Computer-Assisted Learning Using a Dialogue System for Virtual Teacher−Student Communication

Irene Luque Ruiz,* Gonzalo Cerruela García, and Miguel Ángel Gómez-Nieto

Department of Computing and Numerical Analysis, University of Córdoba, Campus Universitario de Rabanales, Building Einstein, Plant-3, E-14071 Córdoba, Spain

Received February 9, 2003

Over the past decade, computer-assisted learning in the field of chemistry has given rise to a large number of systems that approach this objective from different viewpoints: static courses aimed at specific concepts, tutorial systems, 2D and 3D virtual environments, and so on. Correct structuring and representation of the knowledge to be taught, the building of a suitable student interface, and the adaptation of the learning process to the knowledge of the student are but a few of the challenges to be faced in the development of efficient computer-assisted learning systems. The present study tackles the use of computerized dialogue systems as a viable alternative for the simulation of teacher−student interaction and proposes an ontology for the characterization of such interaction, employing the object-oriented paradigm in the modeling of both the knowledge to be taught and the actual level of the student. The proposed solution is based on the representation of the knowledge to be taught through a network of multiconnected knowledge frames (chunks), where each chunk may be specialized in more specific frames (prerequisites and subobjectives), contain associated explanations of varying complexity and with a range of explanatory models, and be associated with one or a set of possible questions the student might ask; to this end, a constantly evolving knowledge model is maintained throughout the explanatory process. Based on the proposed model, Java was used to develop and manage an explanatory system that could be used in any type of teaching system based on student-dominated dialogues. Here, the system has been applied to the teaching of chemistry laboratory practice by its integration into the Virtual Chemistry Laboratory (VCL), a system designed by the authors to simulate chemistry techniques in a virtual 3D world.

1. INTRODUCTION

The development of computer-assisted learning systems has exploded in recent years; such systems are widely used in fields such as chemistry, where numerous tutorial and learning systems have been developed to support both theoretical and practical work.[1−10]

Lately, advances in computing and resources have allowed the development of systems that aim to place the student in a virtual world—the closest thing to a real chemistry lab—in order to carry out experiments that are either considered too dangerous or expensive or when there is simply a lack of laboratory resources and/or time for large numbers of students. These virtual systems provide the (virtual) material and experimental processes required for a chemistry experiment to be performed either manually or automatically.[8,10,11]

One of the main limitations of such teaching systems is the low level of explanation normally supplied. Despite the student being provided with a virtual environment that presents complete textual or graphical information about the experiment, little student intervention is permitted in the sense of asking questions about certain concepts, operations, or methods involved in the experiment. Questions such as "Why do this?", When should I do this?", "Is it dangerous?", "What is this?", or "How do I calculate this?" frequently arise in real-world teaching and cannot be effectively answered by a virtual system.

The inclusion of this kind of functionality in teaching aids involves the modeling and development of an explanatory subsystem which, when incorporated into the virtual environment, is able to conduct a dialogue with the student and that will explain the concepts involved in the knowledge being imparted, while answering any questions or doubts about the experiment and the concepts that have been explained.

The nature of human dialogue has been studied by researchers working in many fields;[12−20] these have been directed toward learning the structure of human dialogue in order to extract patterns for use in its reproduction and for the possible construction of a computerized simulation system using artificial intelligence (AI) techniques. Current technology will only permit a satisfactory realization of this objective if the terms or area of a dialogue are restricted and already known to the system, if major progress is made in the dialogue characterization process, and if models are created to enable the design and construction of systems for specific knowledge frames or areas which could be integrated into *Intelligent Tutorial Systems* (ITS) for use as teaching aids.

For the greater part, ITS are based on AI developments and may be defined as computer-based teaching/learning programs whose ultimate aim is to provide personalized learning systems and whose main difference with respect to

* Corresponding author phone: +34-957-212082; fax: +34-957-218630; e-mail: ma1lurui@uco.es.

10.1021/ci0340254 CCC: $25.00
Published on Web 07/04/2003

classical tutorial systems lies in the way in which the two designs are conceived.[21−24] Often, given the implicitly ambitious nature of both projects and products, ITS remain somewhat limited owing not only to the difficulty in communicating with the student but also to the subsequent drawing of conclusions about his or her knowledge and characteristics in order to develop a more personalized form of training.

There would appear to be certain agreement about the structure and components of ITS, as far as the instructional component, the human-machine communication component, and the expert knowledge component are concerned. Two basic models are mainly used in the building of the knowledge component: the rule-based model and the diagram-based model. Most systems so far developed do, however, include features from both approaches—production rules and conceptual hierarchies. With regard to the learning process, the current tendency is to produce tutorials that make use of all the techniques to control initiative in the process, from the totally led (program-driven) approach, to systems that allow the student free initiative with respect to tutorials, student model, content, etc.

In the present study, an explanatory dialogue was developed and incorporated into the Virtual Chemistry Laboratory,[10] a system previously developed by the authors. Discussion is made of both the nature of the dialogue process and the models proposed for its reproduction in a computer system. The proposed theoretical model and its development under the object-oriented paradigm are presented, alongside its application to the building of an explanatory system for use in the help feature of a chemistry laboratory teaching application.[10] The tests and validation of the system from the point of view of its viability and consistency is presented in this work, leaving for a future work the study and results of our proposal from the educational point of view, and its applicability and the improvements the system can introduce in the computer-aided learning chemistry experiments.

The paper is laid out as follows: Section 2 gives a brief background of the concept of explanation through a dialogue process, establishing the basic requirements for a human-machine dialogue and detailing the components involved in the architecture of a computer system for the generation of explanatory dialogues, in addition to describing the existing theoretical models for this task. Sections 3 and 4 then present the proposed model for the construction of a dialogue generator that may be applied to chemistry teaching, In Section 5 is shown the dialogue generator system, describing its interface, functionality, how the knowledge involved in the dialogue is manipulated, and the integration of the dialogue system in the VCL. Finally, the developed work is discussed, analyzing the contribution proposed for the development of computer explanations.

2. BACKGROUND

In general terms, an explanation is something which clarifies a point of knowledge for the interlocutor. Examples might be how to prepare a solution, work out a concentration, or set up a buret. The explanation is deemed completed when the recipient is satisfied with the answer and understands the concept. While an explanation could be just a simple sentence, at other times it might be rather more complex, as long sentences, containing graphics, formulas, due to questions about a previous explanation, etc.[14,17,19,25]

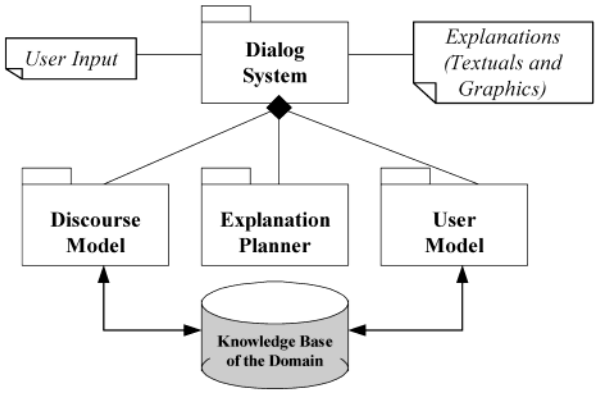

Figure 1. Generic diagram of an explanation-generating system.

Verbal explanations are intrinsically interactive. The person providing the explanation will often have no precise idea of how much the student knows or the degree of detail required to achieve a full understanding. The student will usually have to ask questions until he gets a full grasp of the concept. Short explanations will result in an *explanatory dialogue* consisting of an initial, and potentially complete, explanation, followed by a set of questions and answers.[13,14]

However, when an extensive explanation is required, the degree of the student's understanding must be determined as the explanation progresses and allow the student to interrupt in order to clarify certain points. This interaction enables the person explaining to reappraise their belief of the student's knowledge and continue the explanation in the most appropriate way. Thus, interactive explanations may be seen as a kind of *objective-oriented dialogue*[14,26,27] in which the aim of the person explaining is to clarify a point for the interlocutor.

