ONTOTracED: A Framework to Capture and Trace Ontology Development Processes

Marcela Vegetti[1], Luciana Roldán[1], Silvio Gonnet[1], Gabriela Henning[2] and Horacio Leone[1]
[1]*Ingar (CONICET/UTN), Avellaneda 3657, Santa Fe, Argentina*
[2]*Intec (CONICET/UNL), Güemes 3450, Santa Fe, Argentina*

Keywords: Ontology Development Processes, Ontology Engineering, Support Tools, Design Processes, Design Rationale.

Abstract: In the last two decades several methodologies to assist the ontology development process have been reported in the literature. However, despite important advances, there are no computational tools supporting them yet. Thus, when an ontology development process ends, the things that remain are just design products (e.g., competency questions, class diagrams, implementations, etc.), without an explicit representation of how they were obtained. This paper presents a framework meant to explicitly capture and trace ontology development processes (the activities carried out, the actors executing them, etc.), along with their associated products.

1 INTRODUCTION

In the last decade, many ontology development processes have changed from the traditional ones, performed by isolated knowledge engineers or domain experts, into collaborative processes executed by mixed teams. In such teams, experts in knowledge acquisition and modeling, domain specialists, and experts in the ontology implementation languages collaborate to build ontologies, according to well-established methodologies. The expertise and background of each team member, as well as the executed activities, and the decisions made during the development process might be of great importance in future ontology development processes. However, current tools supporting ontology development processes do not capture such information; thus, the process trace is lost, and any new ontology development process would start from scratch. In fact, once a given ontology development process is finished, the things that remain are mainly design products (e.g., requirement specifications, competency questions, ontology class diagrams, implementations in specific languages, etc.), without an explicit representation of how these products were obtained, and with no capture of the rationale behind the process. In addition, ontology building is turning into a more professional engineering process that needs to be managed and measured in order to obtain high quality results. All these characteristics constitute essential challenges that require to be addressed by the ontology engineering field.

In order to tackle these issues, this contribution proposes ONTOTracED, a framework to represent, capture and trace ontology development processes. The following section presents the framework components. Finally, Section 3 concludes the paper and offers paths to future work.

2 ONTOTracED

Once an ontology development process is concluded, those things that remain are mainly design products (e.g. the requirements specification, competency questions, ontology class diagrams, the ontology implementation in an specific language, etc.) without an explicit representation of how these products were obtained, and lacking the capture of the history and rationale behind the development process. More specifically, there is no trace of the activities that have originated any of the products, the requirements that have been imposed, the actors that have performed each of the activities, and the underlying rationale behind each decision that was made. In order to overcome these drawbacks, this contribution proposes a comprehensive framework

Vegetti M., Roldán L., Gonnet S., Henning G. and Leone H..
ONTOTracED: A Framework to Capture and Trace Ontology Development Processes.
DOI: 10.5220/0004173304190422
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2012), pages 419-422
ISBN: 978-989-8565-30-3

to represent, capture and trace the ontology development process.

Fig. 1 shows the main components of the proposed framework, including: (i) a *Conceptual Model* that is able to represent generic design processes; (ii) an *Ontological Engineering Domain Model (OEDM),* which specifies the concepts that are required to describe ontology development processes, and (iii) a support computational environment, named *TracOED (Tracking Ontology Engineering Designs)*, that implements both the conceptual model and the OEDM to enable the capture of specific ontology design processes, along with their associated products.

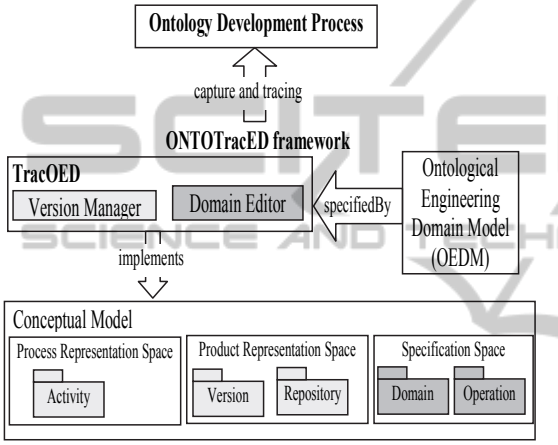

Figure 1: Components of the proposed framework.

The supporting *Conceptual Model* is based on an operational-oriented approach that envisions the ontology development process – as any engineering design process – as a sequence of activities that operates on the products of the development process. The proposal defines two representation spaces to model generic design process concepts: the *Process* and *Product* spaces. In addition, a third component (the *Specification Space* in Fig. 1) is included to fully specify a flexible model that is able to represent and capture design processes pertaining to specific engineering fields.

