Journal of Computing and Information Technology - CIT 2, 1994, 2, 123–138

Using Hypertext to Implement Multiple Tutoring Strategies in an Intelligent Tutoring System for Music Learning

Marios C. Angelides and Amelia K.Y. Tong

Information Systems Department, London School of Economics and Political Science, London, G. Britain

Variation in tutoring strategies plays an important part in Intelligent Tutoring Systems. The potential for providing an adaptive Intelligent Tutoring System depends on having a range of tutoring strategies to select from. In order to react effectively to the student's needs, an Intelligent Tutoring System has to be able to choose intelligently among the strategies and determine which strategy is best for an individual student at a particular moment. This paper describes, through the discussion pertaining to the implementation of SONATA, a music theory tutoring system, how an Intelligent Tutoring System can be developed to support multiple tutoring strategies during the course of interaction. SONATA has been implemented using a hypertext tool, HyperCard II.1.

Keywords: Intelligent Tutoring Systems, Tutoring Strategies, Music Learning, Hypertext

Introduction

Intelligent Tutoring Systems (ITSs) facilitate the provision of a one-to-one tuition between student and teacher. For an ITS to offer a valuable educational experience on this individualized basis, it must be able to adjust its tutoring style to the student's changing needs. Variation in tutoring strategies is fundamental in this respect, as tutoring strategies are the means through which the tutor imparts tutorial material to the student. The purpose of this paper is through the discussion pertaining to the development of SONATA, to illustrate how an ITS can switch from one strategy to another by incorporating the agents responsible for helping the system to make the 'switch' decision. SONATA was developed using the hypertext approach [Angelides and Gibson, 1993]. This involves decomposing and storing logically the different types of knowledge required in SONATA as a collection of 'cards', and setting up links to integrate the stored information together in a desired manner.

SONATA is an ITS aimed at primary level music students. The assumption underlying the use of the system is that the user does not possess any prior knowledge of music theory. The system treats every new student as an absolute beginner. It aims to assist students with their learning of music theory, by providing guidance as well as assessment to the student within the knowledge domain. The long term goal of SONATA is to contribute towards a teaching environment an ITS which offers multiple tutoring resources, enabling tutorial material to be presented from alternative teaching viewpoints.

The paper first gives an overview of intelligent tutoring systems, including discussions about current practices with tutoring strategies and ITSs for music learning. It then presents the development approach used to implement SONATA, followed by a full description of SONATA's architecture, functionality and the mechanisms it deploys in making its strategy selection decisions.

1. Intelligent Tutoring Systems

For a Tutoring System to be classified as Intelligent, it must pass three tests of intelligence [Angelides and Doukidis, 1990]. First, the system must know the subject matter well enough to be able to draw inferences or solve problems in the domain of application. Second, it must be able to deduce a user-learner's approximation of the domain knowledge. Third, the tutorial strategy must allow the system to implement strategies that reduce the difference between the expert and the student performance. Therefore, at the foundation of an ITS one expects to find three special kinds of knowledge: domain, student, and tutoring knowledge.

The first key place for intelligence in an ITS is in the knowledge that the system has of its subject domain [Anderson, 1988]. There are three approaches to encoding knowledge into the domain model which gives rise to the three different types of domain models. The first approach, which gives rise to a black box model of the domain knowledge, involves finding a method of reasoning about the domain that does not actually require codification of the knowledge. A black box model generates the correct input-output behaviour over a range of tasks and so can be used as a judge of correctness. However, the internal computations by which it provides this behaviour are either not available or are of no use in delivering instruction. Such a domain model can be used in a reactive tutor that tells the students whether they are right or wrong and possibly what the right move would be. This is known as surface-level tutoring. The second approach, which gives rise to a glass box model of the domain knowledge, involves reasoning about the domain by applying codified knowledge. A glass box model is the standard knowledge based systems approach to reasoning with knowledge. Because of its nature, the emerging system should be more amenable to tutoring than a black box model because a major component of this expert system is an articulate representation of the domain knowledge. The third approach, which gives rise to a cognitive model of the domain knowledge, involves making the domain model a computer simulation of human problem solving in the domain of application.

The second key place for intelligence in an ITS is in the knowledge that the system infers of its student [VanLehn, 1988]. An ITS diagnoses a student's current knowledge of the subject matter and uses this to individualise instruction according to the student's needs. The ITS component that holds the student's current knowledge is the student model. The input for diagnosis is garnered through the interaction with the student. The output of diagnosis depends on the use of the student model. Nevertheless, it should reflect the student's current knowledge state. Common uses for the student model include advancing the user to the next curriculum topic, offering unsolicited advice when the student needs it, generating new problems, and adapting explanations by using concepts that the student understands. A student model usually consists of three kinds of information: bandwidth (i.e. quality and amount of student input), the type of domain knowledge (i.e. declarative, procedural or causal) and differences between the student and domain models in terms of missing conceptions (i.e. as an overlay model) and misconceptions (i.e. as a list of bugs).

The third key place for intelligence in an ITS is in the principles by which it tutors students and in the methods by which its applies these principles [Halff, 1988]. Tutor models may incorporate many different instructional techniques. A tutor model must exhibit three characteristics: (a) It must exercise some control over curriculum, that is, the selection and sequencing of material to be presented to the student, and some control over instruction, that is the process of the actual presentation of that material to the student, (b) it must be able to respond to student's questions about the subject matter, and (c) it must be able to determine when students need help in the course of practising a skill and what sort of help is needed. Some tutors are primarily concerned with teaching factual knowledge and inferential skills. These are the expository tutors. Some tutors are primarily concerned with teaching skills and procedures that manipulate factual knowledge. These are the procedural tutors. Curriculum can be broken down into, formulating a representation of the material in the domain model and selecting and sequencing concepts from that representation. A tutor model must also incorporate some form of propaedeutics, that is knowledge which is needed for enabling learning but not

for achieving proficient performance. The underlying assumption is that skilled performance will be achieved only with practice. As a result, propaedeutics serve, firstly, to relate theory to practice, secondly, to justify, explain, and test possible problem solutions, thirdly, as a stepping-stone to more efficient problem-solving strategies and, fourthly, as strategies for management of the working memory during intermediate stages of learning. Curricula serve several functions: (a) they divide the material to be learned into manageable units which should address at most a small number of instructional goals and should present material that will allow students to master them, (b) they sequence the material in a way that conveys its structure to students, (c) they ensure that the instructional goals presented in each unit are achievable, and (d) they enable the tutor model to evaluate the student reaction to instruction on a moment-to-moment basis and for reformulating the curriculum.

Tutoring Strategies Currently in Use Some of the common tutoring strategies currently in use include: apprenticeship, successive refinement; learning through exploration; practice; and socratic diagnosis.

