapping* and *interpretation mapping.*

(1) *Representation mapping*: This step evaluates whether the real-world semantics can be represented accurately with UML constructs. During this *representation mapping,* each and every ontological concept will be represented with an accurate UML construct (Figure 1).

(2) *Interpretation mapping*: This step evaluates the capability of UML constructs to interpret the real-world semantics accurately. During this *interpretation mapping,* all the unmapped UML constructs remaining from the representation mapping process will be identified, and then will be represented using suitable ontological concepts (Figure 1).

This methodology ensures the mapping of all the ontological concepts with the existing grammatical constructs. The way how the research methodology (i.e. representation and interpretation mappings) is used to represent the interactional and external environmental semantics is explained in the next section.

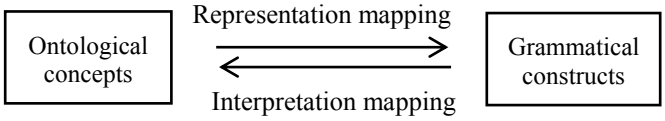

Figure 1: Representation and interpretation mappings.

MODELLING INTERACTIONAL AND EXTERNAL ENVIRONMENTAL SEMANTICS

As specified in the introduction section, real-world semantics fall into four categories: static, behavioural, integration and external environment semantics. *Static* semantics represent the static structure of real-world entities whereas the *behavioural* semantics represent the state changes of those entities. *Interaction* semantics represent the communications between real-world entities. The *external environment* semantics represent the communications between internal and external entities of a real-world system. As mentioned previously, our research study emphasises on interactional and external environmental semantics of real-world phenomena.

Instead of representing the interactional and external environmental semantics using UML in two separate phases, we focussed on representing both types of semantics together with UML. In this regard, we have chosen the use case diagram for our study because, out of the fourteen UML diagrams the use case diagram is the only diagram which is capable of representing both internal and external environment of an IS. According to the latest UML specification documents (Object Management Group, 2012), a use case diagram provides (1) a high-level view of the internal interactions of a system and, (2) an understanding of a system from the viewpoint of external users who are intended to interact with the system. In addition, the use case diagram is capable of representing the system characteristics very systematically and categorically, which makes it easier to understand the system and its interactions with external users (Object Management Group, 2012).

Initially we carry out *representation mapping* process where ontological concepts will be mapped with necessary use case constructs. Next, the remaining use case constructs will be interpreted with necessary ontological concepts during the *interpretation mapping*. By doing so, we will try to match all the constructs of use case diagram in an accurate way with necessary ontological concepts, which ultimately makes use case diagram appropriate for representing real-world semantics.

Representation Mapping

System

Primary concept in Bunge's ontology which is used to represent a set of *things* involve ontological *system.* Bunge's ontology has stated two specific factors to be fulfilled, if a set of ontological *things* to be considered as an ontological *system*. (1) Each *thing* in the set of *things* needs to be coupled at least with one other *thing*. (2) Partitioning the set is impossible, as the history of one partition depends on the history of the other partition.

The use case diagram also possesses a similar construct namely *system boundary,* which represents the boundary between the internal and external environment of an IS. Internal environment consists of object interactions whereas external environment contains the external users (i.e. *actors*) of the system. Figure 2, depicts an example of use case diagram, where the system boundary is represented by a rectangle named 'ATM system usage'.

Few differences can be observed between use case *system boundary* and ontological *system* however. Ontological *system* consist of all the internal *things* of a given real-world scenario. Nevertheless, as discussed above, the use case *system boundary* consists of the interactions of the objects. The objects of a system are represented using UML class diagram, where the classes possess objects. A high-level view of the interactions of those objects is represented within the use case *system boundary*. Therefore, ultimately all the objects which are internal to a system has appeared indirectly in the use case *system boundary*. Accordingly, following semantic mapping rule can be proposed.

Rule 1: Ontological system must be modelled as system boundary.

