MESAM: A Protégé Plug-in for the Specialization of Models

Nadjet Zemirline[1], Yolaine Bourda[1], Chantal Reynaud[2], Fabrice Popineau[3]

[1] SUPELEC/Department of Computer Science, Plateau de Moulon, 3 rue Joliot-Curie, 91192 Gif sur Yvette Cedex, France
{Nadjet.Zemirline, Yolaine.Bourda}@supelec.fr
[2]Université Paris-Sud XI, CNRS (LRI) & INRIA - Saclay Île-de-France / Projet Gemo,
Bât. G, 4 rue Jacques Monod, Parc Orsay Université, 91893 Orsay Cedex, France
chantal.reynaud@lri.fr
[3]SUPELEC/Metz Campus, 2 rue Édouard Belin 57070 Metz, France
Fabrice.Popineau@supelec.fr

1 Introduction

Nowadays, several efforts are focused on re-using generic platforms to create new systems, in order to make the design process easier and faster. Often, the designer has his own models and resources and would like to reuse the generic system over his resources. That means, he has to integrate his models and resources in the system, and then to directly reuse the generic system. But many problems occur. One of them is that the designer needs to translate his models into the specific format that understood by the system and to use the vocabulary specific to that system. Furthermore, he also needs to translate all the instantiations of his models (i.e. the resources and their metadata). We think that this task is tedious and time-consuming and we want to avoid it. Our objective is to allow the designer to reuse his models (his vocabulary) and his models' instantiations without any change of format or vocabulary. For example, a generic Adaptive Hypermedia System (AHS) is made of a generic adaptation model relying on generic user and domain models. The designer would like to integrate his models and instances in the generic models in order to reuse the generic adaptation engine.

Specific systems can be obtained by specializing the generic models. However, this specialization process is not always easy to perform. It has to be supported to make the design process easier and faster. This paper focuses on assisting designers to specialize generic models using their own models. We aim to automate this process which has been so far entirely manual. Our objectives are twofold: to create a support for defining mappings between elements in generic models and elements in the designer's personal models and to help creating consistent and relevant models integrating the generic and specific ones and taking into account the mappings between them. The proposed approach relies on OWL[1], a W3C standard and SWRL[2], a W3C proposal.

2 MESAM plug-in

2.1 The approach and MESAM Plug-in functionality

Given two models, a generic model used in a generic system and a specific model provided by a designer, we propose an approach to support the construction of a model that would integrate all the particularities of the specific model and be usable in the generic system. The approach needs the definition of mappings between elements of both models and a validation process at the structural and semantic level. It relies on the designer who has a very good understanding of his model. He will be responsible for semantic validation while all the structural verifications will be done automatically by the plug-in. The main steps of the approach are described below, for more details see [1]:

1. Specification, by the designer, of the equivalence and specialization mappings between classes of the generic model and the specific model, merging the whole generic model and the mapped classes of the specific model (together with the associated mapping links) in order to obtain a new model.
2. Automatic computation of additional mappings between classes, the mappings and the linked classes being added in the model being built.
3. Automatic computation of mappings between elements different from classes, i.e. between attributes and between relations.
4. Validation by the designer of the deductions made by the system in step 3.

The designer interacts with the system in steps 1 and 4, these are detailed in section 3.

In order to propose a generic solution that can be used whatever the generic and specific models are, we modelled structural knowledge in a meta-model based on the OWL meta-model[1], and we performed inferences on knowledge modelled in the meta-model using SWRL rules. The meta-model is loaded inside the plug-in, and the specific and generic models are transformed in instances of the meta-model. We have chosen to keep these processes invisible to the designer. So, in our plug-in, the designer will have the illusion of working on the specific and generic models.

---

[1] http://www.omg.org/docs/ad/05-09-08.pdf

2.2 MESAM Plug-in architecture

As described in the Figure 1, the plug-in includes two parts. First, a knowledge part gathers the meta-model and deduction rules, for more details see [1]. All these components are reusable across applications. Second, the processing part is made of some components performing interaction with an inference engine (in our case Jess) and the OWL Protégé editor. We have used the OWL Protégé API to manipulate OWL models, as editing OWL models or the generation of meta-model instances from OWL models, and the SWRL Jess Bridge[2] to execute SWRL rules using the Jess inference engine[3].

