Evaluating the Expressiveness of a Conceptual Model Represented in OntoUML and UML

Joselaine Valaski, Sheila Reinehr, Andreia Malucelli

PPGIa – Pontifical Catholic University of Paraná (PUCPR)
Curitiba – PR – Brazil

joselaine.valaski@pucpr.br, sheila.reinehr@pucpr.br, malu@ppgia.pucpr.br

Abstract. *The expressiveness of a conceptual model depends on the set of language symbols used for representation. UML is one of the most commonly used languages for representing conceptual models. However, issues remain regarding expressiveness that the language OntoUML proposes to resolve. Therefore, we performed an experiment involving eight professionals and eighty students to evaluate the expressiveness of both languages. The overall analysis showed that OntoUML was selected by the participants the most expressive language in 42% of the situations, while in 39% it was selected as having the same level of expressiveness as UML. After further analyses, we identified situations in which OntoUML was the most expressive.*

1. Introduction

Requirements elicitation is an activity that seeks to understand stakeholder needs, which are then transformed into software requirements (Pohl, 1997). However, some flaws in this activity exist. These derive from the fact that software engineering area attempts to visualize technology as a solution to a problem without first fully understanding the problem domain (Zanlorenci; Bunett, 1998). Conceptual modeling is the activity of formally describing aspects of the physical and social world in order to understand it fully (Mylopoulos, 1992). Thus, this activity is focused on modeling reality instead of modeling the computing system (Guizzardi, 2005). From this point of view, conceptual modeling can be an instrument that supports this activity of eliciting software requirements, because it aids comprehending a problem domain.

One of the best known conceptual metamodels is entity relationship (ER). However, the popularity of ER is also its main weakness (Castro, 2010): the metamodel is simple, despite the fact that this assists conceptual modelers. However, the metamodel is not highly expressive. UML is also a well-known language for building conceptual models, but it has the same problem of expressiveness (Guizzardi, 2005). The concepts from a universe of discourse are abstract entities that often exist only in the minds of users. To capture these concepts, they must be represented through concrete artifacts. This means a language must represent them in a concise, complete, and unambiguous manner. A language that has flaws of expressiveness may compromise understanding of requirements artifacts in later phases. According to Mylopoulos (1992), the suitability of a conceptual modeling notation is based on its contribution to the construction of models that represent reality, thus enabling a common understanding between their human users. In this regard, Guizzardi (2005) emphasizes using of languages with

ontologically well-founded primitives that help represent the reality of a problem's domain as precisely as possible.

Considering these issues, Guizzardi (2005) proposed OntoUML, which is a language used to represent ontology-based conceptual models. Because the language is ontology-based, the conceptual models constructed in OntoUML are assumed to be more expressive and to represent the real world of the domain more faithfully than do other languages of conceptual representation. The constructs proposed in OntoUML prevent the overload and redundancy found in other languages such as UML.

In his thesis, Guizzardi (2005) presents several specific situations in which the expressiveness of OntoUML is found to be superior to that of other languages, including UML. Although conceptual modeling is critical for an information system and software engineering (Guizzardi & Wagner, 2012), (Melo & Almeida, 2014), few studies have been conducted in this area that examine issues of expressiveness between OntoUML and UML. Therefore, this study evaluated two conceptual models, those represented in OntoUML and UML. Both models represented the same context. They were constructed by specialists in each language and evaluated by professionals and students. The results revealed situations in which OntoUML is more expressive and others in which the two languages showed equal levels of clarity. The results thus revealed the benefits of using OntoUML for conceptual modeling in eliciting software requirements.

The remainder of this paper is organized as follows. In Section 2 we present some basic concepts related to OntoUML. In Section 3, we present our research method. Section 4 discusses the results of our experiment. Section 5 includes final considerations and indication for future studies.

2. OntoUML

OntoUML was proposed by Guizzardi (2005) based on the need for an ontology-based language that would provide the necessary semantics to construct conceptual models using concepts faithful to reality. The classes proposed in OntoUML are representations of the Unified Foundational Ontology (UFO) constructs. These constructs are represented using UML stereotypes.

