Generating ontologies from Intelligent Tutoring System courses. A generic approach.

Hector Escudero [a,*], Ramón Fuentes-Gonzalez [b]
[a] *Department of Mathematics and Computer Engineering, Public University of Navarre*
[b] *Automatics and ComputingAutomatics and Computing, Public University of Navarre*

Abstract. In recent years a great effort has been done in order to create Intelligent Tutoring Systems that get close to the human teaching. Some of the handicaps of the systems already created is the impossibility of sharing the courses between different Intelligent Tutoring Systems and the difficulty to create them. Whereas a great amount of SCORM-compliant learning objects is being created for being imported in educational systems, there is a research current that pleads for ontologies as domain knowledge representation systems. We have created a generic and extensible authoring tool that creates courses for different intelligent tutoring systems. The authoring tool allows the creation of courses for different types of intelligent tutoring systems and saves those courses as ontologies. This allows the reusing of domain models between different Intelligent Tutoring Systems. This paper focuses in the ontology creation and populating process.

Keywords: Intelligent tutoring system, Authoring tool, Ontology

1. Introduction

Intelligent Tutoring Systems (ITSs) are computer based systems that provide individualized tutoring to the students [27]. About 20 years ago, research by Prof. Benjamin Bloom and others demonstrated that students who receive one-on-one instruction perform two standard deviations better than students in traditional classrooms [3]. That is, the average tutored student performed as well as the top 2 percent of those receiving classroom instruction. Besides, research on prototype systems indicates that students taught by ITSs generally learn faster and translate the learning into improved performance better than classroom-trained participants.

Providing a personal training assistant for each learner is beyond the training budgets of most organizations. However, a virtual training assistant who captures the subject matter and the teaching expertise of experienced trainers provides a captivating new option. ITS research has been done for more than three decades by researchers in education, psychology and artificial intelligence.

The good of ITS is to provide the benefits of one-on-one instruction automatically and cost effectively. Like training simulations, ITSs enable participants to practice their skills by carrying out a task within highly interactive learning environments. However, ITSs go beyond training simulations by answering user questions and providing individualized guidance. Unlike other computer-based training technologies, ITSs assess each learner actions within these interactive environments and develop a model of the knowledge, skills and expertise. Based on the learner model, ITSs tailor instructional strategies, in terms of both the content and style, and provide explanations, hints, examples, demonstrations and practice problems as needed [25].

However, ITSs are still seen with scepticism, since they have not been extensively used in real educational settings. The main reason for this limited use

[*]Corresponding author. E-mail: hector.escudero@unavarra.es.

is probably the fact that the task of constructing an ITS is complex, time-consuming and involves a large number of people, including programmers, instructors and experts of a specific domain. Moreover, once constructed, an ITS for a specific domain cannot be re-used for different domains without spending much time and effort. An approach to simplifying the ITS construction is to develop ITS authoring tools that can be used by a wider range of people to easily develop cost-effective ITSs [22].

In order to solve these problems, we propose a configurable ITS course authoring system. We think that the course creation can be independent of the tutoring system. This is, there should be an authoring tool that creates the courses for any ITS, and the courses created should be imported into the ITS, that would represent it. At the time of creating a course, the domain model is the same for all the ITSs, but it must be adapted to the limitations of each ITS. Therefore, there is a basic structure that all the courses share. We have created an authoring tool where the user can configure the characteristics of the ITS where the course will be represented, and create the course using it. Then, the course will be saved in a generic way, so it can be reused in the same authoring tool to create a course for a different ITS. This is the first part of our ongoing research, the creation of an authoring tool capable of create courses for a wide range of Intelligent Tutoring Systems, no matter the teaching strategy they use. This tool is still in prototype phase, but has been tested successfully with two simple ITSs. The second part is taking the most of the knowledge that the course creators generate. To achieve this, we have taken the decision of saving the courses as ontologies. It must be remarked that this authoring tool only creates the courses for the ITSs. The idea is that these courses will be exported to the specific format that each ITS needs, but also will be saved in a generic way, so it can be exported, making little variations, to other formats.

This paper presents the second part of the ongoing research. It will be explained why have we cosen ontologies as saving format and the process that led us to build the ontology. However, for the better understanding of the ontology building process, a brief explanation of the generic authoring tool will be given. If necessary, a deeper explanation can be found in [10].

