Knowledge-Based Systems 23 (2010) 864–874

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Exchanging courses between different Intelligent Tutoring Systems: A generic course generation authoring tool

H. Escudero [a,*], R. Fuentes [b]

[a] Automatics and Computing, Public University of Navarre, Campus Arrosadia, 31006 Pamplona, Spain
[b] Mathematics and Computer Engineering, Public University of Navarre, Campus Arrosadia, 31006 Pamplona, Spain

ARTICLE INFO

Keywords:
Intelligent Tutoring System
Authoring tool
Generic
Hypermedia
Instructional design theories

ABSTRACT

In recent years a great effort has been made in order to create Intelligent Tutoring Systems that get close to human teaching. Some of the handicaps of the systems already created are the impossibility of sharing the courses between different Intelligent Tutoring Systems and the difficulty of creating them. Once the intelligent tutoring system is created, creating a new course is an expensive job that requires the intervention of many people that are expert in different areas. In this paper a generic and extensible authoring tool to create courses for different Intelligent Tutoring Systems is presented. This authoring tool allows the creation of courses for different types of intelligent tutoring systems. Once a course is created it can be exported to another intelligent tutoring system, reusing the domain model that the course represents. The prototype of the authoring tool has been tested with two simple Intelligent Tutoring Systems.

1. Introduction

Intelligent Tutoring Systems (ITS) are computer based systems that provide individualized tutoring to the students [1].

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 [2]. That is, the average tutored student performed as well as the top 2% of those receiving classroom instruction. Furthermore, research on prototype systems indicates that students taught by ITS 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 that 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, ITS enable participants to practice their skills by carrying out task within highly interactive learning environments. However, ITS go beyond training simulations by answering user questions and providing individualized guidance. Unlike other computer based training technologies, ITS assess each learner's actions within these interactive environments and develop a model of the knowledge, skills and expertise. Based on the learner model, ITS tailor instructional strategies, in terms of both content and style, and provide explanations, hints, examples, demonstrations and practice problems as needed [3].

However, ITS are still seen with scepticism due to the fact that they have not been extensively used in real educational settings. The main reason for this limited use 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 reused for different domains without expending 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 ITS [4].

The EON system is a suite of domain independent tools for authoring all aspects of a knowledge based tutor. It can be seen as the father of the ITS authoring tools that are created nowadays. Instead of authoring the ITS as a whole, it contains a set of tools to author different aspects of the ITS, namely the teaching strategies, the domain model, the student model and the learning environment [5].

In recent years several ITS authoring tools have been developed based on the paradigm of EON. VIRTOOL-D [6] is a virtual learning virtual environment for procedural domains. In this case, the tool is a machine-tool process learning environment. It contains a suite of

* Corresponding author.

tools to generate virtual experiences that recreate the cinematic model of a machine. XAIDA is another tool that creates courses for manufacturing engineering education [7], but this one is for instructional domains. This authoring tool contains several editors too in order to generate all the components necessary for the representation of the course. The outline editor creates a general view of the course. Then, the Resource Editor creates the resources that will be presented and the Fact Editor creates the information that will be used for questioning. Finally, built-in templates are modified using the Question Editor. TALHITS (Teaching and Learning by Hypermedia Intelligent Tutoring Systems) is another type of ITS. This system uses hypermedia to facilitate the authoring and learning of people with disabilities [8].

These systems present the same problems. First, the courses they create are not reusable by other ITS. All authoring tools are closed systems. This is, they have their own information representation formats, and specific teaching strategies. Second, even though the possibility of managing the teaching strategies gives more freedom and enhances the possibilities of the authoring tool, it can be difficult for a regular user to understand how they work. REDEEM is an ITS authoring tool that tries to solve this problem using a default teaching knowledge, which will be used for all the ITS. In this way, the user only has to specify the domain model and structure for the Shell to sequence and structure it [9].

Some systems try to decrease the amount of time needed to create new content, using new visual tools that abstract the content creation form programming. Murray [5] said that, after exploring several ITS, an amount of 300 hours of work was needed to create an hour of instruction. Bittencourt et al. [10] propose a computational model to make the development of semantic web-based educational systems (SWBES) easier and more useful for both developers and authors. The ASSISTment Builder [11], for example, is a tool that allows the creation, testing and reusing of the content of the tutor in an easy way. Using this tool, authors have proved that the ratio of development time to instruction time can be decreased to 40:1. Another example is the Internet ITS Authoring tool (IITSAT) [12]. This authoring tool is used to build tutors for military domains. As the domain has been specified, the authoring tool can create new tutors at an affordable cost. These tools reduce content generation costs, limiting the domain models that the ITS will be used for.

This paper presents a new structure for ITS authoring and an ITS course creation authoring tool independent from the ITS that will represent it. In this way, the courses that are created with the authoring tool will be reusable by any ITS.

This paper is organized as follows. Section 2 presents a new architecture for ITS. Section 3 proposes an architecture for the authoring tool that will create courses for any ITS. Section 4 presents the prototype implemented upon the architecture presented in Section 3. Section 5 presents two test cases using the prototype. Finally, Section 6 discusses the main conclusions and future work.

2. Proposed ITS architecture

Even if the course creation authoring tool is only for a part of the Intelligent Tutoring System, the creation of this authoring tool implies a change in the structure of the whole ITS. Usually an intelligent tutoring system is divided into four different components, as shown in Fig. 1: Domain Module, Pedagogic Module, Student Model and Dialog Module [13].

- *Domain Module:* This contains the knowledge about the subject to be taught. It usually is organized pedagogically to ease the interaction with the Pedagogical Module. The knowledge represented in this module is used to determine what must be presented to the student and as an standard to evaluate his/her answers.
- *Pedagogical Module:* The different teaching strategies are represented in this module. Session control methods are implemented by means of the adequate choosing and sequencing of these strategies. It promotes the learning, designing, adjusting and organizing of the instruction that will be presented for each student. This module is able to decide, using the information represented in the Domain Module and the Student Module, which concepts must be presented in each moment, how to present each concept, and when and how to interrupt the student.
- *Student Model:* This represents the image that the system has about the knowledge that the student has acquired during the instruction process. In addition, it incorporates other aspects about the behaviour and knowledge of the student that may affect the students learning. It is used to represent and evaluate the progress of the student acquiring the knowledge of the domain.
- *Dialog Module:* This defines the interface between the system and the user. Initially this module was not part of the architecture of the ITS, but later studies have shown its relevant role as the part of the system that communicates with the exterior. This component translates the interventions of the system in a representation that is legible for the user and transforms the entries of the student to the representation of the information that the system uses internally.

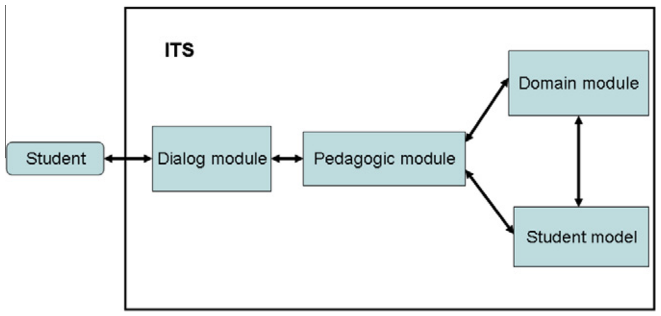


Fig. 1. Main components of an intelligent tutoring system.

