Modeling the Reusable Content of Adaptive Web-Based Applications Using an Ontology

Mária Bieliková and Michal Moravčík

Institute of Informatics and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovičova 3, 842 16 Bratislava, Slovakia
`bielik@fiit.stuba.sk`

Summary. Personalization becomes more common in web-based applications. More and more adaptive web-based applications that adapt information presentation (the content or navigation) employing goals and other characteristics of a user or context are developed. By increasing the number of existing adaptive applications the need for more effective creating and reusing the content among adaptive applications rises. In this chapter we present an approach to creating the content using ontological representation that is suitable for reusing. The ontology includes a domain model and domain dependent part of a user model, both specified on several levels of abstraction. We support the content transformation between several applications and its cross-system use that enables better use of best characteristics of each application without the need of manually creation of the content, which already exists. We evaluated the method using the domain ontology of teaching programming by examples. We claim that the content and structure of the adaptive application represented by the ontology is suitable for sharing and reusing when defining new applications.

1 Introduction

Growths of available information presented by current web-based information systems requires an assistance to a user in the task of finding relevant information. So an intelligent support of navigating in large information spaces by web-based applications is a key requirement today. One approach to achieve this is adapting information presentation (its layout or the content itself) or navigation to each user or a context of the presentation. A need for adaptive web-based applications was recognized a decade ago, and from that time new adaptive web-based applications are being still developed.

With respect to the increasing usage of adaptive web-based applications the need for effective authoring and content delivering becomes increasingly important. However, the content of current adaptive applications is generally not designed for cross-system usage. We can improve situation by providing means for developing reusable models of adaptive applications together with tools for creating instances of the models in various domains.

The base idea of modeling is to deliver general, simple, consistent, portable, reusable representation of the modeled subsystem. These characteristics are in the goals of several existing adaptive hypermedia reference models. By enabling a transformation of the content among adaptive applications the best features of individual applications can be employed. For example existing authoring systems can be reused as authoring tools not only for adaptive applications they were developed for but also for other applications which provide just content presentation. This principle is known as a metaphor *"authoring once, delivering many"* [16]. That is, an author should only to create a piece of information (e.g. educational material) once. This information should be easily re-usable in many other content presentation systems. This allows for the cost-effective creation of information as a user can use for authoring just one application.

Our aim is to support a reuse the content (domain) model of adaptive web-based applications employing an ontology as the knowledge structuring approach used extensively for the Semantic Web applications. The goals and possibilities of the Semantic Web designate the advantage of using the ontology as a domain model of adaptive applications. Several approaches expressing particular models of adaptive applications by means of the Semantic Web technologies exist (e.g., [1, 13, 15]). Nevertheless, the lack of effective reuse of adaptive applications models is still in course.

We present a method for modeling content of adaptive applications using ontology, exporting the content into intermediate format and importing it into (possible more than one) adaptive application. This approach enables also reuse of existing content by its transformation into the ontology. For evaluation of the method we selected the CAF format (Common Adaptation Format, [9]) and adaptive web-based application AHA! [10]. Moreover, we transformed the content of another adaptive web-based system ALEA [4] into our modeling environment, so enable sharing the content between several applications. We present proposed core ontology of the content used throughout the method together with examples from domain of teaching programming.

The chapter is organized as follows. In section 2 we present related work, i.e. existing approaches to adaptive application content modeling with stress on the Semantic Web technologies usage. We devote special attention to web-based application content representation approaches and its transformation between adaptive applications. Section 3 contains description of proposed method for domain modeling, importing existing domain model to the ontology representation and delivering the ontology content into existing adaptive applications. In Section 4 we present ontology models of the content used for domain a nd user modeling. This section is followed by discussing meta-model of the content that aims at delivery the content into the adaptive system with defined concept sequences (section 4). Section 6 includes discussion on proposed method evaluation in the domain of learning programming. The chapter concludes with conclusions and topis for future work.

2 Related work

Content-oriented adaptive applications modeling can be seen from two points of view: modeling the architecture of an application (models of generic web-based applications can be also considered) or modeling the content presented by the application (represented using a domain model). Architecture of adaptive web-based applications can be expressed using methods for modeling generic web-based applications (e.g., OOHDM [17], WebML [6], Hera [13]) even though the specification of adaptive behavior is not accordingly handled by all of these methods. Common architecture of adaptive hypermedia systems reflects reference models such as AHAM [20], Munich reference model [14] or LAOS [7], which define layers of the adaptive application by separating data models (e.g., domain, user, context, goals) from an adaptation engine. Reference models are used as a base for new adaptive applications.

Not all layers of adaptive application model are adequate to be shared and/or transferred between applications. The common for the applications is the content with definitions of its usage in a user model (expressed in domain dependent part of the user model).

2.1 Content representation

Modeling the content of a content oriented web-based application and its efficient representation is as well important as modeling the application itself. For content modeling it is important to analyze to what extent is a particular representation flexible for different domains together with the possibility of reasoning directed to decisions on properties of the information space (e.g., consistency). We do not consider proprietary formats as they almost totally prevent the sharing and reuse of the domain model.

Existing approaches to representing the content of a web-based application include mainly approaches using a relational database or an XML based language. XML offers powerful enough expressiveness. The performance of this solution is limited by the performance of the used file system (it is effective for domain models with few instances and rich structure of concept's characteristics). Reusability and sharing is better than with the database approach, thanks to the platform independence of XML. Using XML has also the advantage that it can be used directly in the Web environment. However, XML as a meta-language defines only general syntax without formally defined semantics, which leads to difficulties when reasoning is required. Moreover, everyone can invent his own names for tags; somebody stores attributes as tags; somebody uses the attributes of tags defined by XML syntax.