This paper is organized as follows. Section 2 outlines the course creation tool. Section 3 presents the process to save the courses as ontologies. In section 4 an example of a course is shown. Finally, section 5 discusses the main conclusions and future work.

2. The generic course creation tool

Every course is created to represent a specific domain model. Our aim is to convertir this model into an ontology. Therefore, the way the authoring tool creates these courses and domain models is important in order to understand how the ontology is created.

An easy to understand way of codifying a domain model is a concept map. A definition of a concept map, given by [18] is ”concept maps are two-dimensional representations of cognitive structures showing the hierarchies and the interconnections of concepts involved in a discipline or sub-discipline.” Once completed, the concept map is a visual graphic that represents how the creator thinks about a subject. Usually, a concept map is divided into nodes that represent concepts and links, that represent relationships (propositions) between concepts [15]. [5] and [28], for example, have used concept-maps in tutoring systems to organize knowledge successfully. As the model will be represented as a concept map, the best way to create and represent it is a graph editor.

Each node of the graph represents a concept to be taught, and the links represent the relations between the concepts. Another thing to have in mind are the attributes of the concepts. Depending on the ITS, concepts contain different attributes. Those attributes can be simple, like a text or a number, or complex. For example, the concepts of the Iris Shell ([1],12) contain two simple attributes, ”Estimated time” and ”Difficulty”, whereas the concepts of the AHA! System [6], contain several complex attributes. The authoring tool allows de definition of complex attribute types combining some basic attributes. There are four basic types in the authoring tool: Text, CheckBox, Combo and List. In Texts the user can write anything, CheckBoxes are for boolean values, Combos offer a set of predefined values and Lists contain several attributes of the same type. Those attributes can be combined to create new attributes.

The authoring tool contains a visual graph editor to create the concepts that will be part of the course. Then, for each concept the information that will be represented to define that concept and the attributes are created.

3. Choosing a language to save the courses

Now that the basis of the authoring tool has been outlined, the process to define the ontology will be explained. The first idea to save the courses in a generic way was to create an XML language for it. As the potential information of the courses is known, it would be easy to create a language able to store all the information created by the authoring tool. Then, making the structure of the language public, anyone can create a parser that translates the information saved in the XML format to the specific format of the ITS.

This approach would fulfill the basic requirements of the tool. Nevertheless, we started thinking if it would be possible to make use of the knowledge created by the users of the tool. The course creation process is based in a set of concepts that have relations between them. This could be seen as an ontology. It is true that for simple courses, like online courses that only have the ”next concept” relations the ontology is nonsense, as the ”next” relation does not contribute to specify any new knowledge about the two concepts that links. However, more complex ITSs use relations like ”is-a” or ”part-of” that can be used to generate knowledge. The authoring tool will be used by people that are expert in a subject, but not necessarily know what an ontology is. When they create the course, they will create a structure of concepts and relations based on their experience and expertise. We know that this is not the most common method to create ontologies, but it could be a good approach for people that is not used to work with ontologies and computers.

In recent years, there is a trend that claims that ITSs should make use of ontologies. Back in 2000, [21] maintained that future ITSs must be built upon ontologies and proposed building the system using three-level ontologies. Level one and two ontologies should define basic terms and definitions about those terms, whereas the ontology on level three should work with software modules. Later, they have built an instructional theory-aware system [13]. The basis of the system is an ontology that contains all the instructional theories. The creator of the course has to make relations between predefined Learning Objects and the theories of the ontology. This way, the tutoring system triggers events, which depending on the theories previously selected, make different actions to happen. The advantage of this system is the possibility of merging different instructional theories. The course creator can define different WAYs. This is, different ways to make the learner acquire knowledge. The disadvantages are that the ontology must be maintained by an expert and that the domain model of the course is not reusable. [2] present a system where ontologies are the core of the system. The system is based in four ontologies: the Domain Ontology, where the characteristics of the domain knowledge are provided, the Student model ontology, the Pedagogical model ontology and the Interaction ontology.

[9] outlines another tutoring system based in ontologies. They subdivide ontologies in three different types, the ones that describe specific domains, the ontologies for creating community-oriented knowledge structures and the ontologies for describing learning communities. They want to take advantage of Semantic Web technologies to enrich semantically the tools for learning using those ontologies. Again, the main part of what would be the course, the domain ontology, must be created specifically for the tutoring system.