In an explanatory dialogue process, the content and structure of explanations are of vital importance if they are to be successful. The building of such explanations using a computer system is highly complex[16−18,28] and requires a set of components or subsystems, each of which we will deal with: knowledge to be imparted, interaction between the student and the system, and planning, generation, and presentation of the explanations with a suitable layout and content.

Figure 1 shows a diagram, by means of UML packages notation, representing a generic architecture for an explanation-generating system. It can be seen that the dialogue is generated using explanations whose aim is to reach a final objective, namely transmit the knowledge to the student. In the course of the dialogue, the student can interact (ask new questions), providing additional input that will lead to the appearance of new objectives.

The generation of explanations is performed through the following:

- *The user model*, that is, the belief of the student's knowledge level; this estimation may vary throughout the dialogue.
- *The discourse model*, representing the previous and partially planned forthcoming presentations.
- *The explanation planner*, which uses a set of rules based on the discourse domain, together with the knowledge and

input of the student, to generate the explanations needed to achieve the objective.

- *The knowledge base for the domain of the discourse*, which is the knowledge set available for the student and upon which the discourse may thus revolve.

Output of the dialogue generator will include explanation frames that may be textual, graphical, or any other type that might be used to furnish an explanation.

A dialogue generator must comprise (see Figure 1) a subsystem (Explanation Planner) to handle planning and generating explanations for the student, working from the student's input (questions) to the system. Generating an explanation means deciding what to say, how to lead the dialogue, and how to reappraise the student model (using this information to direct the formation of new explanations).

Certain authors[14,26,29,30] have proposed simple methods of creating an explanation or description when given a series of assumptions about the knowledge level of the user. The generation of a comprehensible explanation would mean at least being able to choose from several types of explanation and selecting the background information to be provided.

To decide which background information to supply, *prerequisite relationships* may be defined within topics, specifically stating which topics must be known in order to understand others.[14] This is a useful approach but has its issues, since prerequisite topics depend not only on the item to be explained but also on how this is achieved.

The explanation planner will be basically composed of a series of content and dialogue planning rules, together with a planning algorithm. Each content planning rule identifies a knowledge frame to be explained, which in turn corresponds to a domain of the discourse problem. Additionally, for each rule there will be the following:

- A series of *restrictions*, to supply applicability conditions to choose between different dialogue planning rules and which are supplied by the user model.
- A series of *preconditions*, knowledge that must be acquired if absent. These preconditions appear as relationships between information or concept items, where in order to understand one particular concept, the student must know what is explained in the others; equally, these preconditions will only be included when the user model shows that they are unknown to the student.
- *Subobjectives*, representing crucial points in the main explanation. They must be included even if known by the student. For instance, the subobjectives in the explanation of a process will be the different causal events that lead to the process taking place.

When all the restrictions and preconditions of a rule have been satisfied, it can be said that the user has understood the main subtopics.

The planning algorithm continuously decides what to explain and how. For a potentially interactive dialogue, it is important not to fix details of a future explanation prematurely, since assumptions made about the student and the current objective may vary during the explanatory process, meaning that a preplanned detailed may end up redundant.

The dialogue planning rules define the nature of user interaction, building the appropriate basic exchange sequences for the user. According to Sinclair and Coulthard,[15] these exchanges equate to the discussion of a topic (*informative transactions*). These authors state that discussion of a topic begins with an opening exchange, followed by a series of teaching exchanges about the topic in question, and ends with a closing exchange.

As shown Figure 1, dialogue generator must accept user input either in natural language or through a menu-based interface.[12,22,30] If user input is in natural language, the system should be able to interpret it within the context of the interaction. When there is a menu-based interface, like the solution proposed in this paper, the system must be able to display appropriate menu items for the context. In either case, the system must be able to ascertain, to a fair degree of accuracy, what the next user input will be. In a menu-based system, this will enable the system to provide suitable menu choices.

Basically, two types of dialogue may be considered:

1. *Teacher-dominated*: the teacher (here, the system) is responsible for the student learning the material to be explained and dominates the interaction. The teacher can choose exactly what to explain, since the student is showing no obvious need for a particular explanation.

2. *Student-dominated*: the student dominates the interaction and expresses particular needs for information. Responsibility for understanding the material either falls upon the student or is shared. Usually, the student will initiate an explanatory dialogue by posing a question and will lead the dialogue using questions to clarify and elaborate on the subject until reaching a satisfactory stage of learning.

Finally, a model of the user's knowledge of the problem domain plays an important part in the explanation, in several ways:

- *In the content of the explanation*, e.g. by including background information if this appears to be unknown.
- *In the choice of dialogue actions*, e.g. by asking questions to test the user's knowledge.

Thus, the user model decides upon the most suitable strategy for the explanation of a concept (using examples or by classification, etc.) as well as its detail level and what background (or optional) information should be included.

Representation of user knowledge is based on the information elements model. Given a specific element of the knowledge base domain, the user-modeling component works out whether the student understands the information element.

The dialogue generator must distinguish between the concepts that it has decided are known or unknown and those for which no such certainty exists. These certainty measurements provide a simple representation of the system's degree of belief of the user's understanding of a concept.[26,31]

An initial student knowledge model can be based on the user's experience and information based on stereotypes[31] as well as on a series of indirect inferences based on the relationship between concepts. Indirect inferences based on user level and conceptual relationships will only be applied when the system needs to know whether the user knows a certain concept. User interaction should allow direct inference and bring about an immediate change in the representation of the user model, since recent information drawn from a user-system interaction may be reckoned to be more reliable than any other.[32,33]

Table 1. Correspondences between the Real World and the *VCL* Virtual World

real world	↔	virtual world
chemistry lab	↔	3D virtual environment
text/reference book	↔	online help system
lab practice teacher	↔	interactive dialogue system

3. A DIALOGUE SYSTEM FOR THE TEACHING OF LAB PRACTICE IN A VIRTUAL CHEMISTRY LABORATORY

In previous articles, the authors proposed a model and developed a system to permit chemistry lab practice in a 3D environment. *Virtual Chemistry Lab* (*VCL*)[10] is multimedia-teaching software aimed at simulating a working environment (in this case a chemistry lab) where students can work in a virtual environment in a similar way to they would in a real practice lab at any teaching center. The teaching model of *VCL* is not, however, complete, since while it simulates the lab, it fails to simulate the teacher who might clear up any doubts arising as the students work.

The help system included in *VCL*, based on a Web tutorial,[10] could not be classed as an interactive virtual teacher, rather as a complete textbook available at any moment. With this in mind, development of an interactive dialogue system to allow the student to ask questions about any aspect of the lab (materials, working methods, etc.) as well as inquire about theory and/or technique for chemistry experiments would complete the teaching model of the system.

Creation of the dialogue system should provide complete representation of the real teaching model; in other words, all the elements of the real educational process involved in the teaching of practical chemistry technique will have their virtual equivalent, as shown in Table 1

The first objective of the present study was to propose an ontology for the dialogue process between the user and the *VCL*, which could be integrated into the $E(V) = M + m$ model—*Experiment(Virtual)* = *Materials* + *method*—on which it is based.[10] This approach has to take into account the adaptation for the student of the explanatory dialogue with respect to how the dialogue process is fairing (e.g. using more generic explanations if the student displays, through his questions, a significant lack of knowledge about the topic). This ontology must be subject to the restrictions imposed by the very nature of a computer-conducted dialogue (as described above), since given the complexity present in natural language the computer will only be able to reply to a known, previously defined set of questions, and the knowledge network managed by the system will be designed to answer this set of questions.

Creation of an interactive dialogue system incurs the need to model a complete knowledge network composed of multiconnected nodes, such that it may be browsed using the functions connecting the different network nodes. This knowledge network requires permanent storage, remote access to information, acceptably fast information searching, security of information, and so on.[12,20,21]

Information held in the knowledge network must be input by an authority (teacher) to guarantee its accuracy and suitability. For this reason, different user types will have to be identified (for instance, certain functions will be available to teachers but not to students). Likewise, data concerning various lab components should be visually ergonomic, allowing the display, at any given moment, of the structure of the knowledge network and modification of its content (be it adding more information or altering or deleting existing data).

