INT J COMPUT COMMUN, ISSN 1841-9836
Vol.7 (2012), No. 2 (June), pp. 285-301

Authoring Adaptive Hypermedia using Ontologies

H. Jung, S. Park

Hyosook Jung, Seongbin Park
Korea University,
Anam-dong, Seongbuk-gu, Seoul, Korea
{est0718, hyperspace}@korea.ac.kr

Abstract: Adaptive hypermedia has been developed to overcome the problems of disorientation by providing personalized presentation and link structure. An adaptive hypermedia system consists of an adaptation model, a domain model, and a user model. The user model describes various aspects of a user such as interests, knowledge, preferences, etc. The domain model describes the whole knowledge accessible in adaptive hypermedia. The adaptation model consists of adaptation rules that define both how to generate the personalized presentation and update the user model [12]. Authoring adaptive hypermedia typically starts by designing a domain model so that apprropiate adaptation model and user model can be created based the domain model. While there are authoring tools developed for creating domain model of an adaptive hypermedia, authors need to manually create basic concepts as well as their relationships for a domain of interests. In this paper, we present a system that transforms an ontology into the domain and adaptation model of adaptive hypermedia so that the authors can generate adaptive hypermedia easily. The system transforms classes and relationships between the classes defined in OWL ontology [7] into concepts and relationships between the concepts defined in the domain model of AHA! system which is one of best known open source general-purpose adaptive hypermedia systems [1]. Using our system, authors can utilize well-defined knowledge structure in ontologies for authoring adaptive hypermedia. On top of this, since designing domain model is generally an initial step to author adaptive hypermedia, our system can help authors reduce tasks to create adaptive hypermedia by automatically generating domain model from an ontology.
Keywords: Ontology, Adaptive hypermedia.

1 Introduction

A hypermedia system allows users to freely navigate through a large hyperspace. However, the navigational freedom makes the users experience the problems of disorientation and cognitive overload [11]. Adaptive hypermedia has been developed to overcome these problems by providing personalized content and navigational support based on user model, domain model and adaptation model. The user model describes various aspects of a user such as interests, knowledge, preferences, etc. The domain model describes the whole knowledge accessible in adaptive hypermedia. The adaptation model consists of adaptation rules that define both how to generate the personalized presentation and update the user model [12]. Authoring adaptive hypermedia is a difficult and complex task that involves designing different levels of abstraction and multiple links, defining adaptation on an abstract conceptual level, etc. On top of this, the authored materials are hardly reused because there is no standardized approach to adaptive techniques and behavior [8–10].

In this paper, we propose a system that helps authoring adaptive hypermedia using ontologies. Domain model describes concepts related to a domain and their relationships. An ontology also represents the knowledge of a domain by a set of concepts within the domain and the relationships between the concepts [6]. Based on this similarity, our system converts an ontology into a domain model of an adaptive hypermedia system so that authors can easily create domain models using ontologies. Our system has been implemented using AHA! which is an open source general-purpose adaptive hypermedia [1]. The reason that we selected AHA! system is that it is one of the best open source adaptive hypermedia systems which can be easily used in educational environments. Moreover the web site about the system containts a lot of helpful documents that explain various aspects as well as applications of the system [26].

Figure 1 shows the whole structure of the proposed system. Once an OWL document is read, Parser extracts concepts and their relationships by using Protégé-OWL API [17]. It looks for elements of the OWL vocabulary corresponding to the concepts and their relationships of the domain model. For example, <owl:Class> is a concept and <rdfs:subClassOf> is a concept hierarchy. <owl:ObjectProperty> is a prerequisite relationship between two concepts. Its domain class defined in <rdfs:domain> is a prerequisite for its range class defined in <rdfs:range>. Then, Converter creates a domain model by transforming the concepts and their relationships to the format for AHA! Graph Author that is one of authoring tools used in AHA! system.

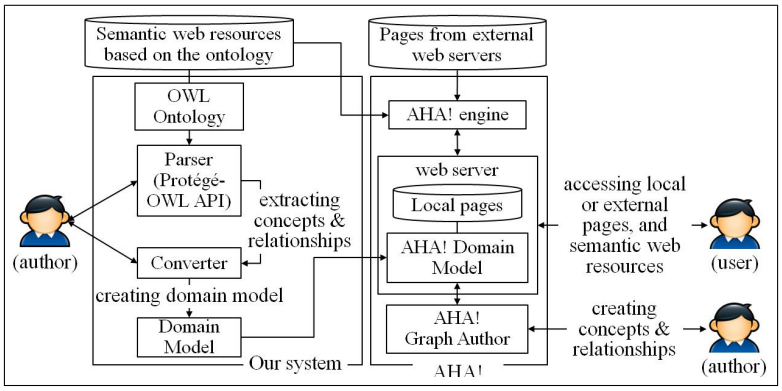

Figure 1: The structure of our system

We have conducted a pilot evaluation for our system with users who had experiences using AHA! system, and their comments were positive in that our system did help easily construct domain models as well as adaptation models using ontologies that could be found on the Web. Since the number of ontologies that are available online is ever increasing [28], it is possible to use well-designed ontologgies for domain model construction even if authors are not experts in the domain of interests.

This paper is structured as follows. Section 2 describes related works. Section 3 describes the structure of the system as well as the transformation steps from an ontology into the domain model and the adaptation model in detail. Section 4 describes illustrative examples that show how the proposed system works. Applications of the system and pilot evaluation about the system are explained in section 5. Section 6 concludes the paper and describes the future works.