ITS contain all the components in the same tool, and depending on the teaching strategies, the type of domain that is going to be taught and the student evaluation capabilities, these four components are subdivided into smaller ones. Each ITS defines its own sub-components and the way they communicate with each other.

All the ITS are closed systems. This means that all the courses they create can only be executed by they own tools. Normally the tool that creates the course is comprised in the tool that represents it, and the saving format is unknown. This makes difficult to compare the efficiency of different teaching strategies. In any case, all tools manage static and dynamic information. The static information is the knowledge about the subject to be taught and the basic information about the student. The dynamic information is the progress of the student and the punctual information that the system generates while the student is learning using the intelligent tutoring system, and this helps the ITS to choose the best teaching strategy for the next learning unit or exam.

The dynamic information is exclusive of each STI, as each one implements its own teaching strategies. But, if the creation of the domain module is correct, that information, described as static information before, could be shared between all the different ITS. It should be noticed that a domain model can be defined in different ways, and all of them can be valid.

This system proposes a generic architecture in which the tool that creates the course and the tool that represents it are separated. In this way, ITS are divided in two parts: there is an authoring tool that creates the static information and codifies it in an standard way (the domain model of the course to be taught and the information about the student), so any course representing tool can read it and use it as the domain to be taught, and there is a tool that represents that course.

Intelligent Nursing Education Software (INES) [14] is an ITS that uses an architecture similar to this one. In INES the core of the ITS is separated from the exercise modules and user interfaces. It is similar to our system because exercises can be provided in XML format to the ITS, and different user interfaces can be added to the system without changing the core. The main difference is that these XML files must be adapted to the format that the core of the ITS needs, whereas in our system this output can vary.

Cabada et al. [15] present another system that has a similar architecture. This is an ITS for mobile systems. The tool that creates the content can import SCORM, pdf, html or doc files into the system, and exports the course in XML format, so the interpreter, installed in the mobile system, can reproduce it. As courses are made with standard file formats, and it is exported in XML format, a parser could be created to transform that XML file into another one. Nevertheless, for the time being, these XML files can only be reproduced by the proprietary interpreter, and the authoring tool does not include any import/export facility. Some systems go further in the separation of the different modules of the ITS and propose a Service Oriented Architecture (SOA) to enhance the interactivity and accessibility of different learning resources [16].

An expert in the subject that is going to be taught would create the course. The course would be saved in a file and then the representing tool would use the file to start the teaching sessions.

As shown in Fig. 2, there are four different characters involved in an intelligent tutoring system:

- The expert in the subject to be taught. Each course should be created by an expert in the subject that is going to be taught. This person must use the course creation tool in order to create the course, but might not be used to working with computers. Therefore, the course creation tool must be as intuitive and easy to use as possible.
- The pedagogical expert. This person decides which rules must be applied for the student's learning to be as good as possible.

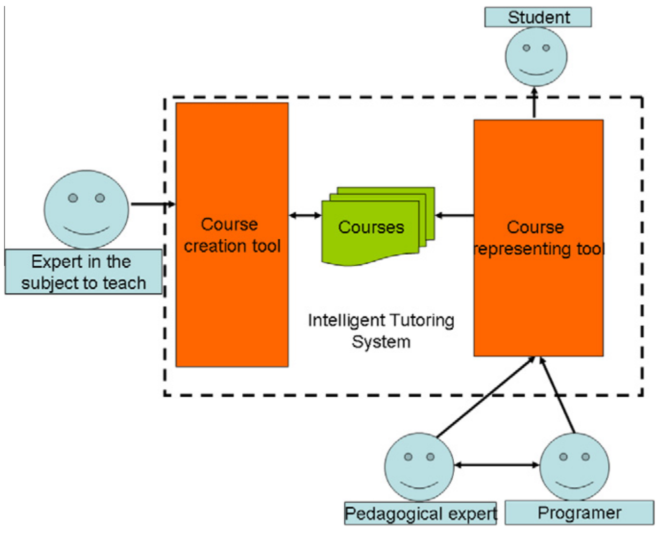


Fig. 2. The general structure of the intelligent tutoring system.

- The programmer. This person codifies the rules decided by the pedagogical expert into the course representing tool.
- The student. This person is the user of the course representing tool.

Nowadays, for each new course, the work of the programmer, the pedagogical expert and the expert in the subject to be taught are needed. Usually, this raises the cost of the creation of the new courses to an unaffordable level. With this new architecture, only the work of the expert in the subject to be taught is required.

Another advantage of this architecture is that it allows the constant recoding of the teaching strategy. In this way, in the initial state of the development of the course representation tool, the feedback of the student can be used to recode the teaching strategy, and the same course can be used to make another test with a different student.

3. Architecture of the generic course creation authoring tool

Different ITS use different teaching strategies. The authoring tool must support the creation of courses from the simplest to the more complex ones. The teaching strategy chosen for the authoring tool is the most complex one, but it's organization has been made flexible enough to allow the creation of simple courses. Instructional Design Theories study in detail knowledge representation paradigms from an educational perspective; they are primarily concerned with prescribing optimal methods of instruction to bring about desired changes in learner knowledge and skills. In particular, these optimal methods must specify what must be learned [17] and some way to represent this knowledge.

Instructional design theories based on Merrill's [18] 'Component Display Theory' present facts, concepts, procedures and principles as Basic Learning Units (BLUs). In order to establish a pedagogical view useful for selecting and/or sequencing the content, Reigeluth [19] references four different kinds of relationships between teaching contents of the same type: requisite relationships, conceptual relationships, procedural relationships and theoretical or principles-based relationships. So, the BLUs include in their representations some attributes which relate them one to another.

Instructional Objectives (IOs) refer to the application of particular skills over BLUs. The most accepted taxonomical classifications in the psycho-educational field have been the taxonomy of teaching objectives [2] and the taxonomy of learning objectives [20]. The former identifies three different learning categories: cognitive, affective, and psychomotor. Inside the cognitive category, six IO have been defined: knowledge, comprehension, application, analysis, synthesis and evaluation. Depending on the domain and the characteristics of the various learning activities proposed by the instructor, each BLU in the domain can be completed by adding the skills (i.e. IOs) that must be developed in the learner.

The authoring tool will use this generic architecture or information representation: there will be a set of BLUs which will be related. Each BLU will contain several IOs, and in each IO some information will be displayed.

In this section the architecture of the generic authoring tool will be presented. The system will be divided into five different modules: the ITS type specification module, the course flow editor module, the content creation module, the evaluation module and the output module.

The most generic definition of an authoring tool is "a system that allows the production and creation of complex and interrelated multimedia objects" [21]. Usually authoring tools have related the creation of web and multimedia content, but their use is widening. However, one of the characteristics that appears in all the definitions is that authoring tools help users that are not experts in computer based multimedia creation applications.