Both above mentioned approaches offer only a way of describing characteristics of domain model concepts and do not offer any added value from the content modeling perspective. Ontology-based approach offers a way of moving content modeling from the low-level describing of domain concept characteristics to a higher-level with additional possibilities (reasoning).

According to the most cited definition of ontology in the Semantic Web community, ontology is an explicit specification of the conceptualization of a domain [12]. The term ontology includes a whole range of models that show varied semantic richness. We represent the ontology by RDF/OWL formalisms (Resource Description Framework; Web Ontology Language). An approach based on RDF and its extension OWL takes the previously mentioned XML representation (syntax) and eliminates its disadvantage by defining a vocabulary for describing properties and classes. OWL serves as a common language for automated reasoning about the content for the vision of the Semantic Web.

The advantages leading to using ontologies for content modeling come from the fundamentals of this formalism. Ontologies provide a common understanding of the domain to facilitate reuse and harmonization of different terminologies. They support reasoning, which is considered as an important contribution of the ontology-based models. Although there exist several approaches where ontology is used as a base for models representation [15], usually specialized ontology for particular case is developed only. We use the ontology as a universal format for representation of models and define a method for their reusability.

2.2 Content transformation

Creating a schema of the domain and instances of it is serious bottleneck of content oriented web-based applications. Using ontology for representation of the domain increases the probability that domain concepts together with their characteristics will be shared among a range of applications of the same domain (especially on the Web, where most ontologies are currently represented using OWL).

Reusing the content can be realized using commonly accepted model for the content of adaptive applications in particular domain, or mapping the content among adaptive applications. A commonly accepted domain model is ideal solution. Since we agree that building common vocabularies is important and useful (we remark the role of standards), considering a large distributed information space (e.g., the Web) we need to make a compromise between enabling diversity and looking for mappings between various models. The idea of commonly accepted domain ontology is simply impossible to reach in such diverse and distributed environment as the Web.

On the other hand, designing converters for each pair of applications is rather ineffective and demanding approach. A compromise lays in defining an intermediate format of the information content for adaptive applications. In this case it is sufficient to convert the content from the intermediate format to the adaptive application format and vice versa. Standards can also help in this process [5].

One of the first attempts to use information content in several applications was a conversion from the Interbook to AHA! [11]. In this case the Interbook system serves as authoring tool and the AHA! system for adaptive presentation. Another approach described in [9] defines a conversion from the MOT system (My Online Teacher, [7]) to adaptive applications of AHA! and WHRULE systems. MOT is used as an authoring tool, where it is possible to define the content of adaptive application and adaptation strategy that specifies personalization of educational content according changing user characteristics. The conversion from MOT to AHA! uses intermediate format CAF (Common Adaptation Format) that defines a hierarchy of concepts and corresponding attributes of concepts using an XML.

3 Development of reusable content

We use the ontology as a mean for representing and maintaining data related to an application domain. Figure 1 depicts our proposal of domain model schema as it is modeled using the ontology. Considering adaptive web-based applications we define also domain dependent part of the user model that is automatically generated from the domain model.

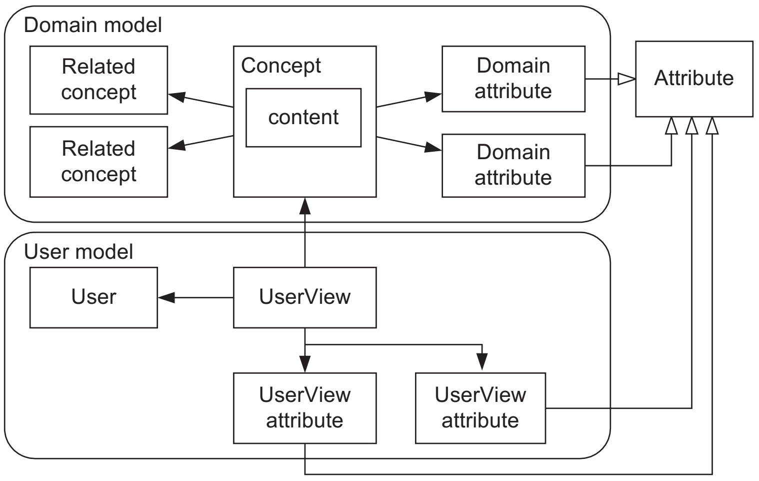

Fig. 1. Schema of adaptive application content model

The domain model defines concepts and their content (for the content of concepts we use term *concept attributes*), domain attributes of concepts and typed *relations* to other concepts. Domain attributes represent information important for adapting but they are not necessarily viewable in the content presentation.

The user model defines for each concept a *User View* and a set of *User View Attributes*. The user view makes a connection among specific user, concept and attributes, which frame together an overlay user model used in most of the adaptive web-based applications and defined in both AHAM and LAOS reference models. Our approach is mainly oriented towards the content modeling. The user model is defined in such a way that can be semiautomatically derived from the domain model. Its actual representation strongly depends on the adaptive application and the means for user characteristics acquisition.

Boxes in Figure 1 represent classes in the ontology (we represent it in OWL/RDF) and every connection (except those deriving the attributes) stands for a relation in the ontology. Specification of a structure of the adaptive application content is done by defining classes and their relations in the ontology. The content of adaptive application itself is represented by instances of classes defined in the ontology with connections specified by relations.

3.1 The method for domain modeling

We proposed a method for modeling the content of adaptive application that uses *core ontology* designed for adaptive application content modeling. It defines steps essential for creating a new domain ontology, which is compliant with the core ontology, describes an export of the ontology into an intermediate format, which is used for importing into existing adaptive applications. We assume also a situation where the domain model already exists and provide a mapping into proposed domain model schema.