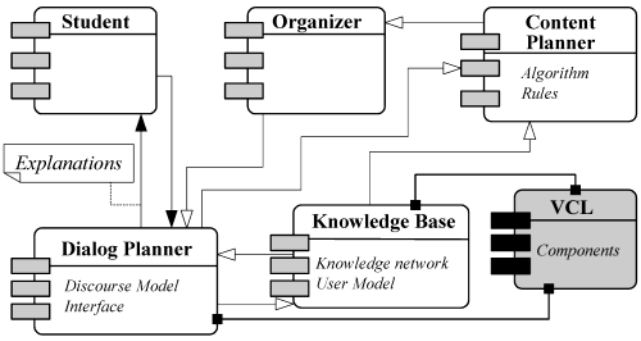

Figure 2. Component diagram of the proposed model.

The present paper describes the proposed model, which has been developed for each of dialogue system components, and a future study will detail the practical application of the dialogue system to the performance of selected experiments.

The proposed model is based on the formulation of suitable paradigms for the representation of the knowledge to be taught, the student model, the content planner (teaching component), and the dialogue planner (interface), in line with existing proposals concerning ITS components.

There are currently several different kinds of ITS, such as simulations, systems with learning capabilities, and knowledge-based paradigms. The proposed model has been developed for incorporation into a knowledge-based ITS, one which is basically required to possess expert knowledge of the problem domain to be taught and how this should be teaching should be undertaken, to lead the explanation "*intelligently*" using teaching strategies while maintaining a dynamic student model.

Figure 2 shows the architecture of the proposed model in a component diagram.[34] This figure shows the different packages of the system organized in components or subsystems and the existent relationships among these subsystems from a static point of view.

Student and knowledge models are represented by a *Knowledge Base* that stores the concepts to be taught and their relationships, together with the belief of the student's knowledge of the concepts. The explanation planner is represented by three components: the *DialoguePlanner*, which takes on the role of teacher and handles communication with the student, extracting explanations from the knowledge base to display them in an order determined by the *Organizer*, whose *ContentPlanner* presents the concepts to be explained, based on the knowledge extracted from the knowledge base.

As shown in Figure 2, the *Knowledge Base* and the *DialoguePlanner* have been integrated with the components of the *VCL*,[10] thus allowing the creation of explanations for any element of the *VCL* and the incorporation of the explanatory system in its virtual interface. There now follows a description of the solution devised for each of the components included in the proposed model for an interactive dialogue system.

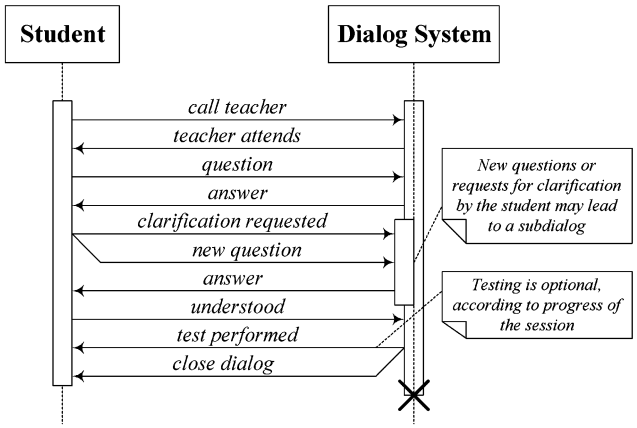

Figure 3. Sequence diagram for a generic system dialogue.

3.1. Representation of the Discourse Model. The proposed discourse model is student-dominated,[16,19,27] where student initiative is limited to asking questions and is based on the use of explanatory transactions[14] whose aim is to simulate the real-world interaction between students and their lab teacher and allows the dialogue flow shown in the sequence diagram[34] in Figure 3.

Given that the content of an explanation may be expressed as a plan and that a hierarchical model of a dialogue structure may produce a useful basis for modeling the organization of interactions, a model can be built where, at a higher level a rule for planning *transactions* will describe an application and will consist of the following:[13,26] an opening delimiting exchange, a sequence of pedagogical exchanges, and a closing delimiting exchange.

This is a basic, clearly plan-based model with simple dialogue planning rules. It has a well-defined intentional structure (where the objective is to help the user understand a specific topic) and contains *dominance* relationships between objectives and subobjectives for planning of content.

Clearly, the model is limited insofar as it cannot provide a sufficiently general and flexible framework that could be easily extended and adapted to generate all possible types of dialogue. Nevertheless, the model obtained is practical and may be applied to a specific type of dialogue (as produced in the *VCL*), while not being a general model.

Student initiative is limited to asking predefined questions about any material or conceptual element appearing in the experiment to be performed, while the system has the initiative when deciding on the form and content of the explanation. The user may pose a question at any point during the discourse when the generator is inactive, e.g. when the system is waiting for user confirmation or for the answer to a question, the user may interrupt the process with a new question. In such a case, the planned explanation will step aside until the new question has been answered correctly, thus giving rise to a subdialogue (see Figure 3).

The student-system interaction is shown in the Figure 3 by means of a sequence diagram. This diagram represents, in a global way, the possible interactions with the system and the answers that the system provides as well as the schedule in that these interactions take place.

Although this dialogue model may appear simple in the sense that it is purely based on the student's questions and the teacher's answers, it enables the student to break in at any moment with fresh questions, which forces the discourse planner to postpone its intended explanation (previously planned objectives) in order to generate explanations for the new question(s); this in turn means forming a new plan that will include any new possible requirements of the objective to be attained with respect to the new question.

In a verbal discourse, the student acknowledgment expected by the teacher after an explanation may take many forms: facial movements, verbal agreement, and so on, but in a computer system these responses must take place through interaction with the system. In the proposed model it is the system that asks the student, directly, if he or she understands a particular topic or has grasped an explanation, and the student must reply with a "Yes", "No", or "Not sure". This kind of exchange is not considered in the studies by Sinclair and Coulthard[15] and will arise whenever the system has no direct or indirect information in the user model as to whether a topic is already known, since this information is needed to make a decision on what to explain.

On occasions where an explanation has not been fully understood by the student, he or she can either ask the system for clarification or ask new questions (see Figure 3). Clarifications on a topic are dealt with thus as follows:

R1: The use of another explanatory paradigm, different from its predecessor; these are described in the next section.

R2: Revision of the prerequisites and explanation of the subobjectives which were not explained.

R3: Asking the student questions to reappraise his knowledge level.

Sometimes, however, the student is not really aware of not understanding what is being explained, and it is up to the system to identify the problem and attempt to correct this situation.[14] This happens when the student does not correctly answer the questions posed by the system, and it is more effective to somehow show the student why the answer was wrong instead of simply supplying the correct answer.

To find out where the student was going wrong, the system places in the discourse model the assumptions about the user's knowledge level that had been used to plan the explanation. If the system has assumed that the user had a good knowledge of certain information, but this has proved to be untrue, it may presume that the root of the problem is that the information was unknown to the user (*reaction R2*) and propose an explanation.

If the system is unable to find a reason for the confusion, it will try to explain the information in another way (*reaction R1*). To achieve this, when there is more than one possible way of explaining something, the discourse model includes links to other forms of explanation.

These clarification subdialogues, which appear as a result of a lack of understanding on the student's part, are dealt with as interruptions and not as part of the prior discourse, so the planned discourse will continue when clarification has been attained.

To add coherence to these clarification subdialogues, there is an interruption transaction rule responsible for planning appropriate opening and closing sentences for the dialogue, which includes the use of discourse markers (see Figure 4) that point to the beginning and end of an interruption (e.g. "Anyway" or "Getting back to where we were...").

