OntoUML Lightweight Editor

A model-based environment to build, evaluate and implement reference ontologies

John Guerson, Tiago Prince Sales, Giancarlo Guizzardi, João Paulo A. Almeida
Computer Science Department, Federal University of Espírito Santo,
Ontology & Conceptual Modeling Research Group (NEMO), Vitória, ES Brazil
{jguerson, tpsales, gguizzardi, jpalmeida}@inf.ufes.br

Abstract - Enterprise information systems are increasingly being conceived as a combination of existing systems and to work as a part of an ecosystem of software products. This change demands methods and tools to deal with the challenging semantic interoperability issues. OntoUML is a well-founded modeling language that allows modelers to formalize world-views in a technologically neutral way, aiding in the solution of such interoperability challenges. In this paper, we present an overview of the OntoUML Lightweight Editor (OLED), our model-based environment to build, evaluate and implement OntoUML models, alongside with its main features and application scenarios.

Keywords – Ontology-driven Conceptual Modeling, OntoUML;

I. INTRODUCTION

High quality information is the key for a rational decision making process within an organization. Without the support of adequate Enterprise Information Systems (EIS), the individuals that participate in these processes in organizations cannot systematically take optimal decisions nor understand the full effect of their actions. An EIS contains structures that represent abstractions over certain portions of reality, capturing aspects that are relevant for a class of problems at hand. Therefore, the quality of an EIS directly depends on how truthful its information structures are to the aspects of reality it is designed to represent. In our current scenario, semantic interoperability has become a pervasive force, driving and constraining the process of creating EIS, which is becoming an increasingly complex combination of domains. More and more, information systems either are created by combining existing independently developed subsystems, or are created to eventually serve as components in multiple larger yet-to-be-conceived systems. To deal with such complexity, we need methods and tools to support us in the tasks of understanding, elaborating and precisely representing the nature of conceptualizations of reality, as well as in tasks of negotiating and safely establishing the correct relations between different ones. Conceptual models produced with this aim are called *reference ontologies* [8].

In [8], Guizzardi defends that reference ontologies should be produced by incorporating the distinctions of a theoretically well-grounded foundation ontology. The Unified Foundational Ontology (UFO) is a foundational ontology that provides a sound ontological basis to evaluate and give real-world semantics to conceptual modeling language's constructs such as UML. OntoUML is a result of such evaluation. The class diagram fragment of UML 2.0 was re-designed and evaluated according to the structural layer of UFO. The result is a well-founded version of UML for ontology-driven conceptual modeling. OntoUML's meta-model has been designed to comply with the ontological distinctions and axiomatization of UFO. It is a highly expressive language, with precise formal semantics and neutrality with respect to implementation technologies, all of which allows modelers to accurately represent how a community (or an organization) understands a particular domain of interest, without being biased by implementation concerns. OntoUML has been successfully employed in a number of industrial projects in several different domains, such as petroleum and gas, digital journalism, telecommunications, and government. Besides the modeling language itself, the OntoUML approach consists in a set of software solutions to enable model construction, formal verification and validation, code generation and verbalization. In this paper, we present the OntoUML Lightweight Editor[1] (OLED), a model-based environment to support Ontology Engineering in OntoUML, in particular the task of formalization, verification, validation and implementation. OLED was designed to aggregate all the aforementioned technologies developed for OntoUML in the long-term research project conducted by the Ontology and Conceptual Modeling Research Group (NEMO).

In the remainder of this paper, we provide a brief overview of the most innovative features in OLED in Section II, followed by some reported uses of the tool in Section III and a brief discussion of the direction OLED is evolving to in Section IV.

II. OVERVIEW

We depict in Fig 1 the main features of OLED w.r.t each phase of the ontology development approach using OntoUML. The environment supports modeling, verification, validation, and implementation of OntoUML models (depicted in grey boxes). Currently, there is no support for requirements elicitation.

Modelers specify their domain ontologies in OntoUML [8], constraining them using the Object Constraint Language (OCL) [13]. The tool provides a set of built-in design patterns to speed up the modeling activity through re-use. To improve the quality of the models built using OLED, it provides an automatic syntax verification alongside two complementary validation features, visual simulation [5] and anti-patterns [15]. To apply the knowledge formalized in the OntoUML in

[1] Available at: https://code.google.com/p/ontouml-lightweight-editor/

semantic web applications, OLED features a number of pre-defined automatic transformations to the Web Ontology Language (OWL) (possibly enhanced with SWRL rules) [1] [3] [17].

