EDU-EX: A Tool for Auto-Regulated Intelligent Tutoring Systems Development Based on Models

P. DOMINGO, A. GARCÍA-CRESPO, B. RUIZ & A. IGLESIAS
Computer Science Department, Universidad Carlos III, Avenida de la Universidad, 30. 28911, Leganés (Madrid) Spain (E-mail: pdgar@ia.uc3m.es)

Abstract. In recent years there has been an upsurge in forms of instruction that envisage a permanent and ongoing involvement in education of novel concepts such as planned and personalised instruction and autonomous learning. A large number of problems that arise in education today may be solved by introducing new technologies into the educational environment, as they allow the form and content of tutoring systems to be tailored to each individual. The application of Artificial Intelligence techniques is helping open up new prospects in the field of teaching and learning. Using Artificial Intelligence techniques in education has the advantage of making it possible to represent expert reasoning and knowledge skills, and to take advantage of this experience in education.

This study has involved the development of a tool to generate auto-regulated intelligent tutoring systems based on models. This form of representation makes it possible to break down, organise and represent information so as to enable the easy creation of functional intelligent computerised tutoring systems. Information about the subject in question, about inference mechanisms, and of a pedagogical nature (independent of any one strategy) is all separated. The tool also enables knowledge acquired by a student to be constantly monitored with a view to auto-regulating the course contents.

Keywords: Artificial Intelligence, authoring tools and methods, autonomous learning, expert systems, intelligent tutoring systems

1. Introduction

The aims of education today could be summed up as: to train individuals to educate themselves, to teach them how to learn and about innovation and creativity, to train them in methods of learning and research, and finally, to develop the motivation to pursue a constant personal enrichment. New forms of education need to be based on relatively short teaching time and extensive and ongoing training. From this point of view, there are currently two clear tendencies in education: the first entails a move away from the transmission of knowledge towards the organisation of students' learning, which requires developing their capacity to teach themselves with the help of the appropriate media. The second tendency is an increase in the use of technology in modern education. This study bears upon the second point by providing a

way to improve the use of technology, which in turn contributes to the first point.

A key development in this field in recent decades has been that of systems that enable teaching methods to be adapted to the characteristics of the student: learning speeds and ratios, psychological features, form and modularity of course contents suitable for each kind of student, etc.

2. Intelligent Tutors

Intelligent tutoring systems are one of the most important tools available for autonomous learning and, moreover, they enable teaching to be personalised. Developing intelligent tutors involves many problems: they are expensive to develop; there are no simple tools to enable non-computing professionals to develop them, and those that there are clearly depend on the subject in question; intelligent tutoring systems are, because of the way they are developed, quite rigid and not easily modifiable, etc.

Intelligent tutoring systems seek to reflect a method of teaching and learning based on one-to-one interaction between student and teacher. For researchers into artificial intelligence, this is the most suitable kind of teaching to tackle initially. Methods of instruction and practice in one-to-one tutoring are the best-understood ways of transmitting knowledge. This method of teaching and learning is moreover widely accepted both in the educational community and in Western culture. This wide popularity is for a good reason: one-to-one education is a highly individual process, so its results are better than with other teaching methods (Bloom 1995).

3. Generic Architecture of an Intelligent Tutoring System

Intelligent tutoring systems separate teaching strategies aimed at the student from information on the subject in question and from the teaching plan. Intelligent tutors have traditionally been based on three models which interact dynamically: the subject model, the student model and the pedagogical model (OREY94). These aims and characteristics are realised by means of an architecture distributed among several modules, as shown in Figure 1.

4. State of the Art

An ITS authoring tool is a generalised framework for building ITSs with a user interface that allows non-programmers to formalise and visualise their

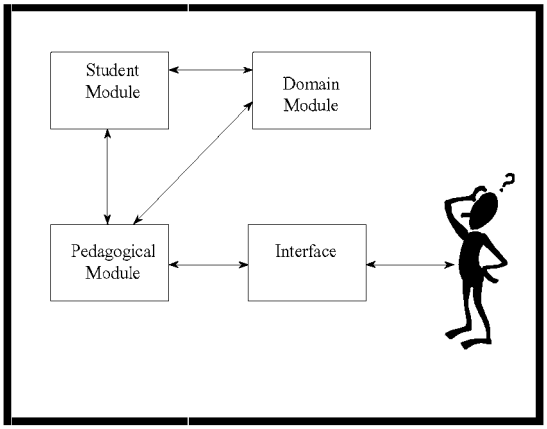

Figure 1. Architecture of an intelligent tutoring system.

knowledge. ITS authoring tools have been used to build tutors in a wide range of domains and targeted toward a wide range of students, however, the key differences among ITS authoring systems are not related to specific domains or student populations, but to the domain-independent capabilities that have.

These systems can be classified according to the type of ITSs they produce, the types of domains and tasks they are suited for, the degree to which they make authoring more easy or efficient, and the depth and fidelity employed to represent the knowledge or skill being taught (Murray 1999).