The method defines following steps for authoring adaptive application domain model represented by ontology:

Method 1 Modeling reusable content for adaptive applications

```
1. Specify the concept classes and the concept class hierarchy
2. Define attributes for the concept classes
   • data attributes
   • domain attributes
   • user attributes
3. Specify and type relations between concepts of specified classes
4. Create concept instances as the class instances of the ontology
```

Steps 1. to 3. are not as a rule performed in this order and in one iteration. Actually they are often mixed and reordered in several iterations with aim to develop consistent model of the content. Developing domain model is followed by the step of *delivering the ontology content into existing adaptive application.*

Step 1: Specify the concept classes and the concept class hierarchy

In this step we classify information fragments which are supposed to be used in the adaptive application. Information fragments serve as a basis for determining the concepts and concept classes, which are identified for example by means of text analysis, i.e. subjects and nouns used in application context are considered (e.g., exercise, example, explanation in programming learning domain).

Concept classes are organized into a hierarchy based on their common properties. In this step the concept class hierarchy is not final and is often changed when discovered duplicate attributes or relations among the concepts. Entry point to the adaptive application is a concept of special type *root* (base).

Step 2: Define attributes for the concept classes

Concept class defines its information fragments, attributes, which have defined their values in the concept instances. For each attribute we specify a type and a flag indicating whether the attribute has single or multiple values. Data or domain attribute value is assigned to the concept instance while the user attribute value is assigned to the concept and the user instance.

If there is a need in the adaptive application to monitor, for example the level of user knowledge about particular data attribute, good practise would be to transform the data attribute to particular concept class and define the user attribute *knowledge* for that class. In this way we can model a domain using existing standards for meta-data, e.g., IEEE LOM (Learning Object Metadata) or SCORM (Shareable Content Object Reference Model).

Step 3: Specify and type relations between concepts of specified classes

There are various ways of concept interconnections and dependencies with significant impact on the adaptive application behavior. We specify a relation between concept classes by defining a name of the relation, sets of source and target concept classes and type of the relation. In defining generic structure we use several obvious relations (such as those depicted on Figure 2. Rectangles represent concept classes, lines represent named relations. Relation of type *fragment* is drawn without the lines. Relations can be specialized according specific need in particular application domain.

Concept classes relations produce a structure of the content. The parent of all concepts is a single concept of the type *root.* If the content is modeled by more than one concept, at least one *child* relation where the parent is the root concept should be defined.

Step 4: Create concept instances as the class instances of the ontology

Performing steps 1-3 leads to a structure of domain ontology. Step 4 is devoted to filling the information content into the ontology, i.e. information fragments

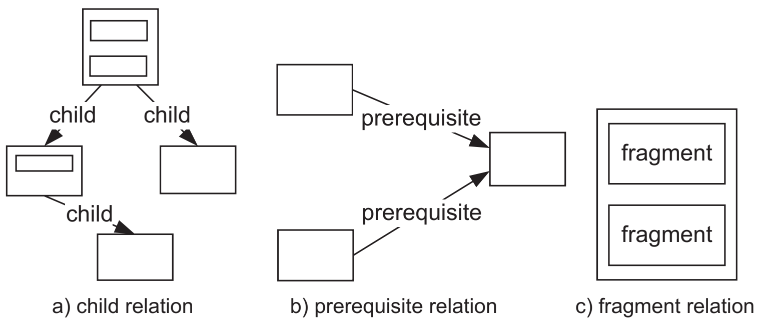

Fig. 2. Examples of types of relationships between concepts

are defined. This is needed for later content exporting into existing adaptive application.

Good practice is first to create an instance of the root concept and thereafter instances of concept classes in relation to the root concept class. Process is similarly performed for other concept classes. Creating a concept instance requires specifying the concept name, filling the values of concept attributes and assigning the other related concept instances.

Our method is not aimed at creating actual content of the adaptive application in the ontology. This can be done by various ways (e.g., manually using an ontology editor, automatically by transformation learning objects knowing core ontology and the source format using an existing OWL API). We concentrate on creating a model of the adaptive application not the application itself, therefore we define only what content should be present in the ontology in order to be able to perform delivering the ontology content into existing adaptive application.

3.2 Importing existing domain model ontologies

Assume that a class in an existing ontology represents a concept class. Then all the data properties of the class represent data attributes and all object properties represent relationships with other concept classes. All the classes in existing ontology including their properties represent the domain model structure of the adaptive application. Instances of all the classes in existing ontology represent the content of domain model of the adaptive application.

This approach helps to use existing sources in form of the ontology as domain models in adaptive applications. All original classes and object properties need to be additionally derived from the core ontology components, which keeps original ontology structure. All instances, data properties and restrictions remain unchanged.

When existing ontology is used as a domain model above described method for reusable domain model development is replaced by the following Method 2.

Method 2 Importing existing ontology domain model

```
1. Transform all classes in the ontology to concept classes
2. Transform all relations in the ontology to relations between concepts
3. Define attributes for the concept classes
   • domain attributes
   • user attributes
4. Specify types of relations between concepts
5. Specify root concept class
```

When transforming ontology classes to concept classes in step 1 all data properties of ontology classes become data attributes of concept classes. Additionally we can define other attributes in step 3.

We assume that the existing ontology includes also classes' instances which are treated as concept instances after performing steps 1 and 2. Consequently we have omitted the step of creating concept instances from the original method. In the case the ontology does not contain instances, we follow the original step 4 after specifying the *root* concept class.