In this study, only the main constructs that comprise the object type category are presented (Guizzardi et al., 2011). In this category, constructs are more closely related to the static conceptual modeling of a domain. The hierarchical structure of these models is presented in Fig. 1. The object type constructs may be sortal and non-sortal. Sortals provide identity and individuation principles to their instances, whereas non-sortals do not supply any clear identification principles. Sortal constructs are classified as rigid and anti-rigid sortals. A sortal is said to be rigid if it is necessarily applied to all its instances in all possible worlds. A sortal is said to be anti-rigid if it is not necessarily applied to all its instances. Rigid sortals include kind and subkind categories. A kind is a rigid sortal and thus has intrinsic material properties that provide clear identity and individuation principles. It determines existentially independent classes of things or beings and are said to be functional complexes. A subkind is also a rigid type that provides an identity principle and has some restrictions established and related to the kind construct. Every object in a conceptual model must be an instance of only one kind.

Two sub-categories of anti-rigid sortals exist: phases and roles. In both cases, instances may change their types without affecting their identities. During the phase construct, changes may occur as a result of changes to intrinsic properties. By contrast, in the role construct, changes occur because of relational properties.

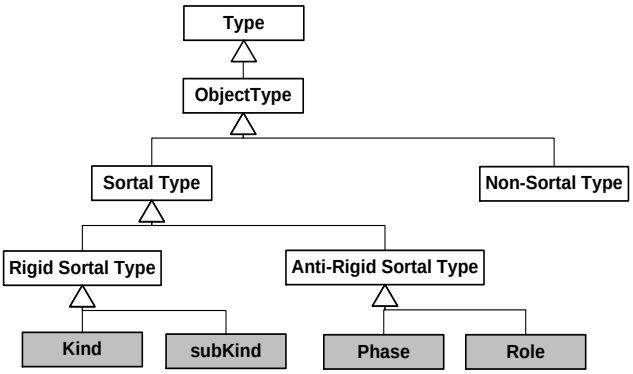

Fig. 1 Fragment of a metamodel (Guizzardi, 2005)

3. Research Method

This section describes the phases of our experiment conducted to evaluate the expressiveness of the OntoUML and UML languages in a specific context.

3.1 Selection of the Domain Description

The first step in our experiment consisted of defining a context for the construction of the conceptual model. The objective was to select an uncommon domain, that is one not commonly known (e.g., a library, a university.) with a smaller scope to lend feasibility to the experiment. We believe that an uncommon domain brings more discussion to find their concepts and relationships.

In accordance with these criteria, the software requirement specifications for an electronic proxy software program were obtained from specialists in the domain. Based on these specifications, a description of the main software features was written. This description is presented in Table 1.

Table 1. Domain description

Text
Only the organization's representative can grant an electronic proxy.
An organization may have one or more representatives.
To allow the grantor to indicate an active user in the Receita-PR database to grant the condition of the grantee.
Only one grantee per proxy.
Only one proxy per grantor and the same grantee.
The granting of a proxy is restricted to organizations with a record in the ICMS database.
To display the services to be granted.
To display a list of organizations (in which the grantor is the organization's representative) to be granted.
To select the organization allowed to perform all services.
To allow the grantor to revoke a proxy.

3.2 Construction of a Conceptual Model in OntoUML

Based on the scope defined in Section 3.1, specialists were selected to construct a conceptual model in OntoUML. As OntoUML is still not a widely used language on the market, few specialists in this language exist. One of the groups trained for this task is the Ontology and Conceptual Modeling Research Group (NEMO). This group works on research related to ontologies as well as OntoUML, and is led by Professor Giancarlo Guizzardi, the creator of OntoUML. Considering their competence in this activity, an e-mail was sent to the NEMO group with a description of the domain (Table 1), and the construction of the respective conceptual model was requested. The constructed model was a collaboration of the three members of the group. Some e-mails were exchanged between the researchers and the specialists until a consensus was reached on the representation of the model.