2 Related Works

The Semantic Web is an environment where Web contents are represented in a form that is machine processable [14]. There are several languages to represent machine interpretable content on the Web. XML offers a surface syntax for structured documents and XML Schema is a lan-

guage for restricting the structure of XML documents. RDF is a data model for objects and their relations and supports a simple semantics for the data model. RDF Schema is a vocabulary for representing properties and classes of RDF resources. OWL adds more vocabulary for describing properties and classes: among others, relations between classes, cardinality, equality, richer typing of properties, characteristics of properties, and enumerated classes [7]. An ontology is a specification of an abstract data model that is independent of its particular form [6]. Ontologies are used by people, databases, and applications in order to share domain information. They contain computer-usable definitions of basic concepts in the domain and the relationships among them [15]. OWL Web Ontology Language is one of languages to represent machine interpretable content on the Web. An OWL ontology consists of a set of constraints on sets of classes and types of relationships between them. A class contains individuals which are instances of the class, and other subclasses. A property specifies class characteristics. It can be either a datatype or object property. Instances are individuals that belong to the classes defined. OWL supports various operations on classes such as union, intersection and complement [7].

Adaptive hypermedia systems such as AHA! [1], Mag [25], AHyCo [27], etc. have been developed to overcome the problem of disorientation by providing appropriate contents for users with different backgrounds and interests based on the user model, adaptation model, and domain model.

In AHA!, an author needs to design the overall conceptual structure by using AHA! authoring tools. They allow the author to define the domain model for an application along with the adaption model related to it, which means that the author defines requirements for each concept and a set of generate rules that represent connections between concepts [2]. In AHA!, most authors do not have to know about the adaptation model because the rules are generated automatically by Graph Author. When a concept is created in Graph Author, a set of attributes and adaptation rules is generated. It has templates for different types of concept relationships [3].

Mag [25] is a tutoring system that supports learning programming languages in various courses. It provides adaptive learning and personalization of a content delivery based on learner model. It consists of the adaptation model, learner model, application model and domain model. It uses ontologies to present a domain, building learner model and presenting activities in the system in order to achieve knowledge sharing and reuse, learner modeling and extension of a system. The domain model presents storage for all concepts, tutorials and tests. It describes how the information content is structured. Instructors can create the domain model based on domain ontology by using an authoring tool.

AHyCo (Adaptive Hypermedia Courseware) [27] is an adaptive educational hypermedia system to create and reuse adaptive courseware. It focuses on adaptive navigation support and lessons sequencing. It consists of the domain model, the student model, and the adaptation model. It allows authors to design the courseware such as creating concepts, linking concepts by prerequisite relationships, and generating test questions. It offers a graphic editor for concept networks that enables the authors to define the prerequisite relationships with a drag-and drop interface like AHA! Graph Author. Authoring a course contains the development of both the network of lessons and the tests that represents the domain model of the course. For constructing a course, the authors create a set of hypermedia fragments that represent the content of a lesson. A set of lessons are connected by prerequisite relationships and grouped into a module. A course is a set of modules connected by prerequisite relationships. The authors define the prerequisite graphs of concepts and modules and determine the difficulty of lessons and tests. They do not have to consider any other adaptation rules because AHyCo automatically generates the adaptation rules based on the definition of prerequisites and the assignment of the difficulty level for lessons and tests. All information about the domain model is stored in a database.

MOT [5] is an adaptive hypermedia authoring system. It implements domain maps, goal and

constraint maps, user and presentation maps, and adaptation maps. The domain maps structure and organize the resources of the learning environment. They consist of hierarchical domain concept maps. The goal and constraints maps contain all resources and links between them. They add all the necessary pedagogic material and linking for students. The user maps contain all necessary variables and initial values to represent the user. The presentation maps make different presentation according to the physical properties and environment. The adaptation maps describe the dynamic of the adaptation process based on the LAG model [4] which is a three-layer model and classification method for adaptive techniques; direct adaptation rules, adaptation language and adaptation strategies. MOT makes authors create a new concept map of a lesson that consists of concepts and their attributes. There are many tasks to create the concept map such as adding each concept, naming them, selecting their attributes, etc. However, our system helps the authors create the concept map without those manual tasks because it provides a hierarchy of basic concepts with their attributes by converting the ontology related to the lesson. The authors just edit or change the created concept map.

The interoperability between different Adaptive Educational Hypermedia (AEH) systems has been investigated by using conversion systems. Interbook to AHA! compiler translates the source format for Interbook to AHA! with Layout model which presents concepts (pages) to authors [20]. MOT to AHA! conversion engine translates from MOT used as a an authoring system to AHA! used as a delivery system for AES. It uses the common adaptation format (CAF) domain and lesson map descriptions and the adaptive strategies written in LAG to create adaptive presentations in AHA! [21]. MOT to WHURLE converter translates MOT to WHURLE used as a delivery system for reusing the authored content on a different system [22]. However, these are limited to conversions between different AEH systems. The conversion should be based on popular standards for reducing re-designing material each time one AEH system moves to another and encouraging the use of AEH systems. [23] presents a solution for AEH interfacing via web services. It uses WSDL for conversion of data between MOT and WHURLE. [24] describes the integration of the generic AH authoring environment, MOT, into a semantic desktop environment.

Our system enables users to access Semantic Web data represented in RDF or OWL format. It stores the URIs of the classes of the ontology as the resource path of the corresponding concept when converting the ontology. The users can access the information about the classes via their URIs. Our system focuses on creating adaptive hypermedia by using an OWL ontology which is one of the core techniques to build the semantic web.

Figure 2 shows related research areas to our work.

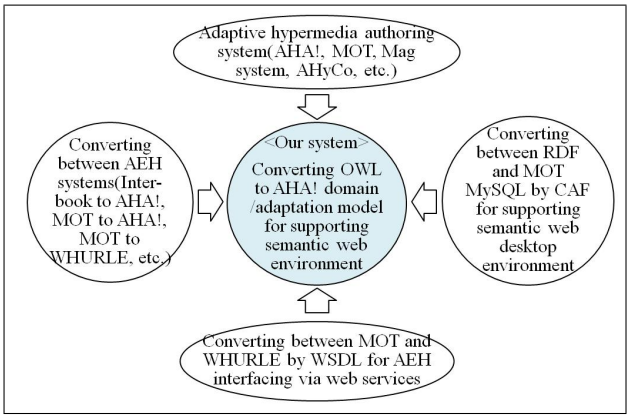

Figure 2: Related works to our research

3 The Structure of the System

In this section, we describe the structure of the proposed system.

