Engineering a complex ontology with time

Jorge Santos
GECAD - Knowledge Engineering and
Decision Support Research Group
Instituto Superior de Engenharia do Porto
Departamento de Engenharia Informática
4200-072 Porto - Portugal
http://www.dei.isep.ipp.pt/~jsantos

Steffen Staab
Universität Karlsruhe (TH), Institut AIFB
D-76128 Karlsruhe, Germany
http://www.aifb.uni-karlsruhe.de/WBS/sst
&
Ontoprise GmbH, 76227 Karlsruhe, Germany
http://www.ontoprise.com

1 Principal Idea

Although most ontologies available on the Web (e.g., consider the DAML ontology library) exhibit only rather simple structures, *viz.* taxonomies and frame-like links between concepts. There are some domains that frequently needs intricate concept descriptions and interactions — in particular ones about time and space.

Despite the last developments in practical theories about time and in the engineering of concept hierarchies and concept frames, the issue of how to engineer complex ontologies with intricate interactions based on time has not been researched very deeply, yet, rendering the engineering of a new complex domain ontology with time a labor intensive, one-off experience with little methodology.

Here, we summarize our ontology engineering methodology, FONTE (Factorizing ONTology Engineering complexity), which pursues a 'divide-and-conquer' strategy for engineering complex ontologies with time. FONTE divides a targeted ontology that is complex and that includes time into two building blocks, viz. a temporal theory and a time-less domain ontology. Each one of the two ontologies can be built independently allowing for a factorization of complexity. The targeted ontology is then assembled from the time-less domain ontology and the temporal theory by an operator $\otimes$.

Thereby, the assembling operator $\otimes$ is very different from existing operators for merging or aligning ontologies [Noy and Musen, 2000; Rahm and Bernstein, 2001]. Merging ontologies is a process that intends to join different ontologies about overlapping domains into a new one and most of its problems and techniques are related to the identification of similar concepts through structure analysis (e.g. graph analysis, path length, common nodes or/and edges and lexical analysis). For instance, car from ontology 1, *O1.car*, and auto from ontology 2 *O2.auto* may be defined to be identical in the merging process because of results of the structure analysis. To formalize the merging and aligning process, Wiederhold proposed a general algebra for composing large applications through merging ontologies of related domains [Wiederhold, 1994] and actually, the operations proposed (*Intersection*, *Union* and *Difference*) are about the similarities and differences of two ontologies.

In contrast, the result of $\otimes$ needs rather to be seen in analogy to the Cartesian product of two entities. For instance, car from ontology 1, *O1.car*, with its frame *O1.licensedInState* is assembled by $\otimes$ with ontology 2 and its *O2.timeInterval* in a way such that every car in the result ontology has a lifetime as well as multiple *O1.licensedInState*-frames with different, mutually exclusive life spans.

$\otimes$ is operationalized by an iterative, interactive process. It starts off with a human assembly — in the sense just explained — between an ontology *O1*, the time-less domain ontology, and an ontology *O2*, the temporal theory. It is then propelled by a set of rules and a set of constraints. The set of rules drives a semi-automatic process proposing combinations. The set of constraints narrows down the set of plausible proposals to valid ones.

In the past a variety of approaches were proposed for reducing the complexity of engineering a rule-based system, e.g. by task analysis [Schreiber *et al.*, 1999], or an ontology-based system, e.g. by developing with patterns [Clark *et al.*, 2000; Staab *et al.*, 2001; Hou *et al.*, 2002] or developing subontologies and merging them [Noy and Musen, 2000; Rahm and Bernstein, 2001]. As different as these methods are, they may be characterized by subdividing the task of building a large ontology by engineering, re-using and then connecting smaller parts of the overall ontology.

Though FONTE shares its goal with these methodologies is its rather different in its operationalization. FONTE does not aim at a partitioning and re-union (by merge or align with recognition of similarities) of the problem space, but rather by a factorization into primordial concepts and a subsequent combination $\otimes$ that is more akin to a Cartesian product than a union of ontologies.

2 Temporal Ontology

The time and general events ontology used embodies many concepts like *Instant*, *Period* or *Process* routinely found in 'standard' ontologies like Time-DAML or SUMO.

Besides the classes *TemporalEntity* and *Eventuality* similar to the ones used in Time-DAML a third class (*TimedThing*) was defined to capture the notion that bridges between temporal concepts and the domain concepts that will be used during the assemble process. In particular, we have included the notion of *Role* as a core concept. While there are concepts that give identity to their instances (i.e. they are *semantically rigid* [Guarino and Welty, 2000]), e.g. while the identity of a particular person depends on being an instance of *Person*, the identity of the same person does not change when