Some systems, at the *Curriculum and Course Sequencing* category, have been built to help instructional designers and teachers design to organise instructional units into a hierarchy of courses, module, lessons, presentation, etc. with relationships between them. The sequencing of the content is being determined dynamically based on student performance, lesson goals and relationship between course modules, when the depth of analysis and feedback in tutors built is limited according to the domain knowledge representation. These systems are more appropriated to teach conceptual and declarative knowledge than procedural or problem solving knowledge. DOCENT (Winne 1991, 1988), IDE (Russell 1988), ISD Expert (Merrill 1998) and Expert-CML (Jones 1991) are some examples of these authoring tools. There exist systems that permit to handle different types of knowledge, defining pedagogic strategies depending on this knowledge, for example CREAM-Tools (Nkambou 1996), DNA (Shute 1998), ID-Expert (Merrill 1998), IRIS (Arruarte 1997) and XAIDA (Hsieh 1999; Wenzel 1998) systems.

Others systems, focused on tutoring strategies, encode fine-grained strategies used by teachers and instructional experts and permit instructional

decisions at a micro level. For example, Eon (Murray 1998), GTE (Van Marcke 1998) and REDEEM (Major 1997) systems.

Demonstr8 (Blessing 1997; Anderson 1991), D3 Trainer (Reinhardt 1995), Training Express (Clancey 1988) are focused at expert systems and cognitive tutors, observing student behaviour and building a fine-grained cognitive student model that can be compared with expert system.

Others systems have special purpose, specialised in particular tasks or domains. For example, systems that simulate devices or train equipment, like XAIDA (Wenzel 1998) system.

Concerning tasks that they can accomplish, at the interface model, the vast majority of authoring systems assure reasonable interface designs simply by pre-defining the student interface, but some of them allow authors to construct the tutoring system's interface from scratch (Murray 1999), for example Eon system. At the domain model, some tools are limited to strict hierarchical representations of the curriculum topics, but other systems include tools for visualising and authoring content objects networks (IDE, Eon, CREAM-Tools, etc.), allowing authors visualise the relationships between curriculum elements and the subject matter. These systems permit include knowledge about the pedagogically relevant properties of topics (importance, difficulty, depth, rules, prerequisites, etc.). The product presented in this article, EDU-EX, has been designed to, dinamically adapt in real time, the content network to the particular profile of the student. The Content Network is not rigid in nature, it works in a way that according to the parameter values captured during the overall student learning process, dinamically changes it's own data structures to achieve maximum efficiency and capacity to adapt itself to the particular student'sKnowledge scenario in execution time.

At the tutoring model have been used a great variety of representational methods to model tutoring expertise (procedures, plans, constraints, rules, etc.), but the vast majority of ITS authoring systems include a fixed and non-authorable tutoring model. For example, Eon uses a rule-based representational method with pull-down menus, REDEEM has a fixed rule set defining the pedagogical behavior, but authors can define new strategies, IDE can justify by a specific theory each planning rule and GTE permits authors to type in plan rules that define a hierarchy of sub-tasks.

At the student model, the vast majority of the ITS authoring tools use overlay student model, but they have been including artificial intelligence techniques, such as fuzzy logic (Goodkovsky 1994), Bayesian Networks (Collins 1996), Neural Networks, etc. Eon system allows student model to be authored. The EDU-EX Tool also includes a software module that provides capabilities for the Developer to define any specific student environement required for this purpose. It is possible to select the most addecuate functional

characteristics to particular student's Knowledge situations, keeping track of all performed iteractions.

An interesting and differential characteristic of this Tool, EDU-EX, is it's visual orientation basis, thus allowing fast and convenient use even by profesionnals with no specific IT background or experience.

5. EDU-EX: A Tool for Generating Intelligent Tutoring Systems

This study has involved developing a model-based system to tackle the problem of representation by breaking down, organising and representing information, and creating functional intelligent computerised tutoring systems. Information about the subject in question, about inference mechanisms, and of a pedagogical nature (independent of any one strategy) is all separated. Information about the subject is later broken down into the categories of system tasks and error detection. EDU-EX is composed of several models that include every task or activity in the learning process: areas to be taught, pedagogical decisions to be taken, evaluations to be performed including pedagogical strategies to be taken, etc. Each model also contains rules that allows to adapt the course to the knowledge level acquired for the student. Objects were developed to model the curricular contents to be transmitted to the student, and further objects can both model the capacity of the student on the basis of ontological characteristics and progressively monitor his or her interaction with the system. Finally, models were generated to make it possible to embrace a multiplicity of teaching strategies. Continuous assessment of interaction with the user enables the system to be steadily adapted to the student by progressively modifying these multiple teaching strategies in line with the student's response to the system.