System composition

As mentioned above, an ontological *system* possesses all the *things* internal to the system. All the set of *things* owned by an ontological *system* is known as ontological *system composition.* Nevertheless, since the use case diagram does not represent any object, ontological system composition cannot be represented by any use case construct. However, all the objects of which interactions are shown in use case *system boundary* are appearing in UML class diagram. Although the focus of this paper is not on class diagram, since we consider the ontological concepts related to ontological *system*, we propose the following semantic

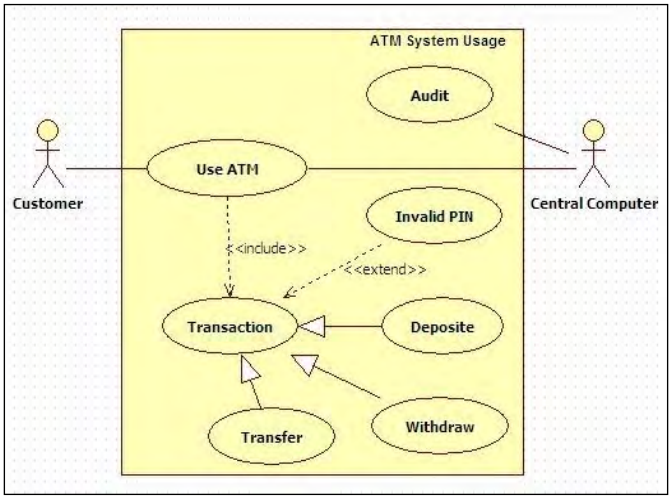

Figure 2: Example use case diagram of an ATM system

mapping rule:

Rule 2: Ontological system composition of a system can be modelled with UML class diagram.

System environment

Ontological *system environment* possesses the *things* that are outside of the *system composition*. Simply, *system environment* represents the external environment of a real-world scenario. UML specification documents specify that (Object Management Group, 2012), the model elements outside the *system boundary* (i.e. use case actors) represent the external environment of an IS. For example, according to Figure 2, 'customer' and 'central computer' stay outside of the use case *system boundary* and hence they belong to the external environment of 'ATM system usage'. Thus, the following semantic mapping rule can be proposed for ontological *system environment* and use case *actors*:

Rule 3: Ontological system environment of a system can be modelled with the entire set of actors of a use case diagram.

System decomposition

Ontological *system decomposition* reflects the breakdown of a given real-world scenario by representing the subsystems own by the system. A similar construct can be identified in the use case diagram namely, the *include relationship* which simplifies a large use case by splitting it into sub-parts.

Although *system decomposition* discusses about the entire *system* and its subsystems, use cases participate in an *include relationship* do not represent a system or subsystems, instead only a use case (i.e. use case which initiates the *include relationship*) and sub-parts of that use case (i.e. emerging use cases of the *include relationship*). According to Rule 6 of interpretation mapping section, use case is only a process and not a system. This can be further explained by an example. As depicted in Figure 2, 'use ATM' is a use case and it does not represent an entire system. Hence, to preserve the meaning of ontological *system decomposition* with the *include relationship,* following semantic mapping rule and corollary are proposed where the rule becomes meaningful only with the satisfaction of the corollary .

Rule 4: Ontological system decomposition of a system can be represented by the include relationship.

Corollary 1: Within the context of an include relationship, the use case which initiates the include relationship must be considered as a system, and the use cases emerged from the include relationship must be considered as subsystems of that system.

Interpretation Mapping

Actor

Use case *actor* is identified as an entity in the external environment of an information system which characterizes the interactions that outside entities may have with the system (Object Management Group, 2012). An *actor* could be a person or hardware, which can be considered as a physical real-world entity. Since Bunge's ontology represents real-world physical entities as *things*, we can interpret the *actor* as an ontological *thing*. Nevertheless, an *actor* could also be an organization or another system. In this regard, we can represent an *actor* by an ontological *system*. However, an external entity always interacts with an information system as a single object. Even if it is another system or an organization, it communicates with the internal environment of the given information system as a single object. Hence, we propose the following semantic mapping rule for use case *actor*:

Rule 5: Use case actor can be represented by an ontological thing.

However, Evermann and Wand (2005) have already mapped ontological *thing* with UML *object*. Accordingly, mapping both UML *object* and *actor* with the same ontological concept *thing* creates construct redundancy. This is a mapping shortcoming. Nevertheless, we propose that UML *object* represents ontological internal environmental *things* whilst UML *actor* represents ontological external *things*. This proposition helps removing the construct redundancy derive from *actor* and *object*.

