This is a draft version of a paper published in *International Journal of Artificial Intelligence in Education*, 12, (2002), pp. 185-211.

Adding an Instructor Modelling Component to the Architecture of ITS Authoring Tools

Maria Virvou, Maria Moundridou

Department of Informatics, University of Piraeus,
80, Karaoli and Dimitriou St., Piraeus 185 34, Greece
E-mail: mvirvou@unipi.gr, mariam@unipi.gr

Abstract. The role of instructors as users/authors of ITS authoring tools is very important for the effectiveness of the produced ITSs. In order for authoring tools to benefit the most from the involvement of instructors, they should provide individualised feedback to them throughout the ITS's life cycle. In most cases the addition of an instructor modelling component to the architecture of an authoring tool may be very beneficial both for the quality of the ITSs to be produced and for the instructor himself/herself. Such an instructor modelling component could acquire information about instructors' preferences, interests and usual activities concerning the courses they author. In addition, it could acquire information about the instructors' level of expertise in particular domains as well as in the authoring process itself. In return it may use this information to increase adaptivity and provision of help to the instructor's needs. The main aim of this paper is to show that an instructor modelling component would be beneficial to all ITS authoring tools. Therefore, we have reviewed the literature and discuss the role that an instructor modelling component could have played in existing authoring tools if it had been incorporated into them. In addition, this paper also describes how an instructor modelling component has been incorporated in an authoring tool that we have developed, which is called WEAR.

INTRODUCTION

Intelligent Tutoring Systems (ITSs) are computer-based instructional systems with the ability to present the teaching material in a flexible way and to provide learners with tailored instruction and feedback. A number of successful evaluations of ITSs (Anderson et al., 1990; Koedinger et al., 1997; Mark & Greer, 1991; Lajoie & Lesgold, 1989; Shute, Glaser, & Raghaven, 1989) have shown that such systems can be effective in improving learning by increasing the students' motivation and performance in comparison with traditional instructional methods. The main flaw of ITSs and possibly the reason for their limited use in workplaces and classrooms is the complex and time-consuming task of their construction. A large number of people such as programmers, instructors and experts of a specific domain must be involved in an ITS development. As estimated by Woolf and Cunningham (1987) an hour of instructional material requires more than 200 hours of ITS development time. Furthermore, an already constructed ITS for a specific domain can neither be reconstructed to function for a different domain, nor can it be altered (i.e. to reflect a different tutoring strategy in the same domain) without spending much time and effort. An approach to simplifying the ITS construction is the development of ITS authoring tools. The main aim of such tools is to provide an environment that can be used by a wider range of people to easily develop cost-effective ITSs.

ITSs have been described as consisting of four main components (Hartley & Sleeman, 1973; Burton & Brown, 1976; Wenger, 1987). These are the domain knowledge, the student modelling component, the tutoring model and the user interface. Accordingly, ITS authoring tools offer to their users the ability to author one or more than one of these components. Hence, a distinction among various authoring tools may be based on the different components that they allow their users to author. Another way to distinguish such systems is in respect to the type of ITSs that they produce. According to Murray (1999), the majority of authoring tools fall into two broad categories: the *pedagogy-oriented systems* which "focus on how to sequence and teach relatively canned content" and the *performance-oriented systems* which "focus on providing rich learning environments in which students can learn skills by practising them and receiving feedback." He also describes seven categories of ITS authoring systems according to the type of ITSs they produce. These are: i) Curriculum Sequencing and Planning, ii) Tutoring Strategies, iii) Device Simulation and Equipment Training, iv) Expert Systems and Cognitive Tutors, v) Multiple Knowledge Types, vi) Special Purpose, and vii) Intelligent/Adaptive Hypermedia. Murray in the same paper classifies over two dozens authoring systems into the above categories to illustrate each system's strengths and contribution to the field of ITS authoring tools. However, he argues that every system classified into a category contains important features from at least one other category.

This is also the case with the authoring tool called WEAR, to be presented in this paper. WEAR observes students while they are working with problems from various Algebra-related domains and provides them with feedback when their actions seem erroneous. In that sense, WEAR falls in the category of Expert Systems and Cognitive Tutors. WEAR also deals with managing the sequence both of the available problems and of the teaching material based on the student's performance and the relationships between course modules, which is distinctive of the Curriculum Sequencing and Planning category. Finally, WEAR is a Web-based system, which provides students with adaptive navigation support by dynamically annotating the links to course modules; as such, it can be seen as an authoring tool belonging to the Intelligent/Adaptive Hypermedia category.

Most of the existing authoring tools, irrespective of the category they belong to, depend heavily on instructors' authoring for the quality of their resulting ITSs. However, instructors may face several difficulties during the design process (e.g. they may not be sure about the structure their course should have) and they may provide inconsistent information to the tool that may lead to the generation of ITSs with problematic behaviour. Furthermore, instructors play a crucial role in the success or failure of any kind of educational software in real school settings. If instructors do not accept the software then this does not stand a good chance of being properly used in class. This is even more the case for authoring tools that are primarily addressed to instructors. However, very few authoring tools provide extra facilities to authors/instructors that would help them with the authoring process. For example, Wu, Houben, & De Bra (1999) describe support tools that help authors create usable and consistent adaptive hypermedia applications. Brusilovsky (2000) introduces a concept-based course maintenance system which can check the consistency and quality of a course at any moment of its life and assist the course developer in some routine operations. A more sophisticated approach is presented in (Nkambou, Frasson, & Gauthier, 1998): in order to provide designers with support that focuses on the expertise for building courses, they propose to use an expert-based assistant integrated with the authoring environment. The expert system reasons on a constraint base that contains constraints on curriculum and course design that come from different instructional design theories. In that way, the expert system validates curriculums and courses produced with the authoring tool and advises the instructional designer accordingly. Another interesting approach is described in (Barra, Negro, & Scarano, 1999). It is about a symmetric model for adaptive WWW-based systems: the model represents users (students) in terms of their "interest" toward information nodes in each topic and vice versa it models information nodes in terms of the perceived "utility" toward users in each category. In this way adaptive behaviour can be presented to the user (student) but also to the author, to assist him/her in designing and tuning the adaptive system.

However, none of the authoring tools incorporates an instructor modelling component that could be valuable towards assisting authors in constructing and tuning the courseware. This remark is also in accordance with an observation made by Kinshuk and Patel (1996) concerning the more general area of computer integrated learning environments:

> "Whereas the work on student modelling has benefited by the user modelling research in the field of HCI, the research on the role of a teacher as a collaborator in the computer integrated learning environments is almost non existent" (p. 222).

Indeed an instructor modelling component could process and record information about instructors' preferences concerning teaching strategies, their interests and usual activities and their level of expertise both in teaching a particular domain and in the authoring process itself. The instructor modelling component, once constructed, could then provide valuable feedback to instructors concerning their own teaching goals and the courses they author.

For example, instructor modelling components may render authoring tools more teacher-friendly, flexible and helpful to instructors. In this way, more human tutors may be encouraged to be involved in the authoring of ITSs. Some of these human tutors may have valuable experience in teaching a particular domain. An ITS could benefit a lot from such experience. On the other hand, if authoring tools are only addressed to very few instructors that would combine excellent software skills with high teaching expertise then they run the risk of being heavily criticised that they do not belong to the mainstream of education and real school settings. This has often been the case with ITSs (e.g. Boyle, 1997). However, as Andriessen and Sandberg (1999) state, instead of criticising the ITS paradigm, we should focus on the role Artificial Intelligence can play in different educational settings.

The addition of an instructor modelling component to the architecture of authoring tools may also contribute to the improvement of the development life cycle of the resulting ITSs. It could provide facilities to authors which would encourage multiple iterations of the authoring procedure. If this was the case, the development life cycle of the resulting ITSs would also be based on multiple iterations as would be recommended by knowledge-based software engineering. Indeed multiple iterations of an ITS life cycle which involves instructors may result in more effective ITSs (e.g. Virvou & Tsiriga, 2000). To achieve this, an authoring tool should provide continuous feedback concerning the constructed or the under construction ITS to its users/authors. In this way, the life cycle of the resulting ITSs may be expanded to include evaluations of them and improvements made by the author based on these evaluations. Such evaluations could be conducted automatically by possible facilities of the authoring tool to keep statistical information on several aspects of the ITSs' usability, learning effects and compliance with the instructor's original goals.

The idea that ITSs may be improved based on the feedback that the progress of students may provide is not new. In fact, O'Shea and Sleeman (1973) made the observation that there is a lot of scope for improvement by the system itself if the system takes into account the feedback from students. This observation gave rise to self-improving tutors such as the QUADRATIC tutor or the PROTO-TEG. The QUADRATIC tutor (O'Shea, 1979) deals with improving teaching strategies that involve parameters about conflicting goals in the teaching task. Parameters that may be modified include the frequency of encouraging remarks, the number of guesses before the system gives out the solution, etc. However, O'Shea suggested that this self-improving mechanism might be best suited for a system that sets up its experiments in collaboration with human teachers. This is even more the case for authoring tools that definitely involve human teachers. In this case an instructor modelling component would serve as the means for collaboration between the human teacher and the authoring tool in the improvement process of the resulting ITSs. The human teacher could set the parameters and the instructor modelling component could automatically monitor the progress of the resulting course with respect to these parameters.

A more recent version of a self-improving tutor is the PROTO-TEG (Dillenbourg, 1990) which is a system that is able to discover the criteria that are useful for selecting the didactic strategies it has at its disposal. These criteria are expressed as characteristics of the student model and are elaborated by comparing student model states recorded when a strategy was effective and those recorded when the same strategy was not effective. In this case too, the criteria used in PROTO-TEG could contribute information to an instructor modelling component that would improve the authoring process.

In the remainder of this paper we will first describe some existing authoring tools and comment on the advantages that an instructor modelling component could have on them. We will next discuss the issues involved in adding an instructor modelling component to the architecture of ITS authoring tools: What are the uses of such a component? Which user aspects can be modelled? What are the sources of information that such a component could use? Then we will report on the authoring tool called WEAR that we have developed and describe how instructor modelling is incorporated in it. Finally we will draw some conclusions about the discussed subject.

AUTHORING TOOLS FOR INTELLIGENT TUTORING SYSTEMS

In the last decade over two dozen ITS authoring tools have been developed. In this section we will describe only a few authoring systems, along with some speculations and considerations concerning various aspects of what we call instructor modelling, in order to facilitate the discussion to take place in the subsequent sections. A thorough and in-depth analysis of the state of the art for ITS authoring tools is beyond the scope of this document; such an analysis can be found in (Murray, 1999).

REDEEM (Major, Ainsworth, & Wood, 1997) is a tool that allows its users to author the *tutoring model* of the ITS that will be produced. It does not deal with the generation of instruction but rather focuses on the representation of instructional expertise. REDEEM expects the human instructor to describe existing teaching material (tutorial “pages”) in terms of their difficulty, their generality, etc., to construct teaching strategies (i.e. when and how to test the students, how much hinting and feedback to offer, etc.) and to identify students. The tool exploits the knowledge provided by the instructor and its default teaching knowledge to deliver individualised instruction to students. Major, Ainsworth, & Wood (1997) mention about REDEEM:

> “… A second focus of our research with the teachers will be to explore aspects of teacher expertise…REDEEM can be used to examine the lessons produced by experienced teachers in comparison with those produced by novice teachers. These environments could be related to any differences in learning outcomes” (p. 335).

