Identifying Conceptual Layers in the Ontology Development Process

Manolis Wallace[1], Panos Alexopoulos[2],
and Phivos Mylonas[3]

[1] Department of Computer Science and Technology,
University of Peloponnese, End of Karaiskaki st., 22100, Tripolis, Greece
wallace@uop.gr
[2] iSOCO, Av. del Partenn, 16-18, 1^o 7^a Campo de las Naciones 28042 Madrid, Spain
p.alexopoulos@gmail.com
[3] Department of Informatics, Ionian University, 7 Tsirigoti Square, 49100,
Corfu, Greece
fmylonas@ionio.gr

Abstract. Whilst a variety of ontological engineering methodologies exist, their actual application is far from trivial, mainly due to the widely diverse nature of the steps involved, that require different forms of expertise, typically possessed by different individuals. In order to address this, in this work we propose the separation between the conceptualization and formalization parts of the process. As proof of concept we apply the proposed approach to the IKARUS methodology, develop a graphical tool to support the resulting methodology and present results from its experimental application. Early results show that the separation of the conceptualization and formalization parts of the ontological engineering methodologies can greatly facilitate the efficiency and effectiveness of the resulting methodologies.

1 Introduction

The field of ontological engineering has played a revitalizing role in various aspects of computer science, by providing a paradigm shift in the way problems are approached and tackled. Such applications range from semantic annotation [16], and document clustering [7] to decision support [5] and knowledge management [11] to list just a few. A more recent and certainly far more challenging field of application for ontologies is that of multimedia processing, where consideration of the semantic layer has provided for the generation of a promising new field, that of semantic multimedia [12]. In this context the complexity and sensitivity of the intended application domain augments demands on the quality of considered ontologies, making development of such ontologies, as for example [3], an even more strenuous task.

On the technical side significant results have been presented with regards to design of ontology representation languages capable of describing the semantics of common, and not so common, human knowledge [15][8], as well as on

I. Maglogiannis, V. Plagianakos, and I. Vlahavas (Eds.): SETN 2012, LNAI 7297, pp. 375–382, 2012.

theoretical aspects such as consistency of representations [4] and computability regarding its implicitly contained knowledge [2] .

But as more progress was made with respect to the technical side of ontological computing, the more evident it became that we were lacking on the methodological side. Although we had the languages to represent human knowledge and algorithms and tools to exploit it, we were still missing the ability to develop extensive, detailed, complete, consistent and correct ontologies. As a response to this gap, we have seen the development of a sequence of methodologies that formalize the ontology development, extension and adaptation processes by organizing them in specific steps and tasks, such as METHONTOLOGY [6], DILIGENT [17], DOGMA[10] and HCOME [13].

Still, even with these methodologies in hand, the actual development of a truly efficient ontology remains a challenging task, not only because tasks comprising these methodologies are quite abstract in their nature, but also because they are quite diverse. Whilst it may be easy to identify experts who can specify fine differences between different types of red wines or others who can select the ideal ontological description structures for every situation, it is quite difficult to find people who combine such skills.

With this in mind, in this paper we shift our focus to characteristics of the ontology development process and to human skills that are associated with them. What we propose is the separation between conceptualization and formalization layers of the process. Based on the characteristics of these layers, any ontology development methodology can be properly adjusted, so that different layers may be implemented by different individuals, who hold different types of expertise needed to be involved for a successful process completion. As proof of concept, we examine a specific ontology engineering methodology, IKARUS-Onto [1], and provide required adaptations. In addition, we develop a simple yet characteristic user interface in order to support users in the cooperative application of the developed methodology.

The remainder of this paper is organized as follows: Section 2 presents the core proposal of this work, i.e. the distinction between conceptualization and formalization layers of ontological engineering. Building on this, in section 3 we review the methodology that will be used in the application example and associate the tasks it comprises to the aforementioned layers. Section 4 presents a graphical tool developed in order to support users in this cooperative ontology engineering task, whilst section 5 lists our concluding remarks.

2 Conceptualization and Formalization Layers

An ontology engineering methodology may be viewed as an abstract description of a process that transfers knowledge from humans to a machine readable formalized structure. Intuitively, we may conclude that for such a process to be successful, we need at least one person that holds the knowledge ontology, a way to extract this knowledge from humans and a way to structure it within an ontology.

Whilst the first component is an individual, commonly referred to as "domain expert", the other two are actions that have to be applied: extraction of knowledge and modeling of the extracted knowledge as a formal ontology. And while identifying a domain expert for a given field is typically a straightforward task, it is these two additional aforementioned components that determine the actual success of the ontology development process. The role of ontology engineering methodologies is to structure the way these actions are implemented, aiming into assisting implementers in achieving better results with less effort.