Apprenticeship is an approach most often used by experts to teach skills in a craft. The expert demonstrates a skill with associated verbal explanation. The apprentice watches the tutor in action and asks questions. As time passes by, the apprentice is allowed to perform small parts of the whole tasks, and eventually the whole task in question. This technique may operate beyond the level of manual skill acquisition, as the same principle applies on processes such as problem solving and reasoning. A tutor using this approach must be able to reason meaningfully with the student. In terms of ITS, this strategy requires the system to have a glass box domain model. SOPHIE [Brown et al, 1975] is an ITS which employs the apprentice strategy. It tutors troubleshooting of electric circuits. SOPHIE mimics the human expert and apprentice relationship by providing a simulated laboratory, in which the student is given a demonstration of how a task should be done by a troubleshooting expert. The student's first activity is to watch the expert locate a fault. Once the expert succeeds in locating a faulty function block, the student is given a chance to locate the particular faulty component within that block.

Successive refinement is often used in cases where the tutorial material contains a substantial amount of details. A tutor using this approach addresses the domain primarily at a global level, by telling a consistent story but omitting the details. Increasing levels of details are presented as the student progresses. A tutor using this technique may have to constantly make justification for the untrue simplicity about the domain which he has previously established, and to actively support the reconstruction of the mental model of the domain within the student. STEAMER [Hollan et al, 1981] is an ITS which employs this approach. It is used to train engineers to operate complex steam propulsion plants in large ships. STEAMER uses the successive refinement approach by initially presenting a top-level view of the steam plant. As the student progresses, more sophisticated materials are tutored. This is achieved through a hierarchical decomposition of the domain which allows the student to explore subsystems in increasing levels of details.

Learning through exploration involves putting a pupil in a situation where he is allowed to act freely and to explore and discover new material for himself. The tutor employing this approach has to be responsible for selecting an area which is new to the student for him to explore, setting up an appropriate environment in which the learning can take place, and monitoring the student's activities, so as to be able to provide guidance to the student when requested, or when the student is in difficulty. WUSOR [Goldstein and Carr, 1977] is an ITS which uses this strategy. It is responsible for instructing the student on how to play the computer game WUMPUS [Goldstein, 1982]. WUSOR uses instructional games to tutor the student, focusing on the role the student creates for himself by playing the game, thus enhancing his own learning environment. There is often no direct knowledge being conveyed to the student. The learning material is disguised as elements of the game, making learning both challenging and more entertaining.

Practice is another strategy that is commonly used. A tutor employing this approach generates practice problems, monitors the student's practice activities and gives feedback. The value of practice activities lies in the stimulation they offer to the student in acquiring new

knowledge, and more importantly, in the application of the knowledge the student has acquired. The Lisp Tutor [Anderson and Reiser, 1985] is an ITS which employs the practice approach. It is a tutoring system for Lisp programming. The system is strongly directed towards recurrent skills and it provides a lot of practice in the basic skills. It has a structured editor and gives on-line help. The system offers a variation in the practice exercises, allows the student to make errors, and gives explanation about errors on request.

Socratic diagnosis is seen as a strategy employed in many ITSs. In its broadest terms, socratic diagnosis is an approach which incorporates some form of 'socratic dialogue'. This involves a question and answer sequence directed towards uncovering the underlying misconceptions of the student. In a stricter sense, this technique requires the tutor to firstly detect the misconception of the student about the domain, and then make the pupil realize that there is an error in his knowledge about the domain through a series of educational interactions. The strategy employed in WHY [Stevens et al, 1982] is commonly described as socratic – when the student makes a mistake at a level involving some concept the tutor will switch to a sequence of questions regarding subconcepts of the erroneous concept. If the student makes an error with one of them, it is further broken into its constituent concepts in turn. The major difference of this strategy from true socratic tutoring is that instead of diagnosing each of the misconception first, the strategy used in WHY is using the remedial action as diagnostic action for the next misconception.

Intelligent Tutoring Systems for Music Learning

Our research results in the field of music education show that while some basic computer-based approaches may have been used to assist music learning, tutoring systems classified as 'intelligent' which are designed for music learning are immensely scarce. Most of the existing computer-based systems for music learning encourage students to explore and to be creative in music learning, but seldom record or monitor the student's progress and diagnose individual's misconceptions.

MC [Holland and Elsom-cook, 1990], which stands for music composition, is a knowledge-based tutoring system which is still under development. This system aims to help novices explore musical ideas through experiments in music composition. It is intended to aid amateur musicians, in an informal setting, to acquire musical knowledge and skills with direct application. The framework suggested in MC involves linking the knowledge-based tutor to one or more 'musical microworlds'. The microworlds together form the 'environment' within which the student is attempting to learn. These microworlds can be used together as free learning or as a tool for composition. The knowledge-based tutor provides a source of guidance and a means of establishing interrelationships between the microworlds.

2. The Hypertext Approach

SONATA has been developed using a hypertext tool. Hypertext is the organisation of information into information nodes and links [Nielsen, 1990a]. Information nodes are linked via information links either sequentially, hierarchically or mixed [Nielsen, 1990b]. The main component of any hypertext system is the stack. A stack is a collection of related cards of information which are seen to logically belong together [Shneiderman and Kearsley, 1989]. These cards depict the information nodes. The link is the core of all hypertext systems [Smeaton, 1991]. With a hypertext framework for specifying an ITS, the domain, tutoring and student knowledge must first be organised into one or more stacks. Secondly, links will be installed to integrate the stored knowledge, not only within individual, but also between, knowledge models.

All the knowledge of SONATA is incorporated in one stack only. The purpose of this design is to shorten the on-line response time of the system, as it takes longer to cross reference across different stacks. A card may be linked to other cards with which it has a class-instance relationship. This establishes a semantic network of cards which are organised hierarchically so that properties can be inherited from generic cards to cards lower in the hierarchy [Marchionini and Shneiderman, 1988]. Links will be set up as

"organizational" hypertext information links to connect a parent card with its children and thus establish a hierarchical tree in this hypertext network. Cards can be linked to other cards with which they are not hierarchically related via "referential" hypertext information links, thus establishing a non- hierarchical structure in this network [Bergeman and Conklin, 1988]. Any information related to a card which cannot be included in the card structure, will be "annotated" to the card as a "typed" hypertext information node, if it is text, or as a "graphical" hypertext information node, if it is an image, or via an "annotation" link. This will establish a part-to-whole relationship with a given card. Within this annotation, there may be further referential, keyword or annotation links to cards which the annotation may relate to. A card may also be linked to another card by a "keyword" link, if two cards have the same value for a given attribute slot. The slot exists in a card in the form of a labelled field. The link names will carry a name which will depict semantic information.