Use case

A *use case* represents externally visible internal function of an information system (Object Management Group, 2012). This function is expressed as (1) interactions exchanged within the internal environment and, (2) interactions exchanged by the internal objects with the actors in the external environment. Different interpretations exist for *use cases*, where a *use case* is considered as a frame which reflects objects, a subsystem, and a frame which reflects object interactions of a system. The most faithful interpretation of a *use case* needs to be identified however, if it is to be accurately represented by an ontological concept.

Firstly, although a use case may be related to a single object, it does not represent the object itself, instead the interactions of that object. For example, we assume that a use case relates to an object named 'ATM'. However, the *use case* is not named with the object name (e.g. 'ATM'), instead, it represents the function associates with that object (i.e. object interactions: e.g. 'Use ATM'). Therefore, a use case cannot be considered as a frame which reflects objects.

Secondly, a *use case* should possess more than one object, if it is to be considered as a subsystem, because a subsystem is a collection of objects of a system. Nevertheless, use cases do not represent objects by themselves, but only the interactions of the objects. Hence, the most faithful explanation for a *use case* is a frame which represents the object interactions of a system. Bunge's ontology also has a similar concept namely *process*, which represents a sequenced set of interactions of an ontological thing. This helps proposing the following semantic mapping rule:

Rule 6: Use case can be represented by an ontological process.

Association relationship

Association relationship is a relationship which exists between two model elements (Object Management Group, 2012). Since it is common for both class and use case diagrams, from the class diagram viewpoint *association* relationship connects two *classes*. From use case diagram viewpoint, *association* relationship connects an *actor* and a *use case*. In their initial mapping, Evermann and Wand (2005) proposed that every *association* relationship must be modelled with an *association class,* since it is a non-substantial *thing* in terms of Bunge's ontology**.**

However, we have a different viewpoint, for *association* relationship. We define it as a link which is used by the system *objects* to interact with other *objects*. For example, we can consider the relationship that exists between the 'customer' object and the 'ATM' object of 'Use ATM' use case (Figure 2). This link makes customer to use ATM, which ultimately affect the status of customers' wealth, and also customers' bank account. Hence, the link between customer and ATM let the customer interacts with the ATM, where ATM also interacts with the customer in return through various activities such as updating customer's bank account. Therefore, we cannot eliminate the existence of such relationships in conceptual level.

Bunge's ontology also stated that every real-world *thing* possesses relationships among themselves by acting on, or by acted upon (i.e. interacts with) via other *things* (Bunge, 1977, p. 256). Bunge's ontology has defined a concept for such relationships namely, *coupling.* He has defined that two different real-world things interact with each other only if each acts on the other, and thus are said to be coupled (Bunge, 1977, p. 259). Therefore, *association* relationships of UML can be represented using the ontological *coupling* concept. Nevertheless, use case *association* relationship exists between an *actor* and a *use case*. Although *actor* is as an ontological thing (Rule 5), *use case* is not an ontological thing (Rule 6). In order to preserve the meaning of ontological *coupling* concept in *association* relationship, we propose the following semantic mapping rule along with a corollary:

Rule 7: UML association relationship can be represented by ontological coupling concept.

Corollary 2: An association relationship occur among an actor and use case represents a communication between an actor and an object, where the object represents its' interactions via the use case.

Extend relationship

Extend is a relationship which helps representing the additional behaviour of a use case (i.e. extending a given use case) using another use case or a set of use cases namely *base* use cases (Object Management Group, 2012). The *base* use cases do not carry the behaviour of extending use case, and are not sub-parts of the extending use case. Besides this, additional and enhanced interactions are represented by base use cases. Nevertheless, extension or enhancement does not have any ontological counterpart in Bunge's ontology. Consequently, the unavailability of a matching ontological concept for extend relationship result in construct excess in the use case diagram (Wand & Weber, 1993). This is a mapping shortcoming.

