A Pattern and Rule-based Approach for Reusing Adaptive Hypermedia Creator's Models

Nadjet Zemirline[1], Chantal Reynaud[2], Yolaine Bourda[1], 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, F-57070 Metz, France
Fabrice.Popineau@supelec.fr

Abstract. The design of Adaptive Hypermedia is a difficult task which can be made easier if generic systems and AH creators' models are reused. We address this design problem in the setting of the GLAM platform only made up of generic components. In this paper, we assume the GLAM platform is used to create a specific adaptive hypermedia. We present a pattern and a rule-based approach helping a AH creator in reusing its user and domain models and instances in order to make them taken into account. This semi-automatic approach takes the creator's models as specialisations of GLAM generic models and requires the creator to express a minimum set of mappings between his models and the generic ones. The process results in a merged model consisting of the generic and the corresponding specific model, being fully compliant with the GLAM adaptation model. A plug-in and experimentations in the e-learning domain have been partially designed.

Keywords: assisting tools, reusing models, models merging, adaptive hypermedia.

1 Introduction

Nowadays, there is a growing demand for personalization and the "one-size-fits-all" approach for hypermedia systems is no longer applicable. Adaptive hypermedia (AH) systems adapt their behaviour to the needs of individual users. The following definition [1] is the more widely used: " by adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user". Thus, adaptive hypermedia systems are tools to access information based upon the user's profile represented in a user's model. They also require a domain model to represent the application domain knowledge. These two kinds of models may be expressed in an AH-specific language or a standard language (RDF, OWL).

Adaptation mechanisms, either rule or trigger based, which are needed in adaptive hypermedia rely on these models.

The creation of an adaptive hypermedia system is too often made from scratch and the re-use of existing models (user or domain) is very rare although more and more annotated resources are available. For example, in the educational domain, many learning resources, developed using a LMS (Learning Management System) or an XML tool, are now available, and are described using metadata (for example using the Learning Object Metadata standard or LOM). But, if a user wants to use a specific AH system, he needs to translate his models into the specific format 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 creator of an adaptive hypermedia to reuse his models (his vocabulary) and his models' instantiations without any change of format or vocabulary.

We are currently working on the GLAM (Generic Layered Adaptation Model) [2] platform defined for an entire class of adaptive hypermedia systems. The platform is made of a generic adaptation model relying on generic user and domain models. Specific systems can be obtained by specializing the GLAM generic user and domain 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 adaptive hypermedia systems creators to specialize the user and the domain model using their own models in the setting of GLAM. We aim at automating this process which has been so far entirely manual. Our objectives are twofold: on one hand, to create a support for defining mappings between elements in GLAM generic models and elements in the creator's personal models, on the other hand, to help creating consistent and relevant models integrating the generic and specific ones and taking into account the mappings between them. The proposed approach is applicable either to user or domain models. It relies on OWL[1], a W3C standard and SWRL[2], a W3C proposal. GLAM's models are expressed in OWL, so we expect that the AH creator's models are also expressed in OWL. This is not restrictive because OWL is widely used.

The paper is organized as follows. In section 2, we present the main aspects of our approach. The pattern-based approach is described in section 3 while the rule-based approach is detailed in section 4. In section 5 we address the validation step and in section 6 the implementation of our plug-in and experimentations made in the e-learning domain are presented. In section 7, we describe close related works. Finally, section 8 concludes the paper.

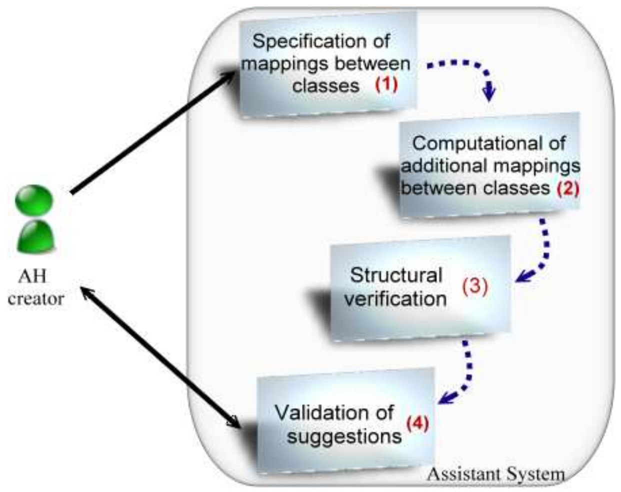
[1] www.w3.org/TR/owl-features/

[2] http://www.w3.org/Submission/SWRL/

2 Main aspects of the approach

Given two models, a generic model belonging to the GLAM platform and a specific model provided by a particular AH creator, we propose an approach to support the construction of a model that would integrate all the particularities of the specific model and be usable by the GLAM adaptation engine. The generic and specific models may be either user or domain models. The same approach is usable for both kinds of models. In the approach, mappings must be defined between elements of both models and then validated at the structural and semantic level. Our approach relies on the AH creator 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 our system. The main steps of the approach are the following:

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



Specification of mappings between classes (1)
Computational of additional mappings between classes (2)
Structural verification (3)
Validation of suggestions (4)
AH creator
Assistant System

Fig. 1 The diagram of the architecture of our assistant system.

