Deriving Domain Functional Requirements from Conceptual Model Represented in OntoUML

Joselaine Valaski, Sheila Reinehr and Andreia Malucelli
Pontifical Catholic University of Paraná (PUCPR), Imaculada Conceição Street, 1155, Curitiba, Brazil

Keywords: Requirements Engineering, OntoUML, Conceptual Model, Functional Requirement.

Abstract: A conceptual model is an artifact that helps to understand a domain and therefore, may contribute with the elicitation of related functional requirements. However, the expressiveness of this model depends on the expressiveness of the language used. Considering that OntoUML is a language that proposes elements that allow more semantics, it is possible to build models with better expressiveness which are more complete than, for instance, models represented in UML language. For evaluating the possibility of deriving domain functional requirements (DFR) from models represented in OntoUML, a heuristic was proposed. This heuristic was obtained by reading and interpreting nine conceptual models represented in OntoUML. Once the heuristic was obtained, it was applied in a systematized manner to six models. According to the results obtained, using a conceptual model represented in OntoUML as a source to derive DFR is possible. In addition to the identification of the DFR, the heuristic can identify possible faults in the model design, or even the incompleteness of the model.

1 INTRODUCTION

Software Requirements Engineering (SRE) can be defined as an iterative process of discovery and analysis for producing a clear, complete, and concise set of specifications about the software to be developed (Robinson & Pawlowski, 1999; Loucopoulos & Karakostas, 1995) During this process, artifacts (documents, models, prototypes, etc.) are built for the analysis of the domain and design of the software that will be developed. However, owing to changes in requirements or in design and development decisions, there may be erosion among these artifacts, and therefore, the loss of conciseness and traceability between them (Landhäußer et al., 2014).

One of the possible reasons for requirement changes is the lack of understanding of the problem's domain for which the software will be developed. In the early stages of SRE, the specification of the software to be developed is often inaccurate and inconsistent (Ding & Marchionini, 1997). Requirement engineers' lack of understanding of the business and communication breakdown among the stakeholders compromise the quality of information (Jureta et al., 2010). Intensifying efforts to better understand the domain before moving on to software design and development is a practice that may minimize future requirement changes.

One of the ways of understanding a problem is to build conceptual models (Jalote, 1997). Conceptual models have been an important resource not only for requirement elicitation, but also for improving the model transformation through the software phases (Valaski et al., 2016). However, good models need good modeling languages (Henderson-Sellers et al. 2015). A language that has flaws in expressiveness may compromise the understanding of requirement 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. Henderson-Sellers et al. (2015) discuss some of the most common problems about software engineering modeling languages. Based on these problems they claim to use a language with an ontological commitment. Languages with an ontological commitment can improve the expressivity and quality of models (Valaski et al., 2016).

In this regard, Guizzardi (2005) emphasizes the use of languages with ontologically well-founded

Valaski, J., Reinehr, S. and Malucelli, A.
Deriving Domain Functional Requirements from Conceptual Model Represented in OntoUML.
DOI: 10.5220/0006290302630270
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 2*, pages 263-270
ISBN: 978-989-758-248-6

primitives that help represent the reality of a problem's domain as precisely as possible. Guizzardi (2005) proposed OntoUML, a language used to represent ontology-based conceptual models. Because the language is ontology-based, the conceptual models constructed in OntoUML are assumed more expressive and to represent the real world of the domain more faithfully than do other languages of conceptual representation do. There are practical situations where OntoUML is more expressive than UML (Teixeira et al., 2014; Valaski et al., 2016b).

Considering that conceptual models represented in OntoUML allow a better representation of a reality, we believe they are an important instrument to identify the first requirements. In this context, the main goal of this study is to evaluate the possibility of using conceptual models represented in OntoUML as a support to derive domain functional requirements (DFR). A DFR is a denomination used in this study to refer to high level functional requirements generated from the representation of the Problem Domain. The present study is organized as follows: Section 2 presents the main OntoUML concepts. Section 3 describes the method applied to obtain the heuristic for the DFR reading and listing from the OntoUML model. Section 4 presents and discusses the result of the heuristic application. Section 5 discusses the limitations of this study. Section 6 presents the landscape of related studies. Final considerations are presented in Section 7.