The first step in the analysis of requirements for a tool to generate Auto-Regulated Intelligent Tutoring Systems (henceforth EDU-EX) is to create *object models*. Object models show the structure of data from the real-world system organised into parts with which the tutoring system may be generated. The information needed to create the objects comes from the requirements of the system, from expert knowledge of the subject, the application and the teaching method, and from general knowledge of the real world. The basic principle is to create a representative model of information based on objects, separating inference mechanisms that will bear upon what is specifically the student's work. Control is external and corresponds to pedagogical criteria defined by the author and governed by the system (pedagogical strategy, teaching style, etc.).

Once the Knowledge Base has being built, the planner is the part of the tool that perform the reasoning process. It processes information in the

Knowledge Base by taking decisions and then managing them on the basis of information contained in the various objects and their properties.

5.1. *EDU-EX knowledge base: Objects and properties*

The representation and management of knowledge has been a basic issue since the birth of artificial intelligence. The purpose of representing knowledge is to organise information so that it may be used in reasoning processes.

EDU-EX models are boxes of objects; they do not, as in classic object orientation, have associated methods applied upon them and executable by them. Thus a state is defined by associating a list of properties with each object, i.e. a list of all properties of an object relevant to a description of its state. The state, and therefore the values of an object's properties, may be updated when the situation changes. The relationship between these objects is via their properties, which cause changes in the way the system operates and describe the states through which the system passes.

Object properties contain certain information about how to ask the student a question or how to carry out an action, about test questions, etc. Each type of object has its own set of properties which make it different from any other. Properties and their values have various purposes, such as:

- Denoting a link, e.g. the *Contains_Subareas* property of the AREA object.
- Denoting links between an area and pedagogical decisions, e.g. the *Decision* property of the AREA object.
- Containing the text that the Planner shows the student when asking questions, giving instructions to carry out an action, etc., as happens with the *Text* property of the DEMONSTRATION object.
- Containing vital instructions for the Planner about how to tell whether an area has been passed or not, as is the case with the *Access_Requirement* property of the AREA object.
- Containing the name of a program to be run by the DEMONSTRATION object, e.g. the property *Program_To_Run*.
- Conditional clauses and logical expressions that make it possible to assess the current state of the student's knowledge, so that the decisions considered appropriate may be taken.

The points below set out the various objects of the model proposed under the following headings:

- Creation the contents network,
- Pedagogical adaptation to the student while varying the structure of the contents network in runtime,
- How to alter the route about the network, and finally,
- Extrinsic needs of the contents network.

5.1.1. *Creation of the contents network*

The most important EDU-EX objects are the AREA objects, as these are the foundations of the structure on which the Knowledge Base is built, and from which the Auto-Regulated ITS is generated.

- *AREA Object*
  Each area corresponds to a part or unit of the subject to be presented to the student. A Knowledge Base is formed by objects interconnected by links or joints which define the relationship between them, and which are created by including the name of one or more objects in certain properties of the AREA object. Areas may not only include the course contents to be presented to the student but also serve as a structure to "hook" onto the various objects with which the pedagogical plan may be formulated.

  Using Area object, a network of contains is build. This network is the skeleton of the future ITS. It holds the rest of objects that allows the autorregulation.
- *DEMONSTRATION Object*
  Demonstration Objects are the contents that, with a different support and different modularity, are presented to the student. Specifically DEMONSTRATION objects also have properties that enable textual information about any particular subject to be incorporated into the teaching system. The properties of a DEMONSTRATION object enable calls to be made to external programs (video or audio files, etc.) which will be used to improve the presentation and clarity of the course contents and as a backup to the texts. It is important to emphasise that a given part of the course could be presented to the student in several ways (using several Demosnstration Objects), loading one or another depending on student knowledge level evaluation.
- *TEST Objects*
  TEST objects are the tests and questions used to assess the knowledge acquired by a student in the areas of knowledge he or she has studied. By evaluating information gathered from these tests and questions it will be possible to alter the pedagogical strategy and to auto-regulate the network to suit the student's characteristics and the continuing process of knowledge acquisition. The answers to test values gathered by the system will be used to determine access to the various areas and DECISION objects associated with the area.

5.1.2. *Pedagogical planning*

Pedagogical planning or adaptation of course contents to the level of the student's knowledge and to his or her characteristics is one of the points