The *Process representation space* models the activities being performed during an ontology development process. Each basic activity performed by an actor during an ontology development process is represented by the execution of a sequence of operations, which transforms the design objects causing their evolution. In order to represent this evolution, each *design object* is specified at two levels: the *Repository* and the *Version* ones, which constitute the *Product Representation Space.*

The operations that can be applied are domain dependent. Therefore, the *Specification Space* allows specifying the building blocks and operations of particular engineering design domains. In the context of the OntoTracED framework, this space has allowed specifying the ontological engineering domain model.

The *Ontological Engineering Domain Model* component can represent and capture particular ontology development projects, based on building-blocks that define the products obtained, as well as the activities carried out during such process. This representation includes those modeling elements that are most commonly used in the methodologies that nowadays guide ontology development projects.

There are several methodologies for building ontologies and no one is yet emerging as a clear reference. In spite of their diversity, most methodologies share structural similarities and have comparable modeling elements. In this proposal, the components that are considered to be part of the proposed domain model are shown in Table 1.

Additionally, in order to show how this proposal may be applied when ontologists want to stick to specific methodologies and/or approaches, the ontological categories proposed by the Unified Foundational Ontology (UFO) (Guizzardi, 2005) have been added to the *Ontological Engineering Domain Model*. UFO is a language to build domain ontologies that preserves the ontological commitment of the domain being modeled. It distinguishes between conceptual entities called *universals* and *individuals*. In particular, due to space limitations, this work focuses on the subsumption hierarchy of sortal universals. Table 1 presents the meanings of the concrete object types *Kind*, *SubKind*, *Phase* and *Role*, which are the leave classes of the mentioned hierarchy. UFO is considered as a Pattern Language; i.e., in this language the choice of a particular design object type causes a whole pattern to be manifested (Guizzardi y colab., 2011). For example, a phase is always defined as part of a partition; a role is always played in relation to another sortal. So, the adopted domain model also includes the following design patterns proposed by UFO: *SubKind Partition*, *PhasePartition* and *RolePattern* (Guizzardi y colab., 2011).

The OEDM also defines the operations required to capture and manage the included design objects. Some of these operations are shown in Table 1.

TracOED is the computational environment that implements the conceptual model and incorporates the OEDM. It is based on TracED (Roldán y colab., 2010), which was conceived for capturing and

Table 1: Ontological Engineering Design Domain.

Design objects	
Competency question	A Competency question plays the role of a type of requirement specification against which a given ontology can be evaluated.
Concept	A *Concept* represents a collection of entities that share a common set of characteristics.
Relation	A *Relation* symbolizes interrelations between concepts.
Individual	An *Individual* is an entity that belongs to a particular concept.
Assumption and Constraint	They represent natural language expressions that restrict the interpretation of concepts and relationships.
Formal Competency Question	A *Formal Competency Question* is a specification in a formal language of an informal competency question that was initially identified.
Axiom and rule	They represent formal expressions that allow ontologists to (i) explicitly define the semantics of an ontological concept by imposing constraints on its value and/or its interactions with other concepts; (ii) verify the consistency of the knowledge represented in the ontology, and/or (iii) infer new knowledge from the explicitly stated facts
UFO Ontological Categories (Adapted from (Guizzardi, 2005)	
Kind	A *Kind* represents rigid, relationally independent object universals that supply a principle of identity for their instances. Examples include instances of Natural Kinds (such as Person, Dog, Tree) and of artifacts (Chair, Car, Television).
SubKind	A *SubKind* is a rigid, relationally independent restriction of a substance sortal that carries the principle of identity supplied by it. An example could be the SubKind MalePerson of the Kind Person.
Phase	A *Phase* represents anti-rigid and relationally independent universals defined as part of a partition of a sortal. For instance, [Child, Teenager, Adult] is a partition of the kind Person. A Phase is always defined as part of a partition.
Role	A *Role* represents an anti-rigid and relationally dependent universal. For instance, the role student is played by an instance of the kind Person.

Proposed Operations					
Basic				**Pattern related**	
addKind	toRole	addConcept	addConstraint	addPhasePartition	addPhase2Partition
addSubKind	toPhase	addRelation	addAssumption	addRolePattern	addSubkind2Partition
addPhase	remKind	addIndividual	remConcept	addSubkindPartition	remPhaseFromPartition
addRole	remSubKind	addICQ	remRelation	remPhasePartition	remRoleFromPartition
toKind	remPhase	addFCQ	remFCQ	remRolePattern	remSubkindFromPartition
toSubKind	remRole	formalizeICQ	remICQ	remSubkindPartition	

tracing engineering designs.

The major components of *TracOED* are the *Domain Editor* and *Versions Manager*. By using the *Domain Editor*, the *OEDM* has been specified in *TracOED*. Furthermore, the editor allows this model to be further specialized, if required. On the other hand, the *Versions Manager* enables the execution of each ontology development project, and captures its evolution based on *operations* that are accomplished and the instantiation of those *design object types* that have been specified in the Ontological Engineering Domain Model by means of the *Domain Editor* tool.