2 OntoUML: BACKGROUND

This section presents a few of the main OntoUML language concepts because presenting all OntoUML language constructs is not possible owing to space limitation. 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, 2005). In this category, constructs are more closely related to the static conceptual modeling of a domain. The Figure 1 shows a fragment of a metamodel OntoUML related to Universals constructs. Universals are constructs related to types (classes) while Individuals are constructs related to instances. The main constructs of the metamodel OntoUML are presented in the follow subsections.

2.1 Substantial

Substantial constructs are applied to represent classes of elements that have high degree of independence. Substantials are specialized in Sortal and Non-sortal. Sortals provide identity and individuation principles to their instances, whereas Non-sortals do not supply any clear identification principles.

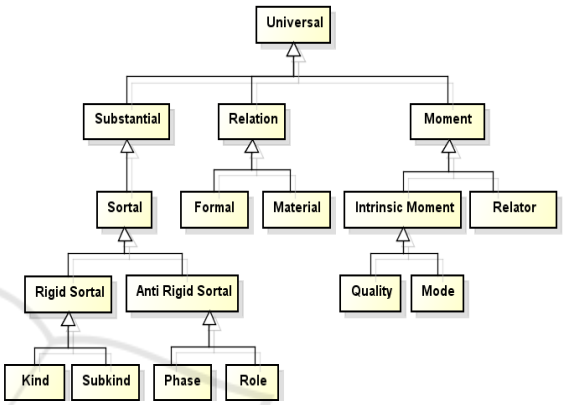

Figure 1 : Fragment of a metamodel OntoUML related to Universals (Guizzardi, 2005).

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 and anti-rigid if it is not necessarily applied to all its instances. Person is an example of Rigid Sortal and Student is an example of Anti-rigid Sortal.

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, for instance Person. 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. Man and Woman are examples of Subkind category. 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 because of changes to intrinsic properties. Teenager and Living Person are examples of Phase category. By contrast, in the Role construct, changes occur

because of relational properties. Student and Husband are examples of role category.

2.2 Moment

Whereas the Moment is a construct used to represent classes of elements which are existentially dependent of other individuals. The Moment class is divided into two categories, Intrinsic Moment and Relator.

The Intrinsic Moment class represents properties that depend only on one individual. The Quality and Mode Universal classes are specializations of the Intrinsic Moment class. Weight, Color and Height are examples of a Quality Universal class. Whereas, Thoughts and Symptoms are examples of a Mode class. The Relator class represents individuals who depends of at least two distinct entities. Sale and Registration are examples of adherent concepts to this class.

2.3 Relation

The Relation construct represents relation categories that may occur between Moments and Substantials. The Relation class is divided into two general categories: Material Relation and Formal Relation.

The Material Relation class has relations mediated by a Relator. Relators (Moments) are individuals with the power of connecting entities. The Formal Relation class represents relations between two or more direct entities without individual mediation. Inherence and association are considered Formal Relations. The Formal Relation can be divided into basic formal and formal comparative relations. For the basic formal relations, three categories are proposed: Characterization, Mediation and Derivation. The Formal Characterization relation occurs between a Mode and a Universal; there is no optional property in this relation. A Formal Mediation relation occurs between a Relator and a Substantial. A Derivation relation is the one between a Material and a Relator from which this relation is derived.

3 INTERPRETING OntoUML MODEL

A method was defined with the purpose of obtaining a heuristic able to extract possible DFRs in a systematized manner from a conceptual model represented in OntoUML.

During the process of obtaining the heuristic, three main activities were executed: the selection of conceptual models represented in OntoUML language, transcription and identification of patterns in the interpretation of the models, and lastly, the definition of the heuristic. Further details on these activities are presented next.

3.1 Selection of Conceptual Models

Initially, the selection of conceptual models that represented a domain was performed. The OntoUML Model Repository was used to obtain such models. The OntoUML Model Repository (http:// www.menthor.net/model-repository.html) is an endeavor made by Menthor to put in one place all the OntoUML models scattered around the web, conferences, journals and books. In total, nine models were found. The represented domain and number of existent elements of the models are summarized in Table 1.