There are two main reasons for following a hypertext paradigm for developing an ITS. The first one stems from the two fundamental limitations with the Expert Systems paradigm [Angelides, 1992]: Firstly, knowledge decomposition, representation and inferencing with Expert Systems is exclusively hierarchical. Secondly, Expert Systems lack explicit information linking since all relationships are established through reasoning. The second limitation raises a serious problem with respect to any possible attempts to interconnect the knowledge models. For instance, how does one represent non-hierarchical and thus non-inferentiable relationships established in the student's knowledge?

Explicit hierarchical and non-hierarchical information linking is regarded as one of the foremost advantages of hypertext. The application of different hypertext information links settles the first limitation of the Expert Systems paradigm to developing ITSs. Organisational links set up inheritance hierarchies and all other links set up non-hierarchical relationships. The use of hypertext information links which are exclusively explicit because they carry semantic information settles the second limitation of the Expert Systems paradigm to developing ITSs and also eradicates the need to perform logical reasoning in order to infer any direct, at least, relationships between related parts in a knowledge model or related knowledge models.

Nevertheless, hypertext on its own does not constitute a framework for developing an ITS because it lacks the logical inferencing mechanisms provided by Artificial Intelligence. Recent research and development on Artificial Intelligence has focused on hybrid models that are made up of Artificial Intelligence and hypertext. These models utilise hypertext's hierarchical and non-hierarchical information linking abilities with Artificial Intelligence's logical inferencing techniques.

The second main reason for using hypertext to develop an ITS is a pragmatic one. ITS development is usually a laborious process which assumes that a lot of human and other resources will be made available during the course of development. The Expert Systems paradigm for developing an ITS can be time-intensive and at the same time inflexible when compared to the hypertext approach. While the Expert Systems version of an ITS took seven months to develop [Angelides and Garcia, 1993], the same ITS developed using HyperCard II [Angelides and Gibson, 1993] only took three weeks, leaving also a lot of room for improvement and re-engineering.

3. A Sample Interaction With SONATA

The system is invoked from the "home card" of HyperCard II.1. When the user clicks the 'START' button on the start-up screen, SONATA asks for his name to check whether he is already a registered user (see figure 1) and, if not, to construct a student record card for him to make him one.

Once this has been concluded the tutorial lesson begins. In the case of a new user, SONATA presents him knowledge in the first area of the domain, in the form of a scenario, as shown in figure 2.

The user clicks around with the mouse to discover the knowledge embedded in the scenario, as shown in figure 3.

When the user clicks the 'OK' button, he is returned to the original senario. Help is available on request when the user clicks the 'HELP' button. The user can leave the scenario by clicking