3.3 Construction of the Conceptual Model in UML

For constructing the conceptual model in UML, three specialists in the language were selected, all of whom held advanced degrees in the field of software engineering and had professional industry and academic experience. The description of the domain (Table 1) was sent through e-mail to each specialist with a request to construct a conceptual model based on the description. E-mails were exchanged between the researchers and the specialists until a consensus was reached on the representation of the model.

3.4 Evaluation of the Expressiveness of the Conceptual Models

The objective of this phase was to evaluate the expressiveness of the two conceptual models constructed by the specialists (OntoUML and UML). Twelve statements were derived from these models. Using these statements, an instrument was prepared to evaluate if the statements were more clearly represented in the conceptual model in OntoUML or UML, or whether both languages exhibited the same level of clarity. The instrument created for the evaluation is wholly included in Appendix A.

After the instrument was prepared, a profile for the participants in the evaluation was defined. Two distinct groups were selected, the first composed of eight professionals educated in the field of computing with experience in UML modeling, and the second group consisted of eighty students from undergraduate courses in the field of computing. The experiment was performed only with classes that had already completed the course on UML. Neither group (i.e., neither professionals nor students) had prior knowledge of OntoUML. The experiment was first performed with the group of professionals, a smaller and more experienced group that could validate the instrument. Suggested improvements and corrections could thus be collected for later use with the group of students. One of the improvements applied to the students was the creation of two models of the instrument. In the first model (Model 1), UML appeared as the first option in the list and this UML model appeared as Attachment 1. In the second model (Model 2) (see Appendix A), OntoUML appeared as the first option in the list and the OntoUML model appeared as Attachment 1. These were necessary to eliminate any bias related to the order in which options in the list and models were presented. Thus, Models 1 and 2 were distributed in alternation to the participants.

4. Results and Discussion

In this section, results are presented and discussed. First, the results concerning the construction of the conceptual models by specialists are presented; afterwards results on the evaluation of the expressiveness of the models by professionals and students.

4.1 Conceptual Model in OntoUML

Fig. 2 presents the final conceptual model constructed by specialists in OntoUML. To finalize this version, these specialists asked researchers four rounds of questions to solve doubts.

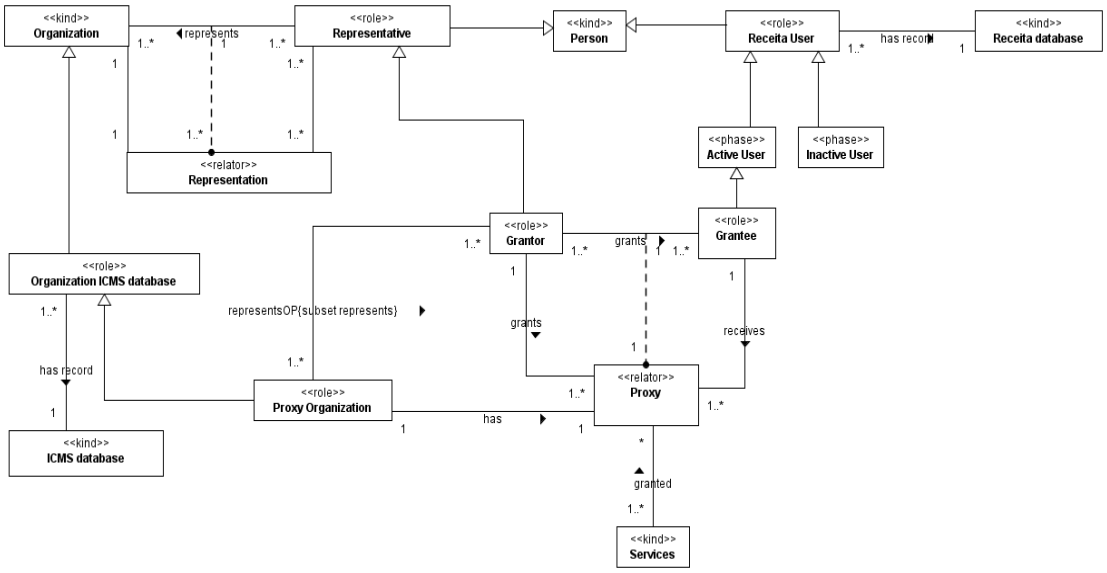

Fig. 2 Conceptual model in OntoUML

The specialists in OntoUML revealed that the language's richer nature generated several questions concerning the domain, even after the scope was sent. The specialists also noted that much of the information that is implied in a model must become explicit when OntoUML is used. In all rounds, specialists revealed information that should be included in the description of the scope so that creating a final model would be possible. Much of the information was implied.