Figure 3 shows how OWL ontology are mapped into AHA! domain / adaptation model.

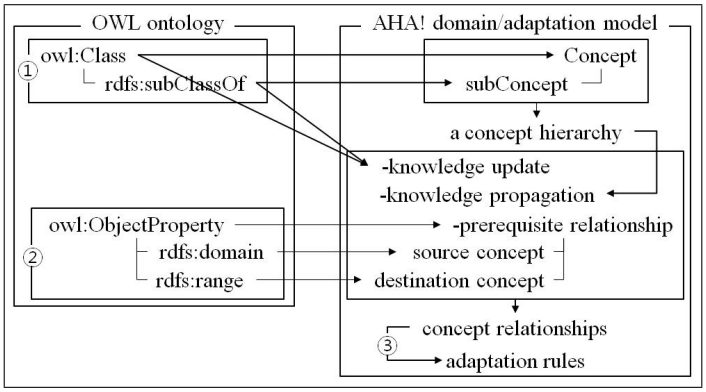

Figure 3: Conversion from an OWL ontology to AHA! domain model and adaptation model

First, our system extracts all named classes in an OWL ontology and transforms the classes into concepts of AHA! domain model in order to generate concepts. A class is transformed into a concept. If the class has subclasses, the system transforms the subclasses into subconcepts of the corresponding concept. It generates a concept hierarchy by using the association between the concepts and their subconcepts. It also defines the information of a concept that is its name and optional description. The system assigns the name of a class to the name of a concept. The URI (Uniform Resource Identifier) of the class is converted to the resource of the concept. It describes a class with subclasses as an abstract concept and the subclasses as page concepts. In AHA! domain model, an abstract concept does not have a resource and a page concept must have resources. However, the optional description can be changed by AHA! Graph Author. It saves the concepts hierarchy and the concept information in AHA! domain model. The Graph Author allows authors to edit concepts and concept relationships of the domain model. The Graph Author also saves the domain model in another authoring format used by Concept Editor (.aha) as translating concept relationships into adaptation rules. Users can edit concepts, attributes, and adaptation rules using Concept Editor which can control the functionality of AHA! such as concepts, attributes and adaptation rules.

Second, our system extracts all OjectProperties in the OWL dontology and their domain and range classes in order to generate concept relationships. It converts an ObjectProperty into a prerequisite relationship of AHA! domain model. In AHA!, when concept A is a prerequisite of concept B, users should read about concept A before continuing with concept B. The concept A is a source concept and the concept B is a destination concept. The system transforms a domain class of a ObjectProperty into a source concept of the prerequisite relationship. It does a range class of the ObjectProperty into a destination concept of the prerequisite relationship. (i.e., the domain class is converted into a prerequisite for the range class.) In addition, the system assigns additional concept relationships to each concept. Each concept with a page has a concept relationship called knowledge update. When a page of a concept is read, its knowledge is updated. The concept hierarchy is used of knowledge propagation. When the knowledge of a concept is changed, the change is propagated to the concepts that are higher than the concept in the concept hierarchy. It saves the concepts relationships in AHA! domain model.

Third, all concept relationships in AHA! are translated into adaptation rules in adaptation model by AHA! Graph Author. The rules are executed conditionally when a page related to a

concept is accessed.

Figure 4 shows how our system fits into AHA! system.

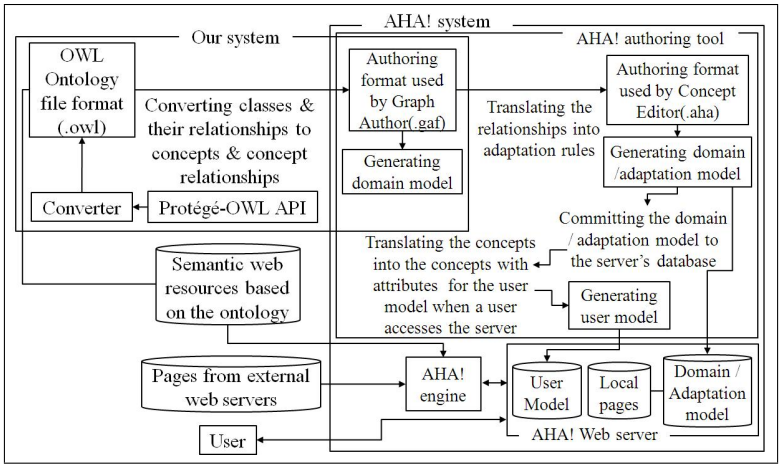

Figure 4: Authoring adaptive hypermedia using our system in connection with AHA! system

When the authors commit the application to the AHA! server's database such as XML or MySQL representation, the AHA! domain and adaptation model are automatically created. If end-users access the application, their user models are created as well. When users access AHA!, it creates their own profiles based on the user model. If a user requests a page, the request triggers the adaptation rules that update the user's profile. The requested page is presented based on the updated profile of the user. The links on the page are also provided based on a ĄŽsuitabilityĄŻ requirement that is part of the domain and adaptation model.

4 Illustrative Examples

In this section, we show two simple examples that illustrate how our system converts OWL ontologies into the domain models and adaptation models of AHA! system. In addition, we show how the user models of AHA! can be created.

4.1 Example 1

Figure 5 shows the structure of an OWL ontology (sweb.owl) about Semantic web that defines 11 classes and 4 object properties. Class Basic has 4 subclasses such as Metadata, Agents, Ontologies, and Logic. Class Technologies has 5 subclasses such as XML, RDF, RDFS, OWL and Rules. ObjectProperty Metadata_is_describedBy has domain class Metadata and range class XML and RDF. ObjectProperty Ontology_is_describedBy has domain class Ontologies and range class XML, RDF, RDFS, and OWL. ObjectProperty Logic_is_describedBy has domain class Logic and range class Rules.

