SOME COMMON PITFALLS IN THE DESIGN OF ONTOLOGY-DRIVEN INFORMATION SYSTEMS

Pablo López-García[1], Eduardo Mena[2], Jesús Bermúdez[1]

[1]*University of the Basque Country, P^o Manuel Lardizabal 1, San Sebastián, Spain*
{pablo_lopez, jesus.bermudez}@ehu.es
[2]*University of Zaragoza, María de Luna 1, Zaragoza, Spain*
emena@unizar.es

Keywords: Ontology Engineering, Knowledge Representation.

Abstract: Ontologies are the mean proposed by the Semantic Web to manage the knowledge of a system. In the software development industry, however, there is another de facto standard set of modeling techniques, methodologies and tools. Reasons for that issue might include technological challenges still to be solved, but also the need of a deep understanding of ontology modeling and the role that the ontology itself should play.
In this paper we show that: 1) building an ontology is an error-prone task that might hide unexpected difficulties even when the ontology is apparently complete and correct, and 2) there is a tendency to misunderstand the role of ontologies when compared to other technologies such as relational databases, forgetting the benefits and strengths of combining both. For such a task, we develop and analyze a case study information system to recommend recipes and menus. We detect, classify and propose solutions to unexpected design pitfalls.

1 INTRODUCTION

The Semantic Web proposes ontologies (Gruber, 1993) as the main knowledge management paradigm. Using Description Logic (DL) and a DL reasoner (Baader et al., 2003), inconsistencies in the ontology model can be detected and current and future concepts, roles, and instances classified. Ontologies are therefore excellent candidates for knowledge representation in information systems (Guarino, 1998). In the software development industry, however, knowledge about the original domain is generally split in a plethora of ad-hoc, hard-coded, and hardly-reusable parts of the system (object methods, database triggers, stored procedures, etc.).

The reasons for reluctance towards ontologies might be argued as practical and technological (Noy and Klein, 2004) but another very important set of conceptual difficulties arises: A deep understanding of ontology modeling and the role that the ontology itself should play are required when using ontologies in information systems.

In this paper we show that: 1) building an ontology is an error-prone task that might hide unexpected difficulties, even when the ontology is apparently complete and correct, and 2) there is a tendency to misunderstand the role of ontologies when compared to other technologies such as relational databases, forgetting the benefits and strengths of combining both. For such a task, we analyze a case study information system to recommend recipes and menus.

The rest of this paper is organized as follows: In Section 2 we introduce the features that the case study should show. In Section 3 an ontology model for that problem is given. In Section 4 we expose the difficulties found, we provide a classification for them depending on their scope, and we show our approach to overcome them. Section 5 discusses related work. Conclusions and future work appear in Section 6.

2 MOTIVATION: A MENU RECOMMENDATION SYSTEM

As a motivating case study, we would like to develop an information system to recommend recipes and menus. The system should make recommendations based on some restrictions given by the user, such as the following:

A) Suggestion of recipes by ingredients. The system should help users in selecting a menu, suggesting recipes that can be prepared given a list of certain ingredients or more complex constraints. The system should suggest all the recipes that satisfy that constraints. The user would also like to know if, given a set of ingredients he can prepare some specific recipe.

B) Suggestion of recipes by guests. The system should help users interested in preparing a menu for some invited guests that might have special food needs (e.g. vegetarians, people with celiac disease, etc.). As in the previous case, the user is also interested in knowing whether a given recipe is suitable for a certain kind of guest.

Given the functionalities described above, the system needs to have some knowledge about the domain to provide suggestions and to tell if a suggestion given by the user meets certain restrictions. As the number or complexity of restrictions grows, the system also becomes increasingly more useful to users.

3 A FIRST KNOWLEDGE MODEL

Throughout the rest of the paper we design the knowledge model of the system. We use the description of functionalities A) and B) and the considerations described in Section 2 as a base to build the knowledge model.

Note that there are some particular issues in our system that advise against the use of some technologies: 1) Storing deep hierarchical structures using relational databases is a hard task (Lien, 1981); 2) In an object-oriented database classes are not based in descriptions and nothing can be inferred automatically. Knowledge could be explicitly programmed in terms of triggers, stored procedures, or business logic. However, we are not interested in building a system for a closed and hard-coded predefined set of restrictions.