In the sequel, we present a brief description of each of these features.

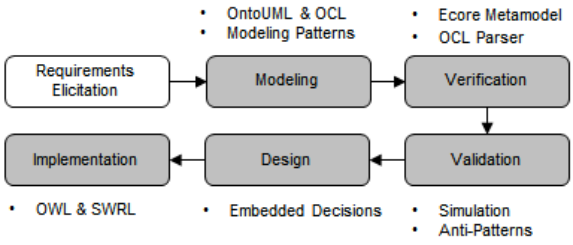

Fig 1. OLED's support for the Ontology Development Process

A. *Modeling Features*

OntoUML [8] is an ontologically well-founded profile of the class diagram fragment of UML 2.0. OntoUML's categories are put forth by the Unified Foundational Ontology (UFO). UFO's ontological distinctions are reflected as UML stereotypes specializing classes and associations according to the ontological notions of rigidity and dependence. In Fig 2, we depict an OntoUML class diagram representing a road traffic accident ontology. In this domain, travelers are people taking part of a travel in a vehicle, possibly becoming involved in traffic accidents; traffic accidents involve victims, crashed vehicles and a roadway; and, accidents may involve a set of fatal victims. A particular sort of accident called rear-end collisions is identified (accidents wherein a vehicle crashes into the vehicle in front of it).

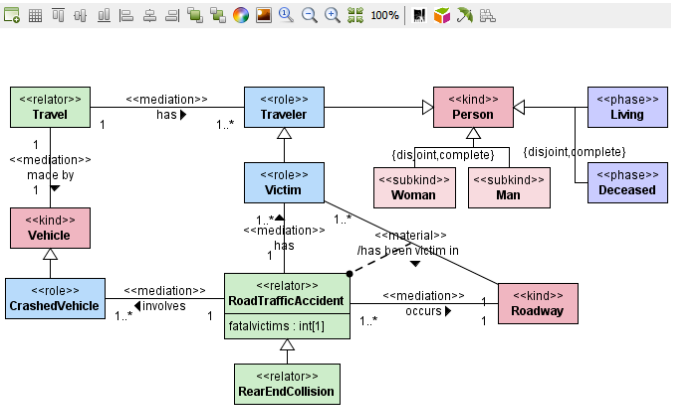

Fig 2. OntoUML Diagram Editor

In order to cover domain constraints that cannot be represented using OntoUML's diagrammatic notation, OLED supports the specification of OCL constraints [13]. OCL express UML class invariants and derivation rules for UML association end-points and attributes. In Fig 3, we depict OLED's OCL editor, which provides syntax highlight, code-completion and syntax verification (parsing) to textual constraints. Fig 3 describes two textual constraints in OCL about the ontology of traffic accidents. First, a derivation rule stating that the number of fatal victims of an accident is derived from the number of deceased people involved as victims in that accident. Secondly, a class invariant stating that a rear-end collision must always involve two crashed vehicles. OLED's support for OCL is developed using the Eclipse framework. In addition to it, OLED supports a temporal version of OCL suitable for modeling with OntoUML [7].

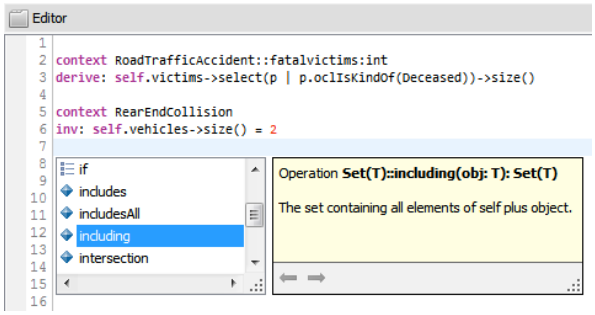

Fig 3. OCL Constraint Editor

B. *Verification Features*

To verify a model means to asses if the model was built correctly i.e., if it is syntactically valid. OLED uses an OntoUML metamodel [6] defined in ECore, which incorporates a set of syntactical rules to reflect UFO's formal axiomatization. OLED has the capability to check automatically if any given model is in pace with these formal constraints, pointing exactly which model constructs break them. In Fig 4, we exemplify an execution of the syntax verification, showing the errors generated when modeling a class stereotyped as «kind» as a super-type of another, stereotyped as a «role». For more details why this construction is not allowed, please refer to [8].