Figure 6 shows the structure of the domain model of AHA! system that is generated from the ontology (sweb.owl) in figure 5 using our system. The name of the OWL ontology (i.e., "sweb" in sweb.owl) becomes the name of an AHA! application or course. Our system extracts all named classes and transforms them into concepts of the domain model such as sweb.Basic, sweb.Technologies, etc. It also generates a concept hierarchy based on the hierarchy between the classes. For example, concept sweb.Basic has subconcepts such as sweb.Metadata, sweb.Agents, sweb.Ontologies, and sweb.Logic. Concept sweb.Technologies also has subconcoept such as sweb.XML, sweb.RDF, sweb.RDFS, sweb.OWL, and sweb.Rules. The system also transforms

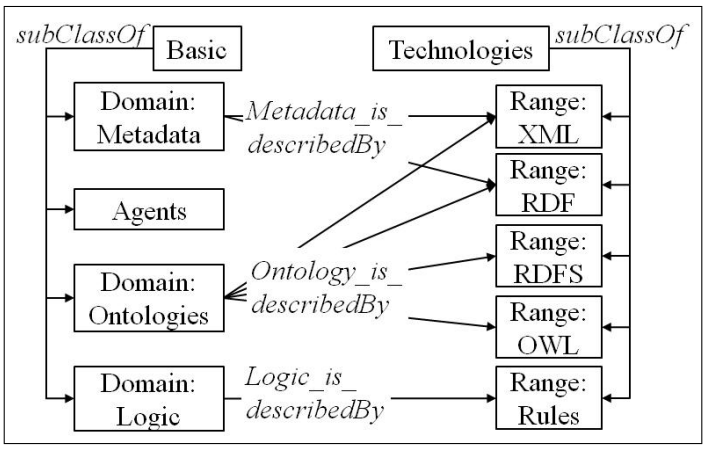

Figure 5: The structure of an OWL ontology about Semantic Web

ObjectPoperties into prerequisite relationships of the domain model. When concept A is a prerequisite of concept B, the concept A is a source concept and the concept B is a destination concept. The system transforms a domain class into a source concept and a range class into a destination concept. For example, ObjectProperty Logic_is_describedBy is transformed into a prerequisite relationship. The domain class Logic of Logic_is_describedBy becomes a source concept and its range class Rules becomes a destination concept.

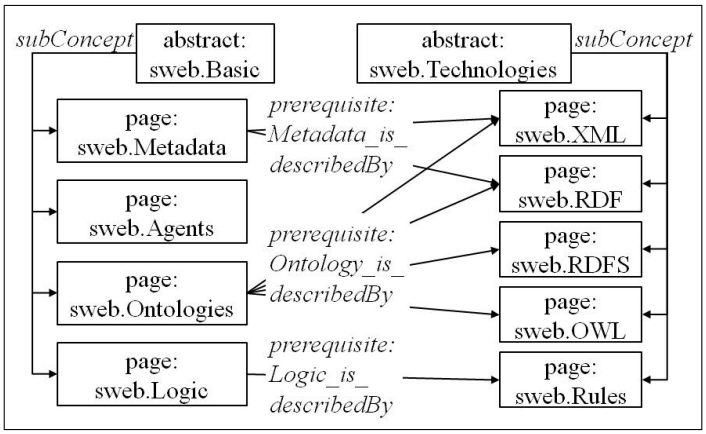

Figure 6: The structure of AHA! domain model transformed from the ontology in figure 5

After the OWL ontology is converted into a domain model, the domain model is saved as a file in an authoring format edited by the Graph Author (sweb.gaf). When an author commits the file to the serverĄŻs database, all concept relationships defined in the domain model are automatically translated into adaptation rules. The adaptation rules and the domain model are saved in a file which is an authoring format edited by the Concept Editor (sweb.aha).

Adaptation rules that are related to concept “sweb.Rules” in “sweb” application which are automatically generated are as follows.

```
<generateListItem isPropagating="true" >
      <requirement>! sweb.Rules.suitability &amp;&amp;
                sweb.Rules.knowledge &lt; 10</requirement>
      <trueActions>
       <action>
```

```
        <conceptName>sweb.Rules</conceptName>
        <attributeName>knowledge</attributeName>
        <expression>35</expression>
       </action>
      </trueActions>
    </generateListItem>
```

The above rule is connected to concept relationship "knowledge update". When the rule is triggered, its condition called "requirement" is checked. It is executed when sweb.Rules.suitability is false and sweb.Rules.knowledge is lower than 10. The action will assign the value 35 to the knowledge of sweb.Rules.

```
<generateListItem isPropagating="true" >
      <requirement>true</requirement>
      <trueActions>
       <action>
        <conceptName>sweb.Technologies</conceptName>
        <attributeName>knowledge</attributeName>
        <expression>sweb.Technologies.knowledge +
               (0.2 * _sweb.Rules.knowledge)</expression>
       </action>
      </trueActions>
    </generateListItem>
```

The above rule is connected to concept relationship "knowledge propagation". When the knowledge of a concept is changed, the change is propagated to the concepts that are higher in the concept hierarchy. Concept sweb.Rules is a child of concept sweb.Technologies in "sweb" application. The action will add the knowledge of sweb.Technologies to 20% of the knowledge of sweb.Rules and assign the value to the knowledge of sweb.Technologies.

```
<attribute name="suitability" type="bool" isPersistent="false"
        isSystem="false" isChangeable="false">
   <description>the suitability of this page</description>
   <default>((sweb.Logic.knowledge &gt; 0))</default>
</attribute>
```

The above rule is connected to concept relationship "prerequisite relationships". sweb.Logic is a prerequisite for sweb.Rules. The suitability of sweb.Rules depends on the knowledge of sweb.Logic. The rule requires the knowledge of sweb.Logic to be higher than 0 in order to access to the page of sweb.Rules.