Exploring aspects of teacher expertise is in accordance with our views about authoring tools. Indeed, the system itself could infer if a teacher is experienced or not based on the differences in the learning outcomes of their lessons. The purpose of doing this would not be to examine the teachers’ competence but rather to offer them help and advice in case they need it. A novice teacher having trouble constructing an ITS would benefit a lot from a system if s/he could see what an experienced (based on the system’s assumptions) colleague of his/her has done. Ainsworth, Grimshaw, & Underwood (1999) conducted a case-based evaluation of REDEEM in which teachers were offered the choice of viewing the courses constructed by other teachers. The results of this evaluation attest to our proposal: teachers working with REDEEM expressed interest in comparing views of the course provided by different authors. To this end a model of each teacher containing information about teachers’ characteristics would be very useful for the provision of individualised assistance.

In the same evaluation as above, teachers were given the opportunity to experience the consequences of their own teaching decisions by playing the videos of a virtual class. All of the teachers took this opportunity and suggested improvements for the courses they had authored. We believe that this could be a facility that REDEEM should incorporate in the authoring tool itself. It could also be expanded to include more automated feedback to instructors. Currently in most ITS authoring tools authors (usually teachers) are responsible for constructing the ITS but they are not receiving any feedback concerning the effect that this ITS had on learners. The purpose of this feedback would be to help teachers improve the course they have constructed and it could have various forms: for example, it could consist of students' progress reports, or of captured students' actions. Furthermore, this feedback could be tailored to each instructor's interests inferred from his/her instructor model. In that way the instructors would be receiving only relevant and useful information and thus be assisted in the course creation. Additionally, the instructor model could be utilised to provide intelligent help to the instructors when their actions are erroneous or in some way incompatible with their inferred or stated goals. For example, an instructor may have stated to the authoring tool that s/he wishes to be "strict" or "lenient" or "neutral" with the students. If a particular instructor has declared that s/he wishes to be "lenient" and the majority of the students fail to solve the problems s/he has provided, then the authoring tool could inform him/her of this incompatibility.

Another system is **EON** (Murray, 1998). EON is in fact a set of authoring tools for building knowledge based tutors. What seems to be important in EON tools is that they allow the authoring of every component of the ITS architecture: the *user interface*, the *domain knowledge*, the *tutoring model*, and the *student model*. Specifically, EON allows authors to construct the ITS's user interface from scratch, using interface objects (called widgets) such as buttons, text, pictures, graphs, etc. These objects can be customised to respond to specific student actions and in that way the authored user interface with EON can be highly interactive. The domain knowledge is authored in EON by defining topics, properties for these topics (i.e. "importance" or "difficulty") and topic link types (i.e. "prerequisite", "part-of", etc.). Then, the author should create a topic network using the information concerning topics that s/he has entered previously. To allow authoring of the tutoring model (teaching strategies) EON uses a flowline-based graphical programming language. In that way the author of the ITS can specify how the teaching material is sequenced, when and how to test the students, give them explanations, hints, etc. To make the constructed ITS adaptive to each student's state of knowledge, EON's teaching strategies refer to values stored in a student model that can also be authored. This is accomplished by allowing the author to assign values (e.g. "mastered", "known", "suspected misconception") to objects at several decision layers (Lesson, Topic, Topic Level, Presentation Contents and Events). The value of these objects is determined by the student model rules written for the specific object's level and specify how an object's value depends on the values of objects at the next lower level.

EON is a rather sophisticated programme requiring some training to be used effectively. Its intended users (on the authoring side) are mainly instructors whose skill levels may vary a lot. To overcome this problem their approach in EON (Murray, 1996) is to provide a three-tiered suite of authoring tools at three levels of abstraction for "three tiers" of user:

> "At the first tier is a general purpose ITS authoring system that requires moderate knowledge engineering and instructional design expertise to use. At the second tier are special purpose ITS authoring systems that require minimal knowledge engineering and instructional design expertise. The third tier involves the average teacher using an ITS in her class" (p. 95).

Similarly to EON, most authoring tools are quite sophisticated pieces of software that require a lot of skills from instructors to use them effectively. One way to facilitate the use of authoring tools (like any other software) is to provide users with intelligent, individualised help. Instructor models can be utilised in this direction as well, if designed to have the appropriate information.

RIDES (Munro et al., 1997) is used for the construction of tutors that teach students how to operate devices through graphical simulations. RIDES allows users to author both the *user interface* and *domain knowledge* components of the ITS to be produced. Both these are achieved by providing tools for building graphical representations of devices and for defining the devices' behaviour. The simulations that RIDES generates are based on the devices that the user has constructed, the devices' properties and their connections. **DIAG** (Towne, 1997) is another authoring system, which adds capabilities to RIDES by delivering instruction on device troubleshooting and maintenance. DIAG is concerned with the creation of *domain knowledge*. In particular, DIAG requests the author's (expert's) qualitative judgements of the possible symptoms (diagnostic signals) produced by equipment faults. Applying fuzzy set theory to these qualitative judgements, DIAG creates a fault table that is used to perform fault diagnosis based on the observed symptoms. Students interacting with the system are presented with a fault and they work at diagnosis; the system, being able to perform fault diagnosis, provides students with necessary feedback and hints.

In DIAG, the domain expert is asked to specify the possible symptoms that failures in a unit will produce. Different domain experts would supply different likelihood judgements, as anyone could expect. Towne's approach to determine the validity of the constructed knowledge base is to ask two or more domain experts to supply the required fault effect judgements and check among those for inconsistencies. This approach could be followed in many ITS authoring tools; but what about the case when conflicts of opinions are present? How should they be resolved? We believe that the existence of user models could help in this arduous task. Taking DIAG as an example, suppose a domain expert has indicated that a symptom rarely occurs, whereas another expert has indicated that the same symptom occurs frequently. A hypothetical user model in DIAG could contain information concerning the users' tendency to overestimate or underestimate the presence of some symptoms. Using such information and some rules, DIAG (and any authoring tool taking this approach) could resolve the conflicts and produce consistent and reliable ITSs.

The **LEAP** authoring tool (Sparks et al., 1998) was developed to be used by authors who were subject-matter experts but they were not programmers. The LEAP system is concerned with teaching customer contact employees (CCEs), such as customer service representatives, the knowledge and skills they need to respond effectively to customer requests and problems. LEAP simulates CCE's work environment, in which they are expected to carry on a dialogue with customers over the telephone while simultaneously interacting with database systems for entering service orders and obtaining information about customers, their accounts etc. Authors must create courseware that may include complex conversation grammars, a description of all the individual actions to these topics, hints for each step of the conversation, audio recordings and textual representations of customer and expert CCE actions, etc. The developers of the LEAP authoring tool had set the following goals for its design: i) ease of use, ii) rapid prototyping and iterative development, iii) reuse of courseware components, iv) multiple representations and input mechanisms and v) interactive visualisations of the courseware structure and content. In particular, ease-of-use meant both easy to learn for novice authors and easy and efficient for experienced users. In addition, the developers had conducted a requirements analysis based on potential LEAP authors, a CCE training expert and human factors experts. The subject matter experts who served as courseware authors during the development of LEAP led the designers to the conclusion that different authors would have different individual authoring styles and different starting points for the authoring process. For example, some authors may have been adapting existing training materials while others were starting from scratch.

The design goals of the LEAP authoring tool and their conclusions from their requirements analysis are in accordance with our views concerning the design needs of authoring tools. In particular, their conclusion about the need for ease-of-use and for multiple representations and input mechanisms for multiple authoring styles points to the direction of the need of an instructor modelling component. However, they have not used one. In case they had, the authoring process and the different authoring styles might have been managed in a more automatic way which would alleviate further the cognitive load of authors-instructors. An instructor modelling component in LEAP could infer and record permanently instructors' preferences and authoring styles and could use this information automatically to help authors in multiple and independent authoring sessions both for the iterative development and reuse of courseware components. In particular, reuse of courseware components could be extended to support groups of authors that could collaborate based on the similarities of their interests as recorded in their instructor models.

The **SmartTrainer/AT** (Hayashi et al., 2000; Jin et al., 1999) is an ontology-aware authoring tool for intelligent training systems. Ontology is defined as "a system of primitive vocabulary/concepts used for building artificial systems" (Mizoguchi, 1993). The activities supported by the authoring tool are the building of an ontology by an ontology author and then the designing of a model by the teaching material author. The developers of the SmartTrainer/AT have identified two phases constituting the authoring process: a composing phase and a verification phase. They have noted that support for the second phase has been overlooked in the existing authoring tools technology as compared to the first phase. Therefore they have developed a component which is called *conceptual-level simulation*. This component is meant to help the author compare what s/he thinks (design intention) with what s/he gets (behaviour). The purpose of the conceptual-level simulation is to show "which part of the training scenario" adds "what type of an educational effect" to "what type of a learner" systematically.

The SmartTrainer/AT addresses the issue of helping authors to verify that their design intentions have been met. This is also one of our concerns although we propose a different kind of component, the instructor modelling component. In fact, the instructor modelling component would be of a more general use, since it could render the system more flexible and helpful to the authors in many aspects both in composing courseware and verifying it, as well as composing subsequent, revised versions of it. The revised versions could be based on collected feedback from all students that would help in the evaluation of the teaching goals of the instructor, as these would have been recorded in his/her model. Moreover, the instructor modelling component would take into account the various levels of expertise of individual authors and would adapt its advice accordingly.

INSTRUCTOR MODELLING IN INTELLIGENT TUTORING SYSTEMS AUTHORING TOOLS

After having reviewed the available literature on ITS authoring tools we thought that the issue of instructor modelling is a rather overlooked one. As we discussed in the previous section the incorporation of an instructor modelling component in ITS authoring tools could significantly add to the strengths of the ITSs to be produced. In particular, an instructor modelling component may help in various directions which will be highlighted and discussed in the subsection to follow. The next subsection will deal with the instructor aspects that can be modelled and the last subsection will specify what the sources of information for an instructor model may be.

Uses of an instructor modelling component

An instructor modelling component can be used in ITS authoring tools to improve their quality through various ways. These are: the encouragement of multiple iterations in the authoring process, the provision of intelligent help and individualised feedback, the encouragement of collaboration among instructors, the ensuring that the produced ITSs are consistent and accurate and the generation of a user-friendlier learning environment for the student.

Multiple Iterations in the Authoring Process

One of the primary aims of ITS authoring tools is to provide environments for knowledge-based software engineering of the resulting ITSs. In this respect, it would be very useful to provide facilities for the iteration of the authoring process. This means that the resulting ITS could be evaluated and then the author-instructor could alter several aspects based on the evaluation results. Therefore, in this case some evaluation aspects, such as the learning outcomes of a particular teaching strategy applied could be recorded in the instructor model of the instructor so that s/he could use this information in improvements of the authoring procedure. This does not mean that the instructor model would be responsible for evaluating the entire teaching strategy for a particular domain and a particular group of students. However, it could record evaluation elements that may be useful to instructors based on their instructor models, so that instructors may conduct the evaluation and possible revision of their courseware themselves. By no means would the authoring tool replace the work of instructors. However, it may well help the instructor in the authoring process and its possible iterations.