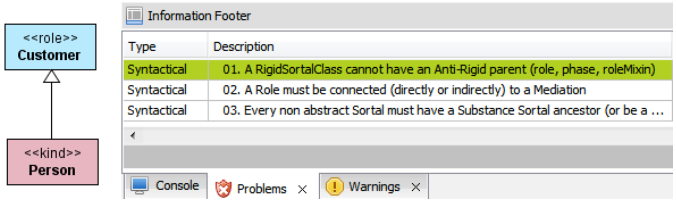

Fig 4. OntoUML Syntactic Checker

Besides checking the OntoUML syntax, OLED allows modelers to verify if the OCL rules restricting the model are specified properly.

C. *Validation Features*

To validate an ontology means to evaluate whether a particular model is the right model for a domain. The task consists in analyzing if the model precisely formalizes the shared conceptualization of the domain at hand. To achieve that, one must check multiple aspects, which include the suitability of meta-categories to classify a given domain concept and the relation between intended and allowed instantiations of the ontology. To aid modelers in the validation process, OLED provides visual simulation and ontological anti-pattern management.

1) Visual Simulation

Visual simulation is obtained by performing a transformation of the OntoUML model (and OCL constraints) to an Alloy [10] specification. The Alloy Analyzer is capable of automatically generating instances of the model as a means to simulate its set of desired/undesired properties confronting the modeler with its decisions. In

Fig 5, we depict a fragment of a possible instantiation of the traffic accident ontology enriched with constraints. It shows a current world (a point in time) wherein a rear-end collision between two crashed vehicles resulted in the death of both travelers of the vehicles. All specified constraints are respected. Differently from an unconstrained model, the number of fatal victims is correct and the rear end collision involves two vehicles.

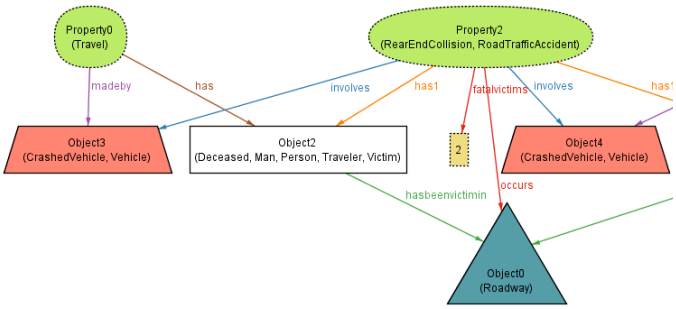

Fig 5. A Fragment of an Automatically Generated Simulation

2) Ontological Anti-Patterns

Ontological anti-patterns are model structures that, albeit producing syntactically valid conceptual models, are prone to result in unintended domain representations. They are configurations that when used in a model will typically cause the set of valid (possible) instances of that model to differ from the set of instances representing intended state of affairs in that domain [9]. In a previous study [14], a library of ontological anti-patterns is presented.

OLED features an anti-pattern management component that consists of three steps: automatic detection, guided analysis and automatic refactoring. The first step, the automatic detection, is meant to relieve modelers from learning all anti-pattern structures and manually inspecting occurrences in their models. Users can request an anti-pattern inspection on a particular diagram or on an arbitrary selection of elements. The second step, the guided analysis, is performed for each identified anti-pattern occurrence. In order to decide whether a particular occurrence entails unintended consequences, a modeler must reason about its consequences. To support this process, OLED provides a wizard for each anti-pattern, which details the elements that participate in the anti-pattern occurrence, provides theoretical background information when necessary, and makes a series of questions, which lead to the appropriate solutions. The third and last step consists in appropriately refactoring the model. Based on the interaction with the anti-pattern wizard, the tool will suggest a plan to rectify the model. To exemplify, consider a tiny fraction of the road traffic model depicted in Fig 2. It states that a traveler travels in a vehicle and that a traveler might be involved in a car accident, which also involves vehicles. This fraction characterizes an anti-pattern deemed Association Cycle (AssCyc) [14] for it allows passengers in a vehicle that has not participated in an accident to be victims in that very same accident.