When a user logs in the AHA! server, the user model of the user is automatically generated. The user model consists of a set of concepts with attributes. It contains an overlay model which means that there is a concept in the user model for every concept in the domain model. After the domain and adaptation model are constructed, the following attributes related to concept sweb.Rules are added to the existing user model when a user accesses the AHA! server.

```
<record>
    <key> sweb.Rules.access</key>
    <type>3</type>
    <persistent>false</persistent>
```

```
    <value>false</value>
  </record>
  <record>
    <key> sweb.Rules.interest</key>
    <type>1</type>
    <persistent>true</persistent>
    <value>0</value>
  </record>
  <record>
    <key> sweb.Rules.suitability</key>
    <type>3</type>
    <persistent>false</persistent>
    <value>false</value>
  </record>
  <record>
    <key> sweb.Rules.visited</key>
    <type>1</type>
    <persistent>true</persistent>
    <value>0</value>
  </record>
  <record>
    <key> sweb.Rules.knowledge</key>
    <type>1</type>
    <persistent>true</persistent>
    <value>0</value>
  </record>
```

4.2 Example 2

Figure 7 shows another the correspondence between an OWL ontology (health.owl) and AHA! domain model. Our system transforms all named classes such as Health, Disease, and Food into concepts of the domain model such as health.Health, health.Disease, and health.Food. It also generates a concept hierarchy based on the hierarchy between the classes. For example, class Health has subclass Disease and Food. Concept health.Health has subconcept health.Disease and health.Food. The system also transforms ObjectPoperties into prerequisite relationships of the domain model. For example, ObjectProperty causedBy is transformed into a prerequisite relationship. Domain class Disease of causeBy becomes a source concept and its range class becomes a destination concept.

After the OWL ontology is converted into a domain model, the domain model is saved as a file in an authoring format edited by the Graph Author (health.gaf). When authors commit the file to the serverĄŻs database, all concept relationships are automatically translated into adaptation rules. A file including the domain model and adaptation rules are saved in an authoring format edited by the Concept Editor (health.aha).

The adaptation rules that are related to concept health.Food in “health” application which are automatically generated are as follows.

```
<generateListItem isPropagating="true" >
   <requirement>! health.Food.suitability &amp;&amp;
         health.Food.knowledge &lt; 10</requirement>
   <trueActions>
```

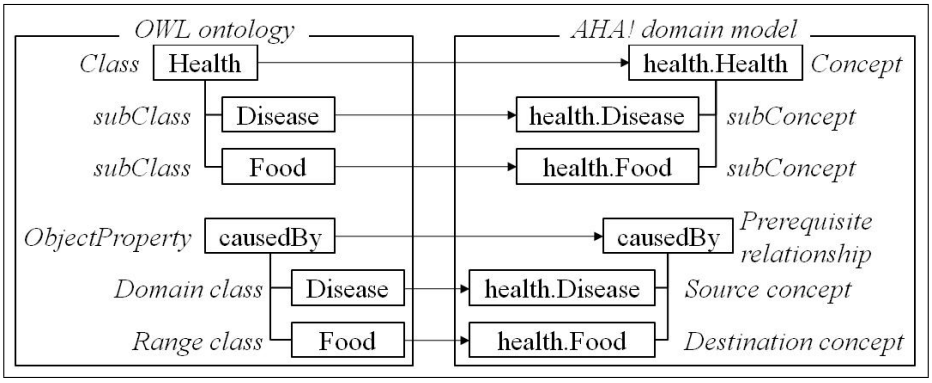

Figure 7: Correspondence between an OWL ontlogy and AHA! domain model

```
      <action>
        <conceptName>health.Food</conceptName>
        <attributeName>knowledge</attributeName>
        <expression>35</expression>
      </action>
   </trueActions>
</generateListItem>
```

The above rule is connected to concept relationship "knowledge update". When the rule is triggered, its condition called "requirement" is checked. It is executed when health.Food.suitability is false and health.Food.knowledge is lower than 10. The action will assign the value 35 to the knowledge of health.Food.

```
<generateListItem isPropagating="true" >
   <requirement>true</requirement>
   <trueActions>
      <action>
        <conceptName>health.Health</conceptName>
        <attributeName>knowledge</attributeName>
        <expression>health.Health.knowledge +
                 (0.5 * _health.Food.knowledge)</expression>
      </action>
   </trueActions>
</generateListItem>
```

The above rule is connected to concept relationship "knowledge propagation". When the knowledge of a concept is changed, the change is propagated to the concepts that are higher in the concept hierarchy. Concept "Food" is a child of concept "Health" in "health" application. The action will add the knowledge of health.Health to 50% of the knowledge of health.Food and assign the value to the knowledge of health.Health.

```
<attribute name="suitability" type="bool" isPersistent="false"
       isSystem="false" isChangeable="false">
   <description>the suitability of this page</description>
   <default>((health.Disease.knowledge &gt; 0 ))</default>
</attribute>
```

The above rule is connected to concept relationship "prerequisite relationships". health.Disease is a prerequisite for health.Food. The suitability of health.Food depends on the knowledge of health.Disease. The rule requires the knowledge of health.Disease to be higher than 0 in order to access to the page of health.Food.

When a user logs in the AHA! server, the user model of the user is automatically generated. The user model consists of a set of concepts with attributes. It contains an overlay model which means that there is a concept in the user model for every concept in the domain model. After the domain and adaptation model are constructed, the following attributes related to concept health.Food are inserted to the existing user model when a user accesses the AHA! server.

```
<record>
    <key>health.Food.access</key>
    <type>3</type>
    <persistent>false</persistent>
    <value>false</value>
  </record>
  <record>
    <key>health.Food.interest</key>
    <type>1</type>
    <persistent>true</persistent>
    <value>0</value>
  </record>
  <record>
    <key>health.Food.suitability</key>
    <type>3</type>
    <persistent>false</persistent>
    <value>false</value>
  </record>
  <record>
    <key>health.Food.visited</key>
    <type>1</type>
    <persistent>true</persistent>
    <value>0</value>
  </record>
  <record>
    <key>health.Food.knowledge</key>
    <type>1</type>
    <persistent>true</persistent>
    <value>0</value>
  </record>
```

5 Application

In this section, we show how the proposed system can be used for authoring adaptive hypermedia. In addition, we describe the pilot evaluation about our system.