In this paper, we only consider equivalence and specialization mappings. As our aim is to reuse GLAM adaptation model, which includes rules expressed on generic user and domain models, equivalence and specialization mappings between elements of the generic and specific models allows the AH creator to reuse directly the GLAM adaptation model. In the following sections, we focus more particularly on the steps 2, 3 and 4 of the approach which are described in the sections 3, 4 and 5 respectively.

Furthermore, we will adopt the following notations: $C_{m,i}$ to represent the class i from the model m, and $R_{m,d,j}$ to represent the relation j with the domain d in the model m.

3 Using a pattern-based approach to deduce additional mappings between classes

Starting from the mappings between classes specified by the AH creator, other mappings can be automatically deduced. We propose to adopt a pattern-based approach to achieve this deduction. Pattern-based approaches for mapping identification across models assume that structural regularities always characterize the same kind of relations.

We have defined 8 patterns which are characterizations of structural contexts composed of 3 classes, either two classes of the generic model and a class of the specific model or two classes of the specific model and a class of the generic model (2 categories). The idea is to deduce the nature of the relation R (equivalence or specialization) between $C_{s,1}$ a class of the specific model and $C_{g,1}$ a class of the generic model, when a third class belonging to one of the two models, $C_{m,2}$, is linked to $C_{s,1}$ by a relation R_1 and to $C_{g,1}$ by a relation R_2, R_1 and R_2 being either equivalence or specialization relations. We identified four patterns per structural context category to represent all possible cases, that is to say 8 patterns all in all.

Given R_{equiv} an equivalence relation and $R_{subClass}$ a specialization relation, the deduction of supplementary mappings is based on the composition (noted o) properties of these two kinds of relations described below:

$R_{equiv} \circ R_{subClass} = R_{subclass}$ $\quad$ $R_{subclass} \circ R_{equiv} = R_{subclass}$

$R_{subclass} \circ R_{subclass} = R_{subclass}$ $\quad$ $R_{equiv} \circ R_{equiv} = R_{equiv}$

The patterns we have defined are generic and usable only to identify mappings between classes. They are expressed using SWRL. Fig. 2 is an illustration of a pattern belonging to the 1st category. Thanks to this pattern, one can deduce that there is a subclass relation between $C_{s,1}$ and $C_{g,1}$.

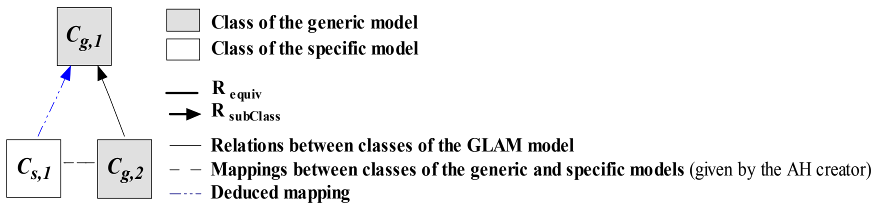

$C_{g,1}$
$C_{s,1}$
$C_{g,2}$
Class of the generic model
Class of the specific model
R equiv
R subClass
Relations between classes of the GLAM model
Mappings between classes of the generic and specific models (given by the AH creator)
Deduced mapping

Fig. 2 An example of a pattern

4 A rule-based approach

In this section our objective is twofold. First goal is to automatically deduce mappings between relations and between attributes of classes of the generic and specific models.

Second goal is to check the consistency of the new model created by the merging process. To do so, our system uses structural knowledge applicable to whatever the model is (user or domain model) (cf. section 4.1). As detailed in section 4.2, as models are expressed in OWL, structural knowledge has been modelled in a meta-model based on the OWL meta-model[3]. Inferences on knowledge modelled in the meta-model are performed using SWRL rules (cf. section 4.3, section 4.4).

4.1 Structural knowledge

First of all, let us note that we only consider OWL models. In OWL, a model includes a set of classes and a set of properties. A property is a binary relation. It is either a relation between an individual and a datatype (representing an attribute) or a relation between two individuals (representing a relation between two instances). Property restrictions such as cardinality constraints (OWL:maxCardinality, OWL:minCardinality or OWL:Cardinality) and characteristics (functional or inverse functional) can be used in the description of classes or of properties.

The exploitation of structural knowledge aims at defining the nature of mapping links between OWL properties which are referred to in this paper by relations because relations (in its usual meaning) and attributes are both represented by properties in OWL.

In our approach, the deduction of mappings between relations is inferred from information characterizing the compatibility of the relations. A mapping between two relations is possible only when the relations are compatible. A mapping may be either a potential or a probable link according to the compatibility information (inferred from mappings between classes and from properties restrictions) associated to the mapped relations.

Definition 1: Two relations $R_{s,i,j}$ and $R_{g,k,l}$ are linked by a *potential link* if a mapping is defined between their domain and between their range.