Existing ontology is treated as the pure domain model which is extended by other properties of an adaptive application content by performing steps 3-5. Relation types in step 4 are needed to be refined in order to accurate interpretation of concept relations. The *root* concept class in step 5 is necessary for specifying the entry point of the adaptive application.

3.3 Delivering ontology content into existing adaptive application

Our aim is to reuse the domain model across several adaptive web-based applications. Creating reusable domain model is used for delivering the content into existing adaptive application. The process results in creating the intermediate format for adaptive application content and importing the content into particular adaptive application. The intermediate format contains concept instances, attribute values and concept relationship bindings. Next step is specific for each adaptive application as a transformation should be developed depending on actual domain model representation. In general following steps are needed:

- Create concepts and generate concept content from data and domain attributes.
- Create concept user attributes in user model.
- Link concepts according to concept relationships.

The intermediate format has to be efficient for reuse and delivering to adaptive systems, consistent and general enough to store the content of adaptive application. Our proposal for an intermediate format was inspired by the already mentioned CAF (Common Adaptation Format) used for a conversion from the MOT system do adaptive applications AHA! and WHRULE [9]. We have extended CAF to the CAFE (CAF Extended) format, which simplifies CAF format and introduces possibility of complete and consistent definition of concept attributes and relations.

The CAFE format defines the domain model by listing concepts and their data elements, domain attributes, user model attributes and relationships with other concepts. The relations among concepts may be more complex than only parent-child relation type represented in original CAF format. The lesson model from the CAF format is no longer needed, while the concept hierarchy can be derived from concept relations.

The attributes in CAFE format are explicitly divided into concept (data), domain and user model attribute sets. Each attribute has its value type defined. Each concept in CAFE lists its relationships to other concepts with defined relation type and the set of target concepts (*relationLink*).

An example bellow illustrates main parts of the concept description.

```
<cafe>
<domainmodel>
 <concept id="LispA1418" name="Lesson exercise">
    <conceptattributes>
       <attribute type="string" name="hint">
          <contents type="individual">Use the template ...</contents>
       </attribute>
    </conceptattributes>
    <domainattributes>...</domainattributes>
    <userattributes>...</userattributes>
    <conceptrelations>
       <relation type="linked" name="template">
          <relationlink conceptid="T_PROJECTION" />
       </relation>
    </conceptrelations>
 </concept>
</domainmodel>
</cafe>
```

Our method can be used with other intermediate formats. However, number of various formats obviously leads into explosion of possibilities of transformations that should be specified manually (with little help of software tools developed just for this purpose similarly as wrapper specification tools are being developed).

Export of the content into the intermediate format is performed in a cycle where all concept instances are read and parsed to retrieve values of attributes and relations to other concepts. Importing the content into existing adaptive application is specific for each application based on its domain model representation. One way wrapper for each particular model should be written.

4 Ontology models of the content

4.1 Core ontology of content model

The core ontology is depicted in Figure 3. It defines generic terms of adaptive application content and relations between them. We concentrate on domain modeling and recognize user attributes only on the level of automatic generation of domain dependent part of the user model. Interconnection of both domain and user models is realized using attributes and views.

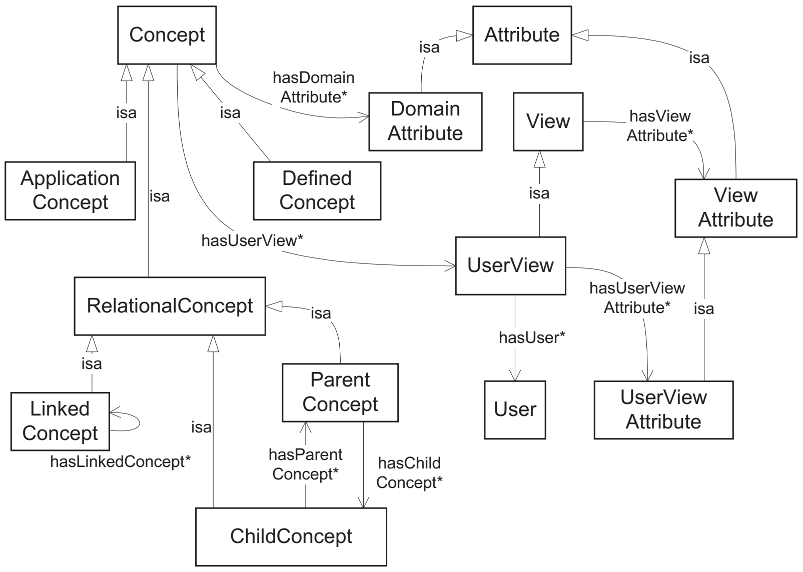

Fig. 3. Core ontology of adaptive application content model

Concepts

The *Concept* class is the base class for all concept classes. It presents abstract information unit (e.g., knowledge in educational application) while concept instance presents concrete information fragment (the content). Concept class defines basic data which include *conceptName*, *conceptDescription* and *conceptText*. We assume bidirectional relations between concepts which are supported and simple to maintain in OWL.

We specify three types of concepts on the highest level (classes derived directly from the *Concept* class):

- *DefinedConcept*: is the base class for all user defined concept classes. By deriving classes from the *DefinedConcept* class we differentiate user defined classes from standard classes from the core ontology.

- *ApplicationConcept*: stands for the root concept class, which is an entry point to the adaptive application. The *ApplicationConcept* class is also derived from the *ParentConcept* class, which defines its relation of type parent-child with some other concepts (application content).
- *RelationalConcept*: is the base class for classes in a relation. It indicates that the concept is in a relation with other concept. Type of the relation is defined by source and target relational concept classes, which are derived from the *RelationalConcept* class.