In this experiment, we observed that a high degree of formality and consistency in OntoUML generated a variety of questions that perhaps would not occur with other languages. This feedback reinforces the belief that conceptual models in OntoUML may lend positive support to eliciting software requirements.

4.2 Conceptual Model in UML

Fig. 3 presents the final conceptual model constructed by specialists in UML. The questions from the specialists were different in this case. Specialist 1 delivered the version of UML with no questions to clarify. Specialist 2 asked a round of questions and delivered the version. Finally, Specialist 3 asked two rounds of questions and delivered the version.

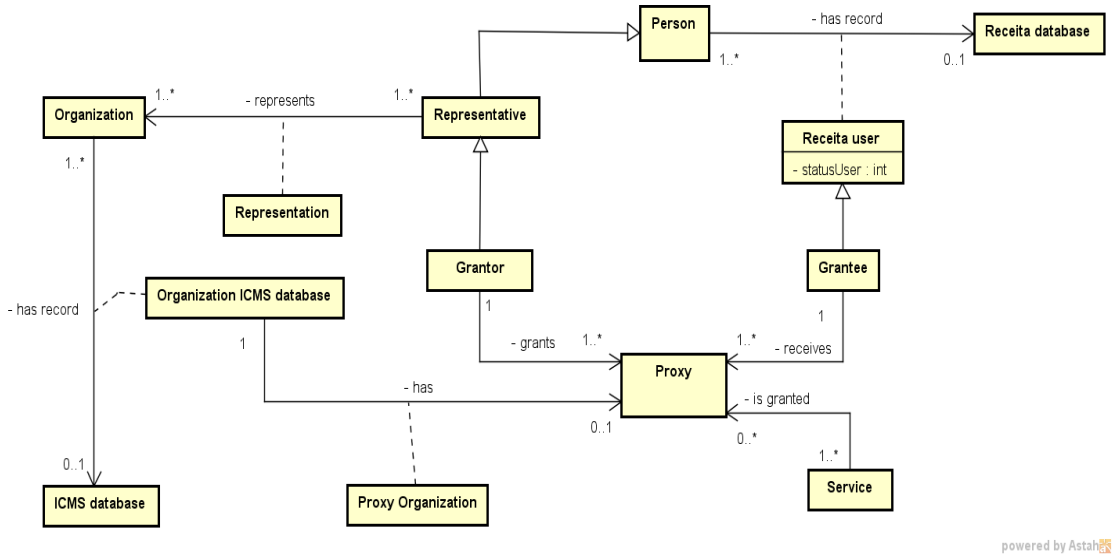

Fig. 3 Conceptual model in UML

The three versions delivered differed considerably. Possible reasons for this include the lower degree of formality of the language allows for distinct representations of the same context, and the lack of semantic restrictions does not encourage questioning during construction. In the delivered versions, a representation focused on data persistence in a software program instead of on the concepts of a domain. This bias may be indicative of the lack of use of conceptual models in UML for understanding a domain. These observations should be studied in greater depth in future studies.

The version delivered by Specialist 3 was the closest to the representation of the scope. An in-person meeting was held among specialists to complete the final version presented in Fig. 3. With the two conceptual models (UML and OntoUML) constructed by the specialists, the next phase of the experiment was to evaluate the expressiveness of the models. The results are presented as follows.