Table 1 : Quantity of elements of conceptual models by domain.

Stereotype	Domain/Quantity								
	A	B	C	D	E	F	G	H	I
Category	0	0	0	1	0	0	0	0	0
Collective	0	0	0	7	2	1	1	2	0
Kind	5	4	4	6	2	7	6	6	3
Mixin	0	0	0	0	0	0	1	0	1
Mode	0	0	2	0	0	0	0	0	0
Phase	2	0	2	5	3	2	5	0	2
Relator	2	6	1	1	3	14	3	3	5
Role	6	2	2	2	4	15	9	15	18
Role Mixin	0	0	1	0	0	0	0	0	2
Subkind	0	0	2	2	0	4	17	3	4
Characterization	0	0	2	0	0	0	0	0	0
Formal	0	0	2	4	0	1	1	0	0
Material	3	0	0	6	6	1	1	7	2
Mediation	6	13	2	2	6	37	7	6	15
Generalization	8	2	8	14	6	8	36	18	30
Unknown	0	0	0	0	0	1	2	0	3
Total	32	27	28	50	32	91	89	60	85
Domain: A: Electronic Proxy; B: Route Bus; C: Project Management; D: Health Organization; E: Conference; F: Library; G: Music; H: Online Mentoring; I: School Transportation									

The class stereotypes: enumeration, nominal quality, non-perceivable quality, and perceivable quality and

quantity, were not evaluated because they were not used in any of the nine models evaluated. The associations stereotypes: componentOf, derivation, memberOf, subColletionOf, and subQuantityOf, were not included then, as they did not present significant facts for this top-level DFR survey step. However, these should be considered in future evaluations.

3.2 Transcription and Identification of Patterns

For each conceptual model, the reading and manual transcription of the interpreted data was performed. The aims of the reading were a) to identify a rule that allowed navigation through all model elements with no repetition; b) to transcribe the interpretation in the reading; and c) to identify design patterns to define a systematized heuristic. After many readings and transcriptions, some patterns were identified, which are summarized as follows:

- <<Relator>> is a class stereotype that always groups the main domain functionalities. If the reading always starts by the Relators, it is possible to define a flow that runs through all elements of the model. Owing to this characteristic, the key word “control” was attributed to describe the requirement;
- <<Category>>, <<Collective>>, <<Kind>>, <<Mixin>>, and <<Subkind>> are class stereotypes associated with entities that require maintenance functionalities; therefore, the key word “maintain” was attributed to describe the requirement;
- <<Mode>> and <<Phase>> are class stereotypes that require updating data from existing entities. For the description of requirements associated with these elements, the key word “inform” was attributed;
- <<Role>> and <<Role Mixin>>: at first, these class stereotypes no need to directly describe a functional requirement, because the identification of the relation between these elements will generate a functional requirement;
- The association relationship (<<Characterization>>; <<Formal>>, <<Material>>, <<Mediation>>) generate requirements to represent the association between two elements (root and node). For this situation, the key word “association” was attributed; and
- The relationship of generalization does not generate DFR when is related to a class that uses the stereotypes: <<Category>>, <<Collective>>, <<Kind>>, <<Mixin>> and <<Subkind>>. For other situation, the key word “association” was attributed.

3.3 Definition of the Heuristic

Based on the patterns mentioned in Section 3.2, the heuristic was defined with the purpose of generating the possible DFR.

In the proposed heuristic, all classes that use stereotype <<Relator>> in the model are selected and stored in an array. For each of these classes, a domain functional requirement is generated. The classes that use a stereotype <<Relator>> leads to the identification of the relations between the dependent elements. For each relation, a domain functional requirement is generated. A domain functional requirement is also generated according to the dependent element (following the previously mentioned patterns).