Relations

Hierarchy of *RelationalConcept* classes is depicted on Figure 4. It demonstrates a technique of defining relation types using the ontology. It is possible to define new types of relations by adding new classes and relations. Each relation is complemented by an inverse relation that is used as reference and is treated automatically. Inverse relations are not depicted in Figure 4.

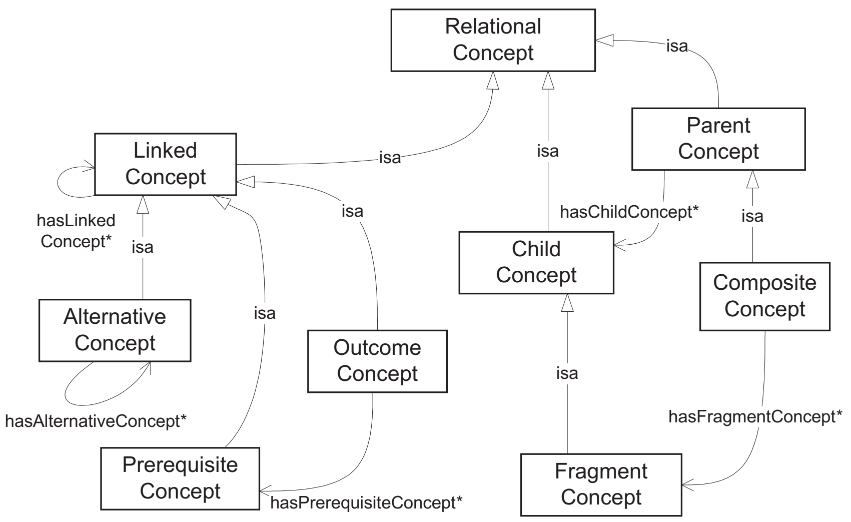

Fig. 4. Hierarchy of relational concept classes

For each relation type in the ontology there is a source and target relation class which are connected by object relation. Figure 5 demonstrates a creation of relationships between the concept classes. When we are creating a relationship between two concept classes we need to name the source and target relation classes and the relation itself. Base relation concept class is the *RelationalConcept* class and the base relation is the *hasRelatedConcept* object relation. They are used to derive more specific relations and relation classes like the *ParentConcept*, *ChildConcept* classes and the *hasChildConcept* object relation on the Figure 5.

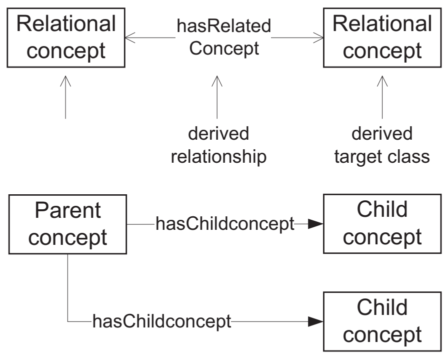

Fig. 5. Creation of relationships between the concept classes

To define a relationship of type parent-child we need to derive the source concept class from the *ParentConcept* class, derive the target concept class from the *ChildConcept* class and create the object relation derived from the *hasChildConcept* relation, which interconnects the source and the target concept classes. For each relationship we consider also an inverse relationship, which is considered only due to completeness of the model and for assisting reasoning in the ontology.

Views

The *View* class represents an abstract view on the concept. *View* defines a set of attributes concerning the adaptive application entity, which can be used for adaptation purposes. For basic modeling we consider one type of view, the view of a user on the concept represented by the *UserView* class.

Views (represented as defined attributes sets) are defined to conform the consistence of modeling. In one of our drafts we considered to insert attributes directly into concept classes. This approach would define only the user meta-model and would not allow to store actual user attribute values. Moreover, the problems with determining attribute sets by ontological restrictions would not be trivial. By defining views we separated the definition of attributes from the content, which simplifies the ontology modeling, the ontology itself becomes usable as a model of adaptive application content for direct use and it is possible to employ actual standards of meta-data representations for particular domain.

User view on the concept implements the overlay user model in the ontology. *UserView* defines the set of attributes for the user model by relational property *hasUserViewAttribute*. The *hasUser* relation connects the view with factual user or with the set of users (in this case stereotype user model). *User view* defines the *ConceptVisited* attribute, which is common for all concepts. It specifies information whether the user has already visited the concept. This attribute is a standard in adaptive web-based applications. More such often used attributes (e.g., showability, suitability, knowledge) according existing adaptive applications can be defined by extending the *UserView* with additional attributes.

Attributes

Attribute is a base class for all attributes except concept data attributes. It presents information processed by adaptive application with purpose to perform the adaptation. Attribute class represents identification and definition of information fragment for particular concept class, user or other entity from a domain model. For example, in the domain of learning programming the attribute can represent the information about solving the programming exercise or understanding exercised programming concepts. Detaching the attributes into separate classes we have enforced an abstraction from the attribute source and processing attribute values at the level of instances.

We mention three attribute types represented in core ontology:

- *DomainAttribute*: differentiates domain attributes from other types of attributes. Typical usage of domain attribute is meta-data related to a concept, which is not displayed in presentation but influences the adaptation process (e.g., difficulty of the programming exercise concept).
- *ViewAttribute*: deriving the attribute class from the *ViewAttribute* specifies the attribute of a view. Deriving is not stright but through additional attribute class specifying the type of the view, e.g., *UserViewAttribute* for the *UserView* view.
- *UserViewAttribute*: specifies that derived attribute class is valid for particular view class derived from the *UserView* class. Default user model attribute valid for all concept classes is the *ConceptVisited* attribute.