Assume that an author wants to create an AHA! application that introduces basic concepts of biology. Although it is possible to design a domain model by using AHA! authoring tools, building a well organized domain model is not a simple task because it involves choosing proper concepts and logically making relationships between them. If the author can use existing ontologies written

by domain experts, creation of a domain model might be easier. So, the author decides to use our system and finds ontologies on the Web using ontology search engines such as Swoogle [18]. Figure 8 shows the structure of an OWL ontology about biology in the tree-view of Protégé-OWL [16]. Our system transforms the OWL ontology document to an XML document that is

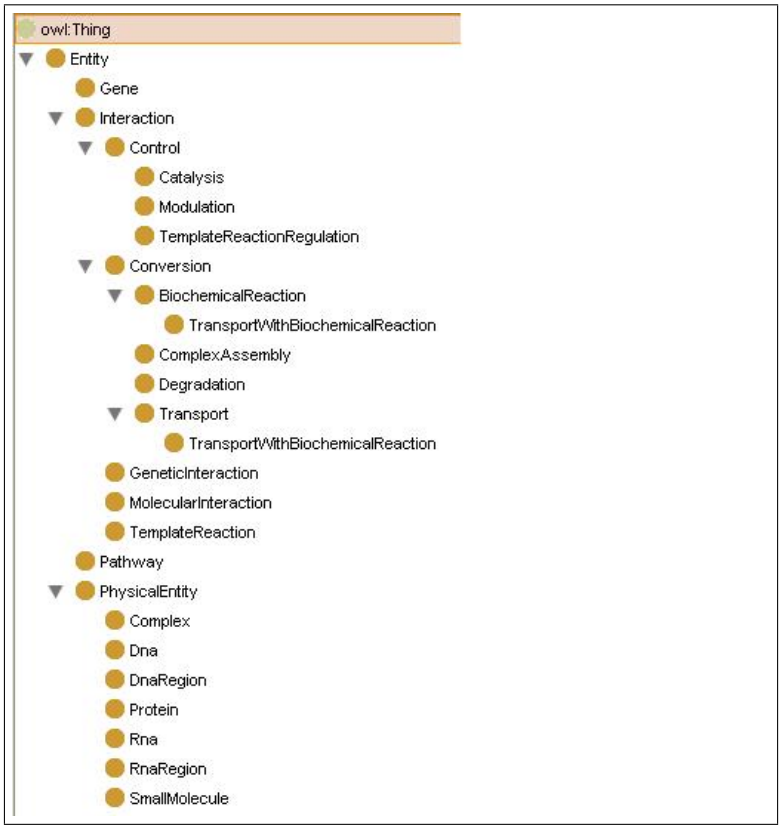

Figure 8: biopax.owl ontology

recognized in AHA! Graph Author. If the author opens an OWL document (biopax.owl) and clicks a button "Converter" on the top, the OWL document is transformed to the XML document (biopax.gaf). (figure 9) The author opens the transformed document in AHA! Graph Author and edits parts of the file appropriately. (figure 10) Similarly, if an author wants to create a domain model about a koala, the author can start with a simple ontology (koala.owl) provided by Protégé site [17]. Figure 11 shows the constructed domain model (koala.gaf) in Graph Author.

We conducted a pilot evaluation with a few PhD students who had experiences using AHA! system. They also had general understanding about OWL ontlogies. They were asked to create domain models about data structure and heals sciences. While they could use Graph Author, they found it difficult to model concept hierarchies for the domain areas. On the other hand, they felt it easy to use existing ontologies found on the Web to create domain models since these ontogies are often times defined by experts in the fields and contain basic concepts and their relationships. Here are some of their comments.

1. Authors can easily find the essential concepts about a domain and their relationships using ontologies defined by domain experts if the authors are not familiar with the domain of interests.

2. Authors can save time and efforts authoring the domain model and adaptation model because they are generated quickly with our system once an OWL ontology is given.

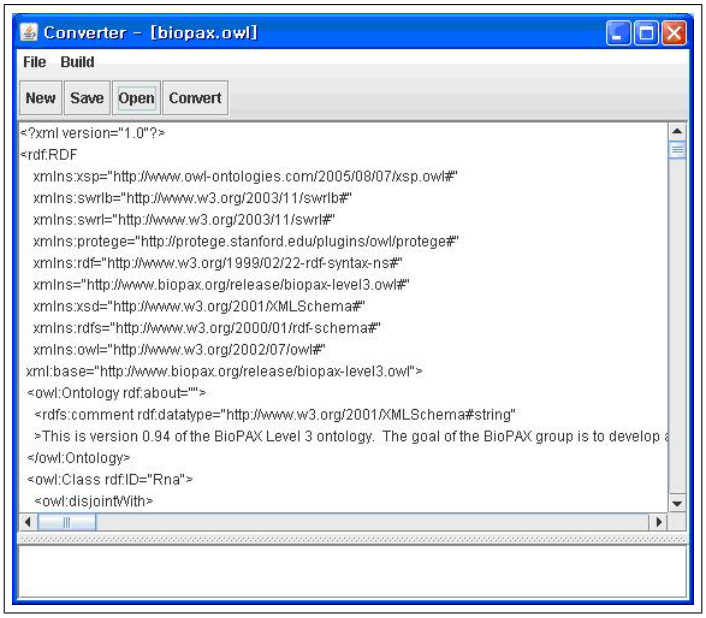

Figure 9: biopax.gaf that was transformed from biopax.owl ontolgy using our system