D. *Code Generation Features*

A reference ontology can produce application ontologies according to a set of specific computational purposes and requirements. Application ontologies have been usually represented in computationally tractable subsets of first-order logic such as OWL or F-Logic. OWL is an extension of RDF based on Description Logics to represent content in the context of Semantic Web. It has been used to implement reference ontologies to discover knowledge, annotate semantically its content and to publish it on the web. SWRL is the language used to express logic rules over OWL specifications.

OLED provides three code generations to OWL. Each one takes different design choices in account. The first named *Simple* [1] maps basically OntoUML classes, associations and attributes to OWL classes, object properties and data properties, respectively. It considers generalization sets and its disjointness properties plus model cardinalities. The second named *Temporal* [17] represents temporally changing information in OWL and encompasses four approaches named *Reification*, and *Worm Views A0*, *A1* and *A2*. Reification reifies the contingent and mutable information of an individual into moments (e.g., anti-rigid objects reified as qua-individuals) where an individual's existence is represented as a temporal extent. In Worm Views, entities are considered spatiotemporal worms where individuals are composed by (i) a concept representing that individual (IC) and (ii) its temporal parts as worm temporal slices (TSs). Finally, the third is named *OOTOS* [3] and it considers OCL constraints as SWRL rules. It also considers structured datatypes, cardinalities, transitivity of material and parthood relations (as SWRL rules) and disjointness between substance sortals.

E. *Model Verbalization Features*

Model verbalization stands for the activity of generating a documentation of the ontology in (controlled) natural language. This process is very useful, for example, to allow domain experts that are not well-versed in the modeling language (the most common situation), to access the knowledge embedded in a conceptual model.

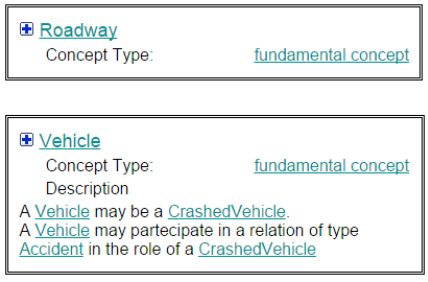

Fig 6. Verbalization of an OntoUML model in SBVR using OLED

OLED provides two types of model verbalization: (i) the generation of a document in Semantic Business Vocabulary

Rules (SBVR), an OMG standard for representing a vocabulary of a business in a structured natural language representation [6] as depicted in Fig 6; (ii) the generation of a technical glossary of the conceptual model in natural language (currently, only in Portuguese-BR). For each concept in the ontology, the glossary generator uses the concept's meta-category, alongside its relationships and properties to assemble a definition.

F. *Integration Features*

It is safe to assume that every organization that does conceptual modeling already adopts a particular tool. Usually, there is an investment in acquiring commercial CASE tools and, thus, there is always resistance when introducing a new tool in the organization. We did not design OLED to be a competitor of these tools. Conversely, we designed it to work with them. Since OntoUML borrows the class diagram syntax from UML, it is possible to build OntoUML models using virtually any UML tool that supports stereotyping. Enterprise Architect[2] (EA) is one of them. We developed an OntoUML plug-in for EA that allows modelers to draw their models in EA, but still use the validation, code generation and verbalization features available in OLED.

III. Applications

From its very conception, we designed and evolved OLED in the context of industrial and academic projects. In the context of our research group, (i) an initiative to produce a reference ontology for the legal domain [4] with the purposes of interoperability and automation of the management of the Brazilian norms; (ii) the development of a core reference ontology for services called UFO-S [12] and a (iii) core reference ontology for organizations called O3 with the purpose of organizational structure description and communication [16]. In all three cases, OLED was used for semantic validation (O3 solely with semantic anti-patterns management). In the case of (i), it was additionally used for OWL generation. Examples of application cases of OLED without the involvement of our research group include: (i) an approach to use reference ontologies for the multidimensional design in Business Intelligence for a domain of electrical systems. In this case, our EA plugin was used in tandem with OLED for verification and validation [11]; and (ii) an approach to pre-process, organize and query high quality meteorological data in the context of silico experiments in which OLED was used to generate an application ontology in OWL [2].