Domain attributes are not contained directly in the concepts, but rather assigned to concepts using object relations between the concept classes and the domain attributes classes. The *Concept* class is in relation *hasDomainAttribute* with instances of the *DomainAttribute* class. To define new domain attribute for the concept class we create a new attribute class derived from the *DomainAttribute* class and assign it to the concept class with new object relation derived from *hasDomainAttribute* relation (see Figure 6).

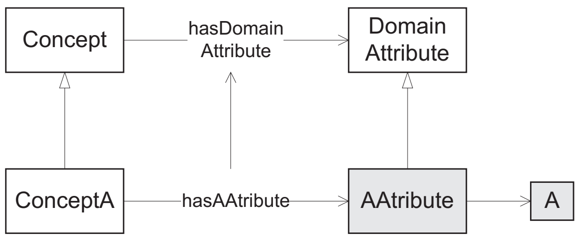

Fig. 6. Definition of a domain attribute

Users

Instance of the *User* class represents a user of adaptive application. The *User* is referenced by multiple user views for multiple concepts. Creating view instances, setting view attribute values and their modification is the role of the adaptive application. Domain model ontology defines default user attribute values for particular concepts. Creating the instances of users is out of the scope of domain modeling. It is in common provided manually (filling forms by the user) or (semi)automatically employing knowledge discovery techniques [18].

4.2 Domain and user models interconnection

Domain model defines concepts including their information content, relations between concepts and domain attributes for the concepts. User model consists of user views for each concept class from the domain model. The user view defines a set of attributes for domain concepts which are stored in the user model. We do not consider the domain independent user model while we concentrate on domain modeling. We model the user only to such extent that can be automatically gathered from developed domain model. Domain independent part of the user model is often defined in separated ontology or it can be accessed from shared source [2].

Interconnection between a domain model and a user model is depicted on Figure 7. The *Concept*, *UserView*, *User* and *ConceptVisited* classes demonstrate the basic interconnection. The *DefinedConcept* does not define any attributes for the user model. As an example we consider e-learning domain where the *EducationalConcept* is specified. It defines two user model attributes *EducationalKnowledge* and *EducationalInterest* for which we have derived new view class *EducationalUserView*, which defines relations for connecting these attributes.

5 Meta-model of the adaptive application content

In ontology it is possible to define data and object properties as functional or non-functional (multiple). Non-functional properties are represented and handled as sets with no regard on ordering the elements contained. This can result into unsuitable attributes order after delivering the content ontology into adaptive application where sequences of concepts or information fragments present important knowledge on the presentation (beside dynamically generated sequences). Explicit specification of elements ordering in the ontology would uselessly embarrass the domain model. We have proposed a meta-model, which contains a list of all concept classes and corresponding attributes and relationships. Each attribute and relationship in the meta-model has defined an order index and a flag of its visibility in the presentation.

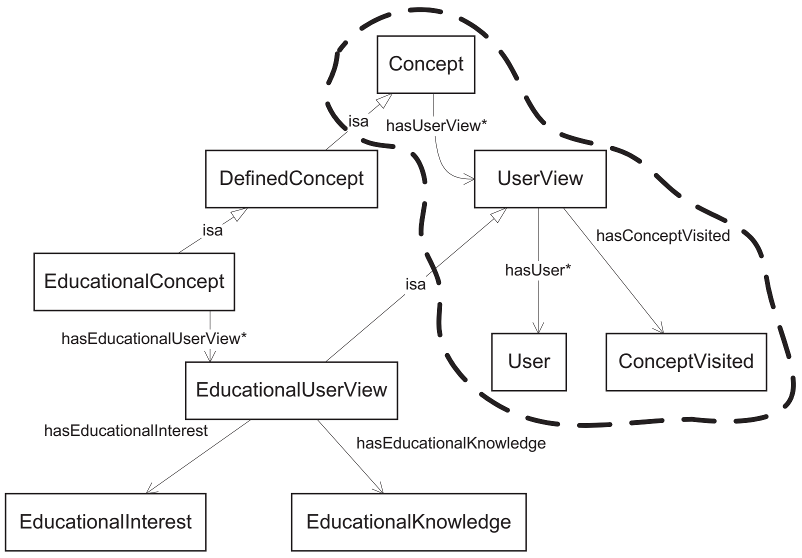

Fig. 7. Interconnection between domain and user models

After generating the meta-model from the ontology the ordering of attributes and relationships conforms their physical ordering in the ontology which can be random in general. Our method for delivering content into existing adaptive application uses the meta-model in the process of exporting the adaptive application context from the ontology into the intermediate format where the attributes and relations are stated in such order as will be presented after importing into the adaptive application.

Bellow is an example of meta-model for sequencing information fragments for a programming exercise in e-learning domain.

```
<meta_caf>
   <meta_concept name="ExerciseConcept">
      <meta_concept_attributes>
         <meta_attribute type="concept" show="true" order="1"
                           name="exerciseHint"/>
         <meta_attribute type="domain" show="false" order="2"
                           name="exerciseDifficulty"/>
      </meta_concept_attributes>
      <meta_concept_relations>
         <meta_relation type="fragment" show="true" order="1"
                          name="hasExerciseDefinition"/>
      </meta_concept_relations>
  </meta_concept> ... </meta_caf>
```

6 Evaluation of proposed method

In our experiments we focused on adaptive web-based system AHA! [10] and authoring tool for adaptive applications MOT [9]. The AHA! system is partially compliant with the AHAM model and the MOT system is based on the LAOS model. Both provide authoring tools for creating adaptive applications. Level of authoring is superior in MOT, which provides simple and powerful definition of domain concept maps, lesson model (defined in the LAOS model) and adaptation strategies (LAG programs, [8]). On the other side, AHA! provides superior adaptive techniques for defining an adaptive presentation. This resulted to a proposal to use MOT as an authoring system and AHA! as a delivery system supported by transforming the content between these two systems (MOT2AHA, [9]). Transformation is either straightforward, or consists of two steps where the intermediate format (CAF, Common Adaptation Format) extended to CAFE (Common Adaptation Format Extended) is used.