There are still research efforts being done on devising methodologies for creating ontologies (Gomez-Perez et al., 2004). These methodologies include a phase for enumerating important terms and relationships of the domain. We do so in the following subsections, following the considerations described at the beginning of this section.

3.1 Concept Menu

We include knowledge about menus by creating a top *Menu* concept in the ontology. We use a *hasCourse* role to model different menus depending on the recipes that compose them. An example of an interesting description for a menu is the following:

$$\begin{aligned}&HealthyLightMenu \equiv\\ &Menu\ and\\ &hasCourse\ some(Recipe\ and\ HealthyRecipe\ and\ MainCourse)\\ &and\ hasCourse\ some(Recipe\ and\ Vegetable\ and\ Dessert)\\ &and\ hasCourse\ max\ 2\end{aligned}$$

Note that menus are mainly described in terms of recipes.

3.2 Concept Recipe

Recipe is another top concept in our ontology. We model the relationship between a recipe and its ingredients with the *hasIngredient* role and the *hasMainIngredient* subrole. In these terms, an example of an interesting recipe is the following:

$$\begin{aligned}&ElaboratePoultryWithVegetablesRecipe \equiv\\ &Recipe\ and\\ &hasIngredient\ only\ (Poultry\ or\ Vegetable)\ and\\ &hasMainIngredient\ some\ Poultry\ and\\ &hasIngredient\ min\ 5\end{aligned}$$

Note that recipes are mainly described in terms of ingredients.

3.3 Concept Guest

We use the *Guest* concept to capture the notion of people with special food needs, being another top concept in our ontology. As an example, a vegetarian can be described as a person who can only eat ingredients from plant origin. This can be modeled with the following description:

$$Vegetarian \equiv Person\ and\ canEat\ only\ PlantOrigin$$

Note that guests are described in terms of ingredients or indirectly in terms of nutrients.

3.4 Concepts Ingredient and Nutrient

Ingredients are needed to classify other concepts (*Recipe* and *Guest*), and nutrients are useful to classify ingredients. The *hasNutrient* role captures this relationship. A sample description of an ingredient is the following:

BalancedIngredient ≡
Ingredient and hasNutrient some Protein
and hasNutrient some Carbohydrate

The top-most terms of the ontology that results after a first attempt are shown in Figure 1.

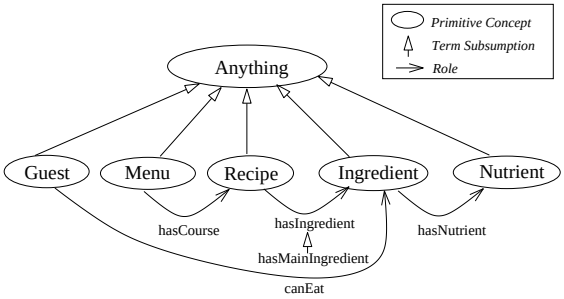

Figure 1: Top ontology for the menu recommendation system.

This knowledge model seems to be adequate as a description of the domain presented in Section 2, since it includes the main concepts and properties needed to respond to the needed functionalities.

4 PROBLEMS IN OUR MODEL

In this section we review our first model in search of hidden pitfalls using the following questions as a guideline to check if a concept is a good candidate to be included in our ontology:

1. Will instances of the concept be considered? How will instances be introduced into the system?
2. Does the concept need to be classified?
3. Does the concept *directly* help to classify another concept? How?
4. What are the relationships between that concept and the rest of the concepts?

4.1 Useless Concepts: Guest

No one is going to introduce instances of *Guest* (question 1). The system is not interested in telling whether a specific person is a vegetarian or not (question 2). *Guest*, as is, is of no use to classify other concepts (question 3). Guests are related with ingredients (question 4) to capture the knowledge that certain guests can or cannot eat. This represented knowledge must not be removed but is not modeled correctly: A *Guest* concept is not needed.

The first steps towards solving the problem are: 1) to identify the useless concept (*Guest*), 2) to identify the featuring concept that needs to be classified (*Recipe*), and 3) to identify the link between both (*Ingredient*). Once all those concepts are identified, the solution is to translate the knowledge of the concept in terms of the concept to be suggested, as in the following example:

IngredientForVegetarian ≡
Ingredient and PlantOrigin

RecipeForVegetarian ≡
Recipe and hasIngredient only IngredientForVegetarian

Once the knowledge has been transferred, the misplaced concept can be removed.