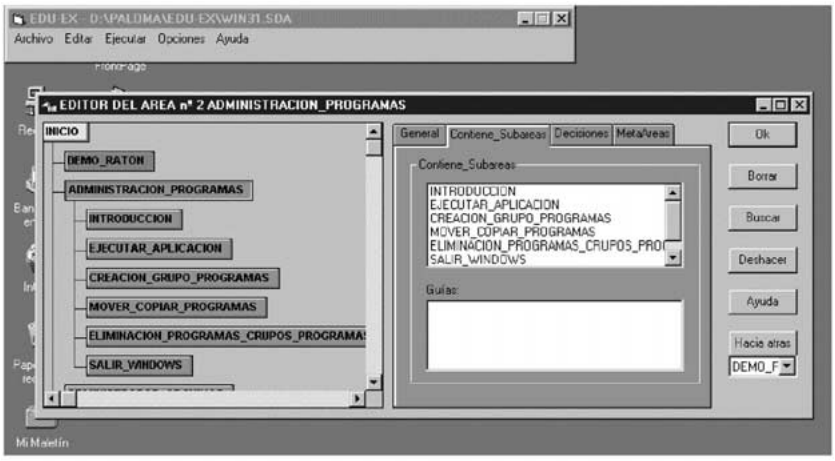

Figure 2. Editor of AREA Object, *Contains_Subareas* property.

that distinguish intelligent tutoring systems from other more traditional computerised teaching methods. This adaptation takes the shape of a series of planning tasks which ensure maximum advantage is taken of the student's efforts. Auto-regulation or adaptation of the system to the student, in EDU-EX system, will proceed two ways: dynamically modifying the contents network and/or modifying the route taken about the contents network. These two forms of pedagogical adaptation will be explained in the following points.

5.1.2.1. *Modification of the structure of the contents network in execution time.*

- *GUIDE Object*
  This object enables new objects to be erased, moved or included in any property that has a list of EDU-EX objects as possible values by using if-then rules stored in the Guide Object. A GUIDE object may, by meeting certain conditions, alter the value of object properties and/or channel the reasoning process. Using this Object it is possible to change the area network structure while the student is studying and also, it could replace one Demonstration object (e.g. video demonstration) by another (e.g. audio demonstration) when necessary, always in execution time. These objects are used to make Knowledge Bases flexible and therefore dynamic, adapting to each situation without the same information having to be rewritten for different cases. GUIDE objects will be able to perform several functions: they may change the order of initial lists, include a new area or remove an area that had previously existed. Any property

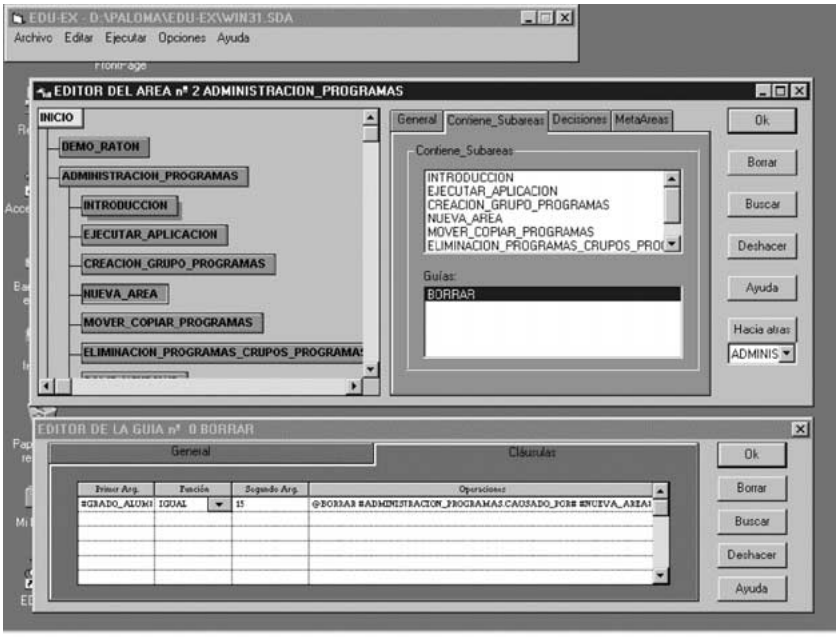

Figure 3. GUIDE object editor.

that has a list of objects as its value will always be able to have one or several associated GUIDE objects allowing the contents of the list to be modified. GUIDE objects look like rules: they are made up of an antecedent and a consequent. When a series of criteria in the premise or left-hand part of the Guide are verified, the actions in the consequent or right-hand part are executed.

5.1.2.2. *Modification of the route about the contents network.*

- *DECISION Object*
  Another form of Auto-regulation, e.g. of continuous pedagogical adaptation, is to alter the normal route about the contents network. The DECISION object is the most complex of all the basic objects in the model proposed; it is the object that enables the route about the contents network to be modified. The Planner always moves about the contents network from left to right and top to bottom unless a property of the DECISION object alters this route. DECISION objects have a series of prerequisites (if-then rules) included in their *Condition* property that ascertain whether this DECISION is to be made or not. The list of decisions is evaluated completely and one by one. Moreover, one aspect that distinguishes this solution from others is a special property called *Strategy*, which enables the normal route about the contents network