We have used the core ontology for authoring described in this chapter together with existing converting tools for delivering the content into the AHA! application (CAF2AHA! tool developed at Eindhoven University).

Evaluation was realized in three stages:

1. Definition of prototype ontology in domain of learning programming by examples.
2. Export the prototype ontology into the adaptive application.
3. Developing software support for authoring the adaptive application content ontology.

In the first stage we developed the ontology describing a programming course using program exercises for languages Lisp and Prolog (based on the adaptive web-based system ALEA used for learning programming in Functional and Logic programming course at the Slovak University of Technology in Bratislava [4]). At the same time we provided manual transformation of a part of the ALEA content into the AHA! application [3] in order to verify developed ontology.

Structure of the domain model is shown in Figure 8. For simplicity we omit here inverse relations between the concepts. *ProgrammingExercises* represents the root concept of the application, which can include a set of program schemata (*TemplateConcept*) and a set of exercises (*ExerciseConcept*). Program schemata include concepts describing the program schema usage (*TemplateUsage*) and concepts containing exercises (*ExerciseConcept*). Each exercise contains one or more concepts defining the exercise (*ExerciseDefinition*) and its solution (*ExerciseSolution*). The *hasSubTemplate* relation enables to build a hierarchy of the template concept instances.

After defining a structure of ontology for learning programming domain, we filled the ontology up with instances of programming templates and exercises to enable further evaluation.

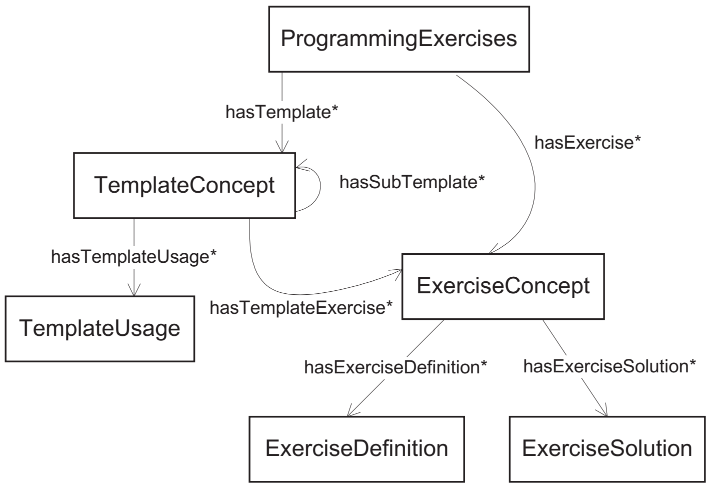

Fig. 8. Programming course domain model

During the second phase in order to solve a problem of unwanted misplacing concept attributes in generated presentation we developed an ontology meta-model, which defines concept classes and their attributes and relations (see Section 5). By editing the meta-model and applying it during exporting to CAF format concept attributes and relations are resorted as specified in the meta-model.

Second and third stages of the evaluation are supported by software tools developed in order to make proposed sequence of steps defined by the method for modeling the content of adaptive application practicable. We developed tools for editing, importing and exporting the ontology defining the content of adaptive application. Figure 9 visualizes a process of transforming content represented by the domain model in ontology to the AHA! system. Moreover we added importing the content from the ALEA system, which was realized by a set of developed software tools for transformation XML representation of the content in ALEA into ontological representation.

7 Conclusions

In this chapter we described a method for adaptive application content modeling using ontology that allows the content reuse between applications. It is based on designed core ontology that is open and can be used for integrating other aspects of adaptive behavior and other layers of reference models. Adaptive applications benefit also from generated domain dependent part of user model.

Our long term goal is a use of ontology as knowledge representation in adaptive web-based applications. It allows building both closed and open cor-

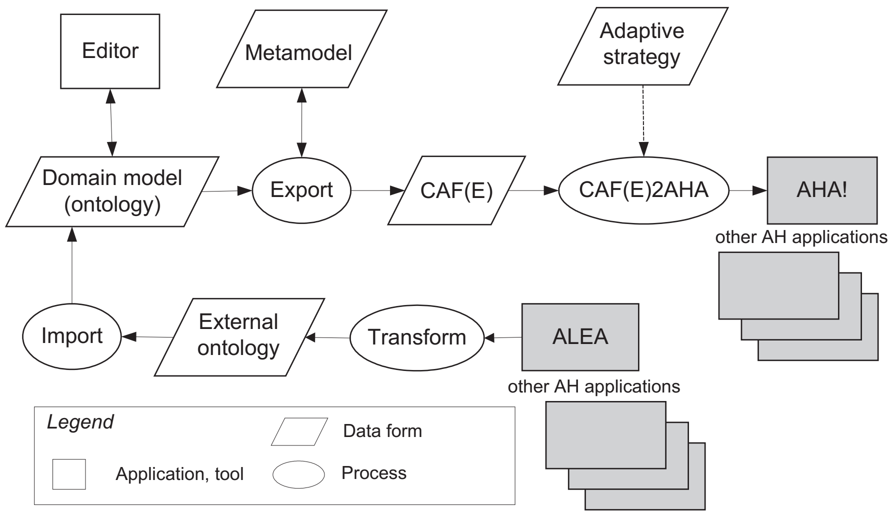

Fig. 9. Content transformation possibilities

pus systems using the same processes. Knowledge representation using the ontology reveals possibilities of reasoning and thus supports defining adaptive behavior. Moreover, automatic means for sequences of content generation (e.g., using knowledge on user behavior [19]) or checking its consistency could be employed.