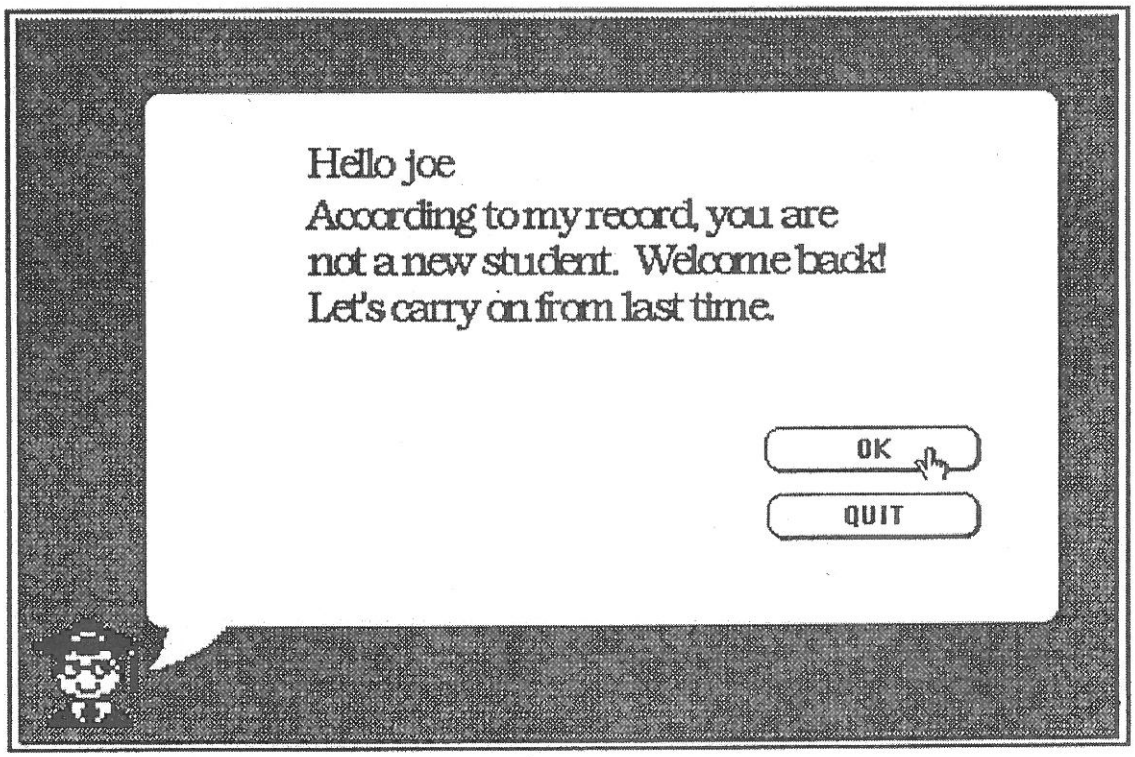

Fig. 1. Student's status checked

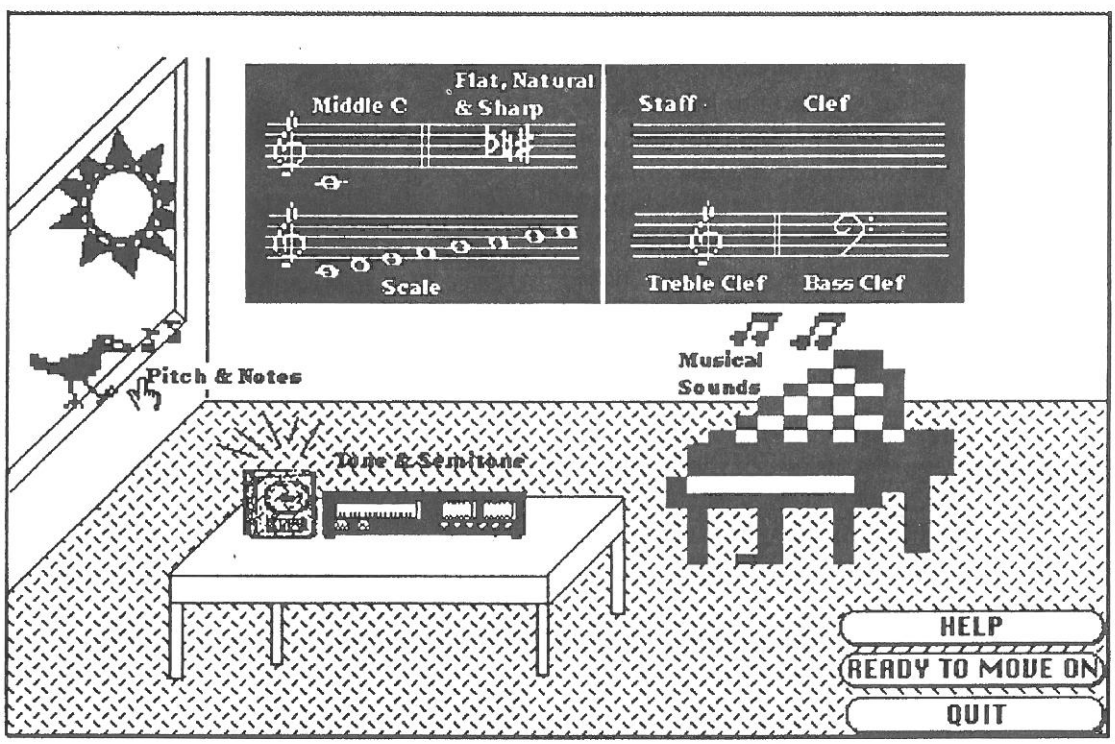

Fig. 2. Scenario of area one of SONATA's domain

the 'READY TO MOVE ON' button. SONATA will prompt him to do so when it 'thinks' that the user is ready.

On leaving the scenario, the user has to go through a series of problem solving activities, one at a time, by applying the knowledge he acquired from the scenario he was previously in. Each activity can be tutored using up to three different strategies. Which strategy is to be used depends on the student's performance and the criteria set by SONATA. The student places a proposed solution in the box provided and informs SONATA of his attempt by clicking the 'COMPLETE' button. If the answer is right the student is advanced by SONATA, based on performance levels, to the next appropriate problem. If the answer is wrong, SONATA diagnoses the next best strategy to be used and tutors the same activity again. Figure 4 shows an interaction of a student engaged in a problem solving activity.

At the end of every attempt SONATA awards the student with a score. This score is kept in a unique student record card. This interaction continues until there are no more problem solving activities available related to the specific area or the student wishes to leave the system. In the first case, the student will be presented a scenario about the second area of the domain where the cycle starts again. In the second case, the student can leave at any point of the lesson by clicking the 'QUIT' button. When the student has completed all the problem solving activities in the various areas of the domain, SONATA will calculate a performance score based on the scores recorded in the student's record card. This overall score will be analyzed and SONATA will advise the student on whether he should continue interacting with the system or that he will no longer gain any benefit from interaction as he is deemed to be as good as the expert, as shown in figure 5.

The prototype was tested with a small group of students who have no previous knowledge in music theory. They commented that the tutorial sessions have been benefitial and the system is 'relatively' easy to use. On the basis of the interaction of that group of students with SONATA it has been estimated that it takes an average student between two and three hours of continuous interaction to complete all the tests.