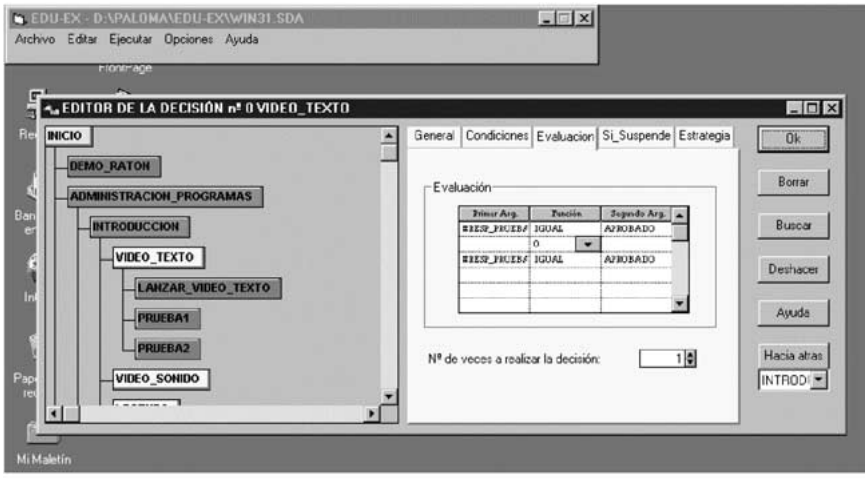

Figure 4. Editor of DECISION object, *Evaluation* property.

to be altered according to the state of the student's knowledge. Other innovative properties make it possible to determine the number of times a lesson may be studied in a certain way, what to do if a student is unable to pass the area, ways of assessing the state of the student's knowledge, etc.

5.2. *Planner*

A Knowledge Base in itself is passive, i.e. it is not able to process information. The Planner is needed so that, together with the Knowledge Base and information obtained by the system from outside (e.g. via a user), the appropriate pedagogical decisions may be taken regarding how to take best advantage of the effort made by the student.

The Planner is the part of the system that processes information in the Knowledge Base by taking decisions and then managing them on the basis of information contained in the various objects and their properties. One of the aspects that differentiates auto-regulated tutoring systems from more traditional teaching systems is precisely this capacity to plan and adapt course contents from a pedagogical viewpoint. As we have seen, auto-regulation is a matter of presenting the student with the most suitable subjects via the most appropriate teaching aids during the learning session or sessions.

The main functions of the Planner are to:

- Assess which course contents are to be shown to the student.
- Adapt the order and rate at which course contents are shown to any one student at any particular time.

- Monitor the knowledge acquired by the student.
- Check that the decisions taken suit and improve the student's learning ratios.
- Issue reports on the knowledge acquired by the student, to be submitted to the teacher.

The Planner will take the following steps to achieve these aims:

5.2.2.1. *Presentation of the system.* When the system starts to run a window appears showing the name of the tool and of the auto-regulated tutor in question.

5.2.2.2. *Gathering relevant information about the student.* The first step the system takes is to ask the student to carry out a series of actions and to respond to questionnaires and tests so that the system may prepare the initial information as considered appropriate by the educator and the expert on the subject.

5.2.2.3. *Student diagnosis.* The Planner identifies knowledge associated with each student characteristic identified at the start and determines which AREA should be run. This diagnosis is the most important stage of the whole process, involving an analysis of the student's prior knowledge and of his or her characteristics as assessed by the system. The Planner assesses the contents of the *Access_Requirement* property of the AREA object, which contains information about the state of the student's knowledge in this area. This property is associated with a logical expression whose evaluation may give the following results:

- True: This means that the area has not been passed, so the Planner goes into the corresponding part of the network and the shows the student what remains to be learned.
- False: This means that the area has been passed, so the Planner does not go into the corresponding part of the network. The student's knowledge is sufficient.
- Unknown: It is impossible to tell whether the area has been passed or not, as there are unknown values as arguments of the functions.

According to the result the property gives, the Planner will continue with this branch of study or pass on to the following ones.

5.2.2.4. *Area resolution.* Once it has been ascertained that an area has not been passed, the student should be presented with the relevant subject information in a form suitable for the type of student in question. The DEMONSTRATION and TEST objects, associated with this area

via the *Action* property of the DECISION object, help establish which demonstrations and tests should be presented to the student to ensure correct comprehension and satisfactory assessment of the subject. It should be noted that the various DECISION objects that may be associated to each area only run if the logical expression allows them to be launched. In the event that the logical expression associated with the decision gives the result "false", this decision will not be carried out, and the system will go on to evaluate the next one.

5.2.2.5. *Assessment of the knowledge acquired by the student.* Once the demonstrations and tests on a certain subject have run, the next stage is to assess the knowledge acquired by the student. This stage is very important, as on occasion a student does not properly assimilate the information, or there are errors in his or her previous knowledge of the subject. During this stage it is decided what is going to happen and at which point of the network the Planner will continue showing new subjects to the student. This assessment is carried out by means of the *Assessment* property of the DECISION object. Once the actions of a DECISION object have been executed, there are two possibilities:

- That the assessment gives the result true. In this case the student has assimilated the required knowledge and may go on to study another area, end the session, etc., depending on the value of the *Strategy* property.
- That the assessment gives the result false. In this case the system will first examine the *If_fails* property, and then the *Strategy* property.