As mentioned before, one of the handicaps of ITS is the non reusability of the domain models created. As each ITS uses a different set of rules for deciding the flow of teaching and stores the domain and student models in different formats, the information sharing becomes impossible. The generic authoring tool proposed in this article is independent of the rules and the representation format of the ITS, and stores the information in a generic and open format, so it can be accessed by any application.

But even if the authoring tool is generic enough to allow the creation of domain and student models for any type of ITS, it would not be helpful if the number of persons involved in the creation of these models is still too large. This would lead to a very high cost in the creation of the new models, which is another of the problems that ITS are suffering nowadays. Therefore, the authoring tool must be as simple and intuitive as possible. In this way, only an expert in the subject to be taught will be needed in order to create the new domain model.

The independency of the domain model from the set of rules that decide the flow of the teaching process brings some other advantages. Firstly, it allows the comparison between different teaching ITS and different teaching strategies. As the same domain model can be applied to different ITS or teaching strategies, once the results of the learning process are evaluated, differences can be attributed to the teaching strategy. Secondly, if any of the teaching strategies proves to be better than the others, only the set of rules of the ITS that has performed worst needs to be changed, whereas the domain model remains the same.

It should be noticed that this generic authoring tool only creates domain and student models, and not the rules that are contained in the pedagogic module. Research has been done to create authoring tools that enable the creation of the whole ITS. We think that this objective is too ambitious. Furthermore, a pedagogical expert is needed in order to create an effective set of rules. The aim of the generic authoring tool focuses on the creation of the different courses once the rest of the ITS is constructed.

3.1. Specification of the tutoring system

As each ITS has different characteristics, first of all these characteristics must be defined. The system must provide a user friendly interface in order to define the characteristics of the tutor the course will be created for. These are the characteristics that need to be defined:

- Basic Learning Unit types. Several instructional design theories present the concepts, procedures and principles as Basic Learning Units (BLU). Not all the ITS use the same BLUs to specify their domains and control the flow of the BLUs.
- Relations between BLUs. Usually, the ITS have the number and type of relations that they can manage restricted. This is because the pedagogic module is prepared to process only some of them. And even if two ITS contain a pedagogic module with the same characteristic, they could have different names for the same relation types.
- Instructional Objectives. These are the different types of capabilities to achieve.
- Information types. This depends on the way that the ITS displays the information to the student. Some of them use simple text, others use web pages created from templates or human like avatars.

This information should be provided by the creators of the ITS and not by the user that is creating the course. However, an easy to understand interface should be provided. People with different knowledge about computers work in the creation of the ITS, and the tool should be designed so any of them can use the tutoring system specification tool.

The definition of the characteristics of the ITS is a very important part of the authoring tool, because the creation of the domain model will vary depending on these characteristics.

3.2. The course flow editor

An easy to understand way of codifying a domain model is a concept map. A definition of concept map, given by Martin [22] 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 [23].

Concept maps have their origin in a learning movement called Constructivism. Constructivism holds that prior knowledge is used as a framework to learn new knowledge. In essence, how we think influences how and what we learn. Concept maps can illustrate faulty views individuals may have and help us better understand how student may construe meanings from subject matter. Finally, the teacher who constructs concept maps for classes is interested in students understanding relationships between facts, not just knowing the facts [24]. Another advantage of concept graphs is that the knowledge bases that are expressed as concept graphs can be validated semantically [25].

In the generic authoring tool this approach will be taken for the creation of the domain model. The BLUs will be the nodes of the concept map, and the relations between the BLUs the relations of the concept map.

Therefore, the core of the course creation authoring tool should be a tool that helps creating the BLUs and their relations visually. Nowadays there are several tools that allow the creation of graphs in a user friendly way. Nonetheless, special attention must be paid to the special characteristics of this graph.

There are different types of links between the nodes. For example, a pre-requisite relation should not be represented in the same way as a co-requisite relation, because, even if the domain is well created, it would not be understandable. Some graph creation tools allow changing the colour of the links, but leaving this decision to the user could lead to confusion, as the user could forget to change the colour of the link. The best option would be the automatic

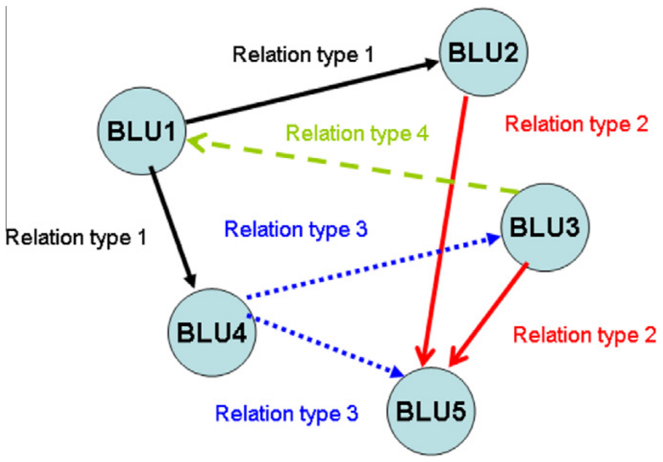


Fig. 3. Representation of the BLU net using a directed graph.