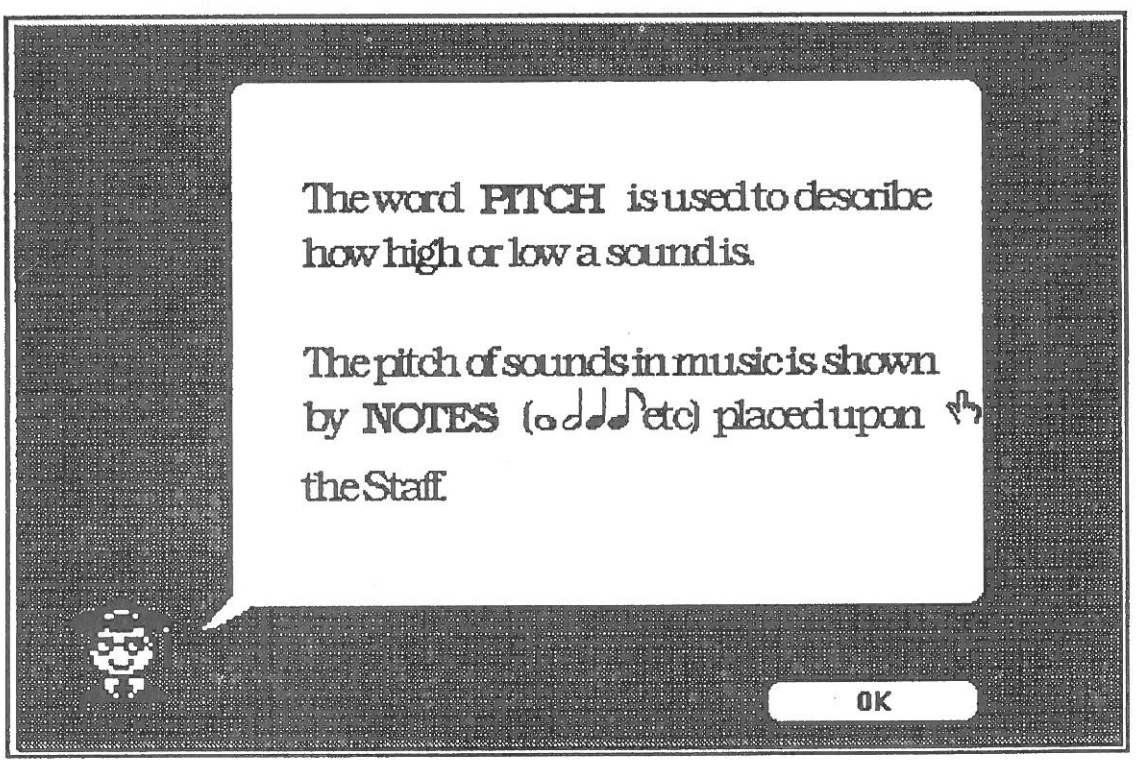

Fig. 3. Association knowledge presented

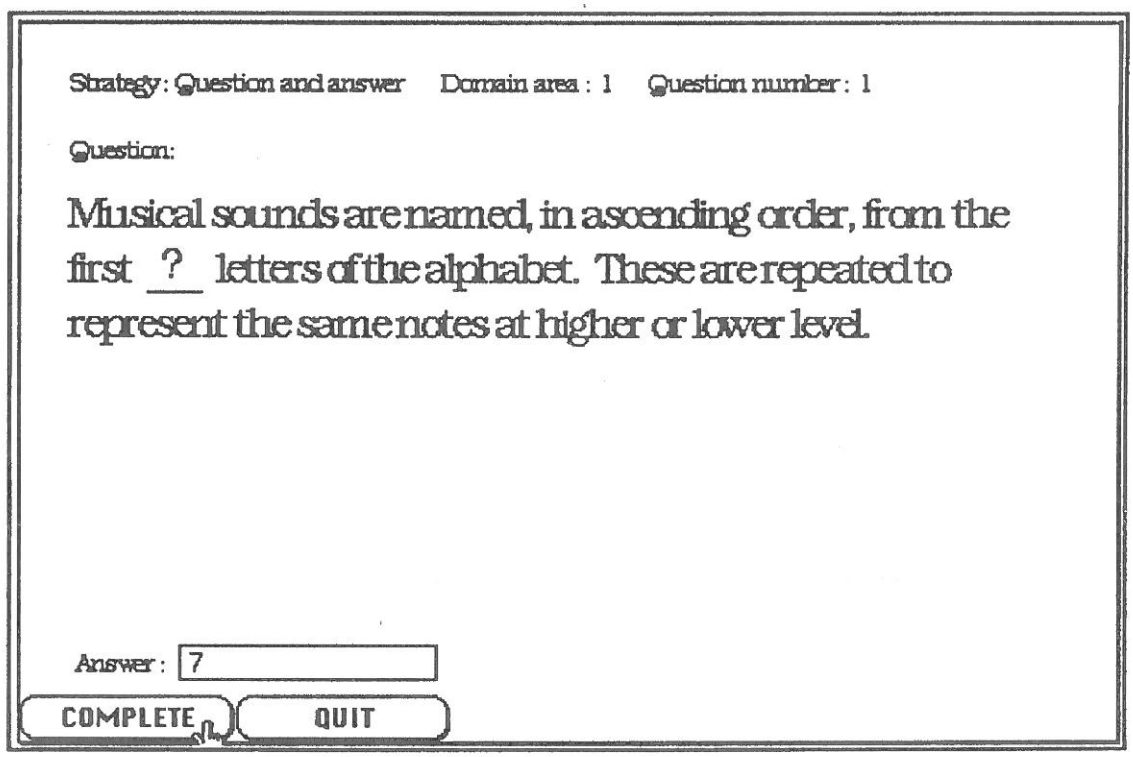

Fig. 4. Student attempts a problem

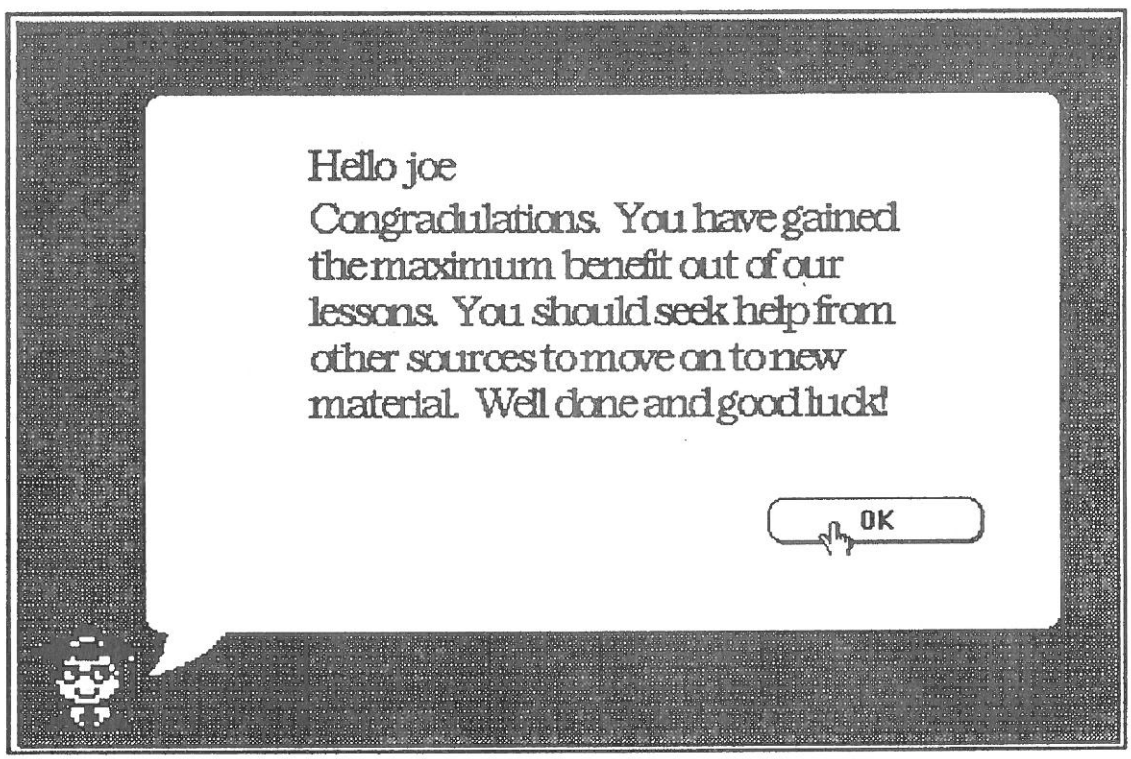

Fig. 5. Student's performance score analyzed

5. Development and Implementation of SONATA

SONATA is a domain independent ITS prototype for music theory learning. The tutorial material it presents is factual knowledge covering the syllabus of the grade one theory of music examination set by the Associated Board of the Royal Schools of Music. The domain is divided into four areas. These areas vary in difficulty and are introduced one after another, with the easiest introduced first. Tutoring in each area involves SONATA initially teaching the student about the area, followed by the student applying his acquired knowledge through twelve successive problem solving activities, before moving into a new area. SONATA provides four actual teaching scenarios and forty eight problem solving activities altogether.

Tutoring Strategies Used in SONATA

The strategies employed in SONATA are: learning through exploration; practice with a hint; multiple choice; and strict question and answering. Learning through exploration presents the student with a scenario full of domain icons, that is, icons containing knowledge about one of the four areas of the domain. The student may choose freely using the mouse. When a domain icon is selected, the associated knowledge is presented to the student. The student is then expected to read and learn the material, and signals the return to the original scenario when he is ready for more exploration. Practice with a hint requires the student to practise by applying his knowledge associated with the material presented by SONATA. A hint is supplied alongside with each practice as guidance. The multiple choice approach presents on the screen an incomplete statement associated with the knowledge domain. The student then has to select an item from a list of alternative choices offered by SONATA in order to provide the missing information. The strict question and answering strategy requires the student to answer questions associated with the domain without the help of any hint or choices.

Each of the strategies used in SONATA is distinguished from one another. Although practice with a hint, multiple choice and strict question and answering may all involve some form of question and answering activity, they differ from one another as they may influence the student's problem-solving behaviour in different ways. While practice with a hint gives indirect guidance to the student, the multiple choice approach explicitly puts the correct answer, mingled with other apparently equally likely but incorrect answers in front of the student, without any clue as to which one is the right answer. Strict question and answering differs from the two, in that no guidance or possible answers are initially given to the student.