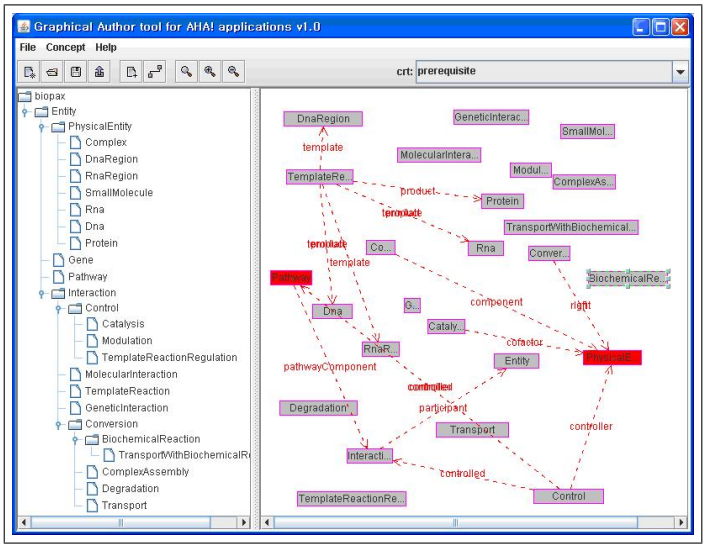

Figure 10: biopax.gaf in Graph Author

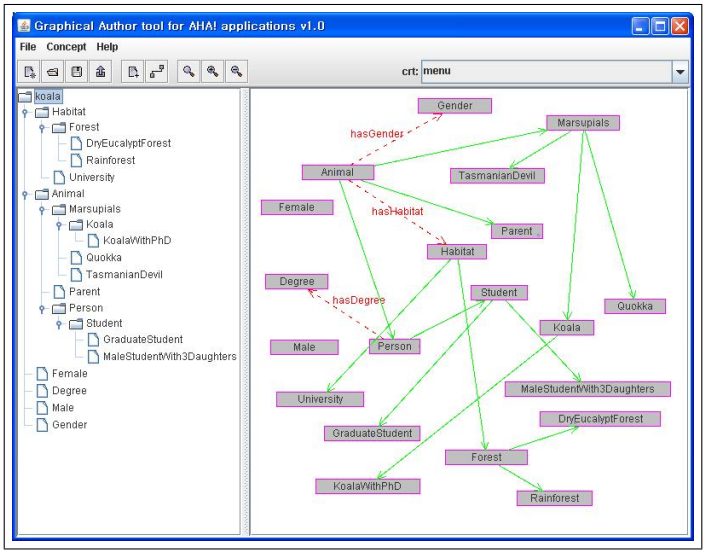

Figure 11: koala.gaf in Graph Author

3. It is helpful when authors have it difficult in conceptualizing a particular domain. In addition, it is easy to reorganize and expand the concepts and concept relationships.

6 Conclusions and Future Works

In this paper, we propose a system that can be used to author adaptive hypermedia using ontologies. Generating the domain model is a basic task for authoring adaptive hypermedia because it influences on generating the adaptation model and the user model. Our system makes it easy to create the domain model using an OWL ontology. The concepts and concept relationships are created automatically by our system and authors do not have to manually create them. When an author designs a domain model, it may not be easy to define concepts of a domain and their relationships. In ontologies, concepts about a domain and their relationships are already defined by domain experts. So, an ontology can serve as a starting point to define a domain model and the author can edit the domain model appropriately.

We plan to extend our system so that it can transform AHA! domain models into OWL ontologies so that it supports authoring a linked data [19]. An AHA! author creates a structure of the domain model consisting of concepts and concepts relationships and writes local resources consisting of a set of xhtml pages. Every page must have a corresponding concept. Both local and remote pages can be associated with a concept. The resources in a domain model can be considered as a linked data. Using our system, authors can gather lots of OWL ontologies transformed from the domain models and create linked data if our system can enable authors to search resources and link one to others. The linked data can be also reused in AHA! system.

We also plan to combine our system with other adaptive hypermedia systems such as Mag system and AHyCo system.

In the case for Mag system, our system helps the instructors define the concepts of the domain model of Mag system by using the domain ontology. It extracts all concepts defined in the ontology and it converts them into the concepts of the domain model. Then, it generates the domain model of Mag system that consists of the converted concepts. If the instructors can use the concepts already defined by domain experts or other instructors, they can easily design the domain model. (Figure 12)

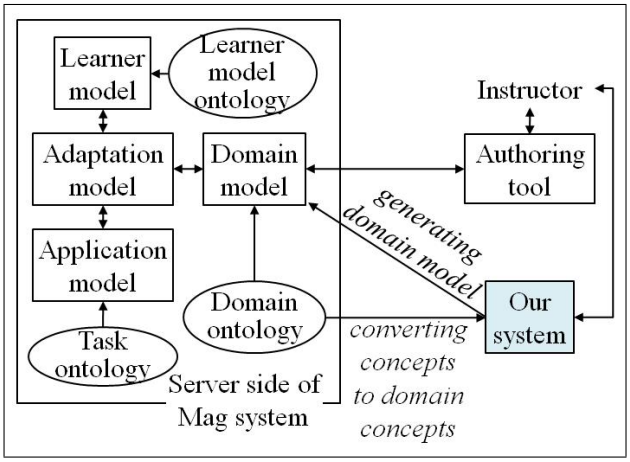

Figure 12: Combination with Mag system

In the case of AHyCo system, our system can help authors define the lessons, group them into modules, and connect between them by the prerequisite relationships. Even if our system cannot support the creation of the real content of each lesson such as text or image, it can offer the structure of the domain model by converting an ontology. It converts a class to a module, and an instance of the class to a lesson. The hierarchy of classes is transformed into the prerequisite relationships between modules. For example, a class is prerequisite for its subclasses. It means that the class is learned before its subclasses. The object property linking between instances of two classes is transformed into the prerequisite relationships between lessons. For example, an instance of its domain class is prerequisite for an instance of its range class. It means that the instance of the domain class is learned before the instance of the range class. Instead of the lessons created by Word, Excel or PowerPoint objects, it stores the URIs of classes or their instances as resources. Our system can provide the authors the predesigned domain model based on the ontology. (figure 13)

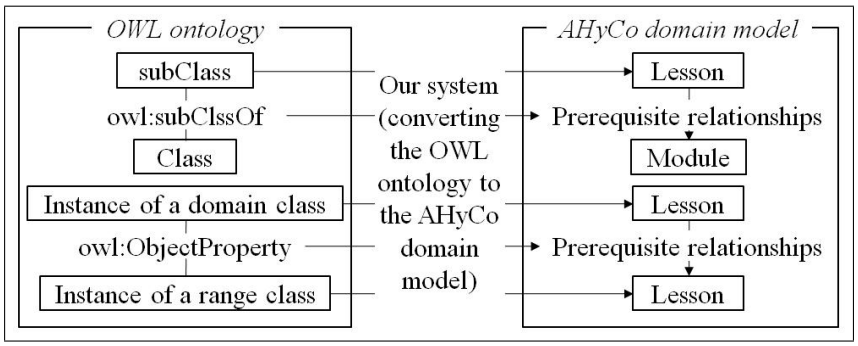

Figure 13: Combination with AHyCo system

Acknowledgement

Seongbin Park is the corresponding author. This research was supported by a College of Education of Korea University Grant in 2010.