However, since extend relationship represents additional or enhanced behaviour of an extending use case, it can be stated that the *base* use cases are *super use cases* (i.e. enhanced *use cases*) of the extending *use cases.* Simply, a normal *use case* represents a process of an information system, whilst a *base use case* represents a 'super' process of the system with enhanced functions. Accordingly, we propose that *base use cases* represent a type of processes of an information system which can be considered as a sub category of a *use case*.

This creates construct excess since both normal *use case* and *base use case* are being mapped with the same ontological concept, i.e. *process.* However, according to our explanation above, since *base use cases* represent a sub category of processes of an information system, the construct redundancy arise with *use case* and *base use case* constructs can be eliminated. Accordingly, following semantic mapping rule can be proposed for *extend* relationship which ultimately remove the construct excess occurred from it.

Rule 8: Extend relationship can be represented by ontological coupling, which helps enhancing the behaviour of an ontological process with another process.

Since extend relationship represent a specific type of *coupling,* it does not create any construct redundancy.

Generalization relationship

According to the newly released UML specification document (Object Management Group, 2012), the use case which initiates the *generalization* relationship is known as the *super-use case,* and the use cases derived from the *generalization* relationship are known as *generalization sets*. For example the generalized use cases depicted in Figure 2 (i.e. deposit, transfer and withdraw) can be considered as three *generalization sets* of 'transaction' super-use case. *Generalization* is not a unique construct of the use case diagram, which is being used in class diagram as well. In the viewpoint of class diagram, *generalization set* reflects a subclass of the super-class, where the *generalization sets* own the common properties of super-class and in addition, possess its own additional set of properties. This representation has to be modified however, to be suited with the *generalization* relationship of the use case diagram as explained below.

Similar to the extend relationship explanation given above, a normal *use case* represents a process of an IS, whilst a *generalization set* represents a 'sub' process of the system with inherited functions. Accordingly, we propose that *generalization sets* represent one type of sub-processes of an IS whilst those *generalization sets* own the common interactions of *super-use case* and in addition possess its own set of interaction. This can be considered as a sub category of a *use case*. Therefore, according to the above explanation regarding the *generalization sets*, the mapping of *generalization* relationship can be proposed for use case diagram as follows:

Rule 9: UML generalization relationship can be represented by ontological coupling, which helps inheriting the behaviour of an ontological process using another process.

Since *generalization* relationship is of a specific type of *coupling,* it does not create any construct redundancy.

CASE STUDY

In order to validate the rules developed above with real-life IS development process, we conducted an empirical assessment. We selected a set of 50 undergraduate students studying information and communication technology degree at University of Colombo, Sri Lanka to participate in the empirical assessment. These students were selected for the following two reasons. (1) They have studied System Analysis and Design (SAD) as one of their course units, whilst UML had been used to exemplify the modelling aspects. (2) After the completion of this course unit, the students had been assigned to work in industry projects where they were required to develop information systems using UML. The industry projects were allocated to groups of students with each group consisting five students.

When our assessment started in June 2013, these students had already completed the development of conceptual models with UML diagrams based on their clients' requirements. During the preparation of the empirical assessment, the new rules along with the details were provided to the students two weeks prior to the assessment. During the assessment, the students were asked to re-model the real-world scenarios explained by their clients using the use case diagrams with the new rules. Their feedback on the new rules proposed for the use case diagram was obtained in group level by way of a questionnaire submitted to them. The questionnaire primarily focussed on the follows: understanding the difference between old and new models in terms of completeness and clarity, understanding applicability and level of complexity of the newly proposed rules, and the appropriateness of the newly proposed rules to represent any real-world scenario. Some example questions are given below:

(1) Completeness: 'According to your viewpoint, out of the two use case models, which one represents all/ most of all the real-world characteristics?' and, 'What are the missing real-world characteristics and which use case model misses those characteristics?'

(2) Clarity: 'Do you think that the new rules uniquely identify each and every real-world characteristic?'

(3) Applicability: 'Based on the completeness and clarity of the two use case models, do you think that the use case diagram with new rules is more capable of representing clients' requirements in a real-world viewpoint?' and, 'If you think so, what are the differences you have figured out in the use case diagram with the new rules, over the normal use case diagram you have used initially?'