The educational and psychological validity of these strategies is not the main focus of this paper. Here we take the view of how appropriate a strategy is to a particular student to be subjective. When applied on a student who knows the domain thoroughly, each of the four strategies may be as appropriate as one another. As for students who are still familiarizing themselves with the domain, while multiple choice may be a more suitable strategy to use than strict question and answering for one individual, another student may find it confusing to have to think about and choose between the alternatives. In that case, strict question and answering may potentially be more appropriate. Hence except for the application of learning through exploration, which is to be used as a first strategy when a new area is to be introduced, no other order of application is presumed for the rest of the three strategies employed in SONATA. Since SONATA has no prior knowledge of a first-time student and it assumes no prior knowledge on the domain on behalf of the student, it is not possible to select an 'appropriate' strategy for a new student before experimenting for a while. The control mechanism in switching between strategies does not follow any extremely rigid presumed pattern.

Criteria for the Strategy Selection in SONATA

At the conceptual level, the factors affecting SONATA's strategy selection decisions are: whether an area is introduced for the first time; the student's prior success with the different strategies; and the suitability of a strategy to a particular type of question involved in a problem solving activity.

When introducing an area to the student for the first time, learning through exploration is always used. This is because the other three strategies are all 'knowledge application' strategies, that is, they tutor through the student's application of his knowledge about the area. Therefore they are not applicable when the student has not even acquired any knowledge on the area. The criteria for switching from learning through exploration to one of the three strategies is based on the number of times the tutorial material in that area has been referred to. This is to prevent the student from 'dwelling' in the exploration scenario for too long without having to apply the knowledge he acquired from his exploration experience. The strategies to be used for the problem solving activities are practice with a hint (Strategy P), multiple choice (Strategy MC), and strict question and answering (Strategy QA). For each activity, SONATA's choice of strategy depends on the student's prior success with the strategies. A strategy is regarded as successful if the student provides the correct solution in a problem solving activity which employs that strategy. This factor affects SONATA's strategy selection at two levels. At a local level, the decision depends on the performance of the student in the previous activity. For instance, given that Strategy MC is unsuccessful as the first strategy used in problem solving activity three of area one, and that Strategy QA is successful as the second strategy to tutor the same activity, then the first strategy to be used in problem solving activity four in area one will be Strategy QA. If it fails, SONATA will switch to Strategy P and tutor the same activity again. Strategy QA is used first because it was successful in the previous activity. Strategy P is preferred to Strategy MC as a second strategy because the former had not been attempted in the last activity whereas the latter failed.

At a higher level, SONATA's strategy selection is influenced by the overall success ratings of the strategies. When twenty five percent of the forty eight problem solving activities in total have been completed, data from the student model will be gathered to infer, for the first time, the overall success rate of each of the three knowledge application strategies. SONATA will then be informed of which strategy is comparatively the most successful, moderately successful and the least successful in general up to the twenty five percent mark. This information is updated after each step in the interaction form then on. The most current information available on the success ratings control SONATA's strategy selection in the next twenty five percent of the problem solving activities. The type of question involved factor reflects the influence of the 'student's prior success' factor on SONATA's strategy decisions at a global level. Here an additional agent is taken into account. Every question in the problem solving activities belongs to one of four types. Each type of question is characterized by a question's physical appearance, and the aim of the question, which is partly content related. When fifty percent of the forty eight problem solving activities have been completed, data from the student model will be collected to infer, for the first time, the success rate of each of the three knowledge application strategies on each type of question. These 'type' success ratings are updated after every step in the interaction thereafter and control SONATA's strategy decision for each question in the problem solving activities according to its type until the end of area four.

The values chosen to checkmark the switch between different strategies in SONATA were initially arbitrary. A substantial amount of laboratory test runs were then carried out on SONATA's intended users, that is, beginner students in music theory. According to the data collected from the resulting student models, the values presented in this paper, namely the twenty five percent mark, the fifty percent mark and the five consecutive question criteria, appear to be the logically appropriate values for the target group of students. There has been no proven teaching theory that lays down any particular rules or guidelines for the determination of such values. In the light of the prototype development of similar systems such as the Lisp Tutor [Anderson and Reiser, 1985], it is suggested that the utilization of experimental test run results can be a good practice with respect to the improvement of the design of an ITS.

Specific rules are used to control the criteria for SONATA's strategy selection. At the implementation level, these rules are translated into HyperCard scripts, which are pieces of program code written in HyperTalk. There are also rules used to manipulate the different types of knowledge in SONATA. All the rules are embedded in different processors within the domain model,