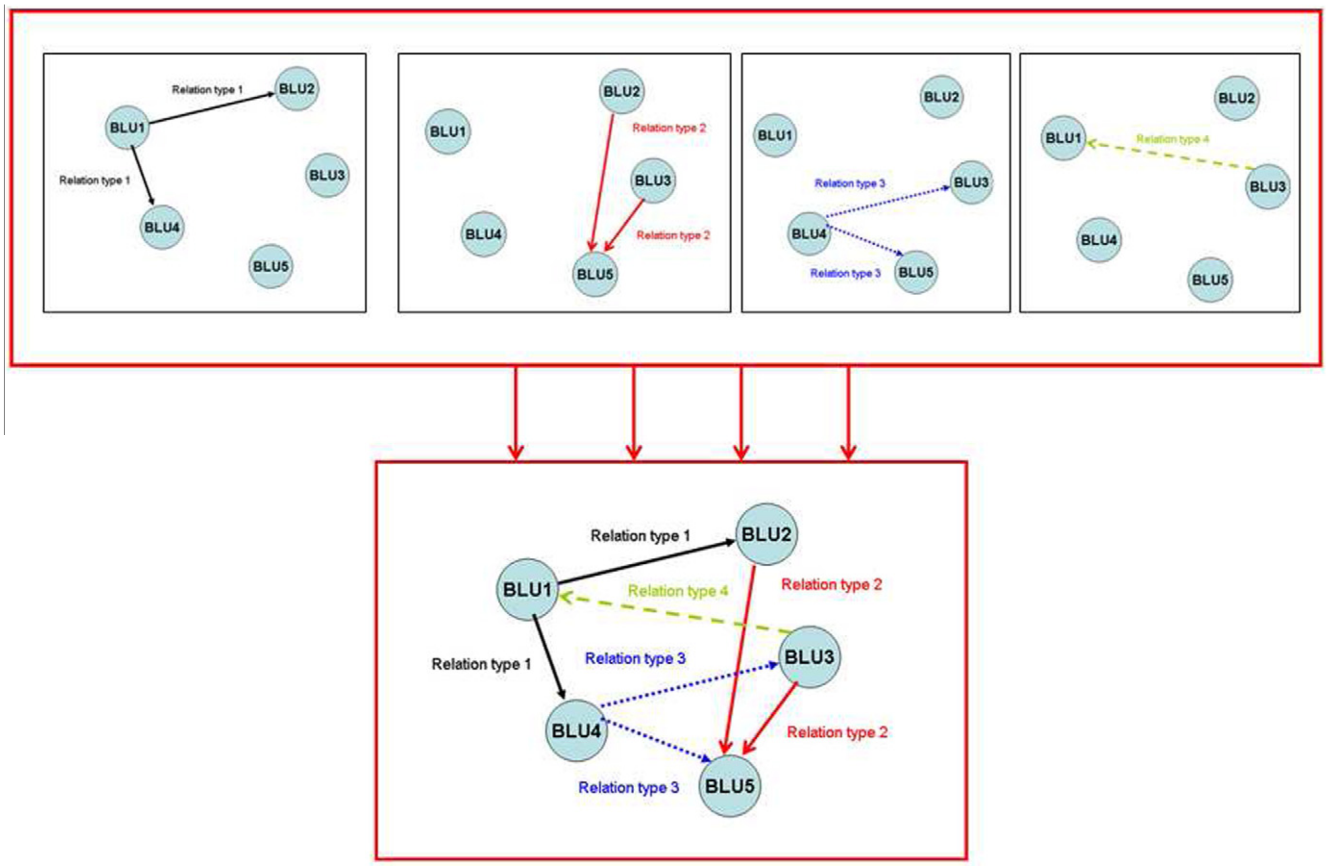


Fig. 4. Relations are crated one by one, and the representation of the whole graph.

creation of the link, which would be represented using a static colour and shape code (Fig. 3).

Another aspect to bear in mind is the great amount of BLUs and links that the course will have. While the graph grows it becomes increasingly difficult to take into account all the information presented in the graph. It would be easier to edit and visualize one type of relation at a time. The edition of one relation each time increases the difficulty of visualizing the hole domain. In order to solve this problem, a visualization tool is suggested. The user would create the graph editing one relation at a time, but with an option to watch the whole graph, even if it is not editable (see Fig. 4).

3.3. The BLU content creation

As shown in the section above, a course contains some BLUs related to each other. For each BLU, the Instructional Objectives (IO) must be defined. Depending on the content to be taught and the type of BLU, the IOs that form the BLU can change. Each IO contains its own specific information, which must be represented by the intelligent tutoring system when the system decides that it is the content that must be shown.

Traditionally, this content has been specified in forms that can be represented easily, as text or images. Sound and video are another possibilities, but having to record the content of each IO makes it expensive to use. Nowadays, a voice synthesizer could be used to create the voice from raw text.

In recent years avatars have come onto the scene as good information presenters. It seems that receiving the information from human like avatars makes us more receptive to that information. Initially, avatars had to be pre-rendered, and the cost of preparing a scene played by avatars was huge. As the processing capabilities of computers have increased, real time avatars have appeared. There are some platforms that, given a file as an input, can create a scene represented by avatars in real time. The CALMsystem is a conversational agent or "chatbot" that can be integrated into an ITS [26]. This chatbot encourages the students to discuss or reflect their knowledge so they can be helped to develop autonomy over their learning.

These days each intelligent tutoring system has its own information creation and representation form. This limits the platforms where the course can be represented. As every intelligent tutoring system has its own information representation system, the information generated by the authoring tool should be as generic as possible.

3.4. The evaluation form editor

In order to decide if the student has acquired the knowledge necessary to continue with the next lesson, the ITS must evaluate the student. Each ITS uses different ways to evaluate the knowledge of the student. Therefore, the evaluation generator must be flexible enough to allow the creation of different ways of evaluation.

The evaluation test could be outside the authoring tool, or even the ITS. Object Oriented Programming System (OOPS) is a problem-solving environment in which students can resolve Object Oriented Programming (OOP) exercises [27]. This system uses SIETTE [28], a web-based system assessment system where students can take tests to check their progress in their acquirements of knowledge.

3.5. The output language

The connection between the course creation tool and the course representation tool will be a file. As the output of the course creation tool could be used by several representation tools, this file must be as generic as possible. It must contain all the information created by the user in an organized way, so that the course representation tool that will use it can adapt it to its own representation form. Of course, this language must be open, that is, the structure must be public.

An XML based language is proposed for the generic authoring tool. In this way, the information will be clearly organized and structured. Furthermore, XML is very versatile. Creating a parser to translate this XML file to the internal representation form of the course representing tool should be easy.

4. Implementation of the tool

As explained above, the tool must contain a set of basic functionalities that can be extended in order to fulfill the requirements of any ITS. There are four main elements implemented for the core of the tool:

- The extensible core.
- The ITS specification tool.
- The course flow editor.
- The evaluation form editor.

The authoring tool has been implemented in Java for several reasons. Firstly, it is a multi-platform language. Secondly, there are many free applications that can be used for the core. And finally, creating a plug-in system is very easy in Java.

4.1. The extensible core

This is a fundamental part of the authoring tool. As explained above, the system cannot cover all possible representation and testing ways. As it is impossible to create editors for all the ITS that exist (and that will exist), a plug-in system has been created. The authoring tool has default information editors, but in case the information representation format of an ITS is not catered for by none of these editors, the authoring tool users can import their own information creation editors. These editors are imported as Java. jar files with classes that must implement some java interfaces already defined in the system. Three different plug-in types have been defined for the authoring tool:

- The information creation plug-in. These plug-ins are so the users can create the information that will be represented in each lesson of the course.
- The test creation plug-in. These plug-ins are so the users can create their own test.
- The save-as plug-in. The authoring tool has a specific information saving format. As each ITS needs a specific information format to reproduce the course, this information must be translated to another format. In this case the authoring tool offers two options. Firstly, a parser that takes the saving file of the authoring tool as input and transforms it into the desired format can be created. The saving format of the authoring tool will be public, so anyone can create a plug-in that translates from the format of the authoring tool into the format of it’s ITS. Secondly, users can use the internal saving structure to create a plug-in that saves the course directly into the desired format. Once the plug-in is imported, a “Save as” option will appear in the main menu of the authoring tool.

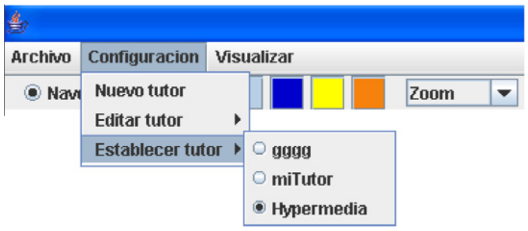


Fig. 5. ITS type menus.