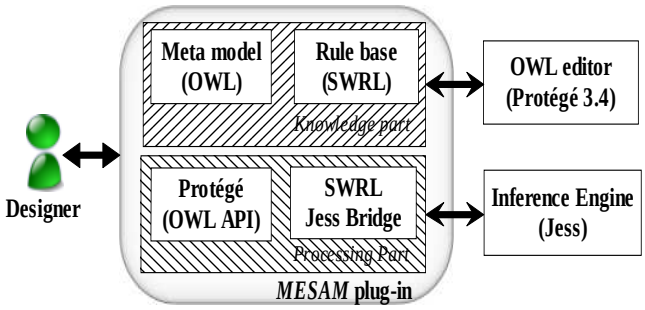

Fig. 1 Architecture of the *MESAM* plug-in

3 Interaction with MESAM Plug-in

Here, we describe how a designer can interact with the plug-in, what facilities the plug-in proposes to guide him in order to obtain a consistent merged model. At first, the developed plug-in provides an interface allowing the designer to indicate the OWL files of both the generic and specific models to be merged (cf. (1, 2) Fig.2). Then, it guides the designer through the other steps of the process. These steps are described below.

3.1 Specification of equivalence or specialization mappings

The developed plug-in proposes a vision of the reused generic and specific models in separate windows, as it can be seen in (cf. (3) Fig.2). The designer can specify either equivalence or specialization mappings between the classes of the two models (cf. (4) Fig.2). Given these correspondences, additional correspondences between classes, properties or relations are generated using a reasoning module. The hypothesis underlying this work is that it is much easier for designers to specify simple correspondences from small numbers of classes in the models, and then evaluate mappings returned by the system. The consistency of the merged model is automatically checked.

3.2 Validation of structural deductions

After running the deduction process (cf. (5) Fig.2), the plug-in deduces mappings and inconsistency problems. The different deductions are presented in separate windows as it can be seen in (cf. (6, 7, 8) Fig.2). The designer can confirm, choose the right one among several solutions, and view inconsistency problems that are in the specific model to eventually modify it outside the plug-in. Problems can be exported in text file (cf. (9) Fig.2). If no inconsistency problems are deduced, the plug-in proposes the creation of the merged model (cf. (10) Fig.2).

Figure 2 describes the different mappings we defined and the results obtained by the deduction process. Specialization mappings have been defined by us between the classes *Historical_Representation* and *History*, and between *User_representation* and *User* (cf. (4) Fig.2). The deduction process has deduced new additional mappings and inconsistency problems. One of these mappings is the relation *has_Required_Attribute*, a sub property of the relation *has_First_Name* (cf. (6) Fig.2). An inconsistency problem has been discovered between the relations *has_Acquired_attribute* and *has_Optional_attribute* (a cardinality problem) (cf. (7) Fig.2) and a missing mapping is indicated for the relation of the generic model *has_Knowledge_attribute* (cf. (8) Fig.2). The user can save all inconsistency problems and missing mappings in a text file.This text file will contain the defined mappings between classes and all found problems.

4 Related works

There are several approaches for performing a semantic integration depending on the degree of integration usually referred to as ontology mapping, aligning or merging. The process of ontology merging takes as input two (or more) source ontologies and returns a merged ontology based on the given source ontologies. Several systems and frameworks for supporting the knowledge engineer in the ontology merging task have been proposed [2, 3, 4]. These approaches are based either on instances of the two given ontologies that are to be mapped (bottom-up) or on concepts (top-down). None of them have been used to merge abstract models with specialized ones. In this paper, we focus on this specific point. The models to be merged are relatively small. The merging process is performed once at the design time. Generic models are composed of abstract classes which have no instances. The designer of the system knows the models to be integrated in the system very well and can then provide simple correspondences between their elements.