Finally, there are some systems that do not use ontologies in their domain model representations, but use them to make content more accessible. Ontologies can be used to search between large amounts of learning objects, which sometimes are not directly understandable. [30] for example, use ontologies to search between the great amount of SCORM ([16] and [24]) objects available in the internet and choose the most appropriate ones for the domain model that is being created. [12] proposes a framework that eases the interaction between e-learning systems based on different ontologies using a mapping ontology. Another example is TM4L [8], an environment that uses ontologies to annotate and search educational objects in different repositories. In [31], data is extracted from raw text and organized in concept maps. Then, ontologies are derived from those concept maps. Finally, those ontologies can be used to sustain the production of e-learning objects.

Those systems record the domain models as ontologies like our system, but still there are some differences. First, the content they create is specific for a platform or ITS, whereas our system allows the generation of ontologies for any type of ITS. This makes the ontology, and the domain model, independent of the ITS. No matter which is the representation system of the ITS (audio, video, web pages...) the ontology can be imported into the system, and only the information to be represented should change for each ITS or platform. Second, the ontology creation is hidden to the user. We expect that the users of the authoring tool will not be used to computers. They only have to care about creating a course about a knowledge domain they are expert in. They will know that they can save and load courses, but will not know that they are managing an ontology.

3.1. *Basic concepts of the ontology*

There is no correct methodology to create an otology. However, there are several rules that must be followed in any ontology creation process independently of the methodology chosen, and that can help making decisions in some points of the process.

- There is no correct way to define a domain. There always are alternatives. Most times, the result depends on the final use of the ontology.
- The ontology creation process is an iterative process.
- The concepts of the ontology should look like the concepts and relations of the domain to be represented. Usually those will be nouns (for concepts) and verbs (for relations) in sentences that describe de domain.

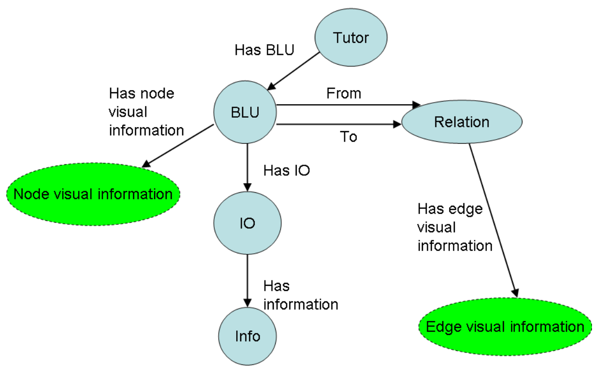


Fig. 1. A first approach to the structure of the ontology.

Deciding the final use of the ontology, and if it will be used for a generic or detailed use can guide the decisions. Finally, the only way to prove if the ontology is correct is to test it in the application it has been created for. This way errors will appear, and the ontology can be redefined. This iterative process will last until the ontology is correct. Usually the ontology creation methodology presented by [23] is followed. The problem of these methodologies is that the domain of the ontology must be clear in order to follow all the steps. In our case, the domain of the ontology varies depending on the course.

If our aim was just to record the information of the course to be loaded later we could create an ontology generic enough to contain all the courses. In the figure 1 there is a representation of an ontology that can contain all the data created in the courses. The problem is that this representation does not contribute to structure the content and create new knowledge from the course.

The conclusion is that we have to create a new ontology for each course. All courses contain several concepts which are related, and those concepts contain attributes and information.

3.2. *Structure of the ontology*

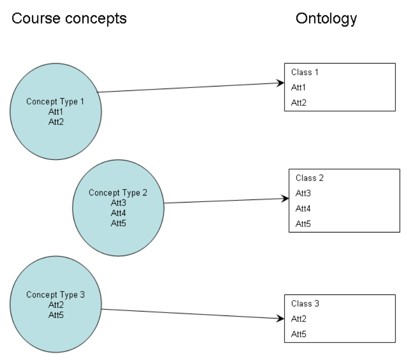
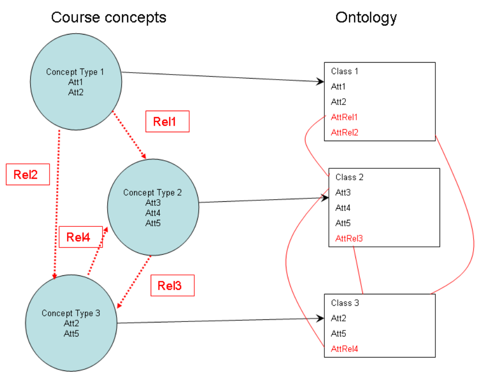


Fig. 2. Transformation of the course concept types into ontology classes.