(4) Complexity: 'Is it difficult to understand the proposed rules?' and, 'Are those rules complex?'

By assessing the level of completeness and clarity achieved by each of the two models developed by each group, and by analysing the answers received for the questionnaires, we have evaluated the improvements achieved by the use case models developed with the newly proposed rules.

An example set of use case diagrams (i.e. initial diagram and the diagram created with the new rules) are depicted in Figure 3 and 4 respectively. This group of students was asked to develop an online library system for an educational institute. Having studied our rules, students observed various shortcomings of their initial use case model. The students identified the importance of showing the internal and external environment of a real-world scenario using *system boundary*. They also observed that all external environmental entities that interact with the internal environment (i.e. as depicted in Figure 4, the 'student' and the 'lecturer') need to be modelled clearly in the diagram. In one of the rules, we proposed to represent the *system composition* with UML class diagram. According to this rule, students re-analysed the real-world scenario and checked whether a high level view of the functions of the *classes* in the class diagram had been represented in the use case diagram. This analysis made students understand that the function of issuing a book should happen in two stages: initially a borrower requests a book, and if it is available, then the borrower borrows the book. That information was not present in the initial use case diagram developed by the students (Figure 3). Representing such information in the conceptual model can improve the understanding because relating each and every use case construct with a unique real-world semantic, and the clear representation of external and internal environment will allow faithful analysis and assessment of the features of real-world library.

Majority of the participants stated that the rules gave a clear real-world meaning to use case constructs, which would thereby facilitate better understanding of real-world phenomena, to those who are even unfamiliar with the given scenario.

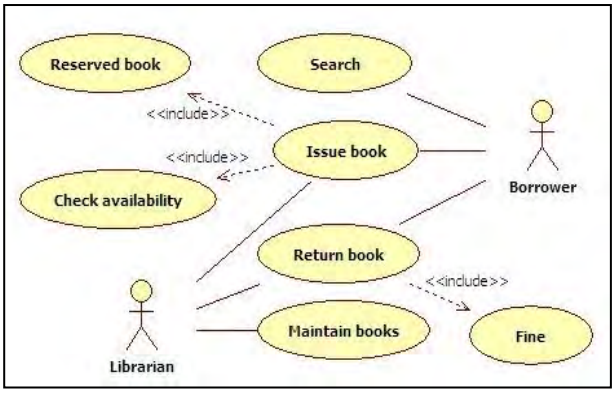

Figure 3: Initial use case diagram for the online library system

All the participants acknowledge that, although initially it was difficult for them to understand and work with the new rules, those were not highly complicated because, those rules (1) appear to be feasible and capable, (2) have not completely changed the meanings of the use case constructs, and (3) have matched the use case constructs with real-world semantics.

When our rules were revealed, students somewhat understood the requirement of using the use case diagram to represent the interactional and external environmental semantics of real-world phenomena. Also they acknowledged that it's a good starting point for the IS development process to use these new rules to represent the clients' requirements seamlessly and smoothly.

With the brevity of the paper, we are only able to provide one example set of use case models to show the improvements achieved by the use case models with the newly proposed rules. However as an average, out of the 10 groups participated in the empirical study, 6 groups showed major improvements, 2 groups showed minor improvements, and 2 groups showed no improvements in their use case models which were developed with the new rules, relative to the previous use case models they have developed. Thus, it could be summarised that the empirical assessment helped evaluating the appropriateness of the proposed rules for accurate understanding and communicating the real-world phenomena with use case diagrams.