Definition 2: Restrictions relative to two relations $R_{s,i,j}$ and $R_{g,k,l}$ are *compatible* if those relations are linked by a potential link and if:

1. ($\mathrm{Cardinality}_{max}(R_{s,i,j}) \leq \mathrm{Cardinality}_{max}(R_{g,k,l})$
   and $\mathrm{Cardinality}_{min}(R_{s,i,j}) \geq \mathrm{Cardinality}_{min}(R_{g,k,l})$)
   or $\mathrm{Cardinality}_{value}(R_{s,i,j}) = \mathrm{Cardinality}_{value}(R_{g,k,l})$.

or

2. $R_{s,i,j}$ and $R_{g,k,l}$ are both functional or not (resp. inverse functional or not) or $R_{s,i,j}$ is functional (resp. inverse functional) and $R_{g,k,l}$ is not.

Definition 3: Two relations $R_{s,i,j}$ and $R_{g,k,l}$ are linked by a *probable link* if they are linked by a potential link and if their restrictions are compatible.

Probable links can be either equivalence or specialization links according to the nature of mapping between the classes corresponding to the range and according to the restrictions associated to the relations.

Definition 4: A probable link between $R_{s,i,j}$ and $R_{g,k,l}$ is an *equivalence probable link* if the two ranges are linked by an equivalence relation and if they have the same restrictions.

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

Definition 5: A probable link between $R_{s,i,j}$ and $R_{g,k,l}$ is a *specialization probable link* if a mapping is defined between their range but the restrictions on $R_{s,i,j}$ are stronger than those on $R_{g,k,l}$ or if they have the same restrictions but the $R_{s,i,j}$ range is a subcategory of the $R_{g,k,l}$ range.
Note: Probable links as their name indicates it, are only probable and are not sure. Thus they will be proposed to the AH creator for validation or eventual modification of the specific model (Section 5).

4.2 Modelling structural knowledge

As the models to be merged are represented in OWL, we propose to represent structural knowledge in a meta-model based on the OWL meta-model. The OWL meta-model was defined by ODM (Ontology Definition Meta Model) of OMG as a MOF2 compliant meta-model. It is composed of several UML class diagrams, a class diagram per element of an OWL model. Our system does not need all the diagrams of the OWL meta-model. We describe the reused diagrams in section 4.2.1. Furthermore, in section 4.2.2, we present how the model coming from the OWL meta-model has been enriched in order to represent the needed structural knowledge described in section 4.1.

4.2.1 Reused Parts of the OWL meta-model

As structural knowledge is relative to classes, properties and restrictions according to the OWL terminology, we reused the *Class*, *Property* and *Restriction* class diagrams in the OWL meta-model. In the *Class* diagram, the *Class* and *Restriction* classes and the *equivalentClass* and *subclass* relations are needed. The *Restriction* diagram has been restricted to the following three classes: *Cardinality Restriction*, *Max Cardinality Restriction* and *Min Cardinality Restriction*. On the other hand, the *Property* diagram has been entirely reused.

4.2.2 Enrichment of the Reused Parts of the OWL meta-model

We enriched our meta-model in order to model structural knowledge by introducing the needed relations (cf. section 3.1). Furthermore, we brought some modifications on the reused part of the OWL meta-model. Indeed, in that meta-model, the XML-Schema datatypes are considered as individuals of the class *Class*. That representation is not convenient for us because some characteristics of classes that we have to represent are not relevant for datatypes. So, we decided to add a new class specialization of C*lass*, denoted *Application Class,* whose individuals are OWL classes different from datatypes. *Application Class has* an attribute *model* which takes *generic* or *specific* value in order to differentiate between individuals being initial elements either of the generic model or of the specific model. The resulting meta-model is presented Fig. 3.

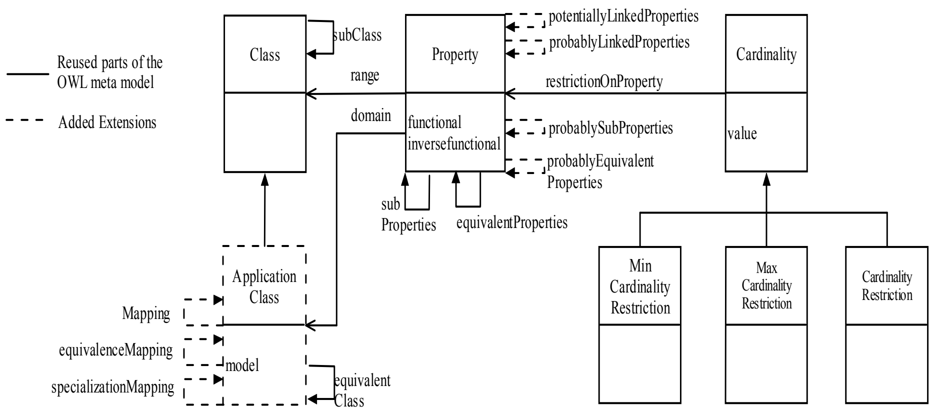


Fig. 3 The proposed meta-model

4.3 Mapping Deduction Rules

In this section, we give the rules to deduce mappings between relations of the generic and specific models. The rules derive from the definitions given in section 4.1 and are based on the proposed meta-model (cf. section 4.2).