To illustrate the heuristic functions, a sketch of the algorithm is presented in Algorithm 1. It is important to emphasize that this algorithm does not represent the implemented and complete version of the proposed heuristic. Several rules were also added to the algorithm to allow the systematized reading of the model. The main rules are presented as follows:

- In the selection (DependElements) of the dependent elements (classes and associations), if the root element is a class that uses the stereotype <<Relator>> and the node element is related to the root element through an association that uses the stereotype <<Material>>, this path is not selected, since the association that uses the stereotype <<Mediation>> meets the functional requirement associated;
- In the selection of dependent elements, if the root element is a class that uses the stereotype <<Role>> and the node element is a generalization related to another class that also uses the stereotype <<Role>>, this path is not selected. If the model design is correct, another path (from another class that uses stereotype <<Relator>>) will reach the specializations of the class that uses the stereotype <<Role>>;
- The recursive call for print requirements verifies if the node element is a class that uses the stereotype <<Relator>>, if yes, the recursive call is not performed. This rule is applied because all relators have already been selected in the Main() procedure. This rule guarantees the extraction of requirements from the most relevant domain functionalities. The recursive call is executed

until another class that uses the stereotype <<Relator>> is reached or there are no more dependent elements in the covered path. With this rule it is possible to generate groups with the requirement and their dependencies; and

- The recursive call for print requirements verifies if the node element has already been covered. If yes, the recursive call is not performed. This rule avoids the duplication of the covered path.

```
procedure Main ()
begin
  relator = SelectAllRelator();
  for each relator do
    Writeln('RF .. control' +
                  relator.name);
    PrintRequirements(relator);
  end-for
end.
procedure PrintRequirements(rootElement)
begin
  nodeElement=DependElements(rootElement);
  for each nodeElement do
    Writeln('RF .. association' +
      nodeElement+ to'+ rootElement.name);
    if nodeElement.type in ['Category',
        'Collective', 'Kind', 'Subkind',
        'Mixin']
      Writeln('RF .. maintain'+
                    nodeElement);
    end-if
    if nodeElement.type in ['Phase,
                            'Mode']
      Writeln('RF .. inform'+
                  nodeElement);
    end-if
    if nodeElement.type not in ['Relator´]
          or not ExistElement(nodeElement)
      AddElement(nodeElement);
      PrintRequirements(nodeElement);
    end-if
  end-for
end.
Continued ...
```

Algorithm 1 : Partial algorithm to read and extract domain functional requirements from OntoUML conceptual models.

4 DERIVATION OF DFR

With the heuristic defined in Section 3, its systematized execution was evaluated. The heuristic was applied in six conceptual models related in Table 1 (A: Electronic Proxy; B: Route Bus; C: Project Management; D: Health Organization; E: Conference; F: Library). The data presented in Table 1 indicates that the six selected models represent different complexities (number of elements), and the use of the different elements considered by the heuristic.

4.1 DFR Listed by the Heuristic

Table 2 partially presents the list of DFR generated by the Conference (ID: E) domain model. The sequence of the requirement identifier helps to verify the dependence and source, among other requirements.

Table 2 : Partial DFR of Conference domain (ID: E).

ID	Requirement description
RF1	System should control Review
RF1.1	System should provide the association of Reviewer to Review
RF1.1.1	System should provide the association of Person to Reviewer
RF1.1.1.1	System should maintain the data of Person
RF1.2	System should provide the association of Paper to Review
RF1.2.1	System should maintain the data of Paper
RF1.2.1.1	System should allow to inform the Not Evaluated Paper
RF1.2.1.2	System should allow to inform the Rejected Paper
RF1.2.1.3	System should allow to inform the Accepted Paper
RF2	System should control Submission
RF2.1	System should provide the association of Author to Submission
RF2.1.1	System should provide the association of Person to Author
RF2.2	System should provide the association of Paper to Submission
continued...	

The extracted requirements were listed in different colors to emphasize the source; in black, requirements extracted from classes that use stereotype <<Relator>>, in blue, requirements extracted from relations (association or generalization), in purple, requirements extracted from stereotypes: <<Category>>, <<Collective>>, <<Kind>>, <<Mixin>>, and <<Subkind>>, and in orange requirements extracted from stereotypes: <<Phase>> and <<Mode>>.