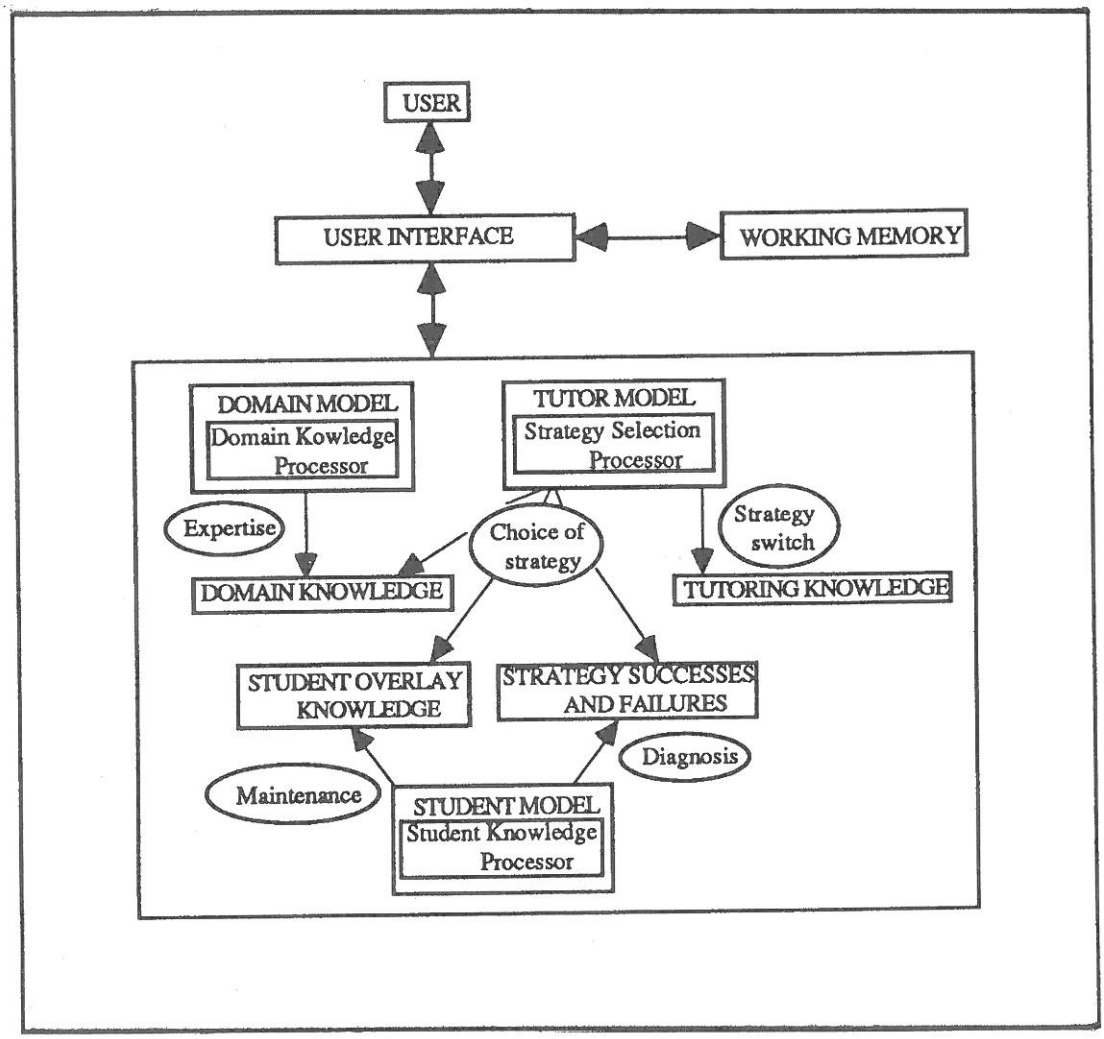

Fig. 6. SONATA's architecture

the tutor model and the student model of the system. The rule based mechanisms behind these processors take the form of rule sets, and are achieved by using IF-THEN-ELSE control structures, together with other 'hypertextual' commands to carry out the necessary deductive reasoning. Figure 6 shows SONATA's architecture in terms of SONATA's components.

Domain Model

The domain model contains the domain knowledge and the domain knowledge processor. The domain knowledge includes knowledge on the four exploration scenarios and the solutions to the forty eight problem solving activities. The processor is responsible for providing an exploration scenario with the associated knowledge, a question for a problem solving activity, an indication of whether a student's answer is correct, and the correct answer to a question when necessary.

Within the SONATA stack, there is one card allocated to the exploration scenario in an area. Another ten cards are allocated to hold the tutorial material associated with that scenario. Each of the ten cards is hierarchically linked with the scenario card. Three cards are allocated to a question for each problem solving activity. Each of the three cards represents either Strategy QA, Strategy MC, or Strategy P.

When the student is engaged in a problem solving activity, the student's proposed answer is stored in a labelled field in the foreground of a question card, overlapping SONATA's model

answer for that question, which is stored in a labelled field in the background in the same position of the same card. The model answer is normally made invisible by the overlapping foreground field. The domain model compares the values in the foreground and the background fields to check and signal for the correctness of the student's answer. If an answer is to be given to the student, the foreground field will be temporarily hidden, revealing the background field with the model answer.

Tutor Model

The tutor model contains the tutoring knowledge and a strategy selection processor. The tutoring knowledge includes knowledge on the four tutoring strategies and the criteria for the strategy selection. The processor refers to the student model, the domain knowledge, and the criteria within the tutoring knowledge, in order to determine which strategy is to be used at a given point for a particular student, and when to switch from one strategy to another. Some mechanisms controlling the major rules in SONATA's strategy selection are:

1. Using the general best strategy first

This mechanism corresponds to the 'student's prior success with the strategies' factor considered at the conceptual level. This mechanism is relevant when the student is engaged in the problem solving activities before the twenty five percent mark. It is used when the information from the student model indicates which strategy is the most preferred at a given point. The tutor model always uses the best strategy first. If it fails, subsequent strategies will be used in order of preference indicated by the student model. Its choice of strategy is then passes onto the domain model, such that the appropriate card will be called upon the screen.

2. Choosing a strategy at random

This mechanism also corresponds to the 'student's prior success' factor considered at the conceptual level. It is used when the information from the student model does not suggest a specific preferred strategy for the next interaction. For instance, when the success ratings of the strategies are the same at a given point. In such a situation, one of the three knowledge application strategies with be picked at random to be used for the next interaction.

3. Unexpected strategy change

This mechanism is related to both the 'introduction of a new area' factor and the 'student's prior success' factor at the conceptual level. One function of this mechanism is to inform the domain model to call up the exploration scenario on the screen when introducing a new one, rather than to continue with one of the knowledge application strategies used at the end of the last area, hence resulting in a sudden change of strategy. The second function of this mechanism is to prevent the student from being trapped in the tutoring with a single strategy for too long. Examples of this situation can be: when a student lingers in an exploration scenario; or when before the twenty five percent mark, a single strategy has been successful in five consecutive problem solving activities within the twelve in an area. In the first case, the student model keeps a count of the number of times the student has clicked each domain icon in an exploration scenario and passes this information onto the tutor model. When the student has referred to each domain icon in an exploration scenario of an area at least once, the tutor model will advise the student to move on to the problem solving activities. If the student indicates he is not ready, he may continue to explore for the second round. When the student has gone through all the icons at least once more again, the tutor model automatically stops using learning through exploration and starts tutoring through problem solving activities by switching to one of the knowledge application strategies.

In the second case where a strategy has been successful in five consecutive problem solving activities, the tutor model will switch to one of the rest of the two knowledge application strategies for the next activity. The tutor checks for the consecutive success by referring to the student model. This is an attempt to make sure that all strategies will have a chance to be employed, such that the purpose of SONATA's multiple strategy tutoring is not undermined.

4. Using the best strategy concerning the type of question

This mechanism corresponds to the 'type of question' factor at the conceptual level. It is similar to the 'using the general best strategy first' mechanism, except that this time, the type of question the current question belongs to is also taken into account. Implementing this extension involves adding another global variable which keeps the information on the type of question. A number representing the 'type' is tagged along each question card so that the 'type' variable is updated each time a different card is called up.

Student Model

The student model consists of the student knowledge and a student knowledge processor. The student knowledge includes knowledge on the score of each problem solving activity of the student and his prior success and failure with the different strategies. The student knowledge is stored in a collection of student record cards. They form a permanent entity which keeps a historical record of each individual student's interactions with the system. Each student record card belongs to an individual student who has previously interacted with SONATA. Checks for new students are made by seeing whether a record card for that student already exists. A new card is created for every new student.