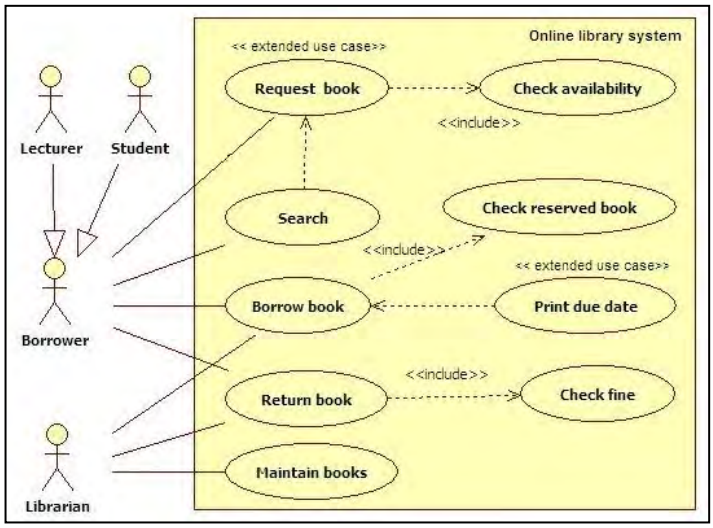

Figure 4: Modified use case diagram for the online library system, which adheres to the newly proposed rules.

CONCLUSION

This paper extends Evermann and Wand's (2005, 2009) works which studied the appropriateness of OO modelling grammars in representing real-world static and behavioural semantics using UML. We have categorised real-world semantics into four categories namely: static, behavioural, interactional and external environmental semantics. The objective of this paper is to evaluate the appropriateness of UML to represent the two remaining real-world semantics: i.e. *interactional* and *external environmental* semantics. We have used the use case diagram to achieve this objective. The results obtained show that ontological concepts can be accurately

mapped with use case constructs that are supported by some additional rules and therefore, the use case diagram is appropriate to model the interactional and external environmental semantics of a real-world phenomenon.

During the mapping between use case constructs and ontological concepts, some of the use case constructs such as *system boundary*, *use case*, *generalization* relationship, and *include* relationship have been mapped with ontological concepts with no mapping shortcoming. However, use case *actor* created construct redundancy and, use case *extend* relationship created construct excess. These are considered as mapping shortcomings. Nevertheless, as discussed in the above sections, we propose systematic approaches (such as modifying UML constructs with new rules) to eliminate those mapping shortcomings. Further, despite the proposal made by Evermann and Wand (2005) to proscribe *association* relationship from conceptual level with *association class* construct, we propose ontological *coupling* concept to represent *association* relationship. The summary of the results of our research study is presented in Table 3.

Table 3: Summary of the mappings between ontological concepts and use case diagram constructs.

Ontological concept	**Use case construct**	**Remarks**
System	System boundary	
System composition	UML class diagram	
System environment	All use case actors	
System decomposition	Include relationship	Within the context of include relationship, it will be considered as a system.
Thing	Actor	Actor only represents external environmental things.
Process	Use case	
Coupling	Association relationship	
Coupling	Generalization relationship	It enhances the behaviour of a use case with another.
Coupling	Extend relationship	It inherits the behaviour of a use case with another.

The results were tested with an empirical assessment which supported the feasibility and appropriateness of the proposed rules in the real-world context. The use of a theoretically sound modelling grammar for conceptual modelling would result in accurate conceptual models. In addition, we have used Bunge's ontological concepts to enhance the UML use case diagram thereby making it suitable to be used as a common modelling diagram in both conceptual and system modelling processes. The usage of a common modelling grammar for both these modelling processes would facilitate seamless integration between the two modelling phases, which would ultimately enhance the quality of the final IS.

One limitation of this study is the empirical assessment which was conducted with a set of university students. This cannot provide sufficient validity, because of their lack of in-depth industry experience in terms of IS development. Hence, an empirical assessment together with industrial experts has to be done. According to Weber (1997), the results of this kind of research study may vary depending on the ontological approach being used, the grammatical constructs that have been chosen, and the way how the mapping process is undertaken. Further Weber stated that these factors are subjective from person to person.

Finally, we briefly outline below the future research directions of this study. Since conceptual modelling is an inherently iterative process, some may argue that each iteration of the modelling process usually leads to improvement. Thus, further empirical assessments with different user groups are required to evaluate the pure improvements a use case model could achieve with the newly proposed rules. Such future assessment with industry experts would provide strong justification for the proposed changes to the use case diagram in a much more substantial way.

REFERENCES

Anglim, B., Milton, S., Rajapakse, J., and Weber, R. 2009. "Current Trends and Future Directions in the Practice of High-Level Data Modeling: An Empirical Study," in *Proceedings of the 17th Europian Conference of Information Systems*, pp. 1-13.