Examples of evaluation elements of the courseware that the authoring tool could record in the instructor model would be the following: statistical information gathered from student models concerning their progress and grades, difficulties in particular exercises where the majority of students may have had problems and so on. The instructor model could compare this information to the wishes of the instructor concerning the courseware, as these would have been recorded prior to the delivery of the courseware. In this way the instructor model could point out certain inconsistencies in case these have been identified (e.g. certain exercises may have turned out to be more difficult than the instructor originally wished).

Moreover, instructors working with ITS authoring tools and acting as authors of ITSs should be receiving adequate feedback concerning the ITSs they are constructing. This will render the ITS construction an iterative process and thus result in more effective systems. This feedback could be tailored to each instructor's interests inferred from his/her instructor model. In that way the instructors would be receiving only relevant and useful information and thus be assisted in the course creation. For example, an instructor may not be interested in the in-depth statistical analysis and possible diagnostic analysis of the students' exam papers. In this case the authoring tool should not take itself the initiative to inform the instructor about any changes concerning such analysis unless the instructor wishes to see such information.

Intelligent and/or Individualised Help

Murray (1996) points out: "…for the foreseeable future, we do not expect the "average" classroom teacher or industrial trainer to be able to author an ITS any more than we expect every teacher to author a textbook in their subject area". To overcome this problem, authoring tools should become more "teacher-friendly". This may be a way to encourage more human tutors to be involved in the authoring of an ITS. Indeed the users of ITS authoring tools are usually instructors who may be experienced in their teaching domain but not very familiar with such environments. While regular "Help" may not be sufficient for this kind of sophisticated software, "Intelligent Help" could prove to be rather beneficial. If help is to be intelligent, it should be individualised to each instructor. To attain this, the authoring tool should maintain and exploit instructor models holding information about each instructor's characteristics and skills. In this case, the instructor modelling component would be crucial for the generation of hypotheses concerning the instructor's difficulties and needs while s/he is using the software, similarly with other projects (e.g. Virvou & DuBoulay, 1999; Virvou & Kabassi, 2000).

Besides the intelligent help that can be provided concerning the use of the software (authoring tool), instructors can be assisted by intelligent help concerning their teaching decisions while authoring a course. The authoring tool could check the compatibility of teaching goals with teaching strategies utilising an instructor model to provide intelligent help to the instructors when their actions are in some way incompatible with their goals. For example, an instructor may have stated to the authoring tool a long-term goal that s/he wishes to keep very high standards in the marking of students' exam papers during a particular semester. This long-term goal could be recorded in the instructor model and then the average marks that students receive for each exam paper of the semester could be compared to average marks of previous exam papers, so that the instructor could be informed about any incompatibility with his/her long-term goal.

Finally, the authoring tool could decide based on the instructor model the extent to which it will offer help and the kind of help it will offer to each instructor. For example, an experienced instructor but not experienced author should receive more help on the actual authoring whereas an experienced author but not experienced instructor may need more help on the actual teaching strategies.

Encouragement of Collaboration among Authors

Collaboration among instructors can be promoted by the presence of instructor models. The idea of enabling the collaboration among instructors in the course building process has already been pointed out in the ITS authoring tools literature. For example, Nkambou, Frasson, & Gauthier (1998) mention that their future plans include a meta-authoring language that will enable co-operating curriculum and course authoring.

Collaboration among instructors may be encouraged by offering instructors less experienced in ITS construction the opportunity to see and even use the ITSs produced by more experienced colleagues. The characterisation of instructors as experienced or not can be based for example on statistical outcomes of their lessons and their own beliefs about themselves. These judgements ("experienced", "novice", etc.) could be kept in instructor models. When information concerning teachers' characteristics is required, the system could consult the instructor models and act accordingly.

Collaboration could also be based on instructors' interests and preferences recorded in their instructor model. For example, instructor models could be compared and instructors could be informed about the existence of other instructors sharing similar interests and preferences. In such cases, instructors could communicate directly to exchange ideas if they wish so. In other cases, the authoring tool may inform instructors about the existence of courses and/or exercises which fit their own interests but have been created by other instructors.

Consistency and Accuracy of the Produced ITSs

A major concern of most researchers in the area of ITS authoring tools is how accurate, consistent, effective and reliable will the produced ITSs be. There seem to be two ways to ensure that the ITSs produced by an authoring tool are accurate, consistent, effective and reliable: either the authoring tool should include mechanisms that examine the under construction course and propose corrections, or other instructors should be consulted. In both cases, the existence of instructor models could be very helpful.

Instructor models may be particularly useful to authoring tools dealing with domains that expect a lot of input which may be subjective and cannot be automatically checked for accuracy and consistency. In this case, the validity of the course may be examined using the information kept in instructor models. For example, the tool may try to check the validity of the instructors' input by consulting multiple instructors; an instructor model containing information about the instructors' characteristics could help the authoring tool overcome a possible conflict that may arise from the different instructors' judgements (see previous section in the paragraph where we comment on DIAG system). In the cases where it is not possible to consult other instructors for the purpose of ensuring the accuracy of a course, the instructor model can provide the tool with evidence about possible inconsistencies. For example, the authoring tool can monitor the students' behaviour when interacting with the system and watch for any suspect evidence (e.g. if the majority of students fail to solve a problem, this could possibly mean that the instructor may have provided a misleading problem statement; if very few students visit a specific topic of the curriculum, this may indicate that this node cannot be easily reached or that this topic is not interesting to students). Such evidence can be contained in the instructor model and used to form advice to the instructors. If for example, an instructor is observed to have frequently constructed problems that the students cannot solve, then s/he could be prompted to check if there is something wrong with the specific problems s/he has constructed.

User Friendlier Learning Environment for the Student

Teaching through the Web may become very impersonal for the students who cannot create a model of their instructor for themselves. Therefore, an instructor could pass certain information to students through his/her instructor model to render the teaching procedure more personal and student-friendly.

Some aspects of the instructor model that could be shown to students could be: what s/he considers most important, what s/he expects from students, what his/her priorities are, how difficult the exercises s/he creates are, whether many calculations are required to solve his/her problems (in case of numerical problems), etc.

Instructor Aspects That Can Be Modelled

A user model may consist of long-term and short-term information concerning the user and thus we may have a long-term and short-term user model (Rich, 1979; 1983). In the case of instructor modelling, the long-term instructor model may record instructors' interests, usual activities, preferences, level of instructor expertise etc. On the other hand, the short-term instructor model may follow the instructor's goals and plans in a particular interaction session with the authoring tool. In particular the instructor aspects that can be modelled in an ITS authoring tool may be the following:

- Level of expertise

  1. Level of expertise concerning the teaching part. This can include both the instructors' ability to create teaching strategies and their knowledge in the particular domain they teach.
  2. Level of expertise concerning the authoring of the course. This mainly concerns the instructors' competence in using the software.

- Interests and activities

  The instructor model may record information about the instructor's special interests. For example, whether s/he is interested in diagnostic analysis of student errors in tests or whether s/he is interested in the collaboration with other instructors and if so, in what particular topics etc. The instructor's usual activities may also be recorded in the long-term instructor model. For example, whether a particular instructor often updates course material or not.

- Preferences in teaching strategies

  The instructor model may record information about the instructor's preferences in teaching strategies. For example, whether they prefer students to be tested regularly or not, what level of difficulty they prefer for the tests, what the average mark for a class should be, how to motivate students, etc.

Sources of Information for an Instructor Model

In order for an authoring tool to maintain and utilise instructor models it should somehow acquire all the relevant information. What is considered relevant information may vary a lot depending on the nature of each authoring tool and the way the instructor model will be exploited. The instructor models may be either explicit or implicit (Rich, 1983). This means that the system can acquire the information it needs to build and maintain the instructor models either by asking the instructor directly, or by observing his/her actions so that it may infer certain attributes concerning the user. In a general case, for the explicit instructor model the instructor's answers to the authoring tool's questions may well serve as a source of information. In the case of the implicit instructor model, the sources of information may be the following:

The student models. Most authoring tools construct student models that can provide useful information for instructor models as well. For example, the students' progress in the course indicating the learning effects may be used to show how effective certain teaching strategies have been, or if certain courses and exercises have been visited by many students many times, then this may indicate popularity of a certain course. In this way, instructor models may give information to the instructors who may use it to improve their own performance if they wish. Of course, the circumstances under which the clues from student models may be used may vary considerably. Therefore, there may be certain parameters concerning the use of the authoring tool that would need to be taken into account, for example, whether students select a course of their own free will or whether the course is compulsory.

Moreover, information from student models may include different aspects that may need to be combined in order to be used for the instructor model. For example, in order to decide about popularity of a course the authoring tool should take into account the progress of students as well as the number of times that these students visit the course. If students visit the course very often but they do not have significant progress then this should not be interpreted as "popularity of the course" but rather as "high difficulty of the course". Similarly, information collected in student models about the number of errors that students make and the types of error that are mostly made, may reveal weaknesses in the tutoring part of the courses. This kind of information, as well, could be used for the instructor model because it may provide useful hints for the revision of courses.

The instructors' actions, habits and preferences when interacting with the system. The instructor model may acquire information from instructors' actions, habits and preferences. For example, if an instructor asks to see the diagnostic analysis about students, then s/he is interested in it. If an instructor takes into account the tool's suggestions about several aspects of the course, then s/he is highly interested in this kind of information.

In many cases implicit information should be combined with explicit information. For example, the categorisation of instructors as expert/novice may be based on their own beliefs about themselves and also on inferences that may be made based on statistical outcomes of their courses in relation with students' performances. In this way the information about instructors acquired by the instructor model may be both explicit and implicit. Explicit questions to instructors may include the following: "How many times have you authored your own courseware using an authoring tool?" or "Do you consider yourself an experienced author, an intermediate one or a novice one?"

Implicit information may be based on the outcomes of statistical information about students' performance in relation to the goals that the instructor had stated when s/he authored the course. For example, if an instructor had stated that certain exercises should be "difficult" and should be solved by 25% of the students and it turned out that indeed 25% of the students solved the exercises then this instructor could be considered experienced concerning the selection of exercises. If this instructor had also stated that s/he was experienced in authoring courses then this could be recorded in his/her instructor model.

Naturally, the information that may be used to categorise instructors into novice/expert may vary considerably, depending on the techniques of construction of the instructor model and the degree of detail of the domain expertise. For example, an authoring tool could use stereotypes for an initial classification of instructors and then use observational techniques to infer more information. Associating students' learning outcomes with instructors' expertise may be an interesting research topic on its own and could provide useful findings to the pedagogic area as well.