4.3.1 Deducing a potential mapping

The rule inferring a potential mapping derives directly from Definition 1, (R1).

```
Property(?Pg) ^ model(?Pg,"generic") ^ domain(?Pg,?Dg)^
range(?Pg,?Rg) ^ Property(?Ps) ^ model(?Ps,"specific") ^
domain(?Ps,?Ds) ^ range(?Ps,?Rs) ^
mapping(?Dg,?Ds) ^ mapping(?Rg,?Rs)
→potentiallyLinkedProperties(?Pg,?Ps)
```

mapping(?Cg, ?Cs) expresses a mapping between a class of the generic model and a class of the specific model. It is either defined by the AH creator or inferred from additional mappings automatically deduced.

4.3.2 Deducing compatible restrictions

Table.1 groups all cases where a relation of the generic model Pg and a relation of the specific model Ps are linked by a potential link and have compatible restrictions. It also includes the corresponding deductions and rules number.

We will not give the associated code for each rule. As an example, here is the R3 rule:

```
potentialLinkedProperties(?Pg,?Ps) ^
functional(?Pg,false) ^ functional(?Ps,true)→
sameFunctionality(?Pg,?Ps) ^
compatibleFunctionality(?Pg,?Ps)
```

Definition.2 relative to compatible restrictions is expressed by the following rule (R10) which takes into account default values for restrictions:

```
compatibleFunctionality(?Pg,?Ps) ^ compatibleCardinality
(?Pg,?Ps) ^ compatibleInverseFunctionality(?Pg,?Ps)
→compatibleRestriction(?Pg,?Ps)
```

Table.1 Compatible restrictions

	Pg	Ps	Associated predicates	Rules
Functional	True	True	`sameFunctionality(Pg,Ps)^ compatibleFunctionality(Pg,Ps)`	R2
	False	False		
	False	True	`restrictiveFunctionality(Pg,Ps)^ compatibleFunctionality(Pg,Ps)`	R3
InverseFunctional	True	True	`sameInverseFunctionality(Pg,Ps)^ compatibleInverseFunctionality (Pg,Ps)`	R4
	False	False		
	False	True	`restrictiveInverseFunctionality (Pg,Ps)^ compatibleInverseFunctionality (Pg,Ps)`	R5
Cardinality	$Card_{min}(P_g)=Card_{min}(P_s)$ $Card_{max}(P_g)=Card_{max}(P_s)$		`sameCardinality(Pg,Ps)^ compatibleCardinality(Pg,Ps)`	R6
	$Card_{value}(P_g)=Card_{value}(P_s)$			R7
	$Card_{min}(P_g)<Card_{min}(P_s)$ $Card_{max}(P_g)=Card_{max}(P_s)$		`restrictiveCardinality(Pg,Ps) ^ compatibleCardinality(Pg,Ps)`	R8
	$Card_{min}(P_g)\leq Card_{min}(P_s)$ $Card_{max}(P_g)>Card_{max}(P_s)$			R9

4.3.3 Deducing a probable mapping

The rule inferring a probable mapping derives directly from Definition 3, (R11).

```
potentiallyLinkedProperties(?Pg,?Ps) ^
compatibleRestriction(?Pg,?Ps)
→probablyLinkedProperties(?Pg,?Ps)
```

Two kinds of probable mappings are distinguished. A rule arising directly from Definition 4 allows deducing an equivalence probable mapping link (R12). The deduction of a specialization probable mapping link can be expressed by the following formula: *Probable link* $\wedge$ (*Restrictive range* $\vee$ *restrictive functional* $\vee$ *restrictive inverse functional* $\vee$ *restrictive cardinality).* As the disjunction operator doesn't exist in SWRL, four rules (R13, R14, R15, R16) are needed to deduce a specialization probable link. Here is one of these four rules (R13):

```
probablyLinkedProperties(?Ps,?Pg) ^ range(?Pg,?Rg) ^
range(?Ps,?Rs) ^ mapping(?Rg,?Rs)
→probablySubProperties(?Pg,?Ps)
```

4.4 Inconsistency deduction rules

Inconsistencies relate to potential mappings and derive directly from restrictions. If a relation from the generic model is more restrictive than the potential mapped relation of the specific model, restrictions of the two relations are incompatible. The deduced inconsistencies are submitted to the AH creator for eventual modification of the specific model (cf. section 5). Table.2 describes all possible incompatibilities, the corresponding deductions and rules numbers.

Table .2 Incompatible restrictions.

Constraints	Pg	Ps	Associated predicates	Rules
Functional	True	False	`incompatibleFunctionality (Pg,Ps)^ incompatibleRestriction (Pg,Ps)`	R17
Inverse Functional	True	False	`incompatibleInverseFunctionality (Pg,Ps)^ incompatibleRestriction (Pg,Ps)`	R18
Cardinality	$Card_{min}(P_g)>Card_{min}(P_s)$		`incompatibleCardinality (Pg,Ps) ^ incompatibleRestriction (Pg,Ps)`	R19
	$Card_{max}(P_g)<Card_{max}(P_s)$			R20
	$Card_{value}(P_g)\neq Card_{value}(P_s)$			R21

5 Validation of the mappings and solving inconsistencies

In this section, we take advantage of deductions made in sections 3 and 4. So, for each class of the generic model, all relations R whose domain is the class of the generic model are analysed according to its mappings. We distinguish two cases. In the first case, there is at least a relation in the specific model linked to R by a probable mapping link. As the deduction of the potential mappings is only based on structural knowledge, it is presented to the AH creator for validation. Table 3 describes the suggestions made to the AH creator depending on the fact there exists only one or several mapped probable relations. In the second case, no relations are linked by a probable link due to incompatible restrictions. This will be interpreted as an error and the AH creator will be asked for modifications.