The proposed model, as seen in Figure 3, has considered the possibility of the system testing or evaluating the

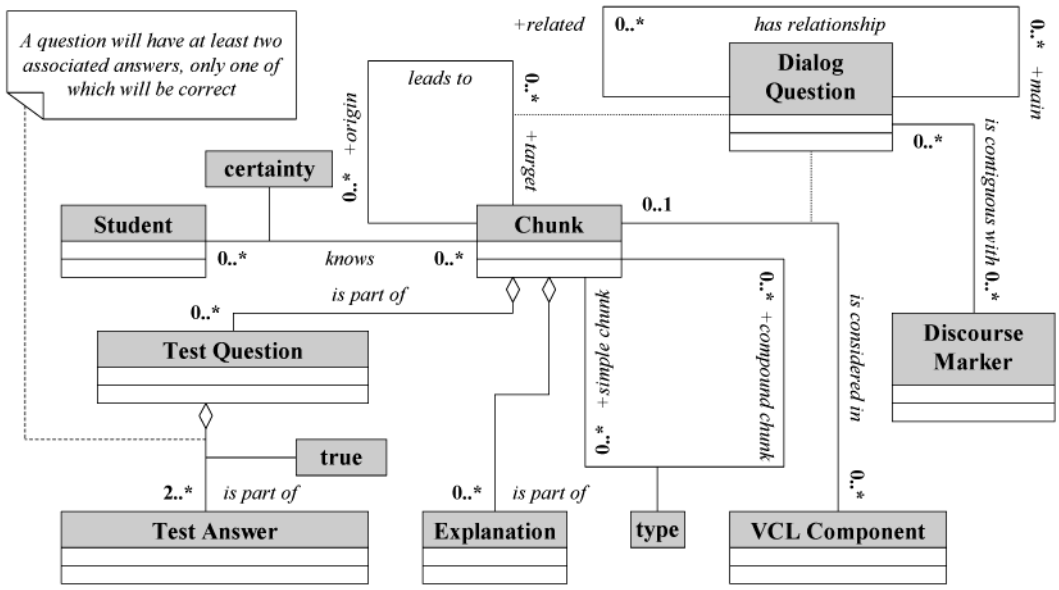

Figure 4. Class diagram of the proposed model.

student's level of understanding. These tests will be carried out by the system once the student has concluded the dialogue and when inconsistency has been found during the dialogue (e.g. when questions corresponding to a knowledge level lower than that assigned to the student are asked). A multiple-choice format is used; tests are assigned to the knowledge domain corresponding to the questions posed by the student in the course of the dialogue, having been previously defined in the knowledge model, shown in the class diagram of the Figure 4. This class diagram shows the defined classes, and their relationships, to represent the knowledge domain that is managed by the dialogue system.

3.2. Representation of the Knowledge Model. To be able to teach, an ITS must work with portions of knowledge and employ the pedagogical features required for each instance, so that it can suitably administrate the portions of knowledge.[21] Several rule-based and frame based models for the representation of this knowledge have been used in the development of ITSs.

Under an object-oriented paradigm[35] (used for the modeling and development of the present dialogue system) the use of frames—represented as classes—allows faithful representation of the knowledge or information domain. Thus, given a discourse domain corresponding to the topic to be taught, the knowledge elements (and their relationships) of which it is comprised may be structured in the form of a network and represented as classes and class associations, as shown in the class diagram[34] in Figure 4.

In the proposed model, these knowledge elements have been named *chunks*, such that a node in the knowledge network may be defined by a name that relates the *chunk* to its teaching material (*explanation*).

The relationships (edges between nodes or *chunks* of the knowledge network) have a specific meaning, an address, and may join pairs of nodes; the following types of relationships are allowed, characterized by the *Type* attribute (Figure 4):

1. *Prerequisite*: the root node must be known for the target node to be taught. This allows us to represent the requirement of certain basic knowledge in order to explain a particular portion of knowledge.

2. *Part-of*: the knowledge in the root node is a subset of the knowledge in the target node. As shown in Figure 4, the proposed model allows the existence of composite *chunks*, that is, *chunks* formed by other *chunks* of lesser significance, which permits modular representation of the knowledge and thus the reuse of the *chunks* representing it.

Thus, taking a node (knowledge *chunk*) as a learning objective, all *prerequisite* relationships arriving at that node may be analyzed, and a check can be made to ensure the student knows each node. In this way, if one of the nodes is unknown, it can be marked as a learning objective.

Teaching material (explanations) should be related to the *chunk* network so that it can be presented to the student when working with the *chunk* in question. The class diagram in Figure 4 shows how a *chunk* may contain zero (composite *chunks*) or several explanations, each of which has a series of distinguishing features and may contain text or graphical information. Two properties have been considered for the classification of explanations:

- *ExplanationLevel*: the difficulty presented by the explanation. Three possible values have been defined: low, medium, or high.
- *ExplanationType*: the way concepts are dealt with in the explanation. The dialogue system can handle four types of explanation: by context, analogy, example, or classification, (following ref 14).

These *chunk* properties allow the *ContentPlanner* to select the explanation best suited to the learning situation and the current level of the student.

One single *chunk* may be associated with several explanations, with different levels of detail. For a *chunk* and a given level of detail there may be several explanations, each using a different explanation paradigm.

Chunks are linked to *VCL* components,[10] such that each object considered in the *VCL* may have an associated *chunk* to enable it to collaborate with the student in the building of an explanatory dialogue based on predefined questions. These questions, as shown in Figure 4, are represented by the *DialogueQuestion* class, which intervenes in the association between the *VCLComponent* and *Chunk* class.

Table 2. Indirect Inferences Considered by the Student Knowledge Model

condition	inference
all subconcepts known/unknown	main concept known/unknown
all subconcepts known/perhaps known	main topic perhaps known
difficulty of concept above user level	concept unknown
difficulty of concept below user level	concept perhaps known
main concept known/perhaps known	subconcepts perhaps known
main concept unknown	subconcepts unknown

Table 3. Direct Inferences Considered by the Student Knowledge Model

dialogue exchange	inference
Systems says X to user, who acknowledges.	User perhaps knows X.
System asks X. User replies correctly.	User knows X.
System asks X. User replies incorrectly.	User does not know X.
User asks X.	User does not know X.

Figure 4 also shows a reflexive association of the *chunk* class, which allows the representation of the directional knowledge (the prerequisites). Additionally, this association is characterized by the *DialogueQuestion* class, which in turn maintains a reflexive association with the relationships that exist between specific frames of the knowledge domain.

3.3. Representation of Student Knowledge. Representation of student knowledge is based on the *Knowledge Model* described above. Given a certain object from the model (e.g. the preparation of 1 M NaOH), the user-modeling component works out whether the student knows the element.

The extent of the student's awareness of the knowledge elements is represented by certainty measurements, defined by the labels "*known*", "*perhaps known*", and "*unknown*", which characterize the association between the student and the *chunk*, as shown in Figure 4. These labels provide a simple representation of the degree of belief assigned to the student's knowledge of a concept.

Since the system requires information about the student in order to produce a personalized dialogue, the student model is based on examination of the *Student* class in which it contains, among others, the following properties:

Base level: initially drawn from the student profile (following the stereotype model[34]) and subsequently reassessed in terms of the history of student-system interaction. Students are placed into one of three levels (high, medium, low) of general knowledge. As the student interacts with the system, the value of the *base level* is refined in terms of the *session level*, on a scale of [0−10].

Session level: student activity during a specific session (mistakes made in tests, help consulted, etc.) is evaluated so that student performance may be measured; the resulting information is used throughout the session not only to plan future dialogues but also to reappraise the *base level* of the student.

The initial user model is set up according to the user's experience and a series of indirect inferences based on the relationships between concepts or *chunks* held in the knowledge network (Table 2). The model is updated according to feedback from user interaction (Table 3).

Indirect inferences based on both student knowledge level and conceptual relationships (between *chunks*) are only applied when the system needs to find out whether the user knows a particular concept. Once the system has deduced this information, it is not, however, explicitly stored in the student model (if the deduced information were the student model, the system would also need to store all the suppositions on which it was based and review the model when any of these beliefs changed). Indirect inferences are applied in order and only when there is no direct information in the user model.