As mentioned previously, the course can be seen as a set of concepts. Each concept will become an instance of the new ontology. As instances need a class to be created from, for each concept type a new class will be created. ITSs are based on concept types. Theses concept types can contain several attributes. Therefore, the first step to create the ontology is to create the structure, so the instances, which will contain the knowledge generated in the course, can be inserted.

The configuration of the ITS determines the number of different concept types and the attributes that each concept type has. Therefore, we have to extract the structure of the ontology from the configuration of the ITS. The tool is based on based on Merrill's "Component Display Theory" (CDT) [20]. This instructional design theory is based in Basic Learning Units (BLUs), which can contain several Instructional Objectives (IOs). When the user defines the ITS, IOs and BLUs are defined. BLUs contain several IOs, and the content of each concept of the course is based on the BLUs. Each IO and BLU contains several at-



Course concepts
Ontology
Concept Type 1 Att1 Att2
Concept Type 2 Att3 Att4 Att5
Concept Type 3 Att2 Att5
Rel1
Rel2
Rel3
Rel4
Class 1 Att1 Att2 AttRel1 AttRel2
Class 2 Att3 Att4 Att5 AttRel3
Class 3 Att2 Att5 AttRel4

Fig. 3. Adding the relations between concepts as ontology properties.

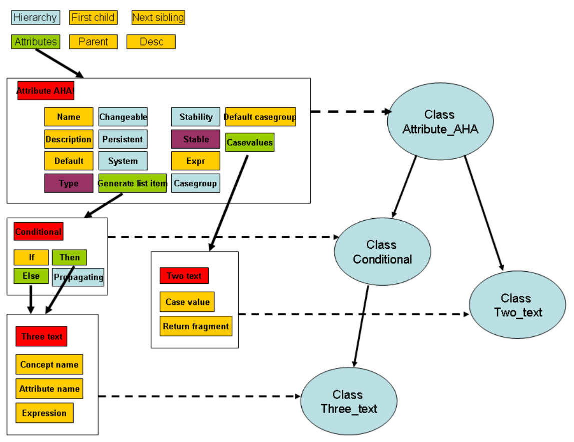


Fig. 4. Transformation of the complex attributes into ontology classes.

tributes, and these attributes are the ones that will define the concept types. Therefore, each concept type becomes a class in the ontology, and the attributes become properties of these classes (figure 2).

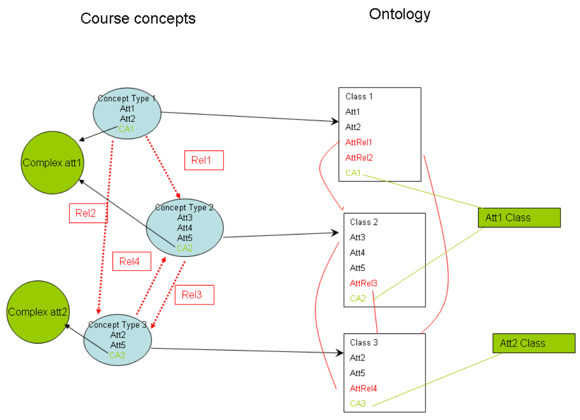
Besides, all the concepts of the ITSs have relations between them. To reflect this in the ontology for each relation a new property is created (figure 3). For example, if the concept ”a” of the ITS has a relation ”rel” which points to the concept ”b”, in the ontology a new property will be added to the class ”a” called ”rel”. The domain of this new attribute will be instances of class ”b”.

In our system attributes can be complex, this is, some attributes contain other attributes. In an ontology attributes can be hierarchised, saying that attributes ”a” and ”b” are part of attribute ”c”. But there is a problem when attribute ”c” can contain several appearances of ”a” and ”b” pairs. In that case a new class is needed. This class will contain two attributes, ”a” and ”b”. The class where the attribute ”c” is contained will have a reference to the new class instances instead of containing the values on it’s instance (figure 4).

For example, in the AHA! System ITS the attributes of the concepts are very complex. In the left side of the figure 4 the hierarchy of the attributes of the AHA! System is shown. The BLUs of the AHA! System contain a list of attributes called Attribute AHA, whereas the Attribute AHA contains attributes called Conditional and Two Text. Finally, the Conditional attribute contains a list of attributes called Three Text. In the right side of the figure 4 the class hierarchy derived from the attributes is shown. A new class is created for each complex attribute, and the relations between the attributes are maintained in the ontology (figure 5).