WEAR: AN AUTHORING TOOL THAT INCORPORATES INSTRUCTOR MODELLING

WEAR stands for WEb-based authoring tool for Algebra-Related ITSs. WEAR is also described in (Virvou & Moundridou, 2000a). It aims to be useful to teachers and students of domains that make use of algebraic equations, such as physics, economics, chemistry, etc. It incorporates knowledge about the construction of problems and a mechanism for student error diagnosis that is applicable to many Algebra-related domains. WEAR deals with the generation of instruction: it offers the ability of problem construction and also the ability of building adaptive electronic textbooks. In that sense it shares the same focus with tools which allow the authoring of the *domain knowledge*. On the other hand, WEAR gives instructors the ability to control the order by which students solve problems and study the teaching material by assigning a value to each problem's attribute called "level of difficulty" and by defining prerequisite relationships between topics of the electronic textbook. Therefore, WEAR is also concerned with managing the sequence of the curriculum on top of generating it. This characteristic is usually met in tools dealing with the creation of the *tutoring model*. What should be noted is that presently WEAR does not deal with authoring teaching strategies in the sense of allowing the instructor to decide when the students should be tested, or be given hints and explanations. In WEAR, when we refer to teaching strategies, we mean the combination of all choices made by the instructor, which affects the behaviour of the generated ITSs. Furthermore, WEAR seems to combine aspects of both the *pedagogy-oriented* systems and the *performance-oriented* ones (Murray, 1999): it is a performance-oriented tool since it provides an interactive environment in which students can practice and learn skills and a pedagogy-oriented one since it also deals with sequencing the teaching material.

WEAR also gives teachers the ability to search for already constructed problems and make them available to their own class. Thus, teachers can benefit from the database of problems that have been previously created using WEAR. The problems stored may have various attributes associated with them. For example, they may be relevant to a wide range of Algebra-related domains, may be addressed to different categories of student, may be associated to several levels of difficulty and so on. These differences in the problems stored indicate a need for WEAR to be adaptable to particular instructors' interests so that it may help them in the selection of the appropriate information from its database. In addition, to avoid the repetition of problems, WEAR closely monitors the instructor during the construction of a new problem and notifies him/her in case a similar problem already exists in its database. Furthermore, WEAR monitors the students' actions when interacting with the system and checks whether the progress of the course is consistent with the instructor's long-term goals; if this is not the case, the instructor is informed. For these reasons, WEAR incorporates an instructor modelling component (Virvou & Moundridou, 2000b).

In the remainder of this section we will first briefly discuss the results of an empirical study we conducted. We will then describe WEAR's architecture, operation and functionality, focusing mainly on the interactions concerning the instructors rather than on the ones concerning the students. Finally, we will describe the functionality of the instructor modelling component as this has been incorporated in WEAR.

An Empirical Study

WEAR's design and development is based on the results of an empirical study we conducted involving about 40 students and 6 instructors. In the first part of the experiment, two groups of high school students were given some problems and were asked to solve them. The first group originated from an economics class and the second from a physics class. Students were also asked to provide their previous year's marks both in the corresponding subject and in mathematics. The tests' results showed that in both domains the errors that were made fell into two major categories: the domain errors and the mathematical ones. In other words, students either could not form correctly the equations that were needed for a particular problem, or they could not solve the equations. Based on the students' errors we further divided these two error categories into sub-categories and found patterns for each of them. Students were also asked to fill in a questionnaire where among other things they specified whether they were interested in knowing their instructor's attitude in teaching. The majority of them answered positively and stated that they were most interested in knowing if the instructor was strict in grading their work and also if s/he was demanding as to the difficulty of the problems s/he gave to them.

The second part of the experiment involved instructors. To those we gave a description of the above-mentioned experiment along with its results and asked them to categorise the most frequent student errors and associate each of them to a stereotype of student. Furthermore, we asked them to grade several student papers and also to define the difficulty level of each exercise that was posed to students. The instructors' answers to all of these revealed that the attitude they had toward the teaching/learning process varied a lot. Different instructors associated different student errors to student stereotypes; they also weighted student errors and graded student papers differently and finally the level of difficulty they assigned to each problem was in some cases dissimilar.

Furthermore, instructors were asked to answer some questions concerning their teaching style and preferences. An interesting finding was that they usually wished to know how students had performed so that they could evaluate their teaching strategies with respect to their teaching goals. Finally, instructors were given a description of a hypothetical authoring tool and were asked to rate its functions and utilities. From their ratings, what is of interest in respect to the subject we are discussing in this paper is that they seemed to like the idea of being modelled by the system if this was to be used to support them in the authoring process, to help them improve their courseware or collaborate with other peers.

System's Architecture

The system's underlying architecture is shown in Figure 1. The *Authoring components* contain the system's modules dealing with courseware construction and management. These are tools for describing a domain in terms of variables and equations, associating domain variables with topics of the electronic textbook, specifying relationships between topics, uploading teaching material, managing student records, constructing new problems and tests and retrieving problems that were previously constructed. The instructor interacts with the *Authoring components*. The information that the instructor passes to the *Authoring components* is put into the database of *Domain knowledge and problems*.

The *Instructor modeller* is responsible for building and updating each instructor model. The *Instructor model* holds: i) information obtained explicitly by asking the instructors (such information may be the instructor's preferences concerning the course and his/her teaching expertise), and ii) implicit information inferred by WEAR (such as the instructor's interest in some categories of problem). The *Instructor model* provides information that is used by the system to individualise the interaction with each instructor; for example, when an instructor browses the categories of problem s/he finds already pre-selected those categories that s/he is interested in.

The *Tutoring components* consist of components that interact with students while they are solving problems, present the teaching material in an adaptive way and form individualised advice for students. To perform these tasks, the *Tutoring components* need to know who each student is and what s/he knows so far, what the structure of the domain being taught is (e.g. which are the prerequisite concepts a student should know before studying a specific concept) and which the correct equations that describe this domain are. The sources for all this information are the *Student models* and the *Domain knowledge and Problems*. A Problem Solver included in the *Tutoring components* is using its knowledge about solving systems of linear equations correctly in order to inform the *Student modeller* about the problem solving activity of the student. The Problem Solver is also using information from the *Domain knowledge and problems*. In case of an error, the *Student modeller* is responsible for diagnosing the cause of it. The *Student modeller* is also responsible for updating the *Student model* based on the student's actions when interacting with the system (reading or not topics of the electronic textbook, solving correctly or not a problem, etc.).

The resulting ITSs from WEAR have a *Student interface* that operates in two modes: the instructor and the co-student. In both cases there is a speech-driven anthropomorphic agent which provides speaking messages to the student (Virvou et al., 2000). The underlying reasoning used for both modes is based on the *Student model* and *Instructor model*. The difference of these modes is restricted to the interface level. In particular, in the instructor mode, the anthropomorphic agent simulates an instructor and provides rather formal messages to the student whereas in the co-student mode, the anthropomorphic agent simulates a co-student that provides messages in a more casual way. We consider the mode of the co-student quite important in order to promote the student's collaborative attitude similarly with other systems, such as (VanLehn, Ohlsson, & Nason, 1994). Furthermore, the presence of an animated speaking character in the interface of an educational application makes the system more appealing and attractive to students and thus increases their engagement and motivation. Indeed, previous studies (Lester et al., 1997) have revealed the so called *persona effect*, which is that "the presence of a lifelike character in an interactive learning environment – even one that is not expressive – can have a strong positive effect on student's perception of the learning experience". On the other hand, the *Instructor interface* does not include any anthropomorphic agent and it operates as a conventional GUI.

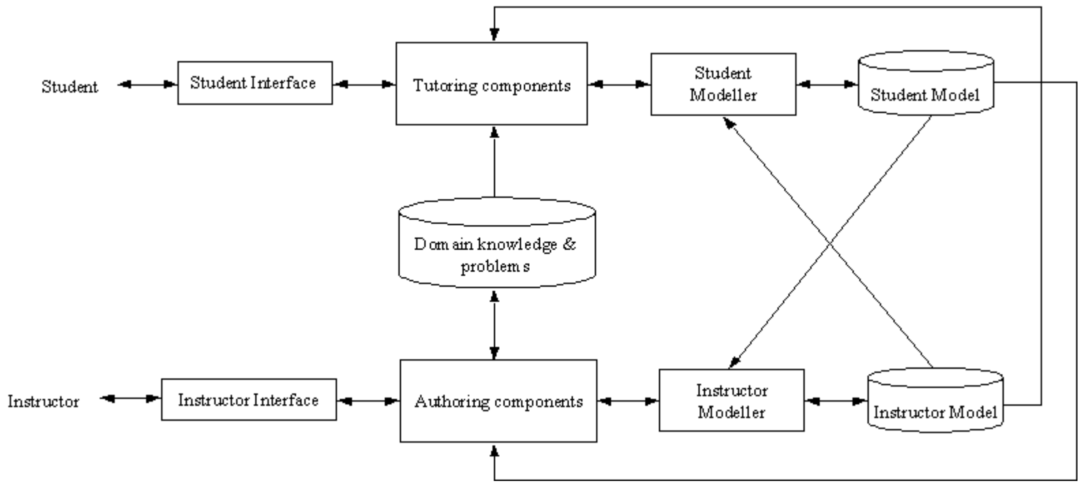
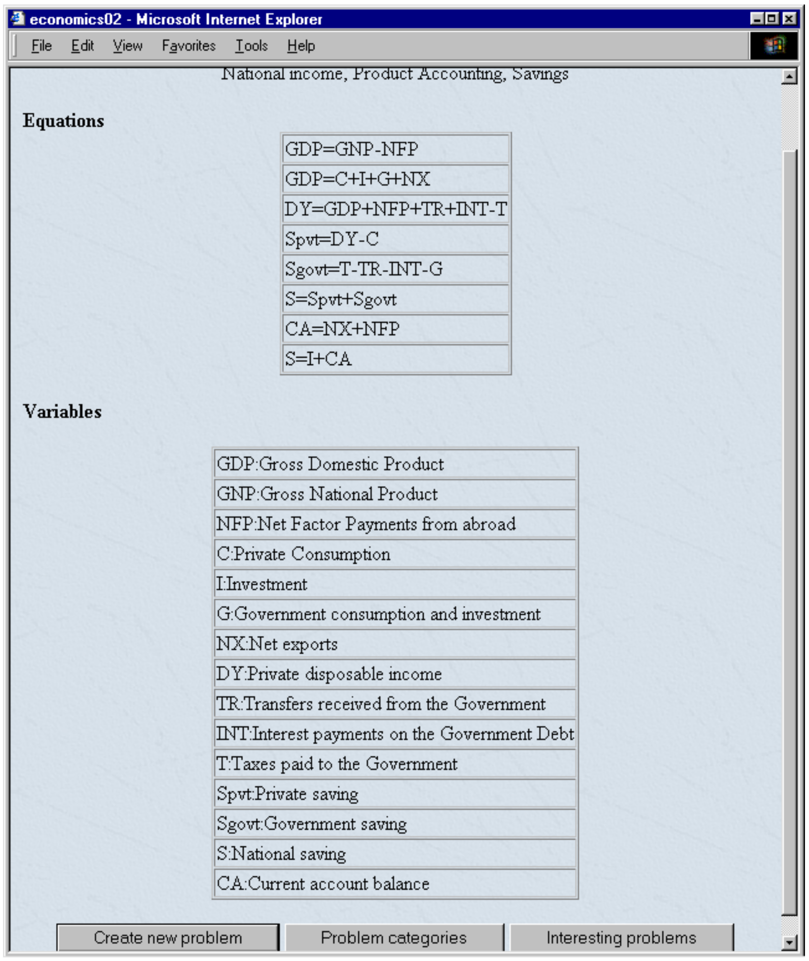


Figure 1. WEAR's architecture

As shown in the above figure, WEAR uses its instructor and student model for instructors and students respectively but also vice versa. This means that the model of each class of user is also used as a source of information to be passed to the other class of user. This is done both explicitly by informing the other class of user and implicitly by affecting the model of the other class of user. For example, instructors may be given some information about students' performance from the student model in order for instructors to evaluate their own teaching strategy. Similarly, students may be given explicitly some information about their instructors' preferences concerning their teaching strategies. In this way students may have a better idea about how to collaborate with their instructor.