4.3 Expressiveness of the Conceptual Models Constructed

First, results are presented for the pilot experiment performed with the professionals. Table 2 presents an overview of the results. Considering that eight professionals evaluated twelve statements, ninety-six choices were derived. Among these choices, twelve (13%) indicated UML the most expressive, forty indicated OntoUML (42%) the most expressive, and forty-four (46%) indicated the languages exhibited the same level of clarity.

Table 2. Consolidated results of the choices by professional group

Language	Number of Choices	Percentage
UML	12	13%
OntoUML	40	42%
Both	44	46%
Total	96	100%

Based on these initial results, we observed situations in which the languages exhibit the same level of clarity, and others in which OntoUML exhibits a greater level of clarity than UML. Only some situations occurred in which UML was more expressive. Considering this first result, each statement was analyzed to identify the situations in which the languages stood out. Fig. 4 presents the results for each statement.

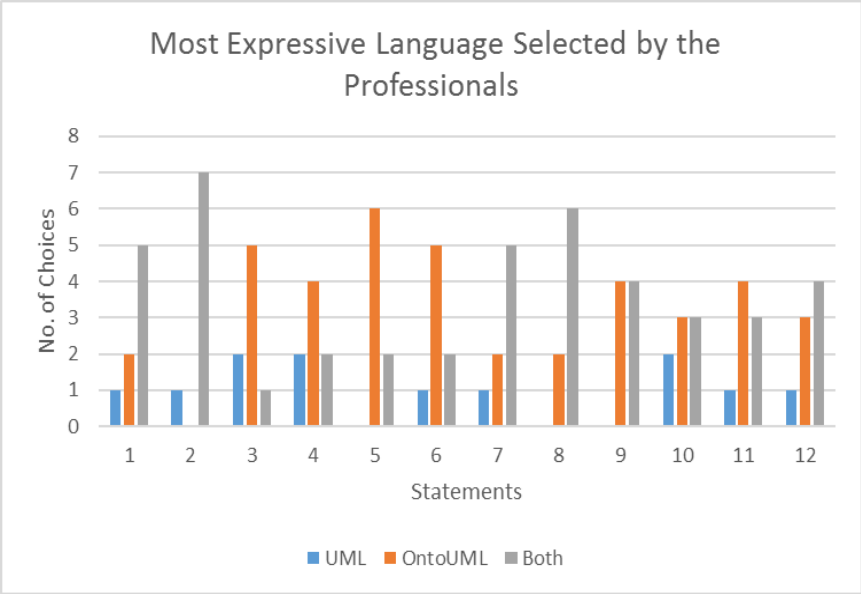

Fig. 4 Number of choices by professionals for each of the twelve statements

Fig. 4 shows that for Statements 1, 2, 7, 8, and 12, both languages exhibited the same level of clarity. For Statements 3, 4, 5, 6, and 11, OntoUML exhibited a greater level of clarity. Statements 9 and 10 revealed that OntoUML and both languages exhibited equal clarity. There has not been any statement in which UML had been the preferred one.

Since the professionals had more practical experience with modeling, they may have had a different viewpoint than students from the field of computing, who have not yet had considerable practical experience. Thus, the same experiment was performed with students from different computing majors to identify their perceptions relative to the expressiveness of the models.

Table 3 presents an overview of the results from the students. Eighty students evaluated twelve statements, thus totaling nine hundred and sixty choices. Among these choices, one hundred and eighty-one (19%) indicated UML the most expressive language, four hundred and six (42%) indicated OntoUML the most expressive, and three hundred and seventy-three (39%) indicated that the languages exhibited the same level of clarity. The perception of the students, despite having less knowledge about modeling, was very similar to those of the professionals. The students also identified situations in which the two languages exhibited the same level of clarity, as well as situations in which OntoUML exhibited a greater level of clarity than did UML. Only some situations occurred in which UML was indicated the most expressive.