Finally, the information represented in the course must be stored. Each BLU contains a bit of information, and that information describes de concept that the BLU represents. For that purpose the ”rdfs:isDefinedBy” property will be used.

”rdfs:isDefinedBy” is an instance of ”rdf:Property” that is used to indicate a resource defining the subject resource (referencia a http://www.w3.org/TR/rdf-schema/). Therefore, the information must be stored in a resource and the the concept created must point to that resource using the ”rdfs:isDefinedBy” property (figure 6).



Course concepts
Ontology
Concept Type 1 Att1 Att2 CA1
Concept Type 2 Att3 Att4 Att5 CA2
Concept Type 3 Att2 Att5 CA3
Complex att1
Complex att2
Rel1
Rel2
Rel3
Rel4
Class 1 Att1 Att2 AttRel1 AttRel2 CA1
Class 2 Att3 Att4 Att5 AttRel3 CA2
Class 3 Att2 Att5 AttRel4 CA3
Att1 Class
Att2 Class

Fig. 5. Adding the complex attributes to the structure of the ontology.

Let’s show an example in order to clarify the whole process. The Iris Shell, mentioned earlier is based in Merrill's 'Component Display Theory', which can contain four different types of BLUs: Concept, Procedure, Principle and Fact. Because of the complexity of managing all BLUs types, the Iris Shell only uses two of them: Concept and Procedure. Therefore, two new classes are created for the ontology, the "Concept" class and the "Procedure" class. The "Concept" class has two attributes, "Significance" and "Summary". Then, a new property is created for each attribute. The domain will be a String and the range the "Concept" class. The "Procedure' BLU has the same two attributes. There is no need to create the attributes, but the class "Procedure" must be added to the range of the attributes. The final step is to define the relations between the BLUs. The Iris Shell supports six different relation types: co-requisite, prerequisite, post requisite, is-a, part of and next. Therefore, six properties must be created, one for each relation type. The range and domain of these properties are the union of all the BLU classes created, "Concept" and "Procedure" for this example.

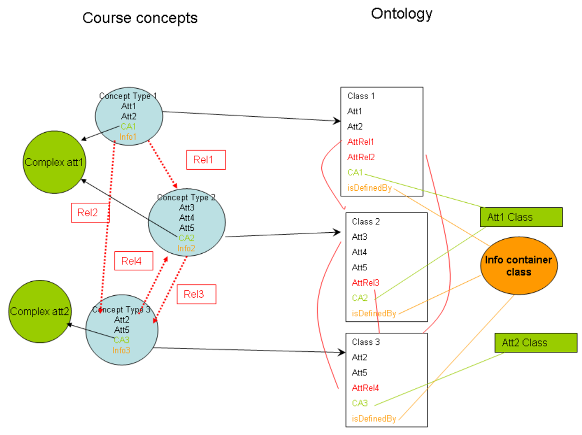


Fig. 6. Adding the information created in the course using the isDefinedBy property.

3.3. *Populating the ontology*

Once the ontology structure has been created it is time to populate it with the data introduced using the authoring tool during the course creation. Each node created using the authoring tool will become an instance of one of the BLU classes created. The second step is to insert the values of the attributes introduced during the course creation process into the properties of each instance.

Besides the attributes, each BLU contains some information that will be shown to the student. During the creation of the structure of the ontology, a class has been created for each type of information representation. For each piece of information that has been added to the BLU (text, image, avatar conversationĚ) a new instance of the corresponding class will be created. Finally, each BLU instance is related with the information pieces that belong to it using the isDefinedBy property.

The last step of the ontology population process is to add the relations between the concepts. For each relation between the concepts created using the authoring tool a new property is created. The subject and the object of these properties will be an instance of the BLU classes.

3.4. *Implementation*

Courses are saved in OWL format [14]. The Web Ontology Languaje (OWL) is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF [17], and RDF Schema (RDF-S) [4] by providing additional vocabulary along with a formal semantics. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms (ontolgies). OWL provides three sub languages:: OWL Lite, OWL DL and OWL Full. OWL Lite supports those users primarily needing a classification hierarchy and simple constraints. OWL DL supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For the ontology that the authoring tool creates, it is enough with the use of the OWL Lite sub language.