One problem with existing ontology engineering methodologies is that both actions are treated in a homogeneous approach and the same person is expected to be in a position to implement both efficiently. This is even the case with collaborative approaches (e.g. see [14] and [9]) where multiple users are considered in order to either divide the volume of work or to verify results, but still all users are expected to work on all parts of the ontology engineering process. Unfortunately, as we explain below, this is inherently problematic in any ontology engineering effort that is referred to any domain other than ontology engineering itself.

The choice of the ontology representation language itself is an important task that has a major influence on the applicability and effectiveness of the resulting ontology driven application. What is needed is an expert that will be able to select from the variety of available representation options (i.e. OWL, RDF, f-SHIN, Fuzzy OWL, etc.) the one that best fits the application requirements to be developed, as well as the nature of the available knowledge. This person is commonly referred to as "knowledge engineer". In addition to this rather simplistic example of language selection, a knowledge engineer makes a series of other critical decisions in the ontology engineering process (e.g. which type of structure to use in each case, what to model as a number and what not, when to consider probabilities, when to consider uncertainty and so on). Consequently, for the ontology engineering process to be successful, the person implementing it needs to combine properties of domain expert and ontology engineer.

Although "intermediate experts" have been considered (i.e. people who to some degree understand both the domain at hand and the ontology engineering procedure and may be able to facilitate the overall exchange of information), this has not been proven efficient in practice, since: i) inclusion of an additional person adds to process overhead and ii) specific person abilities limit the work of both the domain expert and the ontology engineer. An ideal answer would be a methodological tool, i.e. a formal ontology development methodology, that is well defined and designed in a way that is understandable and applicable for both types of experts. This tool would need to provide maximum independence between the work of the two different experts and assist them in understanding each other's role and needs in the process, so that their cooperation may be facilitated.

Our proposal is that the conceptualization part of the process, i.e. the part that involves a knowledge elicitation task, should be disconnected from the formalization part, i.e. the part that involves a representation of this knowledge

using a formal ontology representation language. This would allow for development of ontological engineering methodologies that can be implemented cooperatively by individuals with different backgrounds and types of expertise; both of them required for the overall process to be completed successfully. In such an approach, a domain expert would be allowed to express his knowledge in some generic form, unconstrained from specific formalization limitations. Then, an ontology engineer would examine this knowledge against the intended application and would select a proper formalization. To further elaborate on this statement, in the following section we focus on IKARUS-Onto, a detailed methodology for development of fuzzy ontologies.

3 IKARUS-Onto and Expert Roles

IKARUS-Onto assumes that a conventional ontology is available and describes the actions needed in order to generate its extended fuzzy version. Whilst at first this may strike one as a limited example of the aforementioned approach, it is worth noting that IKARUS-Onto is quite similar to conventional ontology engineering methodologies. In fact, it inherits the structure of METHONTOLOGY, which is one of the most acknowledged and applied ontology engineering methodologies [6]. In order to incorporate in IKARUS-Onto the theory discussed within the previous section, we examine each step in order to identify whether it is a conceptualization or a formalization step. Contrary to what one might expect, it is not possible to simply split the overall process in two distinct phases, one for each layer. Still, it is possible to identify the layers involved and thus the associated roles (domain expert or ontology engineer) that need to implement them. Even if the steps are interleaved, having a clear view of "who should do what" greatly facilitates the effective cooperation of both of them.

In Fig. 1 we summarize the steps comprising IKARUS-Onto. Step 0 corresponds to the development of the original conventional ontology and therefore falls outside the core of the proposed methodology. It is worth noting though, that it is a step that, as has already been mentioned earlier, cannot be perfectly executed by either an ontology engineer or a domain expert alone; their combined expertise is absolutely required. Step 1 refers to establishing the need for fuzziness, and therefore the need to actually apply the rest of the methodology and develop a fuzzy version of the ontology. Broken into distinct actions, this step includes a check that the intended application of the ontology is one where vagueness would play a role, i.e. a task for the ontology engineer, and a check that the specific domain is characterized by vagueness, i.e. a task for the domain expert.

Step 2 is concerned with the actual specification of domain vagueness in fuzzy terms. This forms the core of the work to be performed. When analyzed into distinct tasks, this step may be broken down into:

- Identification of areas of the original ontology where vagueness actually exists, which is a task ideally performed by the domain expert.