Table 3. Consolidated results of the choices by students group

Language	Number of Choices	Percentage
UML	181	19%
OntoUML	406	42%
Both	373	39%
Total	960	100%

Table 3 presents the overall results for the eighty students. However, because these are distinct groups (i.e., with different majors and class schedules) an analysis of each class was also performed. Table 4 presents these individualized results. In addition, Table 4 lists the major, the current semester of each student, and the number of participating students. All classes agreed that OntoUML was more expressive for the majority of statements. The exception was Class 3 in which OntoUML and Both got the same percentage (45%). No classes considered UML to be the most expressive overall. However, the perception of Class 1 and 5, showed a considerable difference relative to UML: 6% and 29%, respectively. In other words, Class 5 considered UML more expressive than OntoUML in at least 29% of the situations analyzed, whereas Class 1 considered UML more expressive in only 6% of the situations. This difference may be related to the extent of student knowledge of UML. However, the reasons behind their decisions cannot be determined only based on the results of this experiment.

Table 4. Choice of languages by students by class

ID	Major	Semester	No. of Students	UML	OntoUML	Both
Class 1	Computer Science	5th	12	6%	49%	45%
Class 2	Information Systems	7th	12	22%	46%	32%
Class 3	Computer Engineering	7th	17	10%	45%	45%
Class 4	Information Systems	6th	9	14%	45%	41%
Class 5	Technology Analysis and Systems Development	5th	30	29%	36%	35%

As it happened with the professionals, the experiment with the students yielded statements in which OntoUML was the most expressive and other statements in which Both (OntoUML and UML) had same level of clarity. Thus, the results per statement were evaluated. The overall results are presented in Fig. 5, which shows that for Statements 1, 2, 7, 9, and 12, the two languages exhibited the same level of clarity. For Statements 3, 4, 5, 6, 8, 10, and 11, OntoUML exhibited a greater level of clarity. No statements were identified in which UML was most frequently selected.

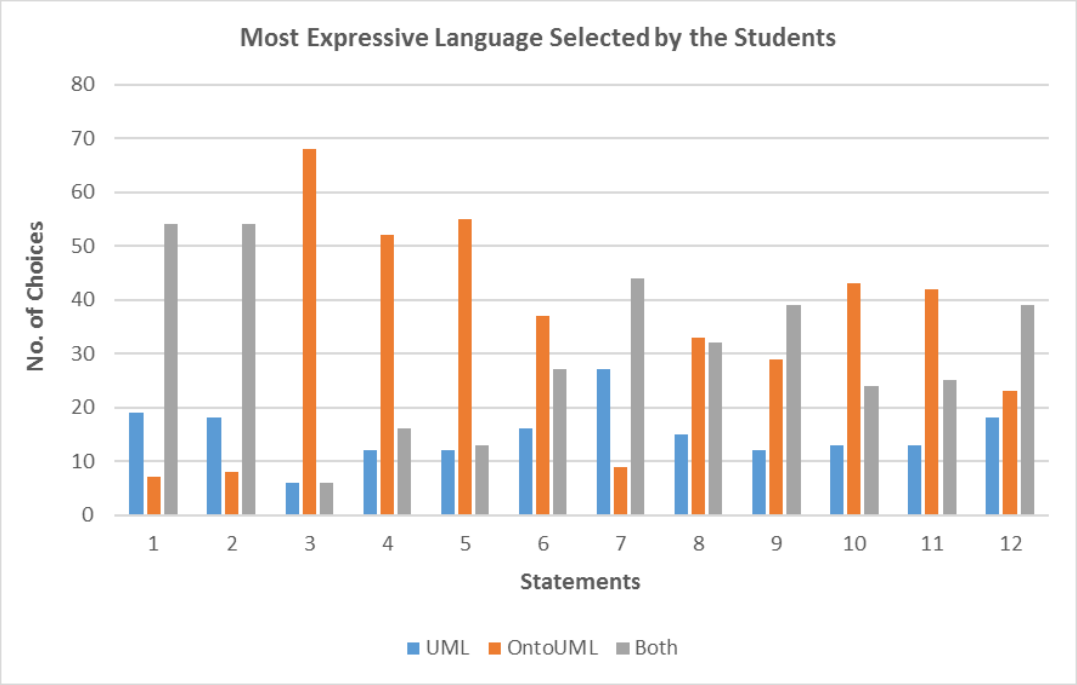

Fig. 5 Number of choices by students for each of the 12 statements

These results indicate situations in which OntoUML is more expressive than UML and situations in which the two exhibit the same level of clarity. To better understand these situations, we examined statements indicating consensus that OntoUML was more expressive. The results were grouped by professionals, students (all eighty), and class. This grouping is presented in Table 5. Table 5 reveals consensus for Statements 3, 4, 5, 6, 8, 10, and 11 (in gray) in which the representation in OntoUML was the most expressive.