IV. Final Considerations

OLED is a constantly evolving environment that aims to aggregate the technological results of a long-term research project of the NEMO research group. In this paper, we provided a "whirlwind tour" of its main innovative functionalities. As future endeavors, we aim to add new code generation features to provide a greater flexibility for users to implement their ontologies using different technologies. We also aim to expand the validation features, since the feedback from the user community evinced a great interest in them. Lastly, we plan to include a support for refactoring models in languages like UML and OWL, for semantic interoperability is one of the main application of OntoUML models. OLED is developed as an academic open source project. Currently, a startup company called *Menthor*[3] is being created which will further explore the commercial development of the tool.

Acknowledgment

This research is funded by the Brazilian Research Funding Agencies CNPq (grants 311313/2014-0 and 485368/2013-7) and CAPES/CNPq (402991/2012-5). We would also like to thank all NEMO members who have participated in OLED's development. In particular, we thank Antognoni Albuquerque, Cásssio Reginato, Diorbert Correa, Freddy Brasileiro, Victor Amorim, Roberto Carraretto and Vinicius Sobral.

References

[1] Albuquerque, A., "Ontological Foudations for Conceptual Modeling Datatypes", MSc Thesis, Federal University of Espírito Santo, 2013.

[2] Barbosa, T.S., Santos, E.O., Lyra, G. B., and da Cruz, S. M. S., "Using Well-Founded Provenance Ontologies to Query Meteorological Data", In IPAW, pp. 267-270, 2014.

[3] Barcelos, P.P.F., Santos, V. A. dos, Silva, F.B., Monteiro M.E., Garcia A.S., "An Automated Transformation from OntoUML to OWL and SWRL", Ontobras, pp. 130-141, 2013.

[4] Barcelos, P.P.F., Guizzardi R.S.S., and Garcia A.S., "An Ontology Reference Model for Normative Acts", Ontobras, pp. 35-46, 2013.

[5] Benevides, A.B., Guizzardi, G., Braga, B.F.B., Almeida, J.P.A., "Validating modal aspects of OntoUML conceptual models using automatically generated visual world structures", J. Univers. Comput. Sci., vol. 16, pp. 2904–2933, 2011.

[6] Carraretto, R., "A Modeling InfraStructure for OntoUML", BSc Thesis, Federal University of Espírito Santo, 2010.

[7] Guerson, J., Almeida, J. P. A., "Representing Dynamic Invariants in Ontologically Well-Founded Conceptual Models", In 20th International Conference, EMMSAD, 2015.

[8] Guizzardi, G., "Ontological Foundations for Structural Conceptual Models", Telematica Instituut, The Netherlands, 2005.

[9] Guizzardi, G., "On ontology, ontologies, conceptualizations, modeling languages, and (meta) models", In Frontiers in artificial intelligence and applications, Databases and Information Systems IV, 2007.

[10] Jackson, D., "Software Abstractions-Logic, Language, and Analysis, Revised Edition", The MIT Press, 2012.

[11] Moreira, J., Cordeiro, K., Campos, M. L., and Borges, M., "OntoWarehousing–Multidimensional Design Supported by a Foundational Ontology: A Temporal Perspective", In Data Warehousing and Knowledge Discovery, pp. 35-44, 2014.

[12] Nardi, J.C., et al. "Towards a commitment-based reference ontology for services" Enterprise Distributed Object Computing Conference (EDOC), 2013 17th IEEE International. IEEE, 2013.

[13] OMG, OCL Specification v2.4, 2014.
Available: http://www.omg.org/spec/OCL/2.4/

[14] Sales T.P., "Ontology Validation for Managers", MSc Thesis, Federal University of Espírito Santo, 2014.

[15] Sales T.P., Guizzardi G., "Detection, Simulation and Elimination of Semantic Anti-Patterns in Ontology-Driven Conceptual Models" In Conceptual Modeling (ER), pp. 363-376, 2014.

[16] Pereira, D.C., Almeida, J.P.A., "Representing Organizational Structures in an Enterprise Architecture Language". FOMI'2014 Formal Ontologies meet Industry, p. 7-16, 2014.

[17] Zamborlini, V., and Guizzardi G., "An Ontologically-Founded Reification Approach for Representing Temporally Changing Information in OWL", In 11th International Symposium on Logical Formalizations of Commonsense Reasoning, 2013.

[2] http://www.sparxsystems.com.au/

[3] http://www.menthor.net/