---

[2] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessTab

[3] http://herzberg.ca.sandia.gov/

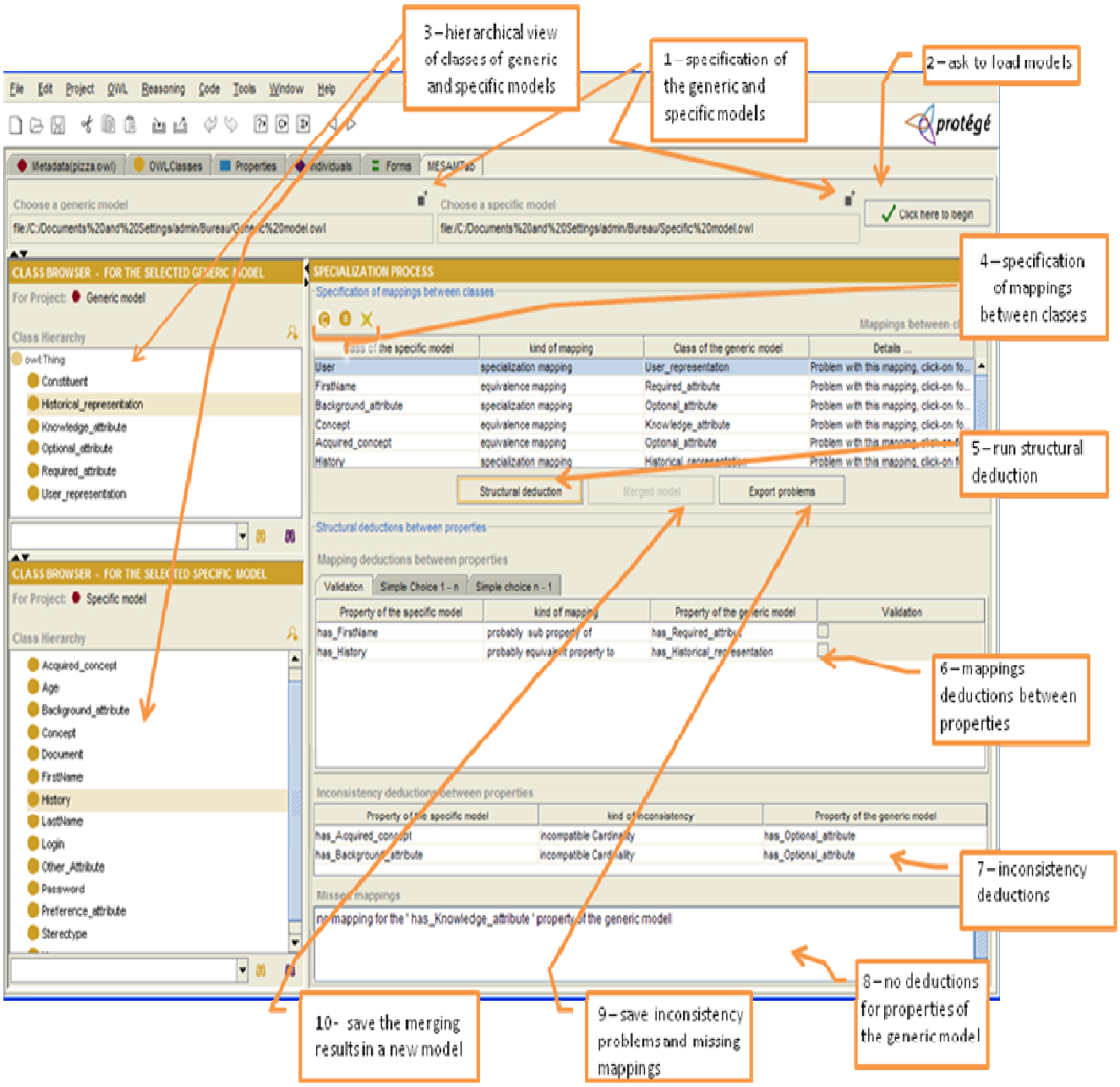

Fig.2 Work space of MESAM plug-in

5 Conclusion and future works

Throughout the MESAM plug-in, we have proposed a solution enabling the designer to reuse its own models and consequently his own resources and their metadata (the instantiations of the models) to be reused by the generic system. The MESAM plug-in allowed us to make some experiments in Adaptive Hypermedia Systems. The first one has been done in the e-learning domain aiming at building a user's model, which is usable in the GLAM system [5]. We have personally played the role of a designer. A user model developed by our team at Supélec [5] has been chosen as a specific model. As future work, we plan to add several extensions for the plug-in. First, we intend to consider inconsistency problems and to help the designer in resolving them. Then, we intend to extend our plug-in, taking into account disjunctions.

REFERENCES

1. Zemirline N., Reynaud C., Bourda Y., Popineau F.: A Pattern and Rule-Based Approach for Reusing Adaptive Hypermedia Creator's Models. In: 16th EKAW, PP. 17-31, Springer, Catania, Italy (2008)
2. Stumme, G., Maedche, A.: FCA-MERGE: bottom-up merging of ontologies. In: 17th IJCAI, pp. 225-234, Seattle, Washington, USA (2001)
3. Gomez-Perez, A., Angele, J., Fernandez-Lopez, M., Christophides, V., Stutt, A., Sure, Y.: A survey on ontology tools. In: OntoWeb deliverable 1.3 Universidad Politecnica de Madrid (2002)
4. Noy, N.F., Musen M. A.: The PROMPT Suite: Interactive Tools for Ontology Merging And Mapping. In: IJHCS, vol. 59, no. 6, pp. 983-1024, Elsevier (2003)
5. Jacquiot, C., Bourda, Y,. Popineau, F., Delteil, A., Reynaud. C.: GLAM: A generic layered adaptation model for adaptive hypermedia systems. In: 4th International AH2006, Springer, pp. 131–140. Springer, Heidelberg, Allemagne (2006)