Through the list presented in Table 2, it is possible to observe that DFRs are generated from classes that used stereotype <<Relator>> grouping other DFR. The list might suggest sub-modules or groups, from which uses cases and interface prototypes, among others, can be generated. Figure 2 partially illustrates the conceptual model represented in OntoUML related to the Conference domain.

4.2 Verification of DFR Listed

With the result from the heuristic, two surveys were performed with the aim to verify the consistency of the results: 1) unidentified DFR due to fault in the heuristic; 2) unidentified DFR due to fault in the model design. To perform these verifications, three variables were used for each one of the six domains evaluated: the number of DFRs listed by the heuristic, the number of DFRs not listed by element, and the number of elements in the conceptual model. Table 3 presents these results.

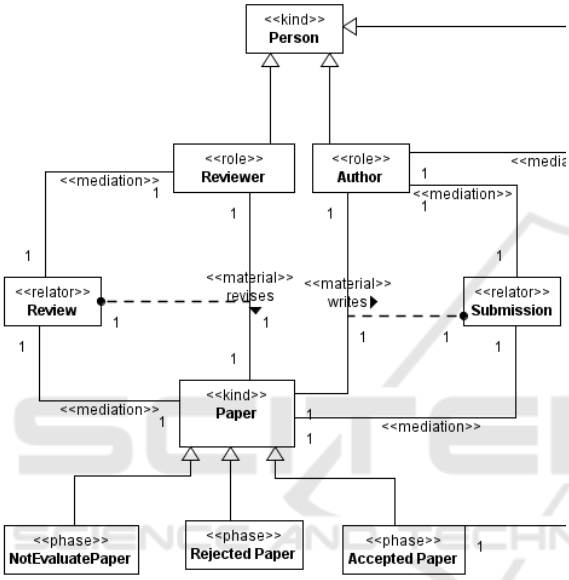

Figure 2: Partial OntoUML conceptual model of Conference domain (ID: E).

Table 3: Quantity of DFR listed, elements without DFR listed and elements from conceptual model by domain.

ID	Description	Domain/Quantity					
		A	B	C	D	E	F
x	DFR listed	25	25	22	35	22	71
y	Elements without DFR	7	2	5	15	10	17
z	Elements from conceptual model	32	27	28	50	32	91

4.2.1 Unidentified DFR Due to Fault in the Heuristic

First, it was verified if any DFR was left unidentified owing to a fault in the heuristic. All "elements without DFR" were identified and grouped by type in Table 4. After the individual and manual verification for each element, the following analysis was obtained: 1) The class that used the stereotype <<Collective>> (domain ID: F) did not generate DFR because it was related to an association that uses the stereotype <<MemberOf>>, as predicted by the heuristic; 2) The classes that used the stereotypes <<Role>> and <<Role Mixin>> do not generate DFR, as predicted by the heuristic; 3) The association that used the stereotype <<Material>> do not generate DFR. In such cases, the association that uses the stereotype <<Mediation>> generates the necessary DFR, as predicted by the heuristic; and 4) The relationship of generalization does not generate DFR, as also predicted by the heuristic.

Table 4 : Quantity of elements without DFR listed by domain.

Stereotype	Domain/Quantity					
	A	B	C	D	E	F
Collective	0	0	0	0	0	1
Role	6	2	2	2	4	15
Role Mixin	0	0	1	0	0	0
Characterization	0	0	0	0	0	0
Material	1	0	0	1	6	1
Generalization	0	0	2	12	0	0
Total	7	2	5	15	10	17

These analyses led to the conclusion that all situations with no DFR generation were predicted by the heuristic. Within the scope proposed for this first version, the heuristic fulfilled its role. Improvements should be proposed as the heuristic is applied to models with different complexity.

4.2.2 Unidentified DFR Due to Fault in Model Design

Considering that the heuristic generates the maximum of one DFR for each element in the model and that the number of unlisted DFR in Table 3 (line y) is correct, we suppose that the sum of the listed DFR of Table 3 (line x) and unlisted DFR of Table 3 (line y) must be equivalent to the number of elements in the conceptual model of Table 3 (line z). According to this premise, in five of the six domains evaluated, this verification was true. Only the Library domain (ID: F) was missing three requirements.