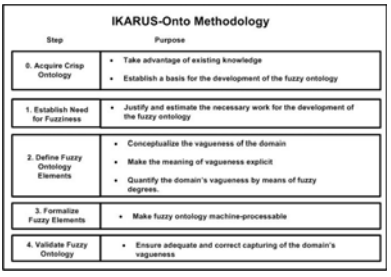

Fig. 1. Outline of the IKARUS-Onto methodology

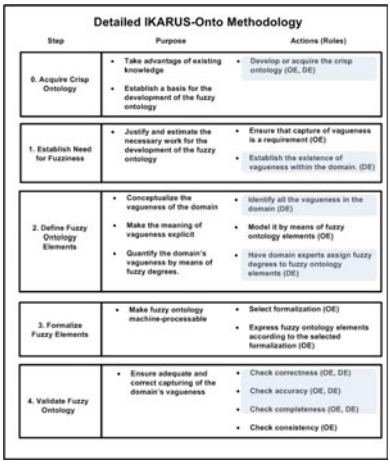

Fig. 2. Detailed IKARUS-Onto methodology with reference to user roles

– Selection of the mathematical model and ontological structure that best match each case of vagueness, when considering both the semantics of the vagueness and the actual limitations or requirements of the intended application; a task that may only be performed by the ontology engineer.
– Specification of the exact fuzzy degree(s) that should be associated with each case of vagueness (a task to be performed by a domain expert), assuming of course that the domain expert is aware of the meaning that these degrees are expected to carry and the way in which these degrees will be interpreted when the ontology is put into practical application.

Clearly, step 2 cannot be performed by an ontology engineer or a domain expert alone. Step 3 refers to the selection of the most suitable ontology representation language for the generated ontology. This selection is determined by the nature of ontological structures that have been used in the previous steps, as well as by technical specification of the application in which the fuzzy ontology will be used. This is a clearly technical step that can be performed by an ontology engineer alone. Finally, step 4, often omitted as optional, is the validation step, in which the output of the aforementioned tasks is checked for correctness, consistency, etc.. These checks range from purely technical ones, such as the consistency check, to heavily semantic ones, such as the accuracy check, and as a result are again performed by a collaboration between domain experts and ontology engineers.

In Fig. 2 we present a graphical representation of the IKARUS-Onto methodology, on which tasks to be implemented by domain experts are highlighted. One may easily observer that the role of the domain expert is not limited to a single continuous segment of the process, but is instead closely intertwined with the work of the ontology engineer. Hence, the need for organization of the coopera-

tion between users performing the two roles is evident. Further to the abstract description of the methodology including the consideration of the conceptualization and formalization layers, a graphical tool that would organize the work would further facilitate the cooperation. We present a first development attempt of such a tool in the following section 4.

4 Ontology Fuzzification Tool

In order to apply the theory proposed herein in an experimental framework we developed a graphical tool that implements the IKARUS-Onto methodology, while also taking into consideration and supporting distinct user roles that implement the different layers of the process. Specifically, the tool aims to organize the ontology fuzzification work around the IKARUS-Onto methodology, while at the same time support domain experts in their role. Emphasis is put on the conceptualization layer, since, on one hand, there is already an abundance of tools to support the ontology engineer in his task and on the other hand, an ontology engineer needs far less support due to the nature of his work.

The ontology fuzzification tool is set up as a standalone web interface in which a crisp ontology may be loaded. This ontology is visualized, so that users may graphically review it and specify any required updates and/or extensions. The visual approach to ontological editing makes it possible for domain experts, who are laymen when it comes to computing, to participate in the process. Additionally, it is also worth mentioning that the visual portion of the ontological editing process is not bound to a specific notation or ontology description language. Therefore the domain expert does not need to comprehend or even worry about specific language characteristics or limitations.

As already discussed, step 0 of the methodology presented in section 3, forms a preparatory step that lies outside the core scope of IKARUS-Onto. Tasks included in steps 1 and 2 of the methodology are supported in greater detail, since they are the main tasks of the ontology fuzzification process. As can be seen in Fig. 2, work in steps 1 and 2 of the methodology can be organized as four intermittent phases of ontology engineer and domain expert work. This exact structure is followed by the developed tool. Specifically, the visualized ontology is first presented to the ontology engineer, so that he may assess the need to capture the related vagueness. Assuming the decision is to go ahead with the fuzzification process, the visualized ontology is presented to the domain expert in the User Interface (UI) shown in Fig. 3. In this UI the domain expert can visually specify elements of the ontology for which some type of vagueness will need to be captured and modeled in the ontology.

Following the structure of the IKARUS-Onto methodology, the work is then transfered to the UI presented in Fig. 4. Here the ontology engineer is presented with elements that have been "highlighted" by the domain expert. For each one of them, the ontology engineer may select the most suitable structure to model its vagueness. Additionally, the ontology engineer "annotates" his work by explaining the meaning of the specified degrees and the way they will be

Fig. 3. The domain expert is assisted in identifying the elements to fuzzify

Fig. 4. The ontology engineer specifies the type of fuzziness for each element and its meaning

interpreted, when the ontology is put to actual use. It is exactly this information that will assist the domain expert during the next step to specify the fuzzy degrees in a meaningful, consistent and efficient manner.