Direct inferences are based on user interaction and produce an immediate update to the user model, since the most recent information resulting from *user-system* interaction should be more reliable than any other.

Throughout the discourse, it will be necessary to review the estimated knowledge level of the user. Should, for instance, the student correctly answer a series of questions considered above his current estimated level, or should the opposite happen, the experience level would have to be revised.

The student model is decisive in three aspects of discourse content:

1. Selection of the most suitable strategy to explain a concept.

2. Detail level of the explanation.

3. Selection of the background (or optional) information that should be included. This material is represented by *preconditions* in the planning rules and will only be included if it is "believed" to be "*unknown*".

Moreover, the student model also plays a part in the choice of dialogue actions. The system uses the model to offer an explanation of a concept to the student or perhaps to carry out a test—since one aim of the system is to increase its certainty of its estimation of student knowledge.

4. GENERATION OF EXPLANATIONS

The sequence diagram[34] shown in Figure 5 details the actions performed during the dialogue process between the student and the components of the proposed model, to give a clear idea of the functionality of the proposed solution. Dialogue processing is modeled in the presence of agents (here implemented as classes): the *ContentPlanner*, at the explanation generation stage, and the *DialoguePlanner* at the student interaction stage.

The *ContentPlanner* decides on the concepts to be explained (prerequisites and subobjectives), according to student knowledge. It determines the knowledge to be included (*part-of* relationships) for each topic (chunk) and deals with *what* should be explained. To do so, it extracts information from the knowledge network (*chunks* and associations) using the content planning rules, whose premises are based on student questions, knowledge or session level of the student, and explanations already displayed (Figure 5).

The *DialoguePlanner* decides *how* to tackle the topics considered by the content planner. It is also responsible for the handling of user interaction (by specifying the points at which the student may interrupt the explanation, carrying out evaluation tests, dealing with misunderstanding, etc.).

These two agents communicate through an *Organizer* (see Figure 5), comprising an ordered list of preconditions and subobjectives that the *ContentPlanner* has decided should be explained, while the *DialoguePlanner* deals with extracting these entries and presenting them to the student.

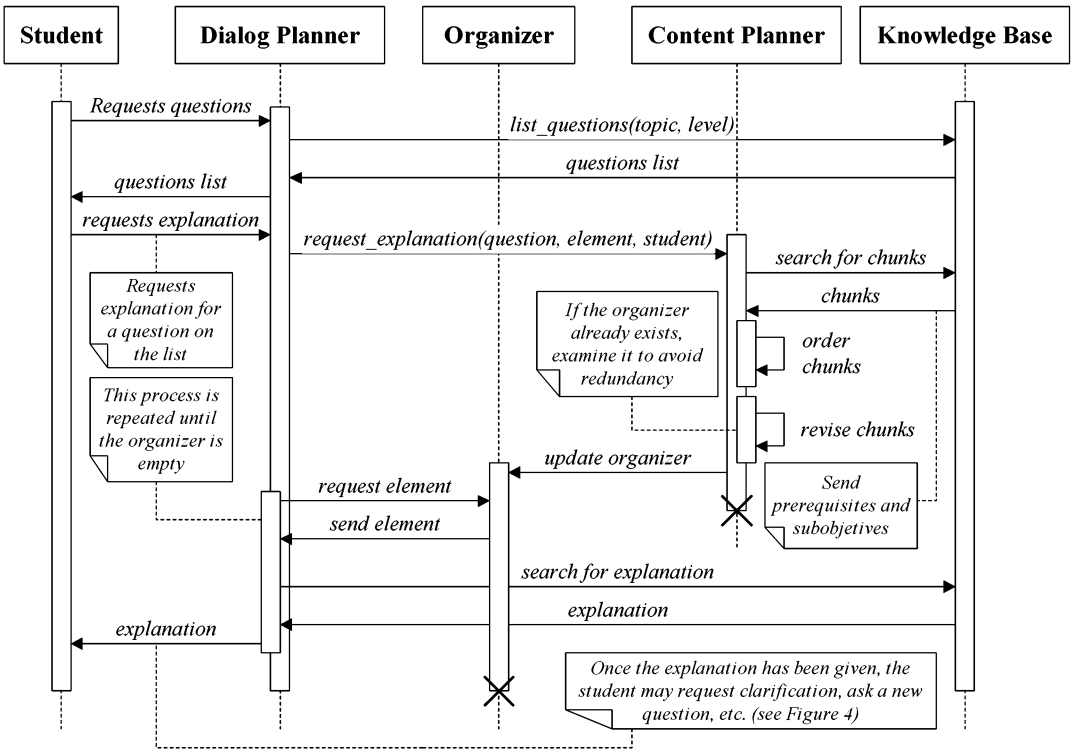

Figure 5. Dialogue flow between the student and components of the proposed model.

The dialogue process is initiated when the student wishes, at some stage during his or her interaction with the learning system (*VCL*[10]), to ask a question, at which point the *DialoguePlanner* will respond by displaying a list of preset questions drawn from the knowledge base.

4.1. Dialogue-Building Strategy. When a student asks a specific question, the *DialoguePlanner* passes it on to the *ContentPlanner*, which uses the following strategy to build the explanation organizer:

1. It accesses the knowledge base and searches the knowledge network for related chunks and the information held about the student's knowledge of those chunks.

2. Prerequisites are ordered so that they will be explained before the subobjectives; this ordering will also ensure there are no unsatisfied dependencies. It is thus assured that when a concept is explained, any knowledge required for this will have been explained beforehand.

3. All subobjectives comprising a chunk are extracted recursively from the knowledge base in top-down order of preference, that is, from simple chunks through to composite ones.

4. Any prerequisites for which there is direct evidence that the student knows (e.g. due to prior student-system interaction) are omitted.

5. For prerequisites where no such direct evidence of prior knowledge exists (i.e. where this information does not appear in the knowledge base), the following indirect inference is applied:

□ If the prerequisite is unknown, but where all the subobjectives and their corresponding prerequisites are known, it is presumed that the prerequisite in question is, in fact, known.

□ If the prerequisite is unknown, and one or more of its subobjectives or prerequisites is also unknown, the indirect inference rule is applied recursively.

6. Once chunk extraction is finalized, the organizer is reviewed to eliminate duplicates, keeping the first occurrence of any duplicate, thus removing redundancy and maintaining the conceptual consistency of the explanation (see Figure 5).

4.2. Creating the Explanation. Once the *Organizer* has been built, the *ContentPlanner* extracts explanation elements from it, employing the following strategy:

1. For each prerequisite and subobjective of the organizer, the knowledge base supplies the explanation whose difficulty level is just below that of the student.

2. Where there are several explanations that satisfy the above criterion for one chunk, the choice is made from the four types defined in the model, depending on the level of the student and considering the complexity level assigned to the explanations.

3. If the student interrupts the explanation to ask a new question, this question will again be sent to the *ContentPlanner*, which suitably updates the *Organizer*.

4. The explanatory process is carried out until the *Organizer* runs out of further explanations.

When the explanation has concluded, the *DialoguePlanner* decides if the student should be presented with a progress test to reappraise his or her estimated knowledge level, for which the following strategy would be used:

1. If through previous interaction there is evidence that the student already knows the chunk, no testing is carried out.

2. Where there is no such evidence, but the level of the explained chunk is below that of the student, the system will ask the student if he or she would like to be tested.

3. Where there is no such evidence but the level of the explained chunk is equal to or higher than the level of the student, a test will always be given.