Table.3 Interactions with the AH creator during the validation phase

Interpretation		Asking the AH creator
Pg and Ps are linked by a probable link	Pg linked by a unique potential mapping link. Ps linked by a unique potential mapping link.	Validation of this probable mapping
	Pg linked by multiple probable or potential mapping links.	Choice of the correct probable mapping

Pg and Ps have incompatible restrictions	--	Modification of the restriction of Ps

6 Implementation

In order to test and validate our approach, we have implemented *MESAM*, a plug-in for Protégé 2000[4]. Currently, MESAM (*Model mErging by Specialization of Abstract and generic Models*) is under test.

In section 6.1, we describe the architecture of the MESAM plug-in. A general overview of its implementation is presented in section 6.2. Finally, in section 6.3 we illustrate through an example the execution of the plug-in

6.1 Architecture of MESAM plug-in

As described in Figure 4, the plug-in includes two parts.

First, a knowledge part gathers generic models (GLAM generic models in our case), the meta-model and deduction rules (4 rules related to patterns and 21 mappings and inconsistency deduction rules). All these components are reusable across applications.

Second, the process 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[5] to execute SWRL rules using the Jess inference engine[6].

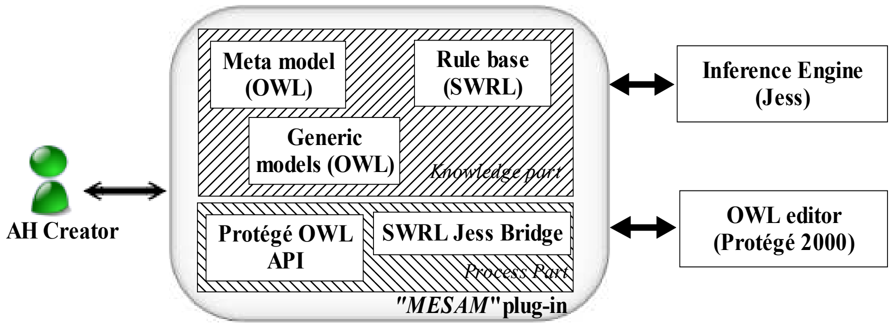


Fig. 4 Architecture of the “MESAM” plug-in

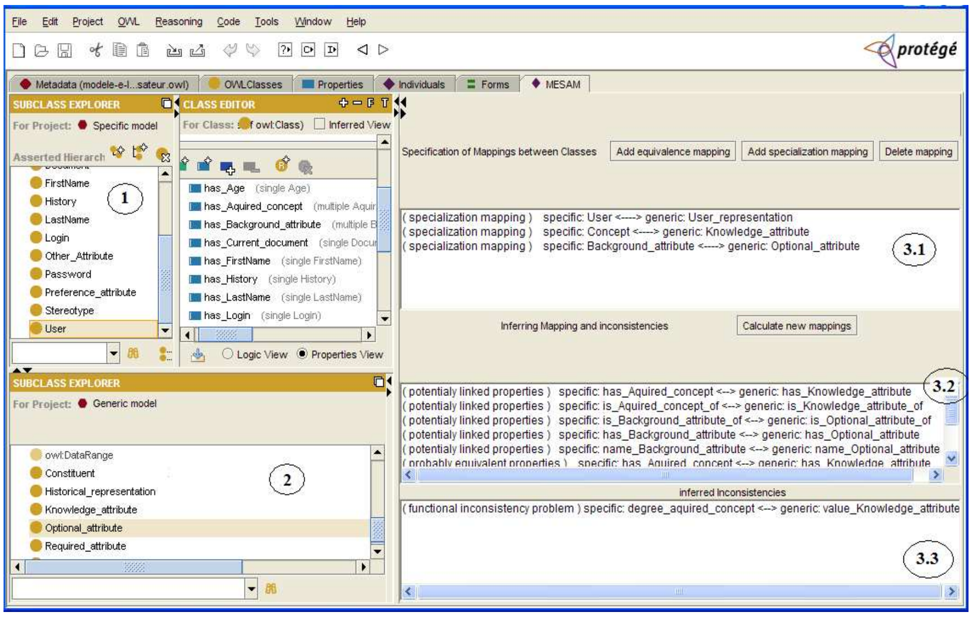
[4] http://protege.stanford.edu/
[5] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLJessTab
[6] http://herzberg.ca.sandia.gov/

6.2 The Protege MESAM plug-in

MESAM plug-in proposes an intuitive work space (cf. Fig. 5) which is divided into three main windows:

A first window (cf. (1) Fig 5) enables the visualization and eventual modification of the specific model. It proposes the same functionalities as the *OWLClasses tab*.