The ontology has been implemented using the Jena Semantic Web Framework [19]. Jena provides a programmatic environment for RDF, RDFS and OWL, SPARQL [26] and includes a rule-based inference engine. The first step to start working with the framework is to create a Model. The Jena framework is based in statements. Therefore, an OWL model is defined as a set of statements. Each statement asserts a fact about a resource, and has three parts: the subject, the predicate and the object. Each time a property is added to a resource, one of those triples is created. The Jena framework supports the distinction between DatatypeProperties and ObjectProperties. DatatypeProperties are properties that relate instances of classes and RDF literals, whereas ObjectProperties relate instances of two classes.

4. An example: how to work with a lathe

For the example, we are going to create a course about how to work with a lathe, which will be compatible with the Irirs Shell.

4.1. Configure the ITS

First of all, the characteristics of the STI mus be defined. The Iris Shell can work with two IO, Knowledge and Application. Therefore, these two IOs must be created using the authoring tool. For each IO, the attributes it contains and the representation formats it can support must be defined. For this example, two representation systems have been loaded into the system: a text editor and the Ekit html editor.

Once the IOs are created is time to define de BLUs. The Iris Shell can manage two BLUs: Concept and Procedure. The Concept BLU can contain Knowledge IOs, whereas the Procedure BLU can contain Knowledge and Application IOs. Finally, the BLUs that the ITS can support (figure 3) and the possible relations between the BLUs must be defined. The relations that the Iris Shel can manage are: co-requisite, post requisite, prerequisite, is-a, part-of and next.

4.2. Create the course

Now that the ITS is configured it is time to create the course. The course will explain the different parts of a lathe and some of the different uses it can have. The first step is to create all the BLUs and the relations between them. The relations must be introduced one by one. In the top of the editor the user has to choose the relation he/she wants to work with. In the figure 4 the "is a" relations are shown.

Figure 4. The "is-a" relations of the lathe course. Once all the relations are introduced a whole view of the concept map can be seen using the model viwer, although this view is not editable (figure 5). Finally the content of the course must be added to the BLUs. For each BLU the attributes it contains and the information that will be presented to student mus be created using the editors defined for this tutor.

4.3. Save the course

First the structure must be created using the procedure explained earlier. There course contains two concept types, therefore two classes must be created (listing 1).

```
OntClass uba_res_class = model.createClass(
    uri+tutorName+"_" + configUBA.getName());

uba_res_class.addProperty(RDFS.subClassOf,
   RDFS.Resource);

uba_res_class.setRDFType(RDFS.Class);
```

Listing 1: Code to create the new classes from the course concept types.

Then the attributes must be added as properties of the newly created classes. The Iris Shell contains two attributes, "Significance" and "Summary". The problem is that both concepts share the same attributes. In the Jena framework adding several classes as the range of a property creates an intersection. Therefore, only one class is allowed to be the range of a property at a time. To allow several classes to contain the same property a union must be created. Then, that union will be the range of the property (listing 2).

```
RDFList unionList = model.createList(ubaNodeList);
OntClass unionClass = model.createUnionClass(null,
     unionList);

OntProperty prop = model.createDatatypeProperty(
    uri+satt.getName());

prop.setDomain(unionClass);
prop.setRange(unionClass);
```

Listing 2: Code to create a union class to set it as a domain or range in a property.

The next step is to create the relations between the concepts as properties of the classes. The Iris Shel can manage six different properties: co-requisite, post requisite, prerequisite, is-a, part-of and next (listing 3).

```
ArrayList relations = configTutor.getRelations();
Ierator rit = relations.iterator();
while(rit.hasNext()) {
  ConfigRelation rel = (ConfigRelation)rit.next();
  ObjectProperty prop = model.createObjectProperty(
    uri+rel.getName());
}
```

Listing 3: Code to create the relations between the concepts.

Once the structure is created it is time to insert the instances into the structure. First, for each node in the graph an instance of it's corresponding class (Concept or Procedure) is created (listing 4).

```
OntResource uba_res_class = (OntResource)ubaMap.get(
     uba.getType());

Individual ubaRDF = model.createIndividual(uri
     +ubaName, uba_res_class);
```

Listing 4: Code to create an instance of a concept of the course.

Then, for each instance, the values of the attributes are inserted (listing 5).

```
DatatypeProperty prop = (DatatypeProperty)this.
     datatypePropertyMap.get(attName);

ubaRDF.addProperty(prop, value);
```

Listing 5: Code to add the value to the properties of the instances.