Table 5. Selection of the most expressive language by statement

Statement	Professionals	Students	Class 1	Class 2	Class 3	Class 4	Class 5
1	*Both*	*Both*	*Both*	*Both*	*Both*	*Both*	*Both*
2	*Both*	*Both*	*Both*	*Both*	*Both*	*Both*	*Both*
3	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**
4	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**
5	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**
6	**OntoUML**	**OntoUML**	*Both*	**OntoUML**	*Both*	**OntoUML**	**OntoUML**
7	*Both*	*Both*	*Both*	*Both*	*Both*	*Both*	*Both*
8	**OntoUML**	**OntoUML**	**OntoUML**	*Both*	**OntoUML**	*Both*	*Both*
9	**Tie**	*Both*	**OntoUML**	*Both*	*Both*	*Both*	*Both*
10	**Tie**	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**	*Both*	**OntoUML**
11	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**	**OntoUML**	*Both*	**OntoUML**
12	*Both*	*Both*	*Both*	**OntoUML**	*Both*	**OntoUML**	*UML*

In OntoUML, the Role construct was used to represent the concept in Statements 4, 5, 6, 10, and 11. Specifically, OntoUML used the role construct to establish that it is relationally dependent on a universal concept, which carries the principle of identity and individuation. Representing a relationship of specialization is then required. In addition, in UML, because of the lack of semantic restrictions, the concepts for these same statements were represented by means of associative relationships, in which the origin of the concept is unclear. This finding became clearer when we evaluated Statement 9. In this statement, the perceptions of the participants were identical. Although in OntoUML, the role construct was also used, in UML a specialization was employed to represent the concept, which, based on the perceptions of the participants, resulted in the same level of expressiveness.

In Statements 3 and 8, the Relator construct was used. This construct allows the multiplicities of a specific relationship to be expressed. In UML, associative relationships were used. These do not allow for the expression of multiplicity in a certain relationship. For example, in UML, representing that a relationship between the same grantor and proxy occurs only once is not possible, even though a grantor may be associated with various proxies and a proxy may be associated with various grantors. Guizzardi (2005) discusses this deficiency in UML and in other languages.

We believe that if the participants knew OntoUML and the meaning of its constructs, the results would have been even more positive. One example is the representation of Statement 9. OntoUML can represent the fact that it is not sufficient for the grantor to be the representative of an organization, but that the grantor must be the representative from the same organization referenced in the proxy. In UML, only the grantor as the representative of an organization is represented, and this may not necessarily be the organization referenced in the proxy.

5. Conclusion

Conceptual models are considered crucial instruments to achieve consensus and understanding of a domain. Thus, conceptual models are allies that support requirements elicitation in unknown domains. However, the use of a certain language to represent the model may undermine its expressiveness. UML is one of the most commercially popular languages. However, according to Guizzardi (2005), flaws exist in terms of its expressiveness. OntoUML is a more academic language, and is designed to correct flaws of expressiveness in languages such as UML. Although Guizzardi (2005) discussed several specific situations in which OntoUML is more expressive, other studies have not been conducted that evaluate the perceptions of professionals and students regarding the expressiveness of OntoUML.

The objective of this study was to collect these perceptions and identify situations in which OntoUML is more expressive in an information systems context. Although our participants lacked knowledge of the constructs of OntoUML, overall it was considered more expressive than UML. In addition, various situations occurred in which consensus was reached between the participating groups that OntoUML better represents certain concepts. In addition, when conceptual models were constructed by specialists during our experiment, OntoUML was determined to have a high degree of formality and consistency. Our study showed that OntoUML causes modelers to

question the situations of a domain that are not explicit. Thus, models that are more consistent and faithful to reality were built.