4.2. The ITS specification tool

As each ITS has different characteristics, the user must define the characteristics of the ITS that the course will be created for. The authoring tool provides a tool to specify and record several configurations for different types of ITS. In the main application there is a menu (Fig. 5) where the user can create a new ITS type, edit an already existing ITS type and establish the active ITS type that will be used to create the next course.

When the user chooses to create or edit an ITS type, a new window appears where the user can define the characteristics of the ITS. This new window provides an easy interface in order to define the BLU types, the Instructional Objects, the relations between the BLUs and the way the information of the ITS course will be represented.

Regarding the BLUs, only their name need to be defined Fig. 6(a). For the way the course will represent the information, the system provides two alternatives, raw text and an avatar editor. In any case, there are ITS that use different type of information representation (like web pages, for example). For these cases, the system allows the inclusion of a jar file that will be automatically detected when left in a specific folder of the application. Furthermore, the user must specify the main class of the jar, so the tool can use it. As shown in Fig. 6(d), the user must insert a name for the information type and the path to the main class of the implementation. This class must implement two java interfaces so it can be properly activated: (see Fig. 1)

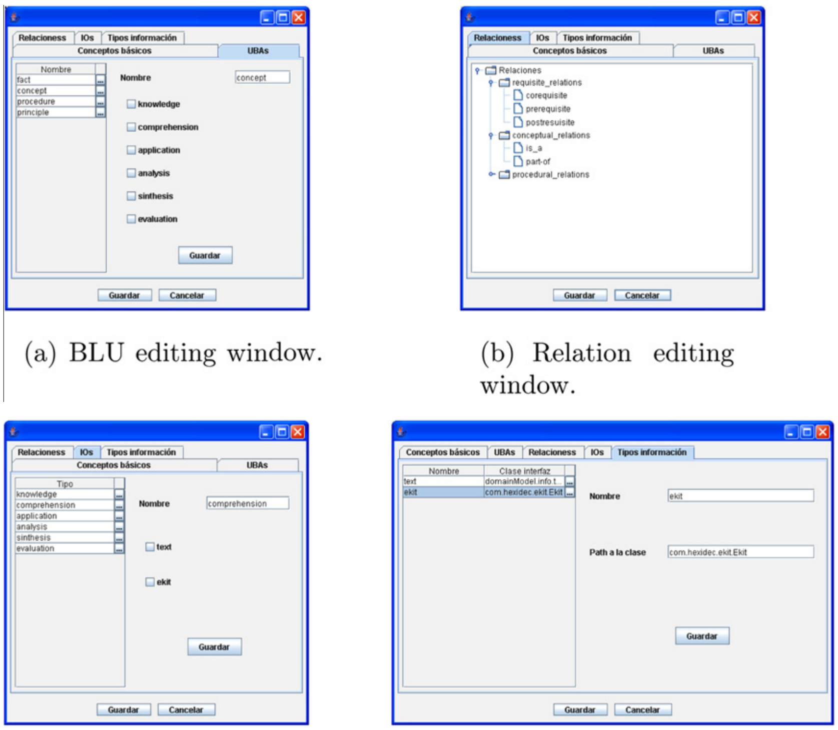
The Info interface is the class that will contain the information and the InfoInterface interface is the class that will provide the visual interface to insert that information.

Once the information types are defined, the user can specify the Instructional Object types. As IOs contain the information that will be presented, the course creator must define which will be the representation format for the IOs of the ITS Fig. 6 c. In addition, there is usually some information added to the IOs. This information helps tracking the learner’s progress. The difficulty of the IO, the expected learning time or the number of times the user has seen the IO are examples of these attributes. The more complex the ITS, the more attributes have the IOs. As every ITS has different attributes, the authoring tool has an attribute editor Fig. 7. There are four basic attributes: text attribute, check box attribute, combo attribute and list attribute. These attributes can be combined to create more complex attributes. Once the desired attributes are created, these can be inserted in the IOs.

Finally, the relation types between the BLUs must be defined. As these relations could be hierarchical, the system provides a tree where the user can include nodes just by clicking with the right button of the mouse.

4.3. The course flow editor

As the model will be represented as a concept map, the best way to create and represent it is a graph editor. Nowadays there is a great variety of tools that allow the creation and manipulation



(a) BLU editing window. (b) Relation editing window. (c) IO editing window. (d) Information types editing window.

Fig. 6. ITS type editing windows.

```
 public interface Info {
        public String saveInfo();
        public void loadInfo();
}

public interface InfoInterface {
        public void addInfoListener(InfoListener infoListener);
        public void setVisible(boolean value);
        public void setInfo(Info info);
}
```

Listing 1. The two interfaces that must be implemented.

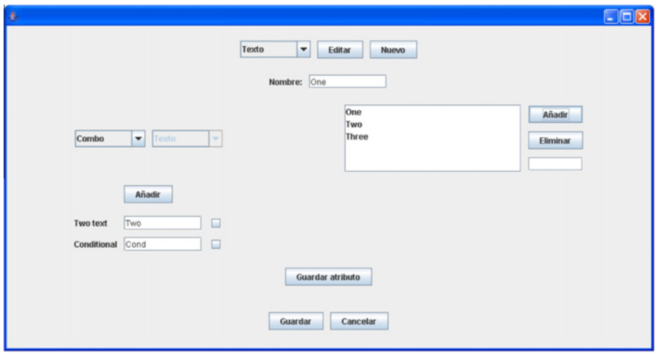


Fig. 7. The attribute editor.

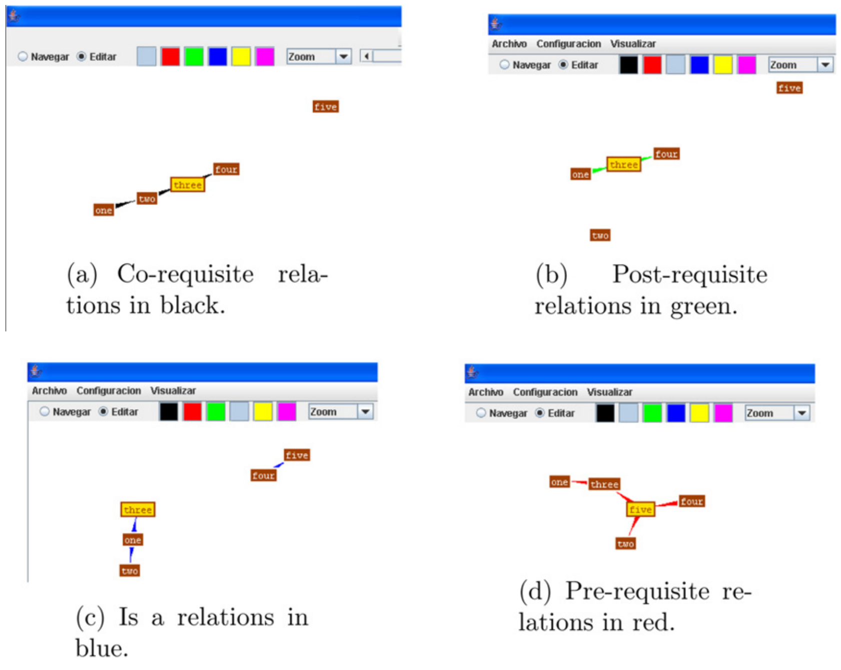
of graphs in an easy and visual way. InXight [29], TheBrain [30] or ThinkMap [31] are professional tools that use graphs to visualize different complex relations between concepts.