The *Versions Manager* enables the execution of a design project. When a new design project is created, an existing design domain has to be selected. Therefore, the evolution of a project is based on the execution of domain-specific operations and the instantiation of the design object types pertaining to the selected design domain. Additionally, TracOED includes in its Versions Manager features that allow to keep information about: (i) predecessor and successor model versions (if any exists) of each model version; (ii) history links which save traces of the applied operation sequences (basic activities), which have originated new model versions; (iii) references to the set of object versions that arose as the result of each operation execution. Thus, it is possible to reconstruct the history of a given model version by beginning the trace from the initial operation. Fig. 2 presents the *Version Manager History Window*, which depicts all the operations that have been applied to evolve from an initial model version to the current one. It is possible to see detailed data about each applied operation. For instance, this panel presents information about the time point at which a given operation was applied, who the

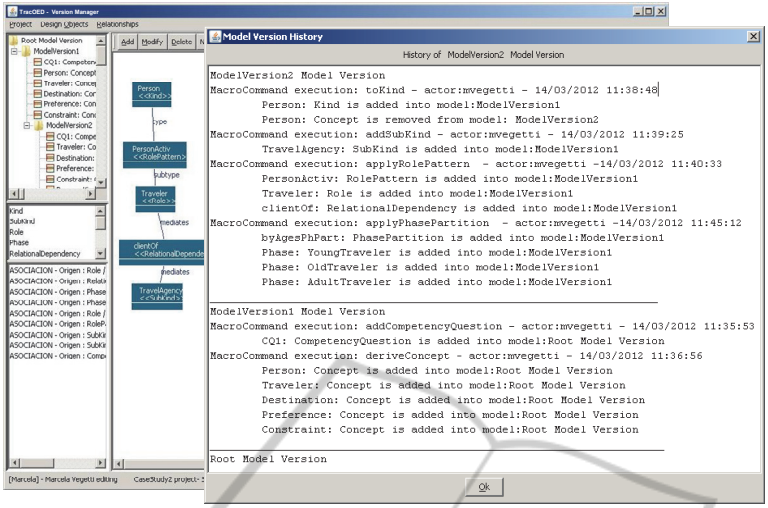

Figure 2: TracOED history window.

involved actor was, and the identification of the successor object versions. In this example, the history window shows that an *applyRolePattern* operation was executed at *ModelVersion2* by mvegetti at time 11:40 -14/03/2012.

It is important to remark that TracOED was developed with the aim of proving the proposed ideas and materializing the ONTOTracED framework. Therefore, this tool is not meant to replace traditional support environments currently used in the ontology domain. On the contrary, in the future TracOED should be integrated with existing ontology development tools, such as the OntoUML editor. In this way, TracOED would perform the capture of all the applied operations by working in a background mode, without being noticed by ontologists.

3 CONCLUSIONS

This contribution presents ONTOTracED, which is a framework aimed at capturing and tracing ontology development processes. The framework is based on a conceptual model of generic engineering design projects, an Ontological Engineering Domain Model, which specifies design objects and operations that are specific to ontology development processes, and a computational environment, named TracOED, which implements these models. The capabilities of TracOED allow the ontologist to keep track of the ontology development process along with its associated products, to store its history, allowing for the future retrieval of knowledge and experience. The proposal is flexible enough to be used in the development of ontologies that rely on particular methodologies and/or approaches, or that address particular fields. If needed, the TracOED domain editor can be used to extend the proposed Ontological Engineering Domain Model or to create a new one.

To further validate the proposal, future work will be oriented to integrate TracOED with existing ontology development tools, like Protégé, the Neon Toolkit or the ontoUML editor, in such a way that it execution takes place in a background mode.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the financial support received from ANPCyT (PAE-PICT-2315 and PAE-PICT-51), CONICET (PIP 2754), UTN (PID 25-O117 and PID 25-0118), and UNL (CAI+D R4 N12).

REFERENCES

Guizzardi, G. 2005. Ontological Foundations for Structural Conceptual Models. PhD with Cum Laude, *Telematica Instituut Fundamental Research Series*, 015, Enschede, the Netherlands.

Guizzardi, G., Pinheiro das Graça, A., Guizzardi, R. 2011. Design Patterns and Inductive Modelling Rules to Support the Construction of Ontologically Well-Founded Conceptual Models in OntoUML. *In: 3rd International Workshop on Ontology-Driven Information Systems (ODISE 2011)*.

Roldán M. L., Gonnet S., Leone H. 2010. TracED: A Tool for Capturing and Tracing Engineering Design Processes. *Advances in Engineering Software*. 41, 1087-1109.