5 Discussion and Conclusions

In this paper we discussed fundamentally different types of tasks that are involved in the ontology development process and may be identified into two distinct layers, namely conceptualization and formalization. As we explained, by separating tasks associated with each layer, any conventional ontology engineering methodology may be modified to facilitate efficient collaboration between a domain expert and an ontology engineer, thus optimizing the overall process with respect to both effort and quality of results.

Continuing, we applied our proposal specifically within the IKARUS-Onto methodology framework. The result of this analysis has been an updated version of the methodology that is designed specifically for cooperative and interdisciplinary ontological engineering, as well as a first version of the graphical tool that supports it. It is worth noting that, although much of the analysis has been focused specifically on IKARUS-Onto, the core of our proposal is directly applicable in any ontological engineering methodology. In fact, as part of our immediate future work we plan to apply our cooperative and interdisciplinary modifications to other methodologies, such as METHONTOLOGY, whereas corresponding versions of our tool will also be developed.

References

1. Alexopoulos, P., Wallace, M., Kafentzis, K., Askounis, D.: IKARUS-Onto: a methodology to develop fuzzy ontologies from crisp ones. Knowledge and Information Systems, doi:10.1007/s10115-011-0457-6

2. Analyti A., Antoniou G., Damsio C.V., Wagner G., Computability and Complexity Issues of Extended RDF. In: Proceedings of the 18th European Conference on Artificial Intelligence (2008)
3. Arndt, R., Troncy, R., Staab, S., Hardman, L.: COMM: A Core Ontology for Multimedia Annotation. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, 2nd edn. International Handbooks on Information Systems, pp. 403–421. Springer (2009)
4. Baclawski, K., Kokar, M.M., Waldinger, R., Kogut, P.A.: Consistency checking of semantic web ontologies. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 454–459. Springer, Heidelberg (2002)
5. Bouamrane, M.-M., Rector, A., Hurrell, M.: Using OWL ontologies for adaptive patient information modelling and preoperative clinical decision support. Knowledge and Information Systems, 1–14 (2010)
6. Fernandez Lopez, M., Gomez Perez, A., Juristo, N.: Methontology: From ontological art towards ontological engineering. In: Proceedings of the Spring Symposium on Ontological Engineering of AAAI (1997)
7. Fodeh, S., Punch, B., Tan, P.-N.: On ontology-driven document clustering using core semantic features. Knowledge and Information Systems, 1–27 (2011)
8. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: the making of a Web Ontology Language. Web Semantics: Science, Services and Agents on the World Wide Web 1(1), 7–26 (2003)
9. Hu, H., Zhao, Y., Wang, Y., Li, M., Wang, D., Wu, W., He, J., Du, X., Wang, S.: Cooperative Ontology Development Environment CODE and a Demo Semantic Web on Economics. In: Zhang, Y., Tanaka, K., Yu, J.X., Wang, S., Li, M. (eds.) APWeb 2005. LNCS, vol. 3399, pp. 1049–1052. Springer, Heidelberg (2005)
10. Jarrar, M., Meersman, R.: Ontology Engineering -The DOGMA Approach. In: Dillon, T.S., Chang, E., Meersman, R., Sycara, K. (eds.) Advances in Web Semantics I. LNCS, vol. 4891, pp. 7–34. Springer, Heidelberg (2008)
11. Jurisica, I., Mylopoulos, J., Yu, E.: Ontologies for Knowledge Management: An Information Systems Perspective. Knowledge and Information Systems 6(4), 380–401 (2004)
12. Kompatsiaris, Y., Hobson, P. (eds.): Semantic Multimedia and Ontologies: Theory and Applications. Springer (March 2008)
13. Kotis, K., Vouros, G.: Human-Centered Ontology Engineering: the HCOME Methodology. International Journal of Knowledge and Information Systems (KAIS) 10, 109–131 (2006)
14. Maedche, A., Staab, S.: Ontology Learning for the Semantic Web. IEEE Intelligent Systems 16(2), 72–79 (2001)
15. McGuinness, D.L., Fikes, R., Hendler, J., Stein, L.A.: DAML+OIL: An Ontology Language for the Semantic Web. IEEE Intelligent Systems 17(5), 72–80 (2002)
16. Sanchez, D., Isern, D., Millan, M.: Content annotation for the semantic web: an automatic web-based approach. Knowledge and Information Systems, 1–26 (2010)
17. Vrandecic, D., Pinto, H.S., Sure, Y., Tempich, C.: The DILIGENT knowledge processes. Journal of Knowledge Management 9(5) (2005)