The above examples concern the case when each class of user may be given explicitly some information from the model of the other class. However, most importantly the models of the two classes of user interact with each other and affect the modelling process itself. For example, the students' performance recorded in the student models is used to calculate the degree of an instructor's tendency to overestimate or underestimate the level of difficulty that s/he assigns to problems. If a high degree of such a tendency seems to exist, it is recorded in the instructor's model and used to provide individualised help to him/her (e.g. to remind him/her of this when constructing new problems). Similarly an instructor model may affect student models. For example, the students' level of knowledge, which is recorded in student models, is assessed taking into account the students' errors. These may either be mathematical or domain errors. By default WEAR considers the two kinds of error equally important; however, if an instructor model indicates a specific instructor's preference to weigh more one kind of error than the other, then the students' level of knowledge is calculated taking into account the instructor's preference.

The implementation of the system is based on the client-server architecture. WEAR resides on a Web server. Both students and instructors are clients who can use the teaching and authoring services offered by the system using a conventional Web browser.


economics02 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Create new problem
Problem categories
Interesting problems

National income, Product Accounting, Savings

Equations

GDP=GNP-NFP
GDP=C+I+G+NX
DY=GDP+NFP+TR+INT-T
Spvt=DY-C
Sgovt=T-TR-INT-G
S=Spvt+Sgovt
CA=NX+NFP
S=I+CA

Variables

GDP:Gross Domestic Product
GNP:Gross National Product
NFP:Net Factor Payments from abroad
C:Private Consumption
I:Investment
G:Government consumption and investment
NX:Net exports
DY:Private disposable income
TR:Transfers received from the Government
INT:Interest payments on the Government Debt
T:Taxes paid to the Government
Spvt:Private saving
Sgovt:Government saving
S:National saving
CA:Current account balance

Figure 2. Part of a domain description (instructor's mode)

System Operation

The tool takes input from a human instructor about a specific equation-related domain (e.g. economics). This input consists of knowledge about variables, units of measure, formulae and their relation. An example of such information from the domain of economics that an instructor has entered is illustrated in Figure 2. The instructor does not have to provide the complete list of variables and equations that describe the domain in a single session. S/he may only enter the ones that are needed to solve the problems to be constructed in the current interaction and add more in the interactions to follow. WEAR accumulates domain knowledge each time that the human instructor gives new information. This means that the instructor may provide input to the tool at the same rate as lessons progress in a course. Examples of input to the system that an instructor could provide to describe a portion of the domains of physics and economics are shown in Table 1 and Table 2 respectively.

Table 1. Input example from the domain of physics

Variable's description	**Variable's name**	
Period	T	
Frequency	f	
Speed	u	
Wavelength	λ	
Time	t	
Distance	d	
Equations		
T=1/f	u=f*λ	u=d/t

Table 2. Input example from the domain of economics

Variable's description	**Variable's name**
Gross Domestic Product	GDP
Gross National Product	GNP
Net Factor Payments from abroad	NFP
Private Consumption	C
Investment	I
Government consumption and investment	G
Net exports	NX
Private disposable income	DY
Transfers received from the Government	TR
Interest payments on the Government Debt	INT
Taxes paid to the Government	T
Private saving	Spvt
Government saving	Sgovt
National saving	S
Current account balance	CA
Equations	
GDP=GNP-NFP	Spvt=DY-C
GDP=C+I+G+NX	Sgovt=T-TR-INT-G
DY=GDP+NFP+TR+INT-T	S=Spvt+Sgovt
CA=NX+NFP	S=I+CA

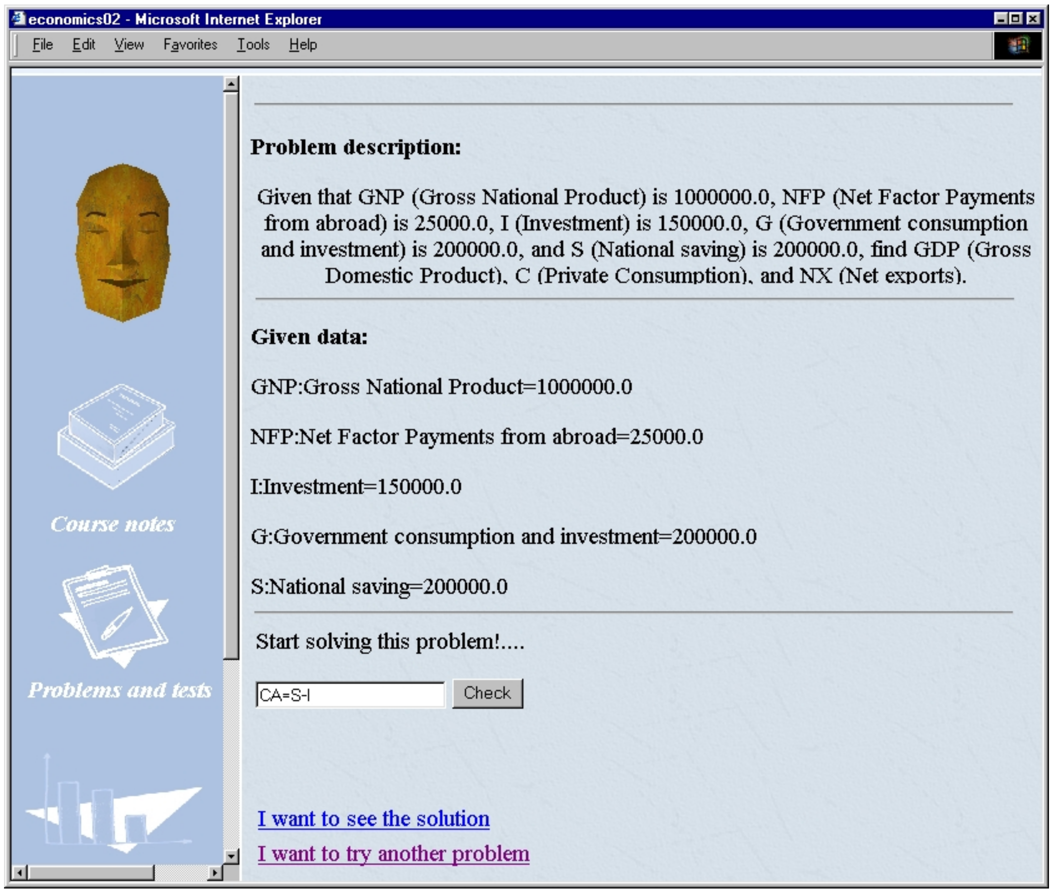
In addition, instructors are requested to classify themselves in one of three categories concerning their teaching expertise: "novice", "having little experience", and "experienced". This information is stored in their instructor model and used by WEAR to provide them with adapted help concerning teaching strategies they may select. Moreover, instructors are asked to explicitly provide their long-term preferences for the course they author in terms of the course's difficulty and popularity. WEAR compares these preferences, which are also part of the instructor's model, with the instructor's short-term goals. If an inconsistency is present, WEAR informs the instructor that the course does not seem to follow his/her long-term goals.

WEAR functions in two different modes: the instructor's mode and the student's mode. The instructor's mode is the authoring tool itself while the student's mode is the ITS that WEAR produces. When entering the initial web page of the tool the user is requested to provide his/her user name and password. Based on this information the system considers the user to be a "student" or an "instructor" and directs him/her either to the web page of the resulting ITS or to the Web page of the authoring tool, respectively. In the student's mode, students are presented with a number of problems to work on and are provided with individualised feedback while they are solving them. They also have at their disposal an electronic textbook and are offered navigation support adapted to their individual knowledge. In the instructor's mode the instructor is able to construct new problems, retrieve previously created ones and author the adaptive electronic textbook. In all cases, WEAR provides automatic assistance, as will be discussed in the subsequent sections.

Resulting ITSs

The student model that WEAR maintains is a combination of a stereotype and an overlay student model, similarly with other systems such as (Hohl, Böcker, & Gunzenhäuser, 1996). The stereotype student model (formed either directly by the instructor or after a preliminary test posed to the student) classifies initially the student according to his/her knowledge of the domain and his/her mathematical skills. As a result of this, each student is assigned to a stereotype (novice, beginner, intermediate or expert). The stereotype model defines initial values for the overlay student model. The latter is represented by a set of pairs "concept-value". The concepts are domain concepts and concepts concerning the equation solving process (e.g. isolating the unknown variable in an equation). Domain concepts include domain variables and topics constituting the teaching material. For example, each variable presented in Table 2 constitutes a domain concept for the economics domain. The value for each concept is an estimation of the student's knowledge level of this concept and it is initialised by the stereotype student model. If, for example, the stereotype model indicates that a student is "intermediate" as to his/her mathematical skills and "beginner" as to his/her knowledge in the domain, then the concepts constituting the overlay student model are given the corresponding values: every concept that concerns the equation solving process and that has not been rated by the instructor as difficult or very difficult is considered known by the student; every domain concept rated as very easy is considered already known. After the initialisation of each "concept-value" pair, the student model is updated taking into account the student's performance in solving the problems associated with this concept and the reading or not of the corresponding teaching material. For example, if a student has successfully solved all problems evaluating the domain concept "Gross Domestic Product – GDP" and s/he has also read the corresponding topics of the electronic textbook, then in his/her student model the concept "GDP" will hold the value 1 and thus it will be considered known.

Information obtained from student models as well as knowledge of the domain being taught, are exploited by WEAR to provide *adaptive navigation support* to students (Brusilovsky, 1996). To achieve this, WEAR makes use of the adaptive link annotation technique: students interacting with the system see visual cues (different icons next to each link) that inform them about the current state both of the available problems and of the topics constituting the teaching material. This is done in order to facilitate the student's choice about which problem to solve next and which topic to study, as well as to provide them with information concerning the already mastered topics and concepts.


economics02 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Course notes
Problems and tests
Check
I want to see the solution
I want to try another problem

Problem description:

Given that GNP (Gross National Product) is 1000000.0, NFP (Net Factor Payments from abroad) is 25000.0, I (Investment) is 150000.0, G (Government consumption and investment) is 200000.0, and S (National saving) is 200000.0, find GDP (Gross Domestic Product), C (Private Consumption), and NX (Net exports).

Given data:

GNP:Gross National Product=1000000.0

NFP:Net Factor Payments from abroad=25000.0

I:Investment=150000.0

G:Government consumption and investment=200000.0

S:National saving=200000.0

Start solving this problem!....