The next step is to create instances to store the content that will be shown to the student in each BLU, and relate it to the instance of the BLU using the "isDefinedBy" property (listing 6).

```
Individual infoRDF = model.createIndividual(uri
     +"Info_" + infoName, info_class);
InfoRDF.addProperty(infoProp1, infoName);
infoRDF.addProperty(infoProp2, infoType);
infoRDF.addProperty(infoProp3, content);

ubaRDF.addIsDefinedBy(infoRDF);
```

Listing 6: Code to create the instances where to store the information of the concepts and the relation with the instances using the isDefinedBy property.

Finally, for each relation in the graph, a property that relates two instances of BLUs is created (listing 7).

```
Relation relation = (Relation)urit.next();
UBA fromUBA = relation.getFromUBA();
UBA toUBA = relation.getToUBA();

Individual fromResource = (Individual)instanceMap
     .get(fromUBA.getName());
Individual toResource = (Individual)instanceMap
     .get(toUBA.getName());
String relType = relation.getType();

ObjectProperty prop = (ObjectProperty)relationMap
     .get(relation.getType());
fromResource.addProperty(prop, toResource);
```

Listing 7: Code to create the relations between the instances.

As the ontology created is valid (not correct, we cannot assure that the concepts and their relations are correctly described in the course), it can be imported into an ontology editor like Protégè [29]. Once inside Protégè it can be exported to other formats or validated with the rules that Protégè incorporates.

4.4. *Loading the course*

As the course is saved as an ontology by default, when the user wants to reload the course the ontology must be read. This could be done going through the whole model reading the structure and instances, but there is an easier way to do it. The Jena Framework provides a programmatic environment for the SPARQL language. SPARQL (Simple Protocol and RDF Query Language) defines a standard query language and data access protocol for use with the Resource Description Framework (RDF) data model. It works for any data source that can be mapped to RDF.

For example, in the listing 8 the query to obtain all the BLUs of a specific type stored in the ontology.

```
queryString =
"PREFIX␣rdf:
␣␣<http://www.w3.org/1999/02/22-rdf-syntax-ns#>␣" +
"PREFIX␣un:␣<http://www.unavarra.es/sti#>␣" +
"SELECT␣?uba␣" +
"WHERE␣{" +
"␣?uba␣rdf:type" + "␣un:" + tutorName
+ "_" + bluType + "␣.␣" +
"␣}";
```

Listing 8: SPARQL query to obtain all the BLUs of ont type of the ontology.

5. Conclusion and future work

In this article an architecture for a generic authoring tool that creates courses for intelligent tutoring system and the first steps of the implementation of the core tool have been shown. Besides, the transformation of the courses into ontologies has been explained. We believe that once it is finished most of the intelligent tutoring system that exist nowadays could make use of this tool in order to share the knowledge they create. Although the freedom of the user is one of the strengths of the tool, there should be a way to validate the ontologies created. This could be achieved using an automatic sematic validator [7] or a collaborative space like in [11] where the authors can discuss the domain models they create. The next steps will be the completion of the core of the authoring tool and the test of it in different scenarios.

References

[1] Arruarte Lasa, A. (1999), Fundamentals and design of IRIS: a shell for building Intelligent Teaching-Learning Systems, *AI Commun.* **12(1-2)**, 113

[2] Bittencourt, I. I., Costa, E., Silva, M., and Soares, E. (2009), A computational model for developing semantic web-based educational systems, *Know.-Based Syst.* **22(4)**, 302

[3] Bloom, B., Engelhart, M., Murst, E., Hill, W., and Drathwohl, D. (1956), Taxonomy of Educational Objectives: Handbook I, Cognitive Domain, Longman

[4] Brickley, D. and Guha, R. (eds.) (10 February 2004), RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, Available at http://www.w3.org/TR/rdf-schema/

[5] Chang, K. E., Sung, Y. T., and Chen, S. F. (2001), Learning through computer-based concept mapping with scaffolding aid, *Journal of Computer Assisted Learning* **17(1)**, 21

[6] De Bra, P., Aerts, A., Berden, B., de Lange, B., Rousseau, B., Santic, T., Smits, D., and Stash, N. (2003), AHA! The adaptive hypermedia architecture, in *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, ACM, New York, NY, USA, pp 81–84

[7] Dibie-Barthélemy, J., Haemmerlé, O., and Salvat, E. (2006), A semantic validation of conceptual graphs, *Know.-Based Syst.* **19(7)**, 498