These results reinforce the need for a conceptual model represented in OntoUML to support software requirements elicitation. This research can evolve many different directions. One is developing a computational environment to support constructing a conceptual model in OntoUML. This conceptual model can then support the derivation of functional software requirements. Some results were present in Valaski et al. (2014)

Appendix A. Instrument: Model 2

Name: _____

Read the statements given below that was extracted from the domain electronic proxy. Analyze the corresponding representation in the conceptual model (Attachment 1 - OntoUML and Attachment 2 - UML) and enter **X** for the model that best represents (represents most clearly) what it is being affirmed.

1. An **Organization** may have one or more representatives.

Option 1 – It is clearer in the **OntoUML** model.	
Option 2 – It is clearer in the **UML** model.	
Option 3 – Both exhibit the same level of clarity.	

2. A **Representative** is a person that represents one or more organizations.

Option 1 – It is clearer in the **OntoUML** model.	
Option 2 – It is clearer in the **UML** model.	
Option 3 – Both **present** the same level of clarity.	

3. A **Representation** is a relationship established between an organization and one or more representatives.

Option 1 – It is clearer in the **OntoUML** model.	
Option 2 – It is clearer in the **UML** model.	
Option 3 – Both **present** the same level of clarity.	

4. **Organization ICMS Database** is an organization having a record in the ICMS database.

Option 1 – It is clearer in the **OntoUML** model.	
Option 2 – It is clearer in the **UML** model.	
Option 3 – Both **present** the same level of clarity.	

5.**Receita User** is a person having a record in the Receita database.

…..

6. **Grantee** is an active user in the Receita database who receives one or more proxies.

7. **Grantor** is an organization's representative who grants one or more proxies.

8. The **Proxy** relationship between the same grantor and grantee occurs only once.

9. The **Grantor** of a proxy must be the representative of the organization associated with the proxy.

10. **Proxy Organization** is an organization associated with a proxy.

11. **Proxy Organization** has a record in the ICMS database.

12. **Proxy** is a relationship established between one grantor, one grantee, one organization, and one or more services.

References

Castro, L. (2010) "Abordagem Linguística para Modelagem Conceitual de Dados com Foco Semântico", Msc Dissertation, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, Brazil.

Guizzardi, G. (2005) "Ontological Foundations for Structural Conceptual Models," Telematica Institut Fundamental Research Series 15, Universal Press.

Guizzardi, G.; Wagner, G. (2012) "Conceptual Simulation Modeling with OntoUML". Proceedings of the 2012 Winter Simulation Conference.

Melo, S., Almeida, M. B. (2014) "Applying Foundational Ontologies in Conceptual Modeling: A Case Study in a Brazilian Public Company". Access in 20 jun 2016, available in: <https://www.semanticscholar.org/paper/Applying-Foundational-Ontologies-in-Conceptual-Melo-Almeida/91b8f1648480a195f4ae6307741c2a76a11d2c78/pdf>

Mylopoulos, J. (1992) "Conceptual modeling and Telos", In P. Loucopoulos and R. Zicari, editors, Conceptual modeling, databases, and CASE. Wiley.

Pohl, K. (1997) "Requirements engineering: An overview". In Encyclopedia of Computer Science and Technology. A. Kent, and J. Williams, Eds. Marcel Dekker, New York, NY, v. 36, suppl. 21.

Valaski, J., Reinehr S. and Malucelli, A. (2014). "Environment for Requirements Elicitation Supported by Ontology-Based Conceptual Models: A Proposal". In Proceedings of the 2014 International Conference on Software Engineering Research and Practice (SERP'14), ISBN 1-60132-286-0, Las Vegas, USA, p. 144-150.

Zanlorenci, E. P.; Burnett, R. C. (1998) "Modelo para Qualificação da Fonte de Informação do Cliente e de Requisito Funcional," In Workshop em Engenharia de Requisitos.