CA=S-I

Figure 3. Solving a problem while in student's mode

When a student attempts to solve a problem, the system provides an environment where the student gives the solution step by step (Figure 3). At first the student is presented with a problem statement like the one shown in Figure 3. Another problem from the domain of physics could be like this: "The annoying sound from a mosquito is produced when it beats its wings at the average rate of 600 wingbeats per second. What is the frequency in Hertz of the sound wave? Assuming the sound wave moves with a velocity of 340 m/s, what is the wavelength of the wave?"[1]. The student is requested to write down the equations that are needed to solve the problem and then s/he is requested to mathematically solve the problem. To detect the erroneous answers the system compares the student's solution to its own at every step. The system's solution is generated by WEAR's Problem Solver, which is implemented in PROLOG. The Problem Solver incorporates knowledge about how to solve systems of linear equations correctly and may generate the solution to a problem using information about the specific domain to which the problem belongs (e.g. physics). During the process of solving a problem the student's actions are monitored by the system. In case of an erroneous action the Problem Solver passes the student's answer to the Student modeller, which is then responsible for diagnosing the cause of the error. The errors that are recognised by WEAR's Student modeller are the following:

- Domain errors. These include errors that are due to the student's unfamiliarity with the domain being taught. Such errors occur when the student enters an equation different from the one needed for the problem to be solved, or when s/he enters an "almost" correct

[1] This example problem statement is taken from: http://www.glenbrook.k12.il.us/gbssci/phys/Class/waves/u10l2e.html

equation (lacking a variable, using an erroneous relationship between variables, etc.). For example, if a student enters the equation u=d*t instead of u=d/t, then the error is attributed to the category of Domain errors and in particular to the sub-category of “erroneous relationship between variables”.

- Mathematical errors. These include errors that are due to the student’s unfamiliarity with solving mathematical equations. Such errors could be calculation errors, errors in isolating the unknown variable, etc. For example, if a student trying to isolate d in the equation u=d/t enters d=u/t instead of d=u*t, then the error is attributed to the category of Mathematical errors and in particular to the sub-category of “wrong isolation of the unknown variable”.

As has been already mentioned, the student interface includes an animated speaking face which is representing either the instructor or a co-student and is responsible for communicating the instructions and any feedback messages to the students.

Authoring by the Instructor: Construction of problems

When an instructor wishes to create problems s/he is guided by the system through a step by step procedure. At each step of this procedure the instructor should specify values for some parameters needed to construct a problem. Such parameters could be for example what is given and what is asked in the problem to be constructed. After the completion of this procedure the tool constructs the full problem text and provides consistency checks that help the instructor verify its completeness and correctness. In case of redundancies in the given data the tool lets the instructor know. After the construction of a problem the tool lets the instructor preview the problem text and the solution of the problem as formulated by the system. At this point, the instructor is asked to assign to the problem the appropriate “level of difficulty”. The system uses this measure in order to suggest to each student (while in student’s mode) what problem to try next.

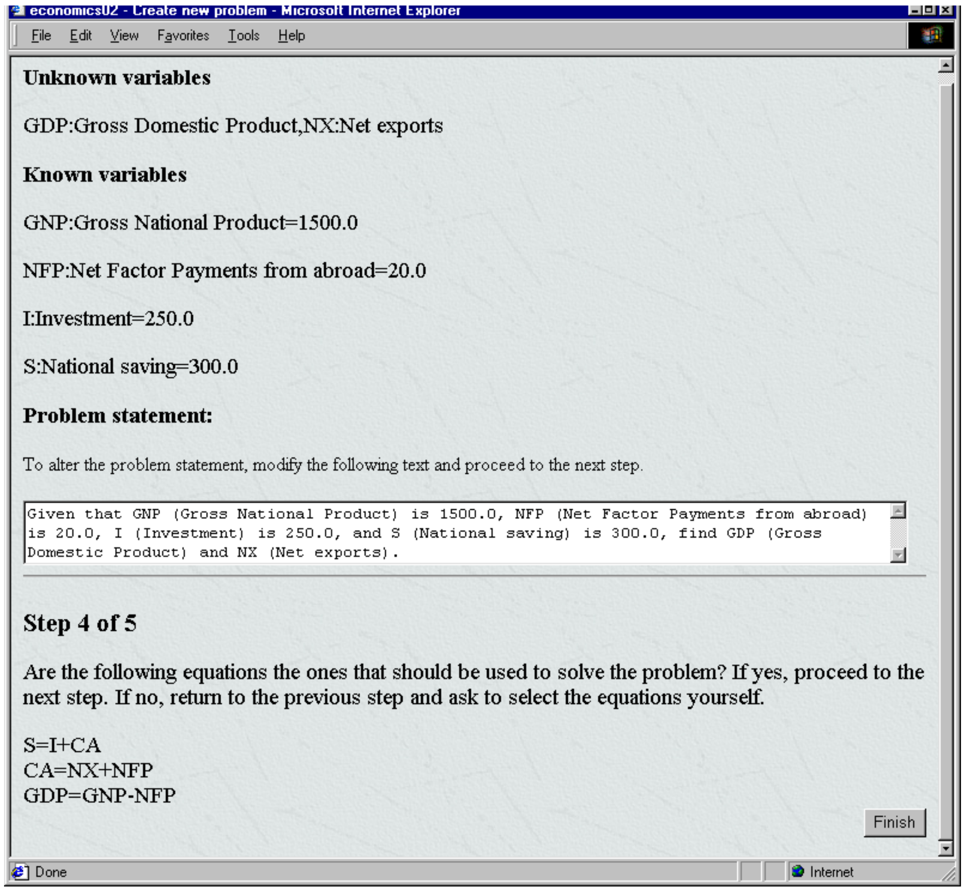
While students are tackling the given problems the system collects evidence about the level of difficulty so that it can provide feedback to the instructor. For example, if the majority of the students of a certain level have failed in solving a particular problem, which has been assigned to this level, then the instructor is informed. In a case like this, perhaps the instructor may wish to reconsider the level of difficulty since there is evidence that the problem may be of a higher level of difficulty. On the other hand, if many students have managed to solve a problem of a higher level of difficulty than the one proposed by the instructor, the level of difficulty may have been overestimated by the instructor. In this case too, the system informs the instructor. In both cases, the tool does not take the initiative to alter the level of difficulty by itself: it suggests the instructor increase or decrease this measure according to the observed students’ performance in a specific problem. In this way an instructor is being assisted by the system in the classification of problems.

There are mainly two types of problems that the system can assist the instructor to construct:

Non-numerical problems. In problems without numbers the system displays every variable that the human instructor has entered when describing the domain. The human instructor should specify which variable(s) is (are) the unknown, which one is given and the type of change. For example in the domain of economics the instructor could select as unknown the variables “exchange rate” and “net exports”, and as given a “decrease” in the level of the “foreign output”. The system would then produce the following problem text: “How will the decrease of the foreign output affect the exchange rate and net exports?”. This kind of problem evaluates the students’ knowledge of the equations involved in each of these problems. In addition, it evaluates the students’ ability to decide about the influence of each variable over the others. In cases like this, students are not requested to solve a particular system of equations, but rather to work with proportions. In this way, such problems might measure the students’ overall understanding in the domain being taught.

Numerical problems. In problems with numbers the system displays again every variable that the human instructor has entered and requests the unknown. The system considers automatically all the variables, which depend on the “unknown” (according to the equations), as possible given data. These variables are shown to the instructor who should now enter their values. The system follows the instructor’s actions and reports any inconsistencies. For example, if the instructor enters values for fewer variables than those needed for the problem to be solvable then the system points out the error. Finally, the system produces a simple problem text describing the given and asked data, which the instructor may change to make it more realistic and comprehensible. The information concerning the known and unknown variables is used by WEAR to examine the domain equations and isolate the ones that are needed for the problem to be solved (Figure 4). In some domains more than one equation exists that defines the same variable. In such cases WEAR finds all the relevant equations to the variables selected. Then it presents them to the instructor to choose by himself/herself the appropriate ones for the problem s/he is currently constructing. In this category of problem, the students are tested over their ability to solve a system of linear equations (mathematical skills) and their knowledge of the equations describing the particular domain.

Beyond these two types of problems, WEAR also offers instructors the ability to create multiple-choice tests. Since there are topics of the curriculum that can not be assessed by problems such as the above mentioned, by using multiple-choice tests instructors can be aware of their students’ performance and understanding in these topics as well.


economics02 - Create new problem - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Finish
Done
Internet

Unknown variables

GDP:Gross Domestic Product,NX:Net exports

Known variables

GNP:Gross National Product=1500.0

NFP:Net Factor Payments from abroad=20.0

I:Investment=250.0

S:National saving=300.0

Problem statement:

To alter the problem statement, modify the following text and proceed to the next step.

```
Given that GNP (Gross National Product) is 1500.0, NFP (Net Factor Payments from abroad)
is 20.0, I (Investment) is 250.0, and S (National saving) is 300.0, find GDP (Gross
Domestic Product) and NX (Net exports).
```

Step 4 of 5

Are the following equations the ones that should be used to solve the problem? If yes, proceed to the next step. If no, return to the previous step and ask to select the equations yourself.

S=I+CA
CA=NX+NFP
GDP=GNP-NFP

Figure 4. Problem construction

Authoring by the Instructor: Selection of existing problems

Beyond constructing a problem by himself/herself, the instructor has the ability to explore the problems constructed by others and choose the ones that s/he desires to be accessible by his/her class. Since new problems (belonging to different domains, involving different variables, etc.) can be continuously added to the system, there is no way for the system to have fixed categories of problem. Every time an instructor constructs a new problem the system performs this problem's categorisation based on some parameters. The problems are first categorised according to the domain to which they belong. At a second level the problems of each domain are categorised according to the variables they involve and their level of difficulty. Every variable of the domain can possibly form a problem category. For example, a problem like: "A force of 100 Newtons is acting on a 25 kg object which is initially stable. After 10 secs how much is the impulse?" belongs to the broad category "Physics" and in the sub-categories "Impulse", "Velocity" and "Acceleration" due to the variables involved in it. The same problem could also belong to the sub-category "level of difficulty 1" based on the problem's level of difficulty as this has been defined by the instructor.

The categorisation of each problem according to the variables involved is achieved by the system through the following algorithm:

1. Assign the problem to the categories of every unknown variable in this problem (In the above example this step results in the category "Impulse" which is the unknown variable).
2. Search the equations that must be solved in order to define the unknown variables' values (in the example, that is the equation $J=m*v$) for the independent variables that are not "given" in this problem and assign the problem to these categories too; in our case that is the variable-category "Velocity". Consider as "unknown" these variables that must be given a value by solving an equation and repeat this step; this will end in assigning the problem to the category "Acceleration", since $v=v_0+a*t$ and "a" is not "given".

Instructors are allowed either to browse the collection of problems by selecting the categories and sub-categories that match their needs and interests, or to search the entire collection using some keywords. An instructor modelling mechanism incorporated in the system is responsible for tailoring the interaction of the instructors with the system to the instructors' needs. The exact way in which this adaptation of the interaction to the instructors' needs is performed is described in the next section ("Instructor Modelling in WEAR").

Authoring by the Instructor: Adaptive textbooks

WEAR allows the authoring of electronic textbooks by instructors and delivers them over the WWW to learners (Moundridou & Virvou, 2001). These textbooks offer navigation support to students, adapted to their individual needs and knowledge. The authoring procedure to create an adaptive electronic textbook with WEAR is quite simple. In particular, the instructor should prepare HTML files for the topics that would be contained in the electronic textbook. The next step is to use WEAR's facilities for uploading these files to the WEAR server. For each uploaded file the instructor must specify a title, a difficulty level and the position that it should have in the topics hierarchy. S/he should also relate topics to the domain variables. Finally, the instructor must edit the *is_prerequisite_of* and *is_related_to* relationships between topics. This information is used to form WEAR's *domain model*. The domain and student models are used by WEAR to generate a table of contents for each student. This table of contents consists of links to each topic of the textbook. These links are annotated in order to inform students about the educational appropriateness of the topic behind them.