Bunge, M. A. 1977. Treatise on Basic Philosophy: Volume 3: Ontology I: The Furniture of the World, D. Reidel Publishing Company.

Evermann, J., and Wand, Y. 2005. "Ontology Based Object-Oriented Domain Modelling: Fundamental Concepts," *Requirements Engineering* (10:2), pp. 146-160.

Evermann, J., and Wand, Y. 2009. "Ontology Based Object-Oriented Domain Modelling - Representation Behaviour," *Journal of Database Management* (20:1), pp. 48–75.

Khoo, B. 2006. "Evaluating the Effectiveness of the Constructionist Approach to Object-Oriented Systems Analysis and Design Pedagogy," *Computer Information Systems*, (13:4), pp. 59-66.

Lee, P., Chen, D., and Chung, C. 1994. "An Object-Oriented Modelling Approach to System Software Design," *Information and Software Technology* (*36:*11), pp. 683-694.

Milton, S. K., Rajapakse, J., and Weber, R. 2010. "Conceptual Modeling in Practice: An Evidence-Based Process-Oriented Theory," in *Proceedings of the 5th International Conference on Information and Automation for Sustainability,* pp. 533-536.

Milton, S. K., Rajapakse, J., and Weber, R. 2012. "Ontological Clarity , Cognitive Engagement , and Conceptual Model Quality Evaluation : An Experimental Investigation," *Journal of the Association for Information Systems* (13:9), pp. 657-693.

Object Management Group. 2012. OMG Unified Modelling Language: Version 2.5 FTF – Beta 1, The Object Management Group, October.

Pieris, D., and Rajapakse, J. 2012. "Logical Database Design with Ontologically Clear Entity Relationship Models," in *proceedings of the IEEE 6th International Conference on Information and Automation for Sustainability*, pp. 209-214.

Rajapakse, J. 1996. *On Conceptual Workflow Specification and Verification,* The University of Queensland.

Rajapakse, J., and Orlowska, M. 1995. "Towards a Graphical Transactional Workflow Specification Language," in *Proceedings of Australian Systems Conference*.

Recker, J. C., Rosemann, M., Induska, M., and Green, P. 2009. "Business Process Modelling: A Comparative Analysis," *Journal of the Association for Information Systems* (10:4), pp. 333-363.

Rosemann, M., Recker, J. C., Indulska, M., and Green, P. 2006. "A Study of the Evolution of the Representational Capabilities of Process Modelling Grammars," *Advanced Information Systems Engineering* (4001:2006), 447-461.

Tilakaratna, P., and Rajapakse, J., (2012a). "Ontological Framework for Object-Oriented Analysis and Design," *American Journal of Engineering and Applied Sciences* (5:3), pp. 251-260.

Tilakaratna, P., and Rajapakse, J., (2012b). "Assessing the Completeness and Clarity of UML for Conceptual Modeling," *International Journal of Innovation, Management and Technology* (3:6), pp. 747-753.

Wand, Y., and Weber, R. 1993. "On the Ontological Expressiveness of Information Systems Analysis and Design Grammars," *Information Systems Journal* (3:4), pp. 217-237.

Wand, Y., and Weber, R. 1995. "On the Deep Structure of Information Systems," *Information Systems Journal* (5:3), pp. 203–223.

Weber, R. 1997. *Ontological Foundations of Information Systems*, Coopers & Lybrand and the Accounting Association of Australia and New Zealand.

Weber, R. 2003. "Conceptual Modelling and Ontology : Possibilities and Pitfalls," *Journal of Database Management* (14:3), pp. 1-20.

Weber, R., and Zhang, Y. 1996. "An analytical Evaluation of NIAM's Grammar for Conceptual Schema Diagrams," *Information Systems Journal* (6:2), pp. 147-170.

ACKNOWLEDGEMENT

We are indebted to Professor Ron Weber (Pro Vice-Chancellor – Monash University, South Africa), Mr. G. K. A. Dias (Head of the Department of Communication & Media Technologies – University of Colombo School of Computing, Sri Lanka), and Mr. H. D. Caldera (Retired Manager – Bank of Ceylon, Sri Lanka) for the helpful comments and reference suggestions given on this paper.

COPYRIGHT