5.2.2.6. *The "If_Fails" property.* If the student has assimilated the required knowledge, the system goes on to examine the *If_fails* property. If this property is empty, the system examines the *Strategy* property. In the event that the *If_fails* property contains the name of an area, the system will take this as the starting area and begin to assess it. Once the *If_Fails* property has been executed, the system evaluates the *Strategy* property of the DECISION object, and if the requirements explained in the previous paragraph have been met, it will go on to evaluate this property.

5.2.2.7. *The "Strategy" property.* The strategy property has four values:

- End: The system will end the session. The area has been passed and there are no outstanding areas.
- Beginning: The Planner begins to examine the system from the START object. This examination involves evaluating and considering all the questions and the session date. The system does not take the same route about the network if it is not necessary.

- Continue: The Planner continues with the evaluation of the contents network in depth and breadth, just as it did before reaching the DECISION.
- Upward: This is the most complex strategy of all. The Planner goes up the same branch of the network by which it came down, re-evaluating the areas' *Access_Requirement* property and also re-evaluating the questions until an area that has not been passed is found, or it arrives at the START object. If it finds an area that has not been passed the system carries out the DECISION strategy again, and if it arrives at the START object it goes on to evaluate the right-hand branches of the network which had not been examined before.

5.2.2.8. *Search for areas not passed and return.* When an area has been entered and the information presented and studied by the student, the course is only considered finished when all the areas have been passed or all the misconceptions that have become apparent during the session have been rectified. But if, having reached any one point in the network, the Planner knows that there are more areas not passed, the system will choose the next area and return to the stage of Diagnosis, Decision, Assessment, etc. Only when all information considered relevant has been assimilated and the student has completed the course will this stage of the system come to an end, followed by the reports stage.

5.2.2.9. *Issuing of reports.* Once all areas have been tackled and assessed and the end of the session has been reached, specific reports will be issued to in order to make an Assessment of the student's learning.

These reports will show data about the student, time spend on the session, test results, etc.

6. Experimentation

Validating educational systems is a complex and difficult process. Anderson, Boyle, Cobert and Lewis remark that: "It is not known exactly which characteristics of tutors produce positive results nor how optimum they are" (Self 1995). To evaluate the system proposed, an Auto-Regulated Intelligent Tutoring System for instruction in Windows 3.1 was developed for this study.

The experimental design used to validate the system was the "non-equivalent control group". The additional comparative group was generated simply by working with students that enrol for a certain subject, with no randomised allocation of students to a particular way of studying (Aiken 1998). This design was used as we have results for students studying for

Table 1. Main characteristics of the trial groups

Characteristics				
Conventional group	Autonomous learning group	No. of elements in the sample	Average age	Academic level
Selection procedure	Selection procedure**	223	19.2	GAP 1st year (24%) LADE 2nd year (76%)
University entrance exam (marks)	Graduates and/or unemployed	351	26.2	42% Graduates* 40% Diploma level* 18% COU (pre-university), FP II (vocational training), etc.
Previous knowledge 10%	Previous knowledge 5%			Graduates and/or unemployed 5%

*The course attendants' degrees and diplomas are mainly in arts subjects (Law, Psychology, Fine Art, History, etc.).
**Knowledge (in%) of the total contents of the Windows 3.1. course.

various degrees at the Carlos III University (who took conventional classes with a teacher) and students on European Social Fund courses (who used an intelligent tutoring system for autonomous learning).

6.1. *Method*

Participants. The course participants were selected from among Business Studies and GAP students in the academic year 97/98. These students, broken down by type in Table 1, took conventional classes for a total of 12 hours. Three hours were devoted to theory and nine to practical sessions with a computer. The comparative group were students on European Social Fund courses, who were taught using the intelligent tutoring system for Windows 3.1. Both the students taking conventional classes and those following the autonomous learning course had little or no knowledge of the subject. The students' characteristics are shown in Table 1.

Description of the sessions. Conventional group: The course contents required to carry out each practical assignment are taught in class by a teacher. The subject is taught with the aid of a personal computer connected to a liquid crystal screen to project images of Windows 3.1. screens. The teacher manipulates the Windows 3.1. environment to clarify his explanations as he goes along. Additional aids used are transparencies and a white board. Each session is a block of three hours. The first hour is devoted to a presentation of the course contents required for the practical work, and the two remaining hours are for students to carry out the practical assignments on the laboratory computers.

Autonomous learning group: No explanations are given before the course is run. Students begin to work individually. The Windows 3.1. course lasts a

total of 12 hours. In each session students work with the system individually. At the end of each session the system saves an analysis of the session and of the state of each student's knowledge.