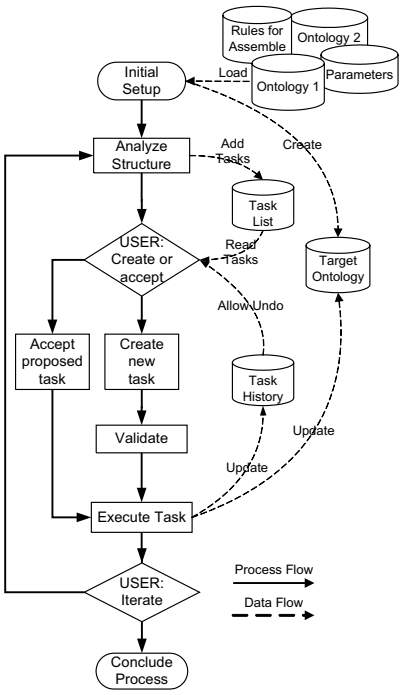

Figure 1: Assembly main process

it ends being a *student* and starts being a *professor*. Thus, the notion of *Role* is important when connecting a temporal theory with a concrete domain.

3 The Assembly Process

The assembly process comprises two main building blocks. First, the specification of temporal aspects for a time-less domain ontology remains dependent on the conceptualization of the ontology engineer. Therefore, it is very important that the engineer may interactively influence the process. Second, in order to facilitate and accelerate the assembly of time-less domain concepts with temporal notions, the interactive process is supported by heuristics asking and pointing the engineer.

The assembling process runs as depicted in Figure 1: It starts by an `Initial Setup`. Some basic operations are performed, namely loading the ontologies to be assembled, loading a set of rules to drive the process and initializing some process parameters. The rules and parameters are defined separately from the tool in order to allow for adaptations to the particular needs of different temporal ontologies. However the rules and parameters do not change when a new domain ontology is to be assembled. The *Target Ontology* initially corresponds to the union of the time-less domain ontology, *O1*, and the temporal theory, *O2*.

In the `Analyze Structure` step a set of tests are performed that restrict the set of possible task instances to plausible ones, which are then proposed by insertion into the *Task List.* As more information becomes available in subsequent iterations, the usefulness of results provided by the structure analysis improves.

In every iteration the engineer decides whether to accept an automatically proposed task instance from the *Task List.* Alternatively, the user may take the initiative and assemble a new task instance from scratch. Then a set of logical tests (`Validate`) are performed in order to detect the existence of any knowledge anomalies (e.g. circularity or redundancy). In contrast, the acceptance of a proposed task instance does not require further checks as the checks are tested for validity before the user sees them.

By the `Execute Task` step the corresponding changes are made to the target ontology. Thereafter, the user decides either to pursue another iteration or to go to `Conclude Process` and accept the current `Target Ontology` as the final version.

4 Evaluation and Conclusion

In order to evaluate the effectiveness of FONTE, we have numerically evaluated the assembly tasks proposed and executed for an ontology about semantic web research community and the temporal ontology briefly presented in section 2. We have investigated how many assembling steps were proposed and evaluated their adequacy. The study results suggest that indeed FONTE provided a way to factorize the complexity of building large applications leading to more reliable and cheaper final products.

Though, so far, we have only studied the assembly of time into a given ontology, we conjecture that FONTE may also be applied to integrate other important concepts like space, trust, or user access rights — concepts that pervade a given ontology in intricate ways such that a method like FONTE is needed in order to factorize engineering complexity out leading to more consistent and cheaper target ontologies.

Acknowledgments This work was supported by a Marie-Curie Fellowship and by FCT (Portuguese Science and Technology Foundation) through the programs ONTOMAPPER (POSI-41818) and SANSKI (POCTI-41830). We thank our colleagues, particulary N. Silva, G. Stumme, and J. Tane.

References

[Clark *et al.*, 2000] P. Clark, J. Thompson, and B. Porter. Knowledge patterns. In *KR2000*, pages 591–600, 2000.

[Guarino and Welty, 2000] N. Guarino and C. Welty. A formal ontology of properties. In *Knowledge Acquisition, Modeling and Management*, pages 97–112, 2000.

[Hou *et al.*, 2002] Chih-Sheng Johnson Hou, N. F. Noy, and M. A. Musen. A template-based approach toward acquisition of logical sentences. In *Procs.Intelligent Information Processing 2002 - World Computer Congress*, Montreal, Canada, 2002.

[Noy and Musen, 2000] N. Noy and M. Musen. Prompt: Algorithm and tool for automated ontology merging and alignment. In *Procs.AAAI-2000*, 2000.

[Rahm and Bernstein, 2001] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[Schreiber *et al.*, 1999] G. Schreiber, Akkermans H., Anjewierden A., R. Hoog, N. Shadbolt, W. Van de Velde, and B. Wielinga. *Knowledge engineering and management. The CommonKADS Methodology*. MIT Press, 1999.

[Staab *et al.*, 2001] S. Staab, M. Erdmann, and A. Maedche. Engineering ontologies using semantic patterns. In *Procs. IJCAI-01 Workshop on E-Business & the Intelligent Web*, 2001.

[Wiederhold, 1994] G. Wiederhold. An algebra for ontology composition. In *Proc.Workshop on Formal Methods*, pages 56–61, Monterey, 1994.