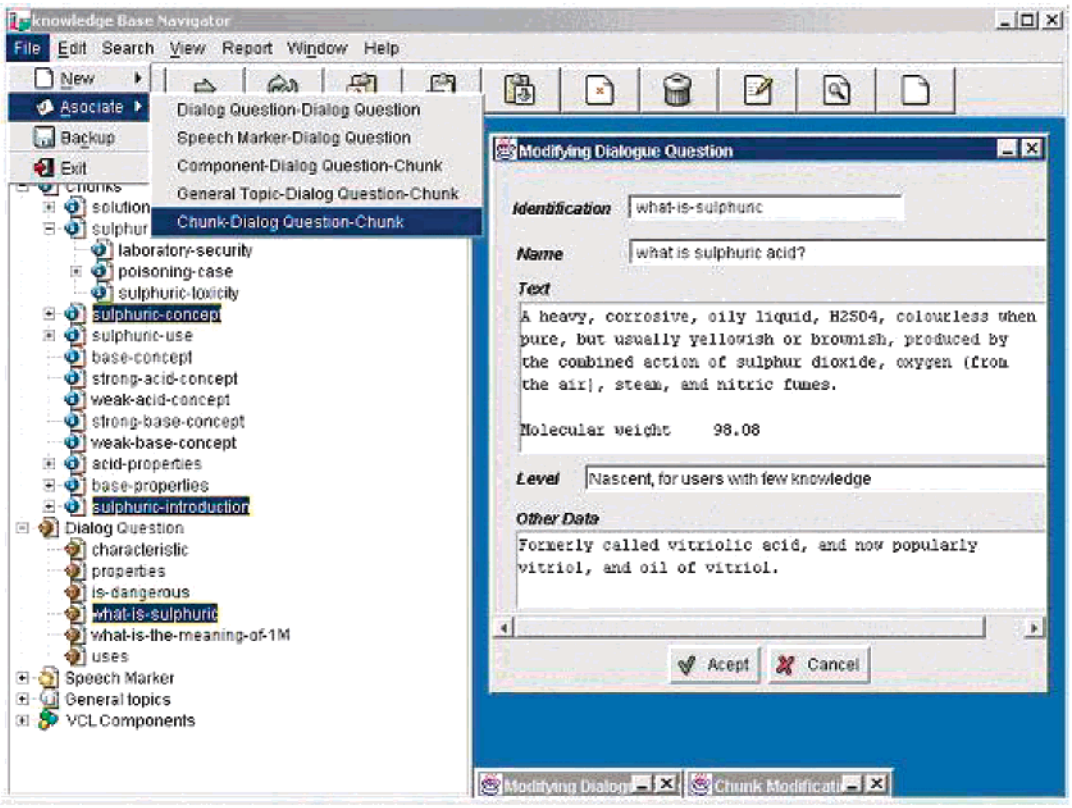

Figure 6. The knowledge-base navigator: association between dialogue questions and chunks.

Tests consist of a semirandom selection of questions, the number of which has previously been defined by the teacher (externally defined as a configuration parameter) and where questions are drawn from all the tests associated with the chunks explained during the session and are of the same or lower level than that of the student (see Figure 3). The *ContentPlanner* on receiving a message from the *Dialogue-Planner* performs selection.

4.3. Reexplanation of the Concept. During the course of the interaction, reexplanation of the selected topic is foreseen for when one of the following conditions occurs:

1. If the student has not passed the appropriate test by correctly answering at least 50% of the questions. Here, indirect inferences are not used to set the prerequisites to be explained; rather, an explanation will be provided for every prerequisite (and its associated chunks) for which there is no direct evidence that the student has properly learned it.

2. If the student explicitly requests an explanation of a specific concept, an alternative and unused explanation for that chunk will be presented; should no such alternative exist, the system will proceed as in the first case.

As in the explanatory process, the *ContentPlanner* is responsible for generating the *Organizer* (this time more extensively than in an ordinary explanation).

4.4. Updating the Student Model. Once the student has been tested, the direct evidence of his or her knowledge is updated, as follows:

1. If the student has passed the test, the direct evidence that the student has learned the main chunk is stored; the system does not, however, store evidence for the objectives and prerequisites since, as described beforehand, this evidence is inferred from the direct evidence of knowledge of the main chunk.

2. Should the student not pass the test, no direct evidence is stored; the chunk remains "unknown" to the user and will remain so until a subsequent test is passed or, as a result of direct interaction with the system, all the subobjectives and prerequisites associated with the chunk have been learned.

3. Finally, the knowledge level associated with the student is updated.

5. IMPLEMENTATION OF THE DIALOGUE SYSTEM

Based on the proposed model and employing the Java language[36] in conjunction with the Oracle 9i DBMS[37] for knowledge-base management, a system was developed to build the knowledge network.

Figure 6 shows the interface of the Knowledge Base Navigator system; a windows style interface where the user (teacher or expert) navigates on the knowledge network containing all the information involved in the dialogue (chunks, dialogue questions, speech marker, etc.).

As shown in Figure 6, the user can define all the items appearing in the dialogue (chunks, questions, etc.) and interconnect them. These elements and their relationships are manipulated by the system through a drop-down hierarchical menu (left-hand side of the interface), enabling the student to easily browse the knowledge network.

The workspace of the interface (midscreen) allows the user (expert) to handle multiple knowledge items simultaneously in independent windows, facilitating the somewhat arduous and painstaking task of building the knowledge network.

In Figure 6 is shown how the user relates a dialogue question (what-is-sulfuric) with two associated chunks (sulfuric-concept and sulfuric-introduction). Clicking in the dialogue question icon in the hierarchical menu, the corre-

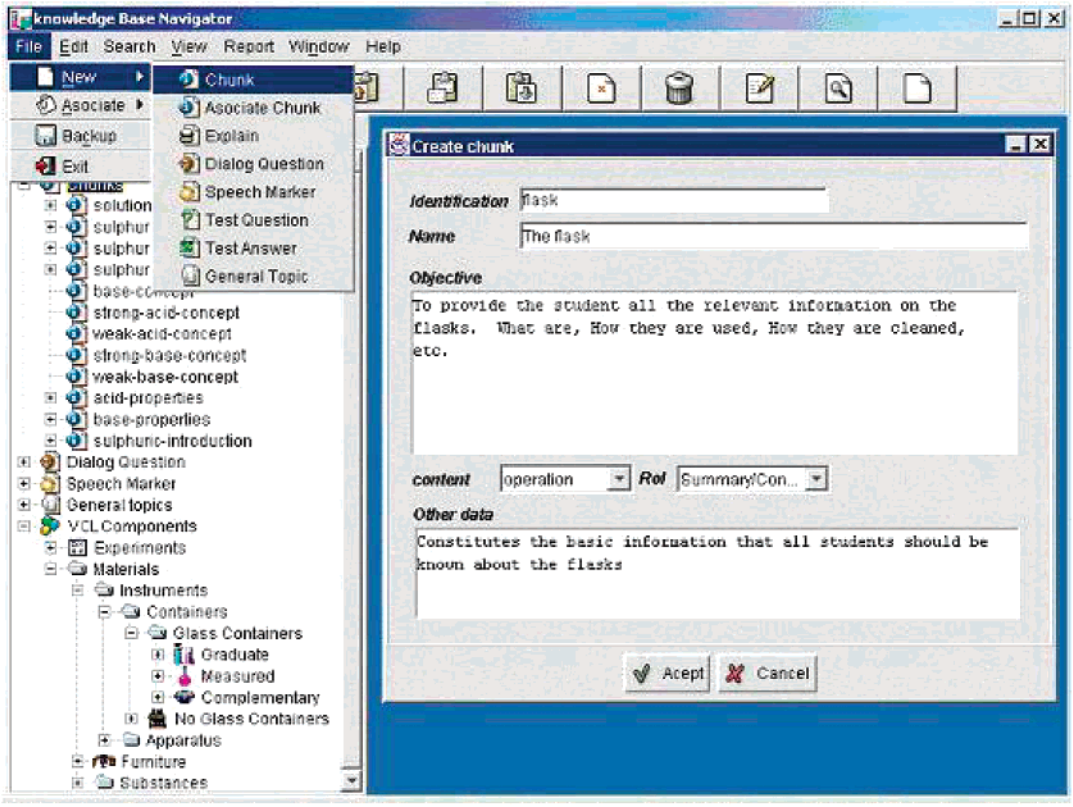

Figure 7. Inserting a chunk related with a VCL component.

sponding window is open in the workspace. The user clicks the chunks to be associated and selects the option (*File: Associate:Chunk-Dialogue Question-Chunk*) in the main menu of the Navigator).