A second window (cf. (2) Fig 5) enables the visualization of the generic model. Classes of the generic model are organized in a hierarchical form.



```
( specialization mapping )   specific: User <----> generic: User_representation
( specialization mapping )   specific: Concept <----> generic: Knowledge_attribute
( specialization mapping )   specific: Background_attribute <----> generic: Optional_attribute
```

```
( potentialy linked properties )  specific: has_Aquired_concept <--> generic: has_Knowledge_attribute
( potentialy linked properties )  specific: is_Aquired_concept_of <--> generic: is_Knowledge_attribute_of
( potentialy linked properties )  specific: is_Background_attribute_of <--> generic: is_Optional_attribute_of
( potentialy linked properties )  specific: has_Background_attribute <--> generic: has_Optional_attribute
( potentialy linked properties )  specific: name_Background_attribute <--> generic: name_Optional_attribute
( probably equivalent properties )  specific: has_Aquired_concept <--> generic: has_Knowledge_attribute
```

```
( functional inconsistency problem ) specific: degree_aquired_concept <--> generic: value_Knowledge_attribute
```

Fig. 5 Work space of MESAM plug-in

A third window (cf. (3) Fig 5) concerns the specialization process, described in section 2 and it includes two parts:

– A first part (cf. (3.1) Fig 5) enables to define specialization and equivalence mappings between classes of the generic and specific models. It can be done in two steps: first, selecting a class of the generic model and another one of the specific model, second clicking on the "*add equivalence mapping"* or "*add specialization mapping"* buttons according to the type of mappings to be defined.
– A second part enables to infer new mappings between classes and between properties, and to infer potential inconsistencies. Results will be printed in the corresponded textbox: (cf. (3.2) Fig 5) for mapping deductions and (cf. (3.2) Fig 5) for inconsistency deductions.

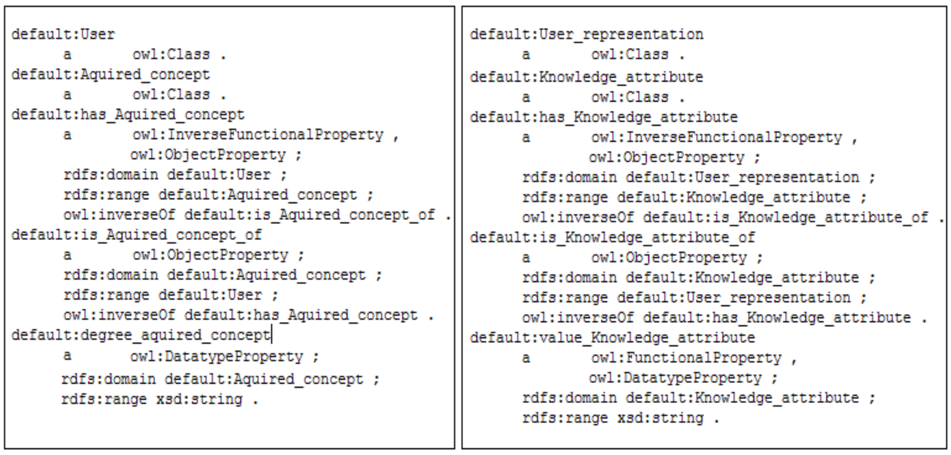
6.3 Example of execution of MESAM plug-in

MESAM plug-in allowed us to make some experiments. The first one has been done in the e-learning domain aiming at building a user's model usable in the GLAM system. We have personally played the role of an AH creator. A user model developed in our team at Supélec [11] has been chosen as specific model.

In order to perform the specialization process of the generic GLAM model by our specific model, we need to launch the protégé OWL editor and to load our experimental plug-in.

The meta-model is loaded inside the plug-in, then meta-model instances are generated from the specific and generic models. We have chosen to keep these processes invisible to the AH creator. So, in our plug-in, the AH creator will have the illusion to work over the specific and generic models.

We have done several tests, testing each time multiple cases and so validating our approach proposed in section 2. Through the figures 5 and 6, we illustrate one of the tests we have done. Figure 6 (A) illustrates a part of the specific model, figure 6 (B) illustrates a part of the generic model.



```
default:User
      a       owl:Class .
default:Aquired_concept
      a       owl:Class .
default:has_Aquired_concept
      a       owl:InverseFunctionalProperty ,
              owl:ObjectProperty ;
      rdfs:domain default:User ;
      rdfs:range default:Aquired_concept ;
      owl:inverseOf default:is_Aquired_concept_of .
default:is_Aquired_concept_of
      a       owl:ObjectProperty ;
      rdfs:domain default:Aquired_concept ;
      rdfs:range default:User ;
      owl:inverseOf default:has_Aquired_concept .
default:degree_aquired_concept
      a       owl:DatatypeProperty ;
      rdfs:domain default:Aquired_concept ;
      rdfs:range xsd:string .
```