[8] Dicheva, D. and Dichev, C. (2006), C.: TM4L: Creating and Browsing Educational Topic Maps, *British Journal of Educational Technology - BJET* **37**, 391

[9] Dzbor, M., Stutt, A., Motta, E., and Collins, T. (2007), Representations for semantic learning webs: Semantic Web technology in learning support, *Journal of Computer Assisted Learning* **23(1)**, 69

[10] Escudero, H. and Fuentes, R. (2010), Exchanging courses between different Intelligent Tutoring Systems: A generic course generation authoring tool, *Knowledge Based Systems*

[11] Gaeta, M., Orciuoli, F., and Ritrovato, P. (2009), Advanced ontology management system for personalised e-Learning, *Know.-Based Syst.* **22(4)**, 292

[12] Gasevic, Dragan; Hatala, M. (2006), Ontology Mappings to Improve Learning Resource Search, *British Journal of Educational Technology* **37(3)**, 375

[13] Hayashi, Y., Bourdeau, J., and Mizoguchi, R. (2009), Using Ontological Engineering to Organize Learning/Instructional Theories and Build a Theory-Aware Authoring System, *Int. J. Artif. Intell. Ed.* **19(2)**, 211

[14] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S. (eds.) (27 October 2009), OWL-2 Web Ontology Language: Primer, W3C Recommendation, Available at http://www.w3.org/TR/owl2-primer/

[15] Jan Lanzing (1997), The concept mapping homepage, http://users.edte.utwente.nl/lanzing/cm_home.htm

[16] Lora Aroyo, Stanislav Pokraev, R. B. (2003), Preparing SCORM for the Semantic Web, in *International Conference on Ontologies, Databases, and Applications of Semantics*, Catania, Italy

[17] Manola, F. and Miller, E. (eds.) (10 February 2004), RDF Primer, W3C Recommendation, Available at http://www.w3.org/TR/rdf-primer/

[18] Martin, D. J. (1994), Concept Mapping as an Aid to Lesson Planning: A Longitudinal Study., *Journal of Elementary Science Education* **6(2)**, 11

[19] McBride, B. (2002), Jena: A Semantic Web Toolkit, *IEEE Internet Computing* **6(6)**, 55

[20] Merril, M. (1983), Component Display Theory,, *Instructional-design Theories and Models, An overview of their current status* pp 279–333

[21] Mizoguchi, R. and Bourdeau, J. (2000), Using Ontological Engineering to overcome common AI-ED problems, *Int. J. Artif. Intell. Ed.* **11 (2)(2)**, 107

[22] Moundridou, M. and Virvou, M. (2002), WEAR: A Web-Based Authoring Tool for Building Intelligent Tutoring Systems, in 2nd Helenic Conference on AI (ed.), *Proceedings, Companion volume*, Thessaloniki, Greece, pp 203–214

[23] Noy, N. F. and McGuinness, D. L. (2001), "Ontology Development 101: A Guide to Creating Your First Ontology", Technical report, Stanford University

[24] of Electrical, I. and Engineers, E. (eds.) (12 June 2002), Draft Standard for Learning Object Metadata, IEEE-Standards Association

[25] Ong, J. and Ramachandran, S. (2003), Intelligent tutoring Systems: Using AI to Improve Training Performance and ROI, *Networker Newsletter* 19(6)

[26] Prud'hommeaux, E. and Seaborne, A. (eds.) (15 January 2008), SPARQL Query Language for RDF, W3C Recommendation, Available at http://www.w3.org/TR/rdf-sparql-query/

[27] Schoksey, S. D. (2004), *Master's thesis*, Worcester polytechnic institute

[28] Shih, B.-J., Shih, J.-L., and Chen, R.-L. (2007), Organizing Learning Materials through Hierarchical Topic Maps: An Illustration through Chinese Herb Medication, *Journal of Computer Assisted Learning* **23(6)**, 477

[29] Standfod, University (2010), "Protegé", http://protege.stanford.edu

[30] Tsai, K. H., Chiu, T. K., Lee, M. C., and Wang, T. I. (2006), A Learning Objects Recommendation Model based on the Preference and Ontological Approaches, in *ICALT '06: Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies*, IEEE Computer Society, Washington, DC, USA, pp 36–40

[31] Zouaq, A. and Nkambou, R. (2008), Building Domain Ontologies from Text for Educational Purposes, *IEEE Transactions on Learning Technologies* **1**, 49