The hierarchical relationship between the chunks proposed by our model (see Figure 4) can be also observed in Figure 6 through the different levels in which the chunks are shown (and therefore are created) in the drop-down hierarchical menu.

While in the dialogue the questions the student might ask are predefined for each chunk and appear in simple text form, explanations may be constructed as .html or .xml files (also stored in the nucleus of the database), thus allowing inclusion of multimedia content (text, graphics, images, and so on).

Although the system only allows the student to carry out questions that have been predefined by the professor and are associated to each chunk of knowledge, do not exist restriction on the number, complexity, and structure of these questions. The professor will support from his teaching experience, use of the system, and level of knowledge and students' type, to insert and modify (and erase) questions to adapt them to the specific teaching conditions.

The system is flexible and may be incorporated into any other object-based teaching aid. This is achieved, using the browser, by importing the classes for which knowledge-explanation items are required and linking those classes to the knowledge (chunks, dialogue questions, etc.), as is illustrated in Figure 7 where *VCL* elements[10] have been linked to knowledge items (see end of the drop-down hierarchical menu), and a chunk is currently created corresponding with a *VCL* component (flask).

To build the explanatory process, a Java applet was written, where an avatar (currently being field-tested) handles teacher−student communication (Figure 8). This applet can be called by any system by instancing its *init()* method and is also used to check the content of the knowledge network while building the system.

In Figure 8 is also shown the integration between the dialogue system and the VCL during the execution of a virtual experiment corresponding with the preparation of 1 M sulfuric acid solution. At any moment the student can invoke to the teacher (avatar), initiating avatar applet and showing a set of questions related to the experiment stage in which the student is working, although the student can select to see all questions defined for the experiment.

Furthermore, as in a textbook, the avatar interface lets the student navigate along all the questions, and their corresponding explanations were defined in the knowledge network, simply using the arrows icons included in the avatar interface.

When student clicking over a specific question from the list, the explanation is shown as a new window (see Figure 8) in which some options are available: print, ask for help, to make another question, etc., continuing the dialogue process until the student accepts the explanation and then the avatar is closed.

DISCUSSION

As in "real-world" teaching, computer-assisted learning seeks to achieve a degree of (virtual) teacher−student communication that will allow suitable personalization of the learning process. AI, in its current state, is still far from a satisfactory simulation of human dialogue, so somewhat reduced models are needed to simulate interactive teacher−student explanations suited to the specific knowledge frames involved in the development of ITSs.

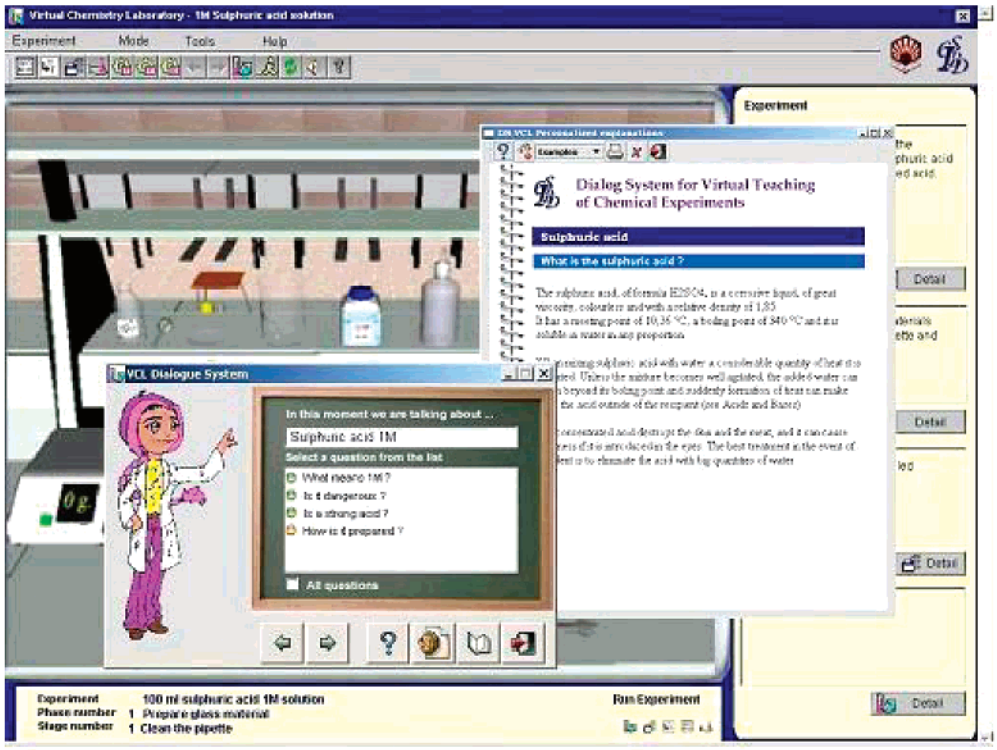

Figure 8. A sample of the dialogue system integrated in the Virtual Chemistry Lab.

Although, in past decades many ontologic models have been proposed for the construction of computer dialogues, its complexity or specificity has not allowed an appropriate implementation. The solution proposed in this work has made use of an ontology that simulates perfectly the man-machine interaction, in which the user is the one that governs the dialogue requiring, when it is necessary, the help (explanation) of the expert.

In this study, the authors have presented a model for the development of dialogues that might arise during the learning process, in this instance for application to the virtual teaching of chemistry experiments, although it could easily be adapted to other similar applications.

The proposed solution is based on the representation of the knowledge to be explained by a network of multiconnected chunks, where each chunk can be further specialized in specific knowledge frames (prerequisites and subobjectives), may contain associated explanations of varying degree of complexity and with different explanatory models, and can be associated with one or a set of questions the student might ask. Recognition and updating of the student's understanding of each chunk is performed throughout the learning process by the use of both stereotypes and direct or indirect inference, enabling the explanation to be adapted to the requirements of each student.

The solution provides a fair representation of how well the topic has been learned, in an environment governed by the student and in which the (virtual) teacher answers any pertinent questions posed by the student.

Both the proposed model and the dialogue generator have been integrated into the Virtual Chemistry Laboratory[10] (*VCL*) for the teaching of selected experiments. However, the dialogue system is open to be used for any knowledge domain to be explained and may integrate any components which have been built in Java.

Furthermore, as the system it has been built in Java, all their components (organizer, dialogue planner, etc.) are Java classes, which can be used, modified, overloaded their methods, etc., what facilitates that the system can be adapted to other teaching problems, on the same or different learning paradigm.

However, one of the most important problems in the generating systems of explanations is to fill it with contents (knowledge) and to validate. Some tests must be done with different kind of students (different level of study) in teaching centers. This work requires time to evaluate the results, comparing a lot of variables (students, levels, teaching centers, teaching using and not the system, and so on).

This validation of the system from the pedagogic point of view will be carried out by the educational experts in chemistry, preparing and introducing in the system the contents based on its method or educational paradigm and according to the level of its students' knowledge.

The authors consider that the results of this validation of the system should be the object of a future work with an eminently educational focus, and for that we are trying to collaborate with teachers and teaching centers, and we expect that these results may be the object of a forthcoming article.

ACKNOWLEDGMENT

The authors would like to thank Mr. Francisco Gabriel Muñoz Rodriguez for his invaluable assistance in the construction of the system, as part of his degree course in Computing Engineering, and to Prof. Dr. Amparo Vila Miranda of University of Granada for his valuable collaboration.

REFERENCES AND NOTES

(1) Cartwright, H. M.; Valentine, K. A Spectrometer in the Bedroom. The Development and Potential of Internet-Based Experiments. *Comput. Educ.* **2002**, *38*, 53−64.