One of the most used graph manipulation tool written in java is GraphViz [32]. This library allows the creation of different types of graphs. The handicap of the library is that due to its complexity the manipulation of the source code in order to satisfy the needs of new applications becomes extremely difficult.

TouchGraph [33] is an alternative to GraphViz. This library allows the creation of different types of graphs, but its advantage is that taking the basic library of TouchGraph the LinkBrowser has been created. The LinkBrowser only allows the creation of nodes and directional links using an user friendly interface of drag&drop. Additionally, the user can zoom in and out, rotate the graph and move it.

On each node of the graph the user must add the content that will be displayed in order to teach a piece of the course. When editing the properties of the nodes, besides their aspect the user can define the type of BLU (which will vary depending on the tutor type chosen) and its content.

Depending on the configuration of the ITS, the user will have different link types available, each one identified by a colour. For



(a) Co-requisite relations in black.

(b) Post-requisite relations in green.

(c) Is a relations in blue.

(d) Pre-requisite relations in red.

Fig. 8. Four different relations edited in the same model.

the sake of clarity, only one will be visible each time. In Fig. 8 four different relation types have been edited in the same model. When the user pushes one of the coloured buttons in the top of the editor, the selected relations appear.

But, as explained above, the user must be able to see the model as a whole, and not as a group of graphs. Showing all the relations in a 2D graph would make the model difficult to understand. This is the reason why a 3D graph representation tool like WilmaScope [34] has been chosen to represent the whole domain model (see Fig. 9).

4.4. The evaluation form editor

Most ITS must evaluate the student in order to decide if he/she can continue and the lesson that should be learned next. The easiest way to evaluate a student using a computer is a multiple choice test. The system provides a tool to create these tests.

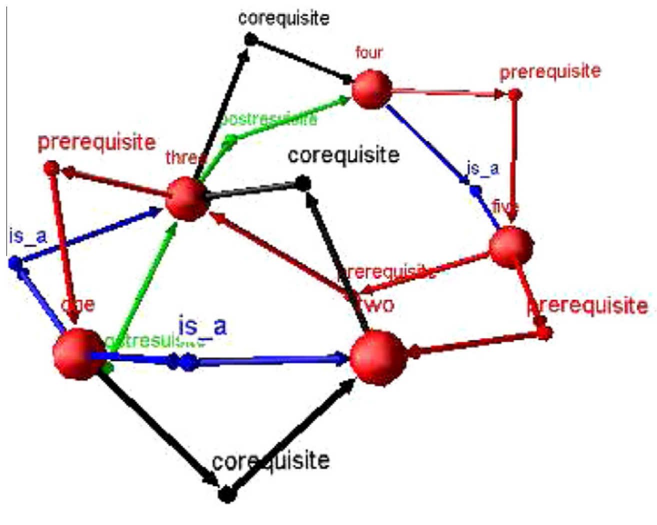


Fig. 9. The whole domain model in 3D.

Nevertheless, as computer technologies evolve, multimedia systems where the user has to interact with the computer in order to pass a test are becoming more common. The authoring tool must be able to import and use any of these multimedia systems, or at least, define the necessary parameters so that the representation system can decide if the student has passed or failed the test, and record what type of mistakes he/she has made in order to create appropriate learning strategies for future lessons.

4.5. Saving and loading the course

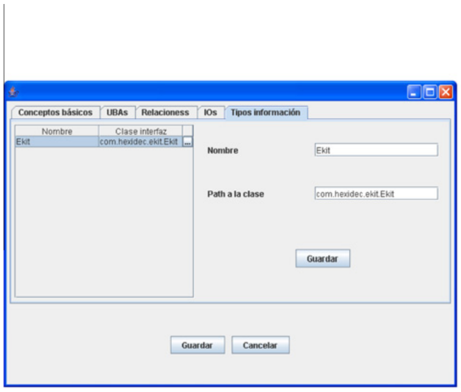
As previously explained, the courses will be recorded in XML format. The specification of the language will be accessible for everybody, so anyone can create a plug-in to record the information in a format that the course representation tool can understand. These plug-ins can be imported into the authoring tool and will be used in a "Save as" way. When the user wants to record the information for the course representation tool, first the course will be recorded in the XML language of the authoring tool, and then converted into the format of the course representation tool using the plug-in.

5. Two simple test cases: a simple online course and a course for an hypermedia system

In this chapter the first tests of the implementation of the tool will be explained. First, the authoring tool will be used to create the simplest ITS, an online course. In this example each BLU will be a web page. Then, the authoring tool will be used to create an Adaptive Hypermedia System. The files generated by the authoring tool can be seen in http://www.unavarra.es/personal/HectorEscudero/english/.

5.1. Creating an online e-learning course

There are two steps to follow to create a course, definition of the characteristics of the ITS and the creation of the course itself.

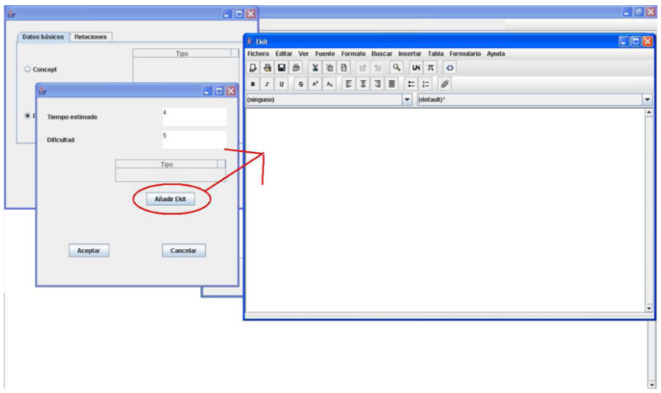


Nombre	Clase interfaz
Ekit	com.hexidec.ekit.Ekit

Nombre: Ekit

Path a la clase: com.hexidec.ekit.Ekit

(a) Specification of the Ekit information type.



Tiempo estimado: 4

Dificultad: 5

(b) As the Ekit information type plug-in has been added into the system, it can be chosen form the authoring tool.

Fig. 10. Adding a plug-in into the system

5.1.1. *Definition of the characteristics of the ITS*

The first step is to think of the characteristics that the final course will have. As it will be an online course, it will be just a set of web pages. The relations between the pages are implicitly defined in the html code, as it is the way to jump from one page to another. This way, each web page can be seen as a BLU. The content of the BLU must be created in HTML code. The information creation tools that the authoring tool has by default cannot create HTML code, so a tool to create web pages must be imported into the tool.

Therefore, in the ITS definition tool, it must be specified that there is only one type of BLU, and that there is no explicit relation. For the information of each page a plug-in must be created. Ekit is a free source Java HTML editor application. It cannot be imported into the authoring tool as it is. As explained above, it must implement two interfaces. The main class of the Ekit application has been modified to implement the InfoInterface java interface (provided by the authoring tool). This way, the Ekit tool can be launched from the authoring tool. Then, when clicking the "Save" button of the Ekit tool, the saving class has been adapted so it implements the Info Java interface, and it returns the information in an appropriate format.