(A) part of the specific model

```
default:User_representation
      a       owl:Class .
default:Knowledge_attribute
      a       owl:Class .
default:has_Knowledge_attribute
      a       owl:InverseFunctionalProperty ,
              owl:ObjectProperty ;
      rdfs:domain default:User_representation ;
      rdfs:range default:Knowledge_attribute ;
      owl:inverseOf default:is_Knowledge_attribute_of .
default:is_Knowledge_attribute_of
      a       owl:ObjectProperty ;
      rdfs:domain default:Knowledge_attribute ;
      rdfs:range default:User_representation ;
      owl:inverseOf default:has_Knowledge_attribute .
default:value_Knowledge_attribute
      a       owl:FunctionalProperty ,
              owl:DatatypeProperty ;
      rdfs:domain default:Knowledge_attribute ;
      rdfs:range xsd:string .
```

(B) part of the generic model

Fig. 6 Parts of the generic and specific user model

Figure 5 describes the different mapping we defined, describes also the results of execution done after launching the deduction process. Among the specified mappings, we defined specialization mappings between the classes *Knowledge_attribute* and *Concept*, and between *User_representation* and *User* (cf. (3.1) Fig 5).

Launching the deduction process has enabled to deduce mappings and inconsistency problems. Among the results, the plug-in has deduced that the relations *has_Aquired_concept* and *has_Knowledge_attribute* may be equivalent, as the relation *is_Aquired_concept_of* and *is_Knowledge_attribute_of* may also be equivalent (cf. (3.3) Fig 5). It has also deduced an inconsistency problem between the attributes *degree_aquired_concept* and *value_Knowledge_attribute* (a functional inconsistency) (cf. (3.3) Fig 5).

7 Closely related works

There is related work both in the Adaptive Hypermedia (AH) and in the Knowledge Engineering communities.
In the AH community, AH have proved their benefits, particularly in the educational domain [3], but authoring an adaptive hypermedia for a particular need is still a difficult task [4]. The AH community is now taking an active interest in this matter, as the series of workshops about that topic testifies (A3H at UM2007 in Corfu, A3H at AH2006 in Dublin, A3H at AIED05 in Amsterdam…).

Some freely available adaptive hypermedia systems, which are, in fact adaptive educational hypermedia systems, like AHA![7] [5] or WHURLE[8] [6], come with an authoring tool but they required to learn how to use the system and it is necessary to adapt the annotated resources to the format used by the system. The lack of standards leads users to be captive of a particular system. It is possible to distinguish two distinct issues: the first one is about interoperability of AH systems (how to use resources developed for one system inside another one), the second one is about the re-use of existing resources.

Considering the first issue, what happens if the system used is no longer maintained or if users want to change? Do they need to re-create all their resources? It is desirable to move away from a “one-to-one” adaptive hypermedia authoring paradigm to a general “many-to-many” [7]. One possible solution today is to use MOT (My Online Teacher) [8], as an authoring tool because it allows a conversion [9] to AHA! or WHURLE. But the AH creator cannot simply reuse his models and consequently the corresponding descriptions of his resources.

Considering the second issue, what happens if annotated resources are pre-existing to the adaptive hypermedia? One possibility is to reuse metadata, for example automatically generated metadata in a semantic desktop environment [10]. A proposed solution is based on Beagle for the automatic generation of metadata, and on the MOT system for the AH system reusing the Beagle metadata. But, it is necessary to translate the Beagle metadata in the Mot format. Another possibility is to rely on the re-use of educational resources developed for a system based on a widespread standard IMS-LD in order to enable authoring of sequencing strategies that can be re-used in any system based on IMS-LD [11] or modeling of adaptive rules by means of IMS-LD [12]. But these solutions are based on the use of IMS-LD.

To conclude, some systems can re-use resources described in a particular standard (like IMS-LD) or format. Other systems can translate resources from one particular adaptive system to another one. But, all those systems have developed “ad-hoc” solutions closed to the adaptive systems used and are not applicable to other adaptive systems. In fact, till now, no system allows the AH creator to directly integrate his resources and his models in an adaptive system. We make a proposal to fill this gap.

In the knowledge engineering community, research in the integration of heterogeneous information systems has been put in perspective since several years. There are several approaches for performing a semantic integration depending on the degree of integration usually referred as ontology mapping, aligning or merging. The

[7] http://aha.win.tue.nl
[8] http://whurle.sourceforge.net/

process of ontology merging takes as input two (or more) source ontologies and returns a merged ontology based on the given source ontologies. As manual ontology merging using editing tools without support is difficult, labor intensive and error prone, several systems and frameworks for supporting the knowledge engineer in the ontology merging task have been proposed [13,14,15,16]. These approaches are based either on instances of the two given ontologies that are to be mapped (bottom-up) or on concepts (top-down).

FCA-MERGE [14] merges ontologies following a bottom-up approach. It extracts instances from a given set of domain-specific text documents (by applying natural language processing techniques) then apply techniques taken from Formal Concept Analysis. The produced result is explored and transformed to the merged ontology by the knowledge engineer.