Our work was aimed at delivering adaptive application content into existing adaptive systems or content conversion between content oriented web-based applications. We have demonstrated authoring of adaptive application content using the ontology and its delivering into existing system in terms of "authoring once, delivering many", which is promising real utilization of value-added possibilities for modeling adaptive applications with means of the Semantic Web. Described approach of modeling adaptive applications content employing the ontology is a step to support the Semantic Web technologies in adaptive web-based systems.

Acknowledgements. This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant VG1/3102/06 and by the Cultural and Educational Grant Agency of the Slovak Republic, grant No. KEGA 3/5187/07.

The authors would like to thank members of PeWe group (`www.fiit.stuba.sk/research/pewe`) and participants of the SMAP 2006 workshop (`www.smap2006.org`) for fruitful discussions and feedback on work presented in this chapter.

References

1. Andrejko A, Barla M, Bieliková M, Tvarožek M (2006) Ontology-based user modeling for web-based information systems. In: Proc. of Int. Conf. on Information Systems Development, ISD'06, Budapest, Hungaria
2. Bieliková M, Kuruc J (2005) Sharing user models for adaptive hypermedia applications. In: Proc. of 5th Int. Conf. on Intelligent Systems Design and Applications, ISDA'05, Wroclaw, Poland, ACM Press, 506–511
3. Bieliková M, Kuruc J, Andrejko A (2005) Learning programming with adaptive web-based hypermedia system AHA! In: Jakab F et al. (eds) Proc. of Int. Conf. on Emerging e-Learning Technologies and Applications, ICETA'05, Koice, Slovakia, 251–256
4. Bieliková M (2006) An adaptive web-based system for learning programming. Int. J. Continuing Engineering Education and Life-Long Learning, Inderscience, 16(1/2):122–136
5. Bureš M, Jelínek I (2005) Reusable Adaptive Hypermedia E-learning Content Using AICC. In: Proc. of the IADIS Int. Conf. WWW/Internet'05, Lisboa, Spain, IADIS Press, vol. I, 376–378
6. Ceri S, Fraternali P, Matera M (2002) Conceptual modeling of data-intensive web applications. In: IEEE Internet Computing, 6(4):20–30
7. Cristea A I, De Mooij A (2003) LAOS: Layered WWW AHS authoring model and their corresponding algebraic operators. In: Proc. of 12th Int. World Wide Web Conf., WWW'03, Budapest, Hungary, ACM Press
8. Cristea AI, Verschoor M (2004) The LAG grammar for authoring adaptive web. In: Proc. of Int. Conf. on Information Technology: Coding and Computing, ITCC'04, IEEE Computer Society Press, 382–386
9. Cristea AI, Smits D, De Bra P (2005) Writing MOT, reading AHA! – converting between an authoring and a delivery system for adaptive educational hypermedia. In: Proc. of 3rd Int. Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia, A3EH'05 at AIED'05, Amsterdam, The Netherlands
10. De Bra P et al. (2003) AHA! - The adaptive hypermedia architecture. In: Proc. of the ACM Hypertext Conf., Nottingham, UK, 81–84
11. De Bra P, Santic T, Brusilovsky P (2003) AHA! meets Interbook, and more. In: Proc. of the AACE ELearn'03 Conf., Phoenix, Arizona, 57–64
12. Gruber TR (1993) Towards principles for the design of ontologies used for knowledge sharing. In: Guarino N, Poli R (eds) Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer
13. Houben GJ, Barna P, Frasincar F (2003) HERA: development of semantic web information systems. In: Proc. of Int. Conf. on Web Engineering, ICWE'03, LNCS 2722, Springer, 529–538
14. Koch N, Wirsing M (2002) The Munich reference model for adaptive hypermedia applications. In: De Bra P, Brusilovsky P, Conejo R (eds) Proc. of Int. Conf. on Adaptive Hypermedia and Adaptive Web-based Systems, AH'02, LNCS 2347, Springer, 213–222
15. Seefelder PA, Schwabe D (2004) A semantic meta-model for adaptive hypermedia systems. In: Proc. of 3rd Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems, AH'04, Eindhoven, The Netherlands, LNCS 3137, Springer, 360-365

16. Stewart C, Cristea AI, Brailsford T (2005) Authoring once, delivering many: creating reusable adaptive courseware. In: Proc. of 4th IASTED Int. Conf. on Web-Based Education, WBE'05, Grindelwald, Switzerland
17. Schwabe D, Rossi G: An object-oriented approach to web-based application design. In: Theory and Practice of Object Systems, Special issue on the Internet, 4(4):207–225
18. Tvarožek M, Barla M, Bieliková M (2007) Personalized Presentation in Web-Based Information Systems. In: Proc. of SOFSEM 2007, J. van Leeuwen et al. (ed), Springer, LNCS 4362, 796–807
19. Velart Z, Šaloun P (2006) User behavior patterns in the course of programming in C++. In: Proc. of the Int. Workshop on Adaptivity, personalization & the Semantic Web, Odense, Denmark, 41–44
20. Wu H, Houben GJ, De Bra P (1998) AHAM: A reference model to support adaptive hypermedia authoring. In: Proc. of the Conf. on Information Science, Antwerp, 51–76