In Fig. 10(a), the importation of the Ekit plug-in into the system is shown. A name for the information type and the path to the class that has implemented the InfoInterface (com.hexidec.ekit.Ekit in this case) are required. Once the plug-in has been imported into the system, the user can specify the information of a BLU using that information type. In the editing options of the BLU a button will appear with the name of the information type, and when you click that button the Ekit HTML editing tool appears (Fig. 10(b)).

Finally, as the result of the course must be a set of web pages, in other words, some HTML files, a converter must be imported into the system. This translator takes the generic saving format of the authoring tool and creates the HTML files. Each BLU contains HTML code to be shown. Therefore, the translator has to create one file containing the HTML code for each BLU.

5.1.2. *The creation of the course*

Once the ITS characteristics have been defined, the course can be created. Each node in the graph will be an HTML page, which will be created with the Ekit editor. In this case there will be no relation between the nodes, as all the relations between pages are embedded in the HTML code.

5.1.3. *Saving and loading the course*

The output of the course must be a set of HTML files. By default the authoring tool saves the course in a generic format, but for this case a "Save as" like plug-in has been imported. This plug-in creates a file for each node of the graph (BLU) and records in that file the information created with the Ekit editor.

5.2. *Creating a course for a hypermedia system*

Bruselovsky defines Adaptive Hypermedia Systems (AHS) as systems that build model of the goals, preferences and knowledge of each individual user, and use this model throughout the interaction with the user, in order to adapt to the needs of that user [35]. The information structure of a typical Adaptive Hypermedia System can be considered as two interconnected networks of "spaces": a network of concepts (knowledge space) and a network of hypertext pages with educational material (traditional hypertext) [36] (Fig. 11).

As noted above, there are two steps to follow to create a course, definition of the characteristics of the ITS and the creation of the course itself.

5.2.1. *Define the characteristics of the ITS*

In AHSs there are two spaces, a network of concepts and a network of hypertext pages. For the hypertext pages, there is no need to specify the relations between them, as they are on the pages as

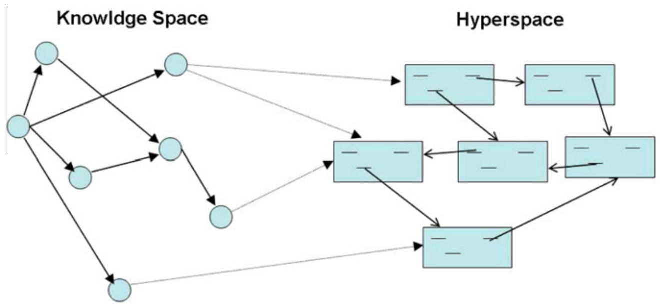
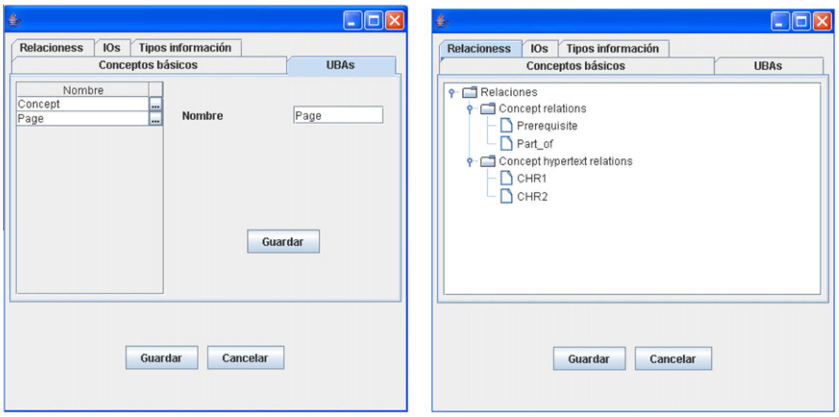


Fig. 11. A typical structure of information space in an adaptive hypermedia system. There is a relation between concepts (circles in the knowledge space) and HTML pages (rectangles in the hyperspace).



(a) BLU configuration. (b) Relation configuration.

Fig. 12. An example of the configuration of an Adaptive Hypermedia System.

hyperlinks. But there can be relations between different concepts and between the concepts and the hypertext pages.

In Fig. 12 the tutor configuration window is shown. The (a) sub-figure shows the BLU configuration window. As there will be two different types of nodes in the course, concepts and hypertext pages, two different BLU types have been defined, Concept and Page. The (b) sub-figure shows the different types of relations. Relations can be added hierarchically or all of them on the same level. For this example, four different types of relations will be created, two for the relations between concepts and another two for the relations between concepts and hypertext pages.

The content of each BLU will be a web page. The plug-in inserted into the system to enable the creation of web pages for the e-learning ITS can be reused for that purpose. But the output will not be just a set of HTML files. In addition to the files, the relations between concepts and the web pages must be codified. This means that another translator must created and imported into the system.

5.2.2. The creation of the course

Once the ITS characteristics have been defined, the course can be created. Each node will contain the information that will be represented to the user (XHTML page created with Ekit), and some attributes that the AHA! ITS needs in order to control the flow of the course. There are two types of relations in this course. The relations embedded in the XHTML code as hyperlinks, and the relations between the nodes (BLUs) that represent concepts of the course. Each node will contain an XHTML page.

5.2.3. Saving and loading the course

There are several files in the output of an AHA! course. First there is a set of XHTML files that will be used to represent the information that the student has to learn. Second, there are two configuration files that record the information of the relations between the BLUs and their attributes. This information is used to decide when has the student learnt enough to go onto the next step. By default the authoring tool saves the course in a generic format, but for this case a "Save as" like plug-in has been imported. For the XHTML files, this plug-in creates a file for each node of the graph (BLU) and records into that file the information created with the Ekit editor. Then two more files are created. A .gaf file that records the graphical information of the concepts of the course, and an .aha file, that records the information about the attributes of the BLUs and their relations.

6. Conclusion and future work

In this article an architecture for a generic authoring tool that creates courses for intelligent tutoring system has been presented. Furthermore, the first steps of the implementation of the core tool have been shown. 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. It has been proven, that with the technologies that exist nowadays, such a generic system can be built. Java offers the opportunity to import any type of plugging into the system in a easy way. The challenge has been to define a core that can support a set of rules of the most complex and the most simple ITS, without loosing usability.

Nevertheless, there is a set of ITS that might get out of the range of the authoring tool. These are the simulation ITS. These tools build the ITS upon 3D machines, and they usually do not have a set of teaching rules. All that the learner must do is to practice until he/she has reached a certain amount of familiarity with the system.

The next steps will be to complete the core of the authoring tool and to test it in different scenarios. Once the two basic tests mentioned above have been made, a more ambitious test is foreseen. The authoring tool will be used to create courses for the IRIS intelligent tutoring system [37]. Furthermore, the XML output language will be enhanced, probably using ontologies, so we can make the most of the knowledge that the content creators produce.