4.2 Concepts with no Instances: Menu

There will be no instances for the *Menu* concept (question 1), because that would imply introducing menus in a system whose goal is to generate menu suggestions. Menus, however, can be described in terms of recipes (question 4) as exposed in Section 3. The system should not classify menus (question 2) and *Menu* does not help in the classification of other concepts either (question 3), so *Menu* is useless as a concept in the ontology.

How can instances be generated and knowledge about menus used? Two solutions arise: A) To outsource that knowledge outside the ontology as an ad-hoc template of the user interface. With this approach, however, knowledge about menus is kept outside the ontology. This is the solution we propose. B) A more interesting solution consists in maintaining the concept in the ontology and building the user interface template automatically. However, this task depends on the concrete language used for descriptions and its semantics, being a task outside the scope of this work.

4.3 Role Fillers and Extension Size

The proposed ontology plays a key role in terms of providing knowledge useful for reasoning. However, there exist additional issues to take into account for an information system to be practical:

1. The size and type of role fillers may severely affect reasoning from a practical point of view. Let us consider, for example, that in our case study a user wants to consult a high-resolution picture for

each recipe. Storing this kind of values in the ontology provides no benefit for reasoning purposes but will unnecessarily increase the size of the ontology.

2. Query-answering information systems, such as our case study, become more useful as the amount of knowledge increases: A menu recommender system with just a few or similar recipes to suggest has no interest at all because human users could manage that knowledge easily by themselves. Therefore, the system needs to take into account that a high number of instances in terms of recipes, ingredients and nutrients will be loaded. These instances should be obtained from external, high-quality, existing sources.

Huge role fillers and a huge number of instances can be stored in a relational database and can be accessed from the ontology.

5 RELATED WORK

In (Allemang and Hendler, 2008) there is a complete chapter dedicated to specific good and bad modeling practices. In our paper, however, we identify situations that can be generalized and we propose a solution for each one. We do not study the problem of correctly expressing what we mean but the problem of detecting if what we mean is useful for the system we are trying to build.

In (Rector et al., 2004) common errors and patterns in the use of OWL-DL are presented by studying a well-known OWL ontology about pizzas. On the contrary, the goal of our paper is focused on discovering pitfalls in complete and correct ontology models.

(Noy and McGuinness, 2001) is centered in constructing a sound ontology from scratch, without references to other popular modeling techniques such as the object model that may result in modeling pitfalls. In our paper we work throughout a complete example and we make a deep analysis to detect hidden conceptual errors.

6 CONCLUSIONS

By creating a system to recommend recipes and menus we have shown, firstly, that correctly representing knowledge of a domain with an ontology does not imply that all parts of the model, as is, are useful for reasoning purposes, since some terms might need to be reformulated in terms of the part of the ontology used for reasoning. Secondly, that sometimes it is not useful to model interesting knowledge using an ontology because the system does not have to access to instances but to generate them. And thirdly, that when designing ontology-driven information systems, ontologies should be used to capture knowledge and relational databases should be used to store huge quantities of data or data that is not useful for reasoning purposes.

As future work, by using new case studies we plan to detect more unexpected pitfalls that are not described in current modeling methodologies.

ACKNOWLEDGEMENTS

This work was supported by grant TIN2007-68091-C02-01 from MICINN (Ministerio de Ciencia e Innovación) of the Spanish Government.

REFERENCES

Allemang, D. and Hendler, J. (2008). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., editors (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Gomez-Perez, A., Corcho, O., and Fernandez-Lopez, M. (2004). *Ontological Engineering : with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition (Advanced Information and Knowledge Processing)*. Springer.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220.

Guarino, N. (1998). *Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy*. IOS Press, Amsterdam, The Netherlands, The Netherlands.

Lien, E. Y. (1981). Hierarchical schemata for relational databases. *ACM Trans. Database Syst.*, 6(1):48–69.

Noy, N. F. and Klein, M. C. A. (2004). Ontology evolution: Not the same as schema evolution. *Knowl. Inf. Syst.*, 6(4):428–440.

Noy, N. F. and McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical report KSL-01-05, Stanford Knowledge Systems Laboratory.

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., and Wroe, C. (2004). Owl pizzas: Practical experience of teaching owl-dl: Common errors and common patterns. In *In Proc. of EKAW 2004*, pages 63–81. Springer.