Bibliography

[1] De Bra, P., Stash, N., Smits, D., Creating Adaptive Applications with AHA!, Tutorial for AHA! version 3.0, Tutorial at the AH 2004 Conference, Eindhoven, pp.4-20, 2004

[2] Stash, N., De Bra, P., Building Adaptive Presentations with AHA! 2.0, *Proceedings of the PEG Conference*, Saint Petersburg, 2003

[3] De Bra, P., Aerts, A., Berden, B., De Lange, B., Rousseau, B., Santic, T., Smits, D., Stash, N., AHA! The Adaptive Hypermedia Architecture, *Proceedings of the ACM Hypertext Conference on Hypertext and Hypermedia*, ACM, Nottingham, pp. 81-84, 2003

[4] Cristea, A., Calvi, L., The Three Layers of Adaptation Granularity, UM'03, LNCS 2702, Springer, 2003

[5] Cristea, A., Smits D., De Bra, P., Towards a Generic Adaptive Hypermedia Platform, a conversion case study, *Journal of Digital Information*, Vol.8, No.3., 2007

[6] Gruber, T.,Ontology, http://tomgruber.org/writing/ontology-definition-2007.htm

[7] McGuinness, D. L., van Harmelen, F., OWL Web Ontology Language Overview, W3C Recommendation, 2004, http://www.w3.org/TR/owl-features

[8] Wu, H., Houben,G. J., De Bra, P., Authoring Support for Adaptive Hypermedia Applications, *Proceedings ED-MEDIA'99*, Seattle, pp. 364-369, 1999

[9] Cristea, A. I., de Mooij, A., Designer Adaptation in Adaptive Hypermedia Authoring, ITCC'03, IEEE Computer Science, pp. 444-448, 2003

[10] Cristea, A., Evaluating Adaptive Hypermedia Authoring while Teaching Adaptive Systems, SAC'04, ACM, pp. 929-932, 2004

[11] De Bra, P., Houben, G.J., Wu, H., AHAM: A Dexter-based Reference Model for Adaptive Hypermedia, *Proceedings of the ACM Conference on Hypertext and Hypermedia*, Darmstadt, pp. 147-156, 1999

[12] Wu, H., De Kort, E., De Bra, P., Design Issues for General-Purpose Adaptive Hypermedia Systems, *Proceedings of the ACM Conference on Hypertext and Hypermedia*, Aarhus, pp. 141-150, 2001

[13] http://protege.stanford.edu/plugins/owl/ontologies.html

[14] Berners-Lee, T., Hendler, J., Lassila, O., The Semantic Web, *Scientific American Special online Issue*, 2001

[15] Heflin, J., Volz, R., Dale, J., Web Ontology Requirements, Proposed W3C Working Draft, 2002, http://km.aifb.uni-karlsruhe.de/projects/owl/index.html

[16] http://protege.stanford.edu/download/protege/3.4/installanywhere (Protégé editor download site)

[17] http://protege.stanford.edu/download/registered.html (Protégé-OWL source code download site)

[18] http://swoogle.umbc.edu

[19] Bizer, C., Heath, T., Idehen, K., Berners-Lee, T., Linked Data on the Web (LDOW 2008), *Proceedings WWW2008*, Beijing, China, 2008

[20] De Bra, P., Santic, T., Brusilovsky, P., AHA! meets Interbook, and more..., *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2003*, pp. 57-64, 2003

[21] Cristea, A., Smits, D., De Bra, P., Writing MOT, Reading AHA! converting between an authoring and a delivery system for adaptive educational hypermedia, *A3EH Workshop at AIED'05*, Amsterdam, the Netherlands, pp. 36-45, 2005

[22] Stewart, C., Cristea, R., Brailsford, T., Ashman, H., Authoring once, Delivering many: Creating reusable Adaptive Courseware, *Proceedings of the 4th IASTED International Conference on Web-Based Education*, WBE 2005, pp.21-23, 2005

[23] Meccawy, M., Celik, I., Cristea, A., Stewart, C., Ashman, H., Interoperable Adaptive Educational Hypermedia: A Web Service Definition, *Proceedings of Sixth IEEE International Conference on Advanced Learning Technologies (ICALT'06)*, pp. 639-641, 2006

[24] Hendrix, M., Cristea, A., and Nejdl, W., Authoring adaptive educational hypermedia on the semantic desktop, *International Journal of Learning Technology*, 3(3):230-251, 2007

[25] Klasnja-Milicevic, A., Vesin, B., Ivanovic, M., Budimac, Z., Integraion of Recommendations and Adaptive Hypermedia into Java Tutoring System, *Computer Science and Information Systems 2010 OnLine-First*, 2010

[26] http://aha.win.tue.nl/publications.html

[27] Hoic-Bozic, N., Mornar, V., AHyCo: a Web-Based Adaptive Hypermedia Courseware System, *Journal of Computing and Information Technology*, CIT 13, 3, p. 165-176, 2005

[28] Thomas, E., Alani, H., Sleeman, D., Brewster, C., Searching and Ranking Ontologies on the Semantic Web, *Workshop on Ontology Management: Searching, Selection, Ranking, and Segmentation, 3rd K-CAP* Banff, Canada. pp. 57-60, 2005