Previous knowledge of the subject. In both groups most of the students' previous knowledge was confined to an ability to use the mouse. In the case of the students taking conventional classes, some of them had a superficial knowledge of the Windows 3.1. working environment, while in the case of the autonomous learning group, some did not know how to use the mouse and almost none had worked with a windows environment.

Gauging the results. Conventional group: The results are gauged at the end of the course via an examination containing a series of tests that the student has to respond to satisfactorily within a set time.

Autonomous learning group: The student takes a series of tests as he or she studies each subject. If they are not passed, full explanations are given again and further tests are set. In one of the system's screens the level of knowledge arrived at appears at all times. If further details should be required, reports may be generated as considered appropriate.

6.2. *Results*

The most important point in this analysis is the gauging of results for each form of teaching.

"Fatigue" on the part of course participants may limit the validity of any evaluation. Different degrees of participant fatigue can jeopardise the validity of group monitoring and processing. Students give up studying at university for a variety of reasons, one of which is sheer tiredness. The conventional group took the course in the first four-month term of the academic year 97/98, so they did not show excessive fatigue. Students in the autonomous learning group took the course over the summer months (June, July, September and October), which may lead one to expect that the results of this group might not have been as good as hoped.

Comparison of group results. These remarks having been made, the next stage is a comparison of the results of the two groups. These results are presented in the form of a table including the subjects covered in both courses and the percentage of students that passed them satisfactorily. As can be seen in Table 2, the course contents for the conventional group were more restricted that those for the autonomous learning group. A further consideration we should stress is that there was no guarantee of student attendance at the conventional classes, while in the autonomous learning group the subjects were studied sequentially.

Table 2. Comparative results of the assessment of the conventional and autonomous learning groups

Areas	Results of the conventional group*	Results of the autonomous leaning group*
Duration	12 hours	12 hours
Program manager	95	100
File manager	90	100
Control panel	75	100
Notepad – Write – Briefcase	50	100
Paintbrush – Card Index	45	85
Printing manager	No	75
Clock – Calculator – Diary	35	70
MS-DOS	No	60
Terminal – Install Windows	No	50
Packager – Map	No	40

*% of students that passed each stage.

7. Conclusions

The models developed in this work enable teaching systems to be developed in various fields and subjects. The EDU-EX tool for generating auto-regulated intelligent tutors allows tutored instruction systems to be developed independently on any subject. Intelligent tutors generated are easy to modify: all that is required is to include the name of the new area (with the associated pedagogical information) in the list of subareas. The area introduced will appear the next time the system is run. It is quite easy for non-computing personnel to develop auto-regulated intelligent tutors, as they only have to fill out record cards and lists for which almost no knowledge of programming is required. This last point only concerns to programming tasks because ITS pedagogical strategies are very difficult to set up.

The tool assists the development of tutors by checking, in development time, that the database is complete and that no compulsory object or property is left undefined. If this happens, the system tells the developer to complete the required information. The time and cost required for developing auto-regulated intelligent tutors are dramatically reduced with the EDU-EX tool (the complete windows 3.1 system domain module was developped in 3 weeks). The improvement entailed by the tool not only reduces development times but also appreciably simplifies the technical knowledge required of personnel involved in the generation of an auto-regulated intelligent tutoring

system. In fact, the developpers of new ITS have not expert programmers and does not have much technical knowledge.

References

Aiken, L., West S., Schwalm D., Carroll J. & Hsiung S. (1998). Comparation of a Randomized and two Quasi-Experimental Designs in a Single Outcome Evaluation. Efficacy of a University Level Remedial Writing Program. *In Evaluation Review* **22**(2): 207–244.

Anderson, J. R., Boyle, D. F., Farrell, R. & Reiser, B. J. (1987). Cognitive Principles in the Design of Computer Tutors. In Morris, P. (ed.) *Modelling Cognition*. Wiley.

Anderson, J. R. & Pelletier, R. (1991). A Development System for Model Tracing Tutors. *In Proc. of the International Conference on the Learning Sciences*, 1–8. Evanston, IL.

Arruarte, A., Fernandez-Castro, I., Ferrero, B. & Greer, J. 1997. The IRIS Shell: How to Build ITSs from Pedagogical and Design Requisites. *International J. of Artificial Intelligence in Education* **8**(3–4): 341–381.

Blanc, F. (1996). Une Utilisation de la Modélisation Qualitative pour la Planification Pédagogique. *Proceedings of the third conference on Intelligent Tutoring Systems, ITS'96*. Montreal, Canada.

Blessing, S. B. (1997). A Programming by Demonstration Authoring Tool for Model Tracing Tutors. *Int. J. of Artificial Intelligence in Education* **8**(3–4): 233–261.

Bloom, Ch., Brigham, B., Meiskey, L., Sparks, R., Dooley S. & Linton, F. (1995). Putting Intelligent Tutoring Systems Technology into Practice: A Study in Technology Extension and Transfer. *In Machine Mediated Learning* **5**(1): 13–41.