(2) Colwell, C.; Scanlon, E.; Cooper, M. Using Remote Laboratories to Extend Access to Science and Engineering. *Comput. Educ.* **2002**, *38*, 65−76.
(3) Barna, N.; Dori, Y. J. Computerized Molecular Modeling as a Tool to Improve Chemistry Teaching, *J. Chem. Inf. Comput. Sci*. **1998**, *36*(4), 629−634.
(4) Borchardt, J. K. Improving the General Chemistry Laboratory. www-chenweb.com/alchem2000/news/nw-000922-edu.html
(5) Rzepa, H. S.; Tonge, A. P. VchemLab: A Virtual Chemistry Laboratory. The Storage, Retrieval and Display of Chemical Information Using Standard Internet Tools. *J. Chem. Inf. Comput. Sci*. **1998**, *36*(6), 1048−1053.
(6) Luque Ruiz, I.; Martinez Pedrajas, C.; Gómez-Nieto, M. A. Design and Development of Computer-Aided Chemical Systems: Representing and Balance of Inorganic Chemical Reactions. *J. Chem. Inf. Comput. Sci.* **2000**, *40*(3), 744−752.
(7) Luque Ruiz, I.; Gómez-Nieto, M. A. Solving Incomplete Inorganic Chemical Systems through a Fuzzy Knowledge Frame. *J. Chem. Inf. Comput. Sci.* **2001**, *41*(1), 83−99.
(8) (a) marian.creighton.edu/∼ksmith/tutorials.html. (b) www.emory.edu/CHEMISTRY/pointgrp. (c) www.chem.uwimona.edu.jm:1104/chemprob.html. (d) www.chemsoc.org.golbook. (e) www.modelscience.com/products.html. (f) www.chemnews.com. (g) website/lineone.next/∼chemie/reviews.html. (h) www2.acdlabs.com/ilabs. (i) www.chemsw.com/10202.htm. (j) www.compuchem.com/dldref/formdem.htm. (k) www.ir.chem.cmu.edu/irProject/applets/virtuallab/applet_wPI.asp. (l) chem.lapeer.org/ChemBatsDocs/Index.html. (m) jchemed.chem.wisc.edu/JCEsoft/Issues/Series_SP/SP8/abs-sp8.html. (n) chem-www.mps.ohio-state.edu/∼lars/moviemol.html. (o) www.chem.ox.ac.uk/vrchemistry/.
(9) Murray, T. Expanding the Knowledge Acquisition Bottleneck for Intelligent Tutoring Systems. *Int. J. Artificial Intelligence Educ.* **1997**, *8*(3), 222−234.
(10) Luque Ruiz, I.; López Espinosa, E.; Cerruela García, G.; Gómez-Nieto, M. A. Design and Development of Computer-Aided Chemical Systems: Virtual Labs for Teaching Chemical Experiments in Undergraduate and Graduate Courses. *J. Chem. Inf. Comput. Sci.* **2001**, *41*(4), 1072−1082.
(11) Crosier, J.; Cobb, S.; Wilson, J. R. Key Lessons for the Design and Integration of Virtual Environments in Secondary Science. *Comput. Educ.* **2002**, *38*, 77−94.
(12) Cook, J.; Oliver, M. Designing a Toolkit to Support Dialogue in Learning. *Comput. Educ.* **2002**, *38*, 151−164.
(13) Moore, J. D. *Participating in Explanatory Dialogues*; MIT Press: Cambridge, MA, 1995.
(14) Cawsey, A. J. *Explanation and Interaction: the Computer Generation of Explanatory Dialogues*; MIT Press: Cambridge, MA, 1992.
(15) Sinclair, J. McH.; Coulthard, R. M. *Towards an Analysis of Discourse: The English Used by Teachers and Pupils*; Oxford University Press: Londres, 1975.
(16) Guinm Curry, I. *Mechanism for Dynamically Changing Initiative in Human-Computer Collaborative Discourses*, Proceedings of the Annual Symposium on Human Interaction with Complex Systems, HICS, IEEE, Los Alamitos, CA, 1996; pp 28−36.
(17) Johnson Gerald, J. Metaphor and the Difficulty of Computer Discourse. *Commun. ACM* **1994**, *37*, 97−102.
(18) Nymeyer, A. Gramatical Specification of Human-Computer Dialogue. *Comput. Languages* **1995**, *21*(1), 1−16.
(19) Tarby, J. C. Human-Computer Dialogue in Learning Environments. *Comput. Appln. Eng. Educ.* **1997**, *5*(1), 29−39.
(20) De Bruin, H.; Bouwman, P.; Van den Bos, J. Modeling and Analysing Human-Computer Dialogues with Protocols. *Eurographics Workshop Design, Specification and Verification of Interactive Systems*, Proceedings, CNUCE-C.N.R., Pisa, Italy, 1994; pp 61−83.
(21) Nussbaum, M.; Rosas, R.; Peirano, I.; Cárdenas, F. Development of Intelligent Tutoring Systems using Knowledge Structures. *Comput. Educ*. **2001**, *36*, 15−32.
(22) Arruarte, A.; Fernández-Castro, I.; Ferrero, B. The IRIS Shell: How to Build ITSs from Pedagogical and Design Requisites. *Int. J. Artificial Intelligence Educ.* **1997**, *8*, 341−381.
(23) *Changing University Teaching: Reflections on Creating Educational Technologies*; Evans, T., Nation, D., Eds.; Kogan Page: London, 2000.
(24) Fernández-Castro, Y.; Díaz Harraza, A.; Verdejo, F. Architectural and Planning Issues in Intelligent Tutoring Systems. *J. Artificial Intelligence Educ.* **1993**, *4* (4), 357−395.
(25) McKeown, K. R. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*; Cambridge University Press: 1985.
(26) Paris, C. User *Models in Dialogue Systems*; Kobsa, A., Wahlster, W., Eds.; Berlin, 1989.
(27) Ishizaki, M.; Crocker, M.; Mellish, C. Exploring Mixed-Initiative Dialogue using Computer Dialogue Simulation. *User Modell. User Interact.* **1999**, *9*(1), 79−91.
(28) Jung Hee, K.; Fredman, R.; Evens, M. W. *Responding to Unexpected Student Utterance in CIRCSIM-Tutor V.3. Analysis of Transcripts*, Proceedings of the Eleventh International Florida Artificial Intelligence Research Symposium Conference; AAAI Press: Menlo Park, 1998; pp 153−157.
(29) Maybury, M. T. *Planning Multisentencial English Text Using Communicative Acts.* Ph.D. Thesis, Computer Laboratory, University of Cambridge, 1991.
(30) Carbonell, J. R. AI in CAL: An Intelligence Approach to Computer Assisted Instruction. *IEEE Trans. Man Machine Systems* **1970**, *11*(4), 190−202.
(31) Rich, E. User Modeling Via Stereotypes. *Cognitive Sci.* **1979**, *3*, 339−354.
(32) Kerpedjiev, S.; Roth, S. F. Mapping Communicative Goals into Conceptual Tasks to Generate Graphics in Discourse. *Knowledge-Based Systems* **2001**, *14*, 93−102.
(33) Ishimaru, K.; Harada, N.; Yamada, K.; Nukuzuma, A.; Furukawa, H. *User model for Intelligent Cooperative Dialogue System*, Proceedings of 1995 IEEE-International Conference on Fuzzy Systems; IEEE: New York, U.S.A., 1995; Vol. 2, pp 831−836
(34) Rumbaugh, J.; Jacobson, I.; Booch, G. *The Unified Modeling Language. Reference Manual;* Addison-Wesley Longman Inc.: U.S.A., 1999.
(35) Booch, G. *The Object-Oriented Analysis and Design with Applications*; Addison-Wesley: 1994.
(36) Campione, M. *The Java Tutorial*, 3rd ed.; McGraw-Hill: 2002. http://java.sun.com.
(37) Loney, K.; Koch, G. *Oracle 9i The Complete Reference*; Oracle Press: 2002.

CI0340254