ODE-MERGE or PROMPT follow a top-down approach based on concepts. ODE-MERGE [15] is integrated in WebODE. It compares two ontologies in a completely automatic way, considering tables of synonymy or hyperonymy. Obtained results are evaluated by the knowledge engineer. I-PROMPT [16] is an interactive tool for ontology merging integrated into the PROMPT framework. It asks users during the merging process. The process goes through the following cycle. It generates suggestions based on the structure of the ontologies, the user triggers an operation by selecting one of its suggestions. Then, I-PROMPT performs the operations and automatically executes changes in the merged ontology based on the type of operation.

So, to conclude, these systems are either based on instances or on concepts. They are either automatic or interactive tools. However, despite they have been used with many different ontologies and in many different domains, they have not been used to merge abstract models with specialized ones and have not been applied to AH systems. In this paper, we focus on this specific point. The models to be merged are relatively small. The merging process is performed once at 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. Given these correspondences, the software implemented our approach is expected to reason about the models and to generate additional correspondences between classes, properties or relations. The hypothesis underlying this work is that it is much easier for AH designers to specify simple correspondences from classes in the models and then evaluate mappings returned by the system, the consistency of the merged model being automatically checked.

8 Conclusion and future work

In this paper, we have proposed a solution enabling the user to create an adaptive hypermedia with the GLAM system re-using his own models and consequently his own resources and their metadata (the instantiations of the models). This approach relies on the AH creator to start the design process. He has to specify a minimum set of mappings between classes from which the system automatically deduces all the other possible mappings and potential inconsistencies. Above all, this approach enables to check and to validate the model resulting from the merging process. That way the merged model can be immediately used by the GLAM adaptation engine. This approach is generic and consequently usable whatever the application domain is.

We now intend to complete the implementation of our plug-in and then to propose it to real AH creators. Future experiments will be done with Supélec teachers. It can also be interesting to consider the relations between the adaptation rules and the user and domain models, in particular when a mapping is not defined for a class of a (generic or specific) model. We envision an extension enabling AH creators to interact with the adaptation model. Finally, our solution is based on the use of OWL to express the models and it is not dependent on the use of GLAM. So, as future work, we plan to consider the application of this approach to other systems.

References

1. Brusilovsky, P.: Methods and techniques of adaptive hypermedia, User Modeling and User Adapted Interaction. vol. 6, no. 2-3, pp. 87-129 (1996)
2. 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)
3. Brusilovsky, P.: Adaptive hypermedia, User Modeling and User Adapted Interaction. vol. 11, no. 1-2, pp. 87-110 (2001)
4. Celik, I., Stewart, C., Ashman, H.: Interoperability as an Aid to Authoring: Accessing User Models in Multiple AEH Systems. In: 1st Adaptive and Adaptable Educational Hypermedia workshop at AH, pp. 71-85. Springer, Heidelberg, Allemagne (2006)
5. De Bra, P., Smits, D., Stash, N.: Creating and Delivering Adaptive Courses with AHA! In: 1st European Conference on Technology Enhanced Learning, EC-TEL 2006, pp. 21-33, Springer, Crete (2006)
6. Zakaria, M.R. Moore, A., Stewart, C.D., Brailsford, T.J.: “Pluggable” user models for adaptive hypermedia in education. In: the Fourteenth ACM Conference on Hypertext and Hypermedia, pp. 170-171, Nottingham, UK (2003)
7. Cristea, A., Stewart, C.: Automatic Authoring of Adaptive Educational Hypermedia. Book chapter in "Web-Based Intelligent e-Learning Systems: Technologies and Applications", ZongMin Ma (Ed.), Information Science Publishing (IDEA group), pp. 24-55 (2006)
8. Cristea, A.I., De Mooij, A.: Adaptive Course Authoring: My Online Teacher. In: ICT'03, Papeete, French Polynesia (2003)
9. Cristea, A.I., Smits, D., De Bra, P.: Writing MOT, Reading AHA! - converting between an authoring and a delivery system for adaptive educational hypermedia. In: A3EH workshop at AIED'05, Amsterdam, Netherlands (2005)
10. Hendrix, M., Cristea, A., Nejdl, W.: Authoring Adaptive learning Material on the Semantic Desktop. In: 4th International Workshop on Authoring of Adaptive and Adaptable (Educational) Hypermedia (A3EH), Dublin, Ireland (2006)
11. Gutiérrez, S., Pardo A., Kloos, C.D.: Authoring of Adaptive Sequencing for IMS-LD. In: 5th International Workshop on Authoring Adaptable and Adaptive Hypermedia, in User Modelling Conference (UM), pp. 12-19, Corfu, Greece (2007)
12. Berlanga, A.J, García, F.J, Carabias, J.: Authoring Adaptive Learning Designs Using IMS LD. AH, In: 4th International AH2006, pp. 31-40, Heidelberg, Allemagne (2006)
13. Mc Guinness, D. L., Fikes, R., Rice, J., Wilder, S.: An environment for Merging and Testing Large Ontologies. In: 7th International conference on Principles of Knowledge

Representation and Reasoning, KR-2000, pp. 483-493, Breckenridge, Colorado, USA (2000)

14. Stumme, G., Maedche, A.: FCA-MERGE: bottom-up merging of ontologies. In: 17th IJCAI, pp. 225-234, Seattle, Washington, USA (2001)
15. 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)
16. 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)