Clancey, W. & Joerger, K. (1988). A Practical Authoring Shell for Apprenticeship Learning. *Proceedings of ITS-88*, 67–74. Montreal.

Collins, J. A., Greer, J. E. & Huang, S. H. (1996). Adaptive Assesment using Granularity Hierarchies and Bayesian Nets. In Frasson, Gautheir & Lesgold (eds.) *Proceedings of the Third International Conference: ITS'96*, 569–577. Springer.

Domingo, P. (1998) Sistema para la Generación de Tutores Inteligentes Basados en Modelos Doctoral Thesis.

Goodkovsky, V. A., Kirjutin, E. V. & Bulekov, A. A. (1994). Shell, Tool, and Technology for Pop Class ITS Production. In Brusilovsky, P., Dikareve, S., Greer, J. & Pertrushin, V. (eds.) *Proc. of East-West International Conference on Computer Technology in Education*, Part 1, 87–92. Crimea, Ukraine.

Gómez-Pérez, A. (1998). Knowledge Sharing and Reuse. *The Handbook of Applied Expert Systems*, edn. by Jay Liebowitz.

Hsieh, P., Halff, H. & Redfield, C. (1999). Four Easy Pieces: Developing Systems for Knowledge-Based Generative Instruction. *Int. J. of Artificial Intelligence in Education*.

Jones, M. & Wipond, K. (1991). Intelligent Environments for Curriculum and Course Development. In Goodyear (ed.) *Teaching Knowledge and Intelligent Tutoring*. Norwood, NJ: Ablex.

Mizoguchi, R., Sinitsa, K. & Mitshuru Ikeda (1996). Task Ontology Design for Intelligent Educational/Training Systems. *Proceedings of the third conference on Intelligent Tutoring Systems, ITS'96*. Montreal, Canada.

Major, N., Ainsworth, S. & Wood, D. (1997). REDEEM: Exploiting Symbiosis between Psychology and Authoring Environments. *International J. of Artificial Intelligence in Education* **8**(3–4): 317–340.

Merrill, M. D. & ID2 Research Group (1998). ID Expert: A Second Generation Instructional Development System. *Instructional Science* **2**: 243–262.

Murray, T. (1996). Special Purpose Ontology and the Representation of Pedagogical Knowledge. *Proceedings of the International Conference on the Learning Sciences*. Evanston IL, July.

Murray, T. (1998). Authoring Knowledge-Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design. *J. of the Learning Sciences* **7**(1).

Murray, T. (1999). Authoring Intelligent Tutoring Systems: An Analysis of the State of the Art. *International J. of Artificial Intelligence in Education* **10**: 98–129.

Nkambou, R., Gauthier, G. & Frasson, M. C. (1998). CREAM-Tools: an Authoring Environment for Curriculum and Course Building in an ITS. *In Proceedings of the Third International Conference on Computer Aided Learning and Instructional Science and Engineering*. New York: Springer-Verlag.

Orey, M. & Nelson, W. (1994). Development Principles for Intelligent Tutoring Systems: Integrating Cognitive Theory into the Development of Computer Based Instruction. *Educational Technology Research and Development* **41**(1): 59–72.

Papert, S. (1995). *Desafío a la mente: Computadoras y educación*. Galápagos. Buenos Aires, Argentina.

Reinhardt, B. & Schewe, S. (1995). A Shell for Intelligent Tutoring Systems. In Greer, J. (ed.) *Proc. of the Int. Conf. On AI in Education*. Charlottesville, VA: AACE.

Russell, D. (1988). IDE: The Interpreter. In Psotka, Massey & Mutter (eds.) *Intelligent Tutoring Systems, Lessons Learned*. Hillsdale, NJ: Lawrence Erlbaum.

Self, J. A. (1995). Computational Mathetics. http://www.cbl.leeds.ac.uk/∼jas/cmch3.html

Shute, V. J. (1998). DNA – Uncorking the Bottleneck in Knowledge Elicitation and Organization. *Proceedings of ITS-98*, 146–155. San Antonio, TX.

Van Marcke, K. (1998). GTE: An Epistemological Approach to Instructional Modeling. *Instructional Science* **26**: 147–191.

Wenzel, B., Dirnberger, M., Hsieh, P., Chudanov, T. & Halff, H. (1998). Evaluating Subject Matter Experts' Learning and Use of an ITS Authoring Tool. *Proceedings of ITS-98*, 156–165. San Antonio, TX.

Winne, P. H. (1991). Proyect DOCENT: Design for a Teacher's Consultant. In Goodyear (ed.) *Teaching Knowledge and Intelligent Tutoring*. Norwood, NJ: Ablex.

Winne, P. H. & Kramer, L. (1999). Representing and Inferencing with Knowledge about Teaching: DOCENT. *Proceedings of ITS-88*. Montreal, Canada.