After the manual analysis, it was possible to verify a non-generated DFR from an "unknown" relation, not predicted in the OntoUML, a situation that must be corrected in the model. Two DFRs were not generated due to the presence of an association that used the stereotype <<Mediation>> between one class that used the stereotype <<Role>> and one class

that used the stereotype <<Kind>>. It is a problem in the model design because in OntoUML specification is said that an association that use the stereotype <<mediation>> must have in the origin a class that uses a stereotype <<Relator>>. The results obtained indicate that it is possible to use a conceptual model represented in OntoUML as a source of DFR derivation. Moreover, the heuristic can identify possible faults of the model design or even the incompleteness of the model. The results also indicate that the conceptual model may represent an instrument to support the traceability of the DFR, in addition to generate metrics to estimate the complexity of the domain.

5 LIMITATIONS OF THIS STUDY

The presented heuristic had the purpose of evaluating the possibility of deriving DFR from OntoUML conceptual models. Due to the presented results, we consider its application to be possible. However, we also believe that the present study has limitations and that future studies are necessary to improve the proposed heuristic. Although the location of conceptual models with domain represented in OntoUML is not trivial, it is necessary to find other models, more complex, to apply the heuristic and obtain new results.

It is also considered important to perform experiments with domain specialists to evaluate the completeness of the DFR. Although the entire process presented here (reading, interpretation and extraction of requirements from the models) is systematized, it was generated manually. Computational tools are being implemented to facilitate the processing of conceptual models for the generation of the DFR list, identification of sources and dependence, faults in the model design or incompleteness of the model, as well as some metrics. As the last item, we observed that the quality of the terms used by the model designer, as well as the proper use of the OntoUML constructs, interfere directly on the quality of the DFR transcriptions.

6 RELATED WORK

The concept of model transformation has been proposed with the general purpose of maintaining the consistency and traceability of the artifacts produced during the software development. Three types of approaches can be found in this concept: textual requirements into analysis models and textual requirements specification from software engineering models, and textual requirements into analysis models and back. An overview of each approach is presented next.

Yue et al. (2011) performed a systematic literature review with the aim of identifying the proposals related to the generation of analysis models from text requirements. With this purpose, 20 primary studies were identified. Most studies presented proposals for the generation of models in UML language, including class diagrams, state diagrams and sequence diagrams. The texts used for the generation of the models were mostly extracted from use cases. Among the main conclusions indicated by the review, despite a significant amount of research, we still do not have a practical, workable automated solution and most of the approaches do not address traceability.

Nicolás and Toval (2009) present a systematic review of the literature related to the generation of textual requirement specifications from software engineering models. In the present work, 25 primary studies were identified with this approach. Most of these cases utilized use case models or scenarios with textual requirements generation source. Goal oriented as i* and KAOS models were also used for the generation of textual requirements. In this review, five proposals for the derivation of requirements from Software Product Lines (SPL) models were also identified. The models used for the derivation were the Feature model and Variability model. Among the main conclusions from the review, in this approach the effort of specifying those requirements is reduced. However, without proper tools support these approaches are not truly applicable in practice.

The textual requirements specifications into analysis models and back approach proposes the combination of the two previous approaches, i.e., mechanisms to allow the transformation of textual requirements specifications into analysis models and vice-versa. With this purpose, Landhäußer et al. (2014) propose the Requirements Engineering Feedback System (REFS), which automates the process of keeping textual specification and models consistent when the models change. The generated models include class, activity and state diagrams represented in UML. Several NLP tools for the pre-processing of natural language texts are applied to identify the changes and to suggest the update of the models and textual requirements specification.

This brief review indicates that most transformation processes proposed use models represented in UML or languages without compromising the ontology. Hence, there are efforts

to maintain the consistency and integrity of artifacts. Part of the challenge stems from the fact that requirements and architectures use different terms and concepts to capture the model elements relevant to each (Grunbacher et al., 2004). Because OntoUML is ontology-based, the conceptual models constructed are assumed to be more expressive and to represent the real world of the domain more faithfully than do other languages of conceptual representation.