When building an electronic textbook, instructors are provided with tools that verify the consistency of the course and report possible problems or errors, such as the case when the prerequisite relationships imply that a topic indirectly requires the knowledge of itself. To offer more intelligent and individualised help WEAR relies on the information provided by the instructor modelling component that it embodies.

Instructor Modelling in WEAR

The instructor modelling component monitors each instructor's interactions with WEAR and constructs and/or updates his/her user model. As we have already mentioned, an instructor either searches for an already constructed problem or constructs a problem by himself/herself. WEAR infers from these two actions the user's interest or contribution to something, respectively. This is similar to a system called InfoVine (Harvey, Smith, & Lund, 1998), which infers that users are interested in something if they are repeatedly asking the system about it whereas they are expert in something if they are repeatedly telling the system about it. In WEAR, when a user frequently searches for specific categories of problem then it is inferred that this particular user is "interested" in these categories of problem; when a user often constructs problems that belong to the same category, the inference made is that this user is a "major contributor" in that sort of problem. In particular, the instructor aspects that are being modelled in WEAR, are the following:

- *Instructor's preferences*.
  In WEAR the instructor may give some long-term preferences as to whether s/he wishes the course to be difficult, average or easy or whether s/he wishes it to be very popular or fairly popular or whether s/he is not interested in this feature. Each of these preferences is associated with a percentage of failure in performances of class students and student interest in the course (e.g. how many times students visit the electronic textbook, and/or how many problems they have solved). The instructor may also state how important s/he considers each category of student error to be. In that way the students' level of knowledge could be calculated according to the instructor's preferences, assigning higher weight to those errors that the instructor has defined as more important.

- *Instructor's usual activities*.
  Instructor's activities that are frequent are recorded in his/her long-term model. For example, if an instructor constructs exercises frequently then s/he is recorded as a major contributor.

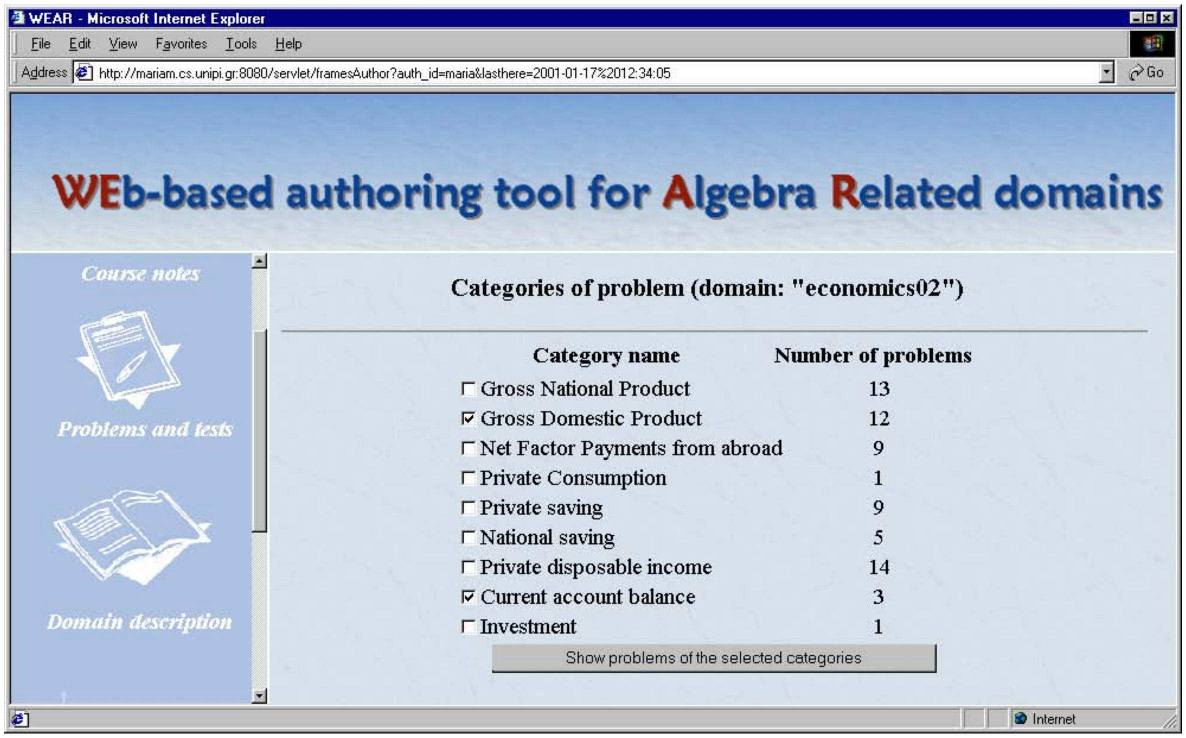
- *Instructor's special interests*.
  The instructor's interests are inferred and recorded in the long-term instructor model. For example, what kind of exercises the instructor is interested in or whether s/he is interested in the diagnostic statistics about students depending on whether s/he asks to see them or not.

- *Instructor's level of expertise in teaching*.
  As we mentioned earlier, the level of expertise of instructors can be recorded along three dimensions: (i) software use, (ii) teaching and (iii) domain. In WEAR the instructor model records the teaching expertise of the instructor. This is explicitly stated by the instructor himself/herself. Each instructor may situate himself/herself in one of three categories: novice, having little experience, experienced. In the case of novice tutors and those having little experience, the authoring tool offers more detailed help concerning the teaching strategies that the tutor may select and shows him/her by default the results of the consistency checks.

The instructor model is utilised by the system in the following ways:

Provide individualised help to the instructor. WEAR uses the instructor model in order to offer individualised help to the instructor with respect to his/her teaching strategies. For example, if an instructor has stated a long-term goal that s/he wishes to render the course popular within the class students then the authoring tool will examine whether the instructor's short-term goals are consistent with his/her long-term goals. Student models provide information about how many students have attempted certain exercises and how many times they have seen certain lectures. This information is used to let the instructor know how well s/he does with his/her predefined teaching strategy. Similarly, if an instructor has been recorded in his/her long-term model as having the tendency to overestimate or underestimate the level of difficulty of problems, s/he will be reminded of that by the authoring tool when inserting new problems. Furthermore, WEAR also performs more thorough checks: for instance, if the majority of students fail to comprehend a specific topic (indicated by low scores in the corresponding tests), then the instructor is informed and given some suggestions concerning this situation (e.g. the underlying reason for the students' failure may be the misplacement of the specific topic in the curriculum, or it may be that the test was too difficult). At the moment WEAR restricts the individualised help that it offers to the teaching strategies. As we mentioned earlier, we believe that individualised help could also be offered with respect to the software use and the particular domain taught. This kind of help is within the future plans of this research.

Adapt the interaction with instructors. When an instructor wishes to find a problem and decides to browse the available categories, s/he will see that in the categories' list the ones that s/he frequently explores are pre-selected for him/her by the system (Figure 5). Of course the instructor is free to select some other categories as well, or even ignore the already selected ones. In addition, if new problems belonging to the categories that a particular user is interested in are added, the system informs the user when s/he logs in. When a user searches the collection of problems using some keywords instead of selecting categories, his/her search is saved and the next time s/he logs in and wishes to search the collection s/he is presented with the option to run again the last saved search.


WEAR - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://mariam.cs.unipi.gr:8080/servlet/framesAuthor?auth_id=maria&lasthere=2001-01-17%2012:34:05
Go
WEb-based authoring tool for Algebra Related domains
Course notes
Problems and tests
Domain description
Show problems of the selected categories
Internet

Categories of problem (domain: "economics02")

Category name	Number of problems
☐ Gross National Product	13
☑ Gross Domestic Product	12
☐ Net Factor Payments from abroad	9
☐ Private Consumption	1
☐ Private saving	9
☐ National saving	5
☐ Private disposable income	14
☑ Current account balance	3
☐ Investment	1

Figure 5. Pre-selected categories of problem according to the instructor's inferred interest

Promote co-operative or collaborative work among instructors. Users are offered the choice of seeing what other users have done along two dimensions: the course structure and the constructed problems. Concerning the former, the information that is presented to the instructor is the structure of a similar course created by another instructor. The similarity of the courses is calculated in terms of the domain to which the course belongs and in terms of the difficulty level assigned to it by its author. In particular, the instructor may see an enriched Table of Contents presenting not only the topic hierarchy but also the *prerequisite* and *is_related* relationships between topics. In that way, instructors who may be novice as course designers could be assisted by more experienced peers who have previously used WEAR. When selecting to see problems constructed by others, the instructor is presented with a list of problems constructed by instructors who are considered by the system as “major contributors” in the categories that this specific instructor is considered “interested”. An instructor is considered a “major contributor” in a particular area if s/he has created many problems in this area and these have been solved by a number of students. In addition, while an instructor is constructing a new problem by himself/herself the system is checking whether there is any similar problem already constructed by another instructor who is considered “major contributor”. If this is the case, the instructor is offered the choice of seeing the similar problems and using them instead of completing the construction of his/her own problem. In that way, the system avoids the repetition of problems, facilitates the instructors’ work and advances the co-operation and collaboration among them.

Provide a more personalised learning environment for the student. As we have already mentioned (in the section named “System’s Architecture”), the interface with the student works through a speech-driven anthropomorphic agent and it operates in two modes: the instructor mode and the co-student mode. In both cases of the instructor and the co-student mode, the instructor model is used to pass some information about the instructor to the student. For example, as the empirical study has shown, students would like to know to some extent who their instructor is; what the instructor expects from the students or what his/her priorities are in terms of the course taught, how difficult the exercises s/he gives are, etc. Usually, they seek such information from older students who had the same instructor in the past, or they ask such questions to their instructors themselves. However, such information may be passed to them from the instructor modelling component through the student interface. This is done in order to render the interaction with the ITS more personalised for the student. Therefore, the virtual co-student uses information from the instructor modelling component to answer questions of students that concern certain aspects of their instructors, e.g. what the success rate was in the previous course given or whether the course is considered by the instructor as hard or not. However, these pieces of information are passed to the student only if the instructor has agreed to this.

CONCLUSIONS AND FUTURE WORK

In this paper we discussed the need of incorporating an instructor modelling component in the architecture of ITS authoring tools. From our experience with WEAR we draw the conclusion that an instructor modelling component could constitute a useful part of almost any ITS authoring tool. Referring to the available literature on authoring systems we pointed out how we believe that such a component (that seems to be absent in most systems) could lead to the construction of more effective ITSs. Furthermore, we commented that enriching the role of instructors by providing them with relevant information throughout the ITS development life cycle could also advance the ITSs’ quality.

We then presented WEAR, a Web-based authoring tool for Intelligent Tutoring Systems in Algebra-related domains. We showed how an instructor modelling component may render the system more flexible and adaptable to particular instructors’ interests and needs. The instructor modelling component interacts with other components of the system in order to form an instructor model which is used for tailoring advice to the individual instructor.