References

[1] S.D. Schoksey, Developing an Affordable Authoring Tool for Intelligent Tutoring Systems, Master's Thesis, Worcester Polytechnic Institute, 2004.
[2] B. Bloom, M. Engelhart, E. Murst, W. Hill, D. Drathwohl, Taxonomy of Educational Objectives: Handbook I, Cognitive Domain, Longman, 1956.
[3] J. Ong, S. Ramachandran, Intelligent tutoring systems: using ai to improve training performance and roi, Networker Newsletter 19 (6) (2003).
[4] M. Moundridou, M. Virvou, Wear: a web-based authoring tool for building intelligent tutoring systems, in: Proceedings of the 2nd Helenic Conference on AI, Companion volume, Thessaloniki, Greece, 2002, pp. 203–214.
[5] T. Murray, S. Blessing, S. Ainsworth, Authoring Tools for Advanced Technology Learning Environments. Towards Cost-effective Adaptive, Interactive and Intelligent Educational Software, Kluwer Academic Publishers, 2003.
[6] A. Lozano, L. Matey, M. Urretavizcaya, B. Ferrero, I.F. de Castro, Virtool-d: entorno virtual educativo para aprendizaje en dominios procedimentales, in: Proceedings of the SIIE 2003, International Symposium in Education Computing, 2003.
[7] S. jen Hsieh, P.Y. Hsieh, Intelligent tutoring system authoring tool for manufacturing engineering education, International Journal of Engineering Education 17 (6) (2001) 569–579.

[8] S. Talhi, M.D. an Salima Ouadfel, S. Zidar, Authoring intelligent tutoring systems for disabled learners, in: Proceedings of the International Conference on Information and Communication Technology & Accessibility, 2007.
[9] S. Ainsworth, B. Williams, D. Wood, Using redeem its authoring environment in naval training, in: Proceedings of the IEEE International Conference on Advanced Learning Technologies, 2001, pp. 189–192.
[10] I.I. Bittencourt, E. Costa, M. Silva, E. Soares, A computational model for developing semantic web-based educational systems, Knowledge Based Systems 22 (4) (2009) 302–315.
[11] L. Razzaq, J. Patvarczki, S.F. Almeida, M. Vartak, M. Feng, N.T. Heffernan, K.R. Koedinger, The assistment builder: supporting the life cycle of tutoring system content creation, IEEE Transactions on Learning Technologies 2 (2) (2009) 157–166.
[12] D. Stottler, D. Fu, S. Ramachandran, T. Jackson, Applying a generic intelligent tutoring system (its) authoring tool to specific military domains, in: The Interservice/Industry Training, Simulation & Education Conference, 2001.
[13] E. Wenger, Artificial Intelligence and Tutoring Systems, Morgan Kaufman Publishers Inc., San Francisco, CA, USA, 1987.
[14] M. Hospers, E. Kroezen, A. Nijholt, R. op den Akker, D. Heylen, Developing a generic agent-based intelligent tutoring system (pdf), in: Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies, 2003, p. 443.
[15] R.Z. Cabada, M.L.B. Estrada, E.U. Barrientos, M.O. Velásquez, C.A.R. García, Multiple intelligence tutoring systems for mobile learners, in: ICALT '08: Proceedings of the 2008 Eighth IEEE International Conference on Advanced Learning Technologies, IEEE Computer Society, Washington, DC, USA, 2008, pp. 652–653.
[16] C. Fang Fang, L. Chien Sing, Collaborative learning using service-oriented architecture: a framework design, Knowledge Based Systems 22 (4) (2009) 271–274.
[17] J. Scandura, Instructional-Design Theories and Models: An overview of their current status, Lawrence Erlbaum Associates, 1983 (Chapter: Instructional Strategies Based on the Structural Learning Theory, pp. 213–246).
[18] M. Merrill, Instructional-Design Theories and Models: an overview of their current status, Lawrence Erlbaum Associated, 1983 (Chapter: Component Display Theory, pp. 279–333).
[19] C.M. Reigeluth, M.D. Merrill, C.V. Bunderson, The structure of subject matter content and its instructional design implications, Instructional Science 7 (1978) 107–126.
[20] R.M. Gagné, L.J. Briggs, W.W. Wager, Principles of Instructional Design, Holt, Rinehart and Winston, 1988.
[21] D.C. Bulterman, L. Hardman, Multimedia authoring tools: state of the art and research challenges, LNCS 1000 (1995) 1–17.
[22] D.J. Martin, The name assigned to the document by the author. this field may also contain sub-titles, series names, and report numbers.concept mapping as an aid to lesson planning: a longitudinal study, Journal of Elementary Science Education 6 (2) (1994) 11–30.
[23] J. Lanzing, The concept mapping homepage, 1997, <http://users.edte.utwente.nl/lanzing/cm_home.htm>.
[24] U. of Tennessee at Chattanooga, Concept mapping and curriculum design, 2002, <http://www.utc.edu/Administration/WalkerTeachingResourceCenter/FacultyDevelopment/ConceptMapping/>.
[25] J. Dibie-Barthélemy, O. Haemmerlé, E. Salvat, A semantic validation of conceptual graphs, Knowledge Based Systems 19 (7) (2006) 498–510.
[26] A. Kerly, R. Ellis, S. Bull, Calmsystem: a conversational agent for learner modelling, Knowledge Based Systems 21 (3) (2008) 238–246.
[27] J. Gálvez, E. Guzmán, R. Conejo, A blended e-learning experience in a course of object oriented programming fundamentals, Knowledge Based Systems 22 (4) (2009) 279–286.
[28] R. Conejo, E. Guzmán, E. Millán, M. Trella, J.L. Pérez-De-La-Cruz, A. Ŕıos, Siette: a web-based tool for adaptive testing, International Journal of Artificial Intelligence on Ediucation 14 (1) (2004) 29–61.
[29] S. Inxight, Inxight, August 2005, <http://www.inxight.com/>.
[30] T.C. TheBrain, Thebrain, August 2005, <http://www.thebrain.com/>.
[31] I. Thinkmap, Thinkmap, August 2005, <http://www.thinkmap.com/>.
[32] E.R. Gansner, S.C. North, An open graph visualization system and its applications to software engineering, Software—Practice and Experience 30 (11) (2000) 1203–1233.
[33] L. TouchGraph, Touchgraph, 2005, <http://touchgraph.sourceforge.net/>.
[34] T. Dwyer, Wilmascope, 2007, <http://wilma.sourceforge.net>.
[35] P. Brusilovsky, Methods and techniques of adaptive hypermedia, User Modeling and User-Adapted Interaction 4 (1) (1996) 1–19.
[36] P. Brusilovsky, Developing Adaptive Educational Hypermedia Systems, From Design models to Authoring Tools, Kluwer Academic Publishers, 2003 (Chapter 13).
[37] A. Arruarte, I. Fernandez-Castro, B. Ferrero, J. Greer, The iris shell, how to build itss from pedagogical and design requisites, International Journal of Artificial Intelligence in Education (8) (1997) 341–381.