7 FINAL CONSIDERATIONS

In the study presented by Henderson-Sellers et al. (2015), one of the items in the authors' wish list is for conceptual models to have more semantics. A recent review (Verdonck et al., 2015) also indicated that, although ontology-oriented conceptual models are proposed, which allows more semantics, people do not know the reason for their application. Our study has a very clear view of the purpose of using OntoUML.

Considering that OntoUML is a language that proposes elements that allow more semantics, there is the possibility of building more expressive models than, for instance, models represented in UML language. The result of a more expressive conceptual model reflects the better domain representation, which can consequently reduce the requirement changes and the efforts to maintain the traceability of the generated artifacts.

With the purpose of evaluating the possibility of deriving DFR in a systematized manner, a heuristic was proposed. This heuristic was obtained with the reading and interpretation of nine conceptual models represented in OntoUML. After the heuristic was obtained, it was applied in a systematized manner to six models. The heuristic allowed the navigation through all elements in the model and the extraction of related DFR. As mentioned in the introduction section, the proposal intention is to use ontological conceptual models to derive the possible high level functional requirements. To generate a detailed and complete list of functional requirements it is needed to move on to the next phases of software development.

REFERENCES

Ding, W., Marchionini, G., 1997. A Study on Video Browsing Strategies. Technical Report UMIACS-TR-97-40, University of Maryland, College Park, MD.

Grunbacher P., Egyed, A., Medvidovic, N., 2004. Reconciling software requirements and architectures with intermediate models. Software and System Modeling (SoSyM), v 3, n 3, Springer, pp 235-253.

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

Henderson-Sellers, B., Gonzalez-Perez, C., Eriksson, O., Ågerfalk, P.J., 1992. Software modeling languages: a wish list. In *Seventh International Workshop on Modeling in Software Engineering*, pp 72-77.

Jalote, P., 1997. An Integrated Approach to Software Engineering, Springer, New York.

Jureta, I.J., Borgida, A., Ernst, N., Mylopoulos, J., 2010. Techne: Towards a New Generation of Requirements Modeling Languages with Goals, Preferences, and Inconsistency Handling. In *International Requirements Engineering Conference*, pp 115–124.

Landhäußer, M., Körner, S. J., Tichy, W. F., 2014. From requirements to UML models and back: how automatic processing of text can support requirements engineering. Software Quality Journal, 22, 1, pp 121-149.

Loucopoulos, P., Karakostas, V., 1995. System Requirements Engineering. McGraw-Hill.

Mylopoulos, J., 1992. Conceptual modeling and Telos, In *Conceptual modeling, databases and CASE: An Integrated View of Information Systems Development*, Wiley, New York, pp 49-68.

Nicolás, J., Toval, A., 2009. On the generation of requirements specifications from software engineering models: A systematic literature review. Information and Software Technology, v 51, i 9, pp 1291–130.

Robinson, W., Pawlowski, S., 1999. Managing requirements inconsistency with development goal monitors. IEEE Transactions on Software Engineering 25(6), pp 816-835.

Teixeira, M.G.S., Falbo, R., Guizzardi, G., 2014. Analyzing the Behavior of Modelers in Interpreting Relationships in Conceptual Models: An Empirical Study, In *3rd International Workshop on Ontologies and Conceptual Modeling*, Rio de Janeiro, Brazil.

Verdonck, M., Gailly, F., de Cesare, S., Poels, G., 2015. Ontology-driven conceptual modeling: A systematic literature mapping and review. Applied Ontology, v 10, n 3-4, pp. 197-227.

Valaski, J., Reinehr, S., Malucelli, A., 2016. Which Roles Ontologies play on Software Requirements Engineering? A Systematic Review. In *International Conference on Software Engineering Research & Practice*, pp 24-30, Las Vegas.

Valaski, J., Reinehr, S., Malucelli, A., 2016b. Evaluating the Expressiveness of a Conceptual Model Represented in OntoUML and UML. In *ONTOBRAS*, Curitiba, Brazil.

Yue, T., Briand, L. C., Labiche, Y., 2011. A systematic review of transformation approaches between user requirements and analysis models. Requirements Engineering, v.16, i.2, pp 75–99.