WEAR’s development phase has now reached its completion. However, WEAR has not been fully evaluated yet. For an exhaustive evaluation to take place we will need to test WEAR’s capabilities during term time in real educational settings. The instructors that will take part in the evaluation will need to be trained first. WEAR's evaluation will involve both students and instructors of various Algebra-related domains. The aspects that will be evaluated for both classes of user include WEAR's consistency, usability, friendliness and completeness. Furthermore, since WEAR's innovative feature is its instructor modelling capability, this will be an important part of the evaluation. In this way, we will test the hypothesis that this novel component in the architecture of an ITS authoring tool is beneficial to both classes of user: students and instructors. However, in order to evaluate certain instructor modelling aspects such as the provision of help for multiple iterations of the resulting ITSs' life cycle as well as the encouragement of collaboration among authors, we will need to conduct experiments that will last for more than one term and will include more than one instructor teaching similar domains.

Nevertheless, results of evaluations of other authoring tools seem to be in agreement with what we believe the instructors' attitude towards such tools is. In addition, the results from the empirical study that we conducted involving instructors and students showed that an instructor modelling component would indeed be beneficiary to instructors.

There is no doubt that each authoring tool, depending on the category in which it belongs, could form an instructor model in different ways and utilise the information stored in it accordingly. However, the specific methods and techniques of acquisition of an instructor model and the particular ways of its use in various authoring tools could be open for further research in the area of ITS authoring tools.

REFERENCES

Ainsworth, S., Grimshaw, S. and Underwood, J. (1999). Teachers implementing pedagogy through REDEEM. *Computers & Education*, 33, 171-187.

Anderson, J.R., Boyle, C.F., Corbett, A. and Lewis, M. (1990). Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42, 7-49.

Andriessen, J. and Sandberg, J. (1999). Where is education heading and how about AI?. *International Journal of Artificial Intelligence in Education*, 10, 130-150.

Barra, M., Negro, A. and Scarano, V. (1999). When the Teacher Learns: a Model for Symmetric Adaptivity. In Brusilovsky, P. and De Bra P. (Eds.): *Proceedings of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW*, Computing Science Report No. 99-07, Eindhoven, Eindhoven University of Technology, 21-28.

Boyle, T. (1997). *Design for Multimedia Learning*. NY: Prentice Hall.

Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3), 87-129.

Brusilovsky, P. (2000). Course Sequencing for Static Courses? Applying ITS Techniques in Large-Scale Web-Based Education. In Gauthier, G., Frasson, C. and VanLehn, K. (Eds.): *Intelligent Tutoring Systems, Proceedings of the 5th International Conference on Intelligent Tutoring Systems-ITS 2000,* Lecture Notes in Computer Science, Vol. 1839, Springer, Berlin, 625-634.

Burton, R.R. and Brown, J.S. (1976). A tutoring and student modelling paradigm for gaming environments. In Colman, R. and Lorton, P.Jr. (Eds.): *Computer Science and Education,* ACM SIGCSE Bulletin, 8(1), 236-246.

Dillenbourg, P. (1990). The Design of a Self-Improving Tutor: PROTO-TEG. *Instructional Science,* 18(3), 193-216.

Hartley, J.R. and Sleeman, D.H. (1973). Towards intelligent teaching systems. *International Journal of Man-Machine Studies,* 5, 215-236.

Harvey, C.F., Smith, P. and Lund, P. (1998). Providing a networked future for interpersonal information retrieval: InfoVine and user modelling. *Interacting with Computers,* 10, 195-212.

Hayashi, Y., Ikeda, M., Seta, K., Kakusho, O. and Mizoguchi, R. (2000). Is What You Write What You Get?: An Operational Model of Training Scenario. In Gauthier, G., Frasson, C. and

VanLehn, K. (Eds.): *Intelligent Tutoring Systems, Proceedings of the 5th International Conference on Intelligent Tutoring Systems-ITS 2000,* Lecture Notes in Computer Science, Vol. 1839, Springer, Berlin, 192-201.

Hohl, H., Böcker, H. and Gunzenhäuser, R. (1996). Hypadapter: An Adaptive Hypertext System for Exploratory Learning and Programming. *User Modeling and User Adapted Interaction,* 6(2-3), 131-155.

Jin, L., Chen, W., Hayashi, Y., Ikeda, M. and Mizoguchi, R. (1999). An Ontology-Aware Authoring Tool - Functional structure and guidance generation -. In Lajoie, S. and Vivet, M. (Eds.): *Proceedings of the 9th World Conference on Artificial Intelligence in Education-AIED'99,* Frontiers in Artificial Intelligence and Applications, Vol. 50, IOS Press, Amsterdam, 85-92.

Kinshuk and Patel, A. (1996). Intelligent Tutoring Tools: Redesigning ITSs for Adequate Knowledge Transfer Emphasis. In Lucas, C. (Ed.): *Proceedings of 1996 International Conference on Intelligent and Cognitive Systems*, IPM, Tehran, 221-226.

Koedinger, K.R., Anderson, J.R., Hadley, W.H. and Mark, M.A. (1997). Intelligent Tutoring Goes to School in the Big City. *International Journal of Artificial Intelligence in Education,* 8, 30-43.

Lajoie, S.P. and Lesgold, A. (1989). Apprenticeship training in the workplace: Computer coached practice environment as a new form of apprenticeship. *Machine-Mediated Learning,* 3, 7-28.

Lester, J., Converse, S., Kahler, S., Barlow, S., Stone B. and Bhogal R. (1997). The persona effect: affective impact of animated pedagogical agents. In Pemberton, S. (Ed.): *Human factors in computing systems, CHI'97 conference proceedings*, ACM Press, New York, 359-366.

Major, N., Ainsworth, S. and Wood, D. (1997). REDEEM: Exploiting Symbiosis Between Psychology and Authoring Environments. *International Journal of Artificial Intelligence in Education*, 8, 317-340.

Mark, M.A. and Greer, J.E. (1991). The VCR tutor: Evaluating instructional effectiveness. In Hammond, K.J. and Gentner, D.Q. (Eds.): *Proceedings of 13th Annual Conference of the Cognitive Science Society,* Lawrence Erlbaum Associates, Hillsdale, NJ, 564-569.

Mizoguchi, R. (1993). Knowledge acquisition and ontology. In Fuchi, K. (Ed.): *Proceedings of KB&KS'93, 1st International Conference on Building and Sharing of Very Large-Scale Knowledge Bases*, JIPDEC, Tokyo, 121-128.

Moundridou, M. and Virvou, M. (2001). Authoring and Delivering Adaptive Web-Based Textbooks Using WEAR. In *Proceedings of ICALT-2001, IEEE International Conference on Advanced Learning Technologies*, to appear.

Munro, A., Johnson, M., Pizzini, Q., Surmon, D., Towne, D. and Wogulis, J. (1997). Authoring Simulation-centered tutors with RIDES. *International Journal of Artificial Intelligence in Education*, 8, 284-316.

Murray, T. (1996). Having it All, Maybe: Design Tradeoffs in ITS Authoring Tools. In Frasson, C., Gauthier, G. and Lesgold, A. (Eds.): *Intelligent Tutoring Systems, Proceedings of the 3rd International Conference on Intelligent Tutoring Systems-ITS'96,* Lecture Notes in Computer Science, Vol. 1086, Springer, Berlin, 93-101.

Murray, T. (1998). Authoring Knowledge Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design. *Journal of the Learning Sciences,* 7(1), 5-64.

Murray, T. (1999). Authoring Intelligent Tutoring Systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education,* 10, 98-129.

Nkambou, R., Frasson, C. and Gauthier, G. (1998). A new approach to ITS-curriculum and course authoring: the authoring environment. *Computers & Education,* 31, 105-130.

O'Shea, T. (1979). A self-improving quadratic tutor. *International Journal of Man-Machine Studies,* 11, 97-124.

O'Shea, T. and Sleeman, D.H. (1973). A design for an Adaptive Self Improving Teaching system. In Rose, J. (Ed.): *Advances in Cybernetics and Systems*, 3, Gordon & Breach Publishers, London.

Rich, E. (1979). User Modelling via Stereotypes. *Cognitive Science*, 3(4), 329-354.

Rich, E. (1983). Users as Individuals: Individualizing User Models. *International Journal of Man-Machine Studies*, 18, 199-214.

Shute, V., Glaser, R. and Raghaven, K. (1989). Inference and Discovery in an Exploratory Laboratory. In Ackerman, P.L., Sternberg, R.J. and Glaser, R. (Eds.): *Learning and Individual Differences*, San Francisco, Freeman, 279-326.

Sparks, R., Dooley, S., Meiskey, L. and Blumenthal, R. (1998). The LEAP Authoring Tool: Supporting complex courseware authoring through reuse, rapid prototyping, and interactive visualizations. *International Journal of Artificial Intelligence in Education*, 10, 75-97.

Towne, D. (1997). Approximate reasoning techniques for intelligent diagnostic instruction. *International Journal of Artificial Intelligence in Education,* 8, 262-283.

VanLehn, K., Ohlsson, S. and Nason, R. (1994). Applications of Simulated Students: An Exploration. *Journal of Artificial Intelligence in Education*, 5(2), 135-175.

Virvou, M. and DuBoulay, B. (1999). Human Plausible Reasoning for Intelligent Help. *User Modeling and User-Adapted Interaction*, 9(4), 321-375.

Virvou, M. and Kabassi, K. (2000). An Intelligent Learning Environment for Novice Users of a GUI. In Gauthier, G., Frasson, C. and VanLehn, K. (Eds.): *Intelligent Tutoring Systems, Proceedings of the 5th International Conference on Intelligent Tutoring Systems-ITS 2000,* Lecture Notes in Computer Science, Vol. 1839, Springer, Berlin, 484-493.

Virvou, M. and Moundridou, M. (2000a). A Web-Based Authoring Tool for Algebra-Related Intelligent Tutoring Systems. *Educational Technology & Society*, 3(2), 61-70.

Virvou, M. and Moundridou, M. (2000b). Modelling the instructor in a Web-based authoring tool for Algebra-related ITSs. In Gauthier, G., Frasson, C. and VanLehn, K. (Eds.): *Intelligent Tutoring Systems, Proceedings of the 5th International Conference on Intelligent Tutoring Systems-ITS 2000,* Lecture Notes in Computer Science, Vol. 1839, Springer, Berlin, 635-644.

Virvou, M., Sgouros, N., Moundridou, M. and Manargias, D. (2000). Using a speech-driven, anthropomorphic agent in the interface of a WWW educational application. In Bourdeau J. and Heller R. (Eds.): *Proceedings of ED-MEDIA 2000, World Conference on Educational Multimedia, Hypermedia & Telecommunications*, AACE, Charlottesville VA, 1724-1726.

Virvou, M. and Tsiriga, V. (2000). Involving Effectively Teachers and Students in the Life Cycle of an Intelligent Tutoring System. *Educational Technology & Society*, 3(3), 511-521.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. Los Altos, CA: Morgan Kaufmann.

Woolf, B.P. and Cunningham, P.A. (1987). Multiple knowledge sources in intelligent teaching systems. *IEEE Expert*, 2(2), 41-54.

Wu, H., Houben, G.J. and De Bra, P. (1999). Authoring Support for Adaptive Hypermedia Applications. In Collis, B. & Oliver, R. (Eds.): *Proceedings of ED-MEDIA '99, World Conference on Educational Multimedia, Hypermedia & Telecommunications*, AACE, Charlottesville VA, 364-369.