A student record card can be divided into two main parts. The first part contains one hundred and forty four fields. They store the overlay scores allocated by the inference rules used in the student knowledge processor. Together they make up the student overlay record. This part of the student record card shows the subset relationship between the knowledge of the student and the expert of the domain. The second part of the record card keeps a history of the success, and the current success ratings of an individual student with the different strategies. The tutor

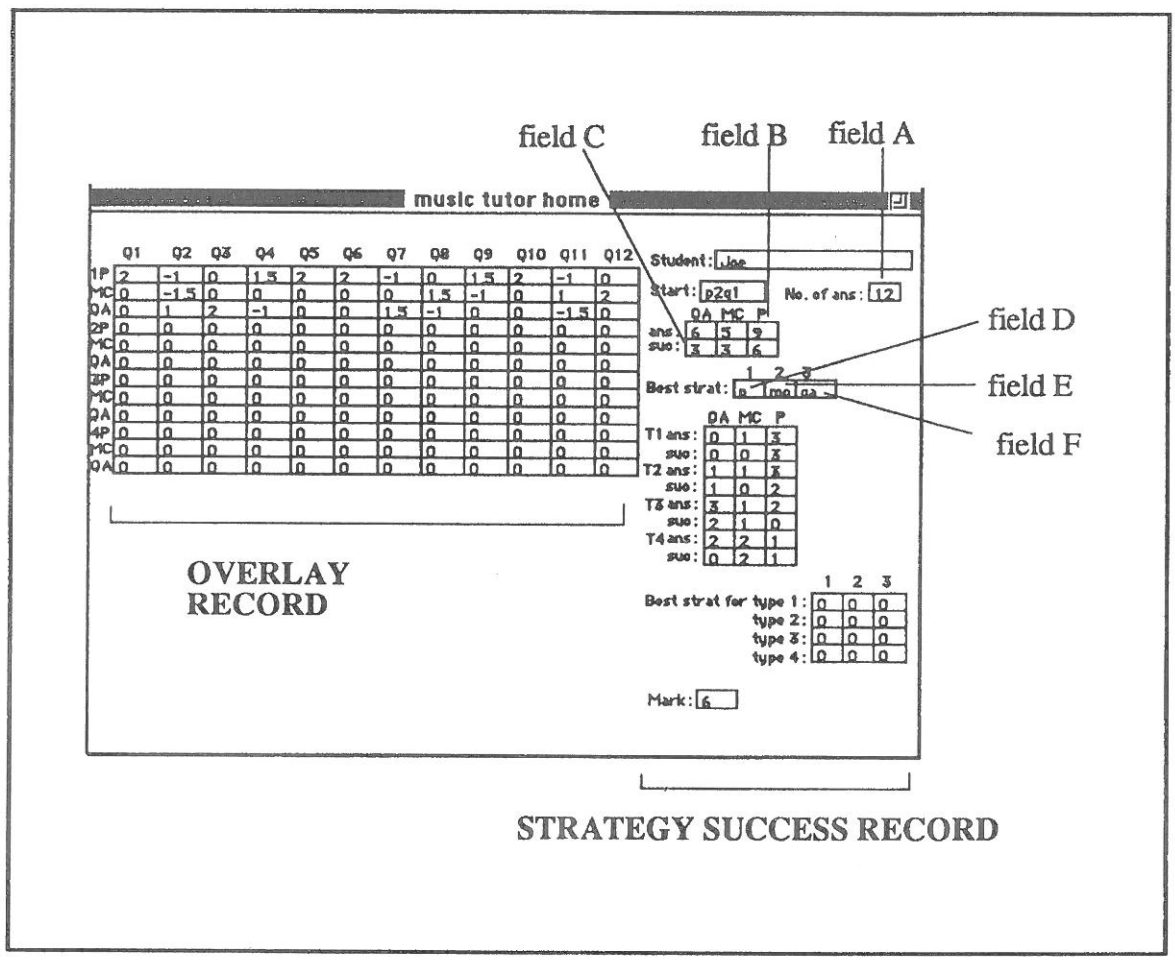

Fig. 7. A sample student record card

model refers to both parts of the student record card when making its strategy decision. Figure 7 shows a sample student record card.

The student knowledge processor is responsible for keeping each student record and subsequently the entire student model up-to-date. The overlay record part of a record card is maintained by inferring the student's current level of understanding of the domain. To this end, the student knowledge processor employs a set of strategy specific overlay rules to allocate an overlay score for each strategy specific question completed by the student engaged in a problem solving activity. Since these overlay scores are strategy specific, three scores are allocated to each problem solving activity, as three different knowledge application strategies can be used for each activity.

The overlay rules governing the score on each question are kept in the script of the 'COMPLETE' button on each question card, which the student has to click to indicate he has completed his attempt at that question. The overlay rules are then triggered. The resulting overlay score will be added to the corresponding field in the overlay record on the student's record card. The overlay rule set covers all the possible concoctions which may be encountered. Each overlay score awarded is within the range of -2 to 2, and is entirely dependent on the combination accured. An example of the combination rules is shown below:

```
IF (studentAnswer=modelAnswer) and
(this is the 2nd strategy used)
THEN depending on other conditions
checked by other parts of SONATA
put 1.5 into background field
X of the corresponding student record card
ELSE IF . . .
```

Before twenty five percent of the total number of problem solving activities are completed, the tutor model bases its strategy selection on information provided by the student overlay record.

The second half of the student record card concerns the student's prior success with the different strategies. The student knowledge processor maintains this half of the record card by keeping a count of the number of problem solving activities the student has completed, the number of questions the student has attempted with each strategy, and the number of successes the student has with each strategy. The counts are kept in individual fields on the student record card (for example, fields A, B, and C respectively as shown in figure 7). When the count in field A shows that the twenty five percent mark has been reached, the student knowledge processor calculates the general success rate of each knowledge application strategy by dividing the number of successes with strategy X by the number of questions attempted with strategy X. The three resulting success rates are compared. The most successful, the moderately successful and the least successful strategies are kept separately in another three fields on the record card (for example field D, E and F respectively in figure 7). The tutor model bases part of its 'unexpected strategy change' and its 'using the general best strategy first' decision mechanisms on this set of ratings. Each time a question is attempted, all the counts are updated, yielding a different set of success ratings. Information in the corresponding fields continue to be updated accordingly until the number of problem solving activities completed reaches fifty percent. The tutor will then base its strategy selection decisions on another set of ratings, that of the success ratings concerning the type of question.

When the corresponding count indicates that the fifty percent mark is reached, the success rate of each strategy for each type of question concerned is calculated by dividing the number of successes with strategy X on type Y questions by the number of type Y questions attempted with strategy X. The associated counts and fields are updated after the attempt of every single question until the end of area four. The tutor model bases its 'using the best strategy concerning the type of question' decision mechanism on this set of ratings.

There is a record kept by the student model which is not registered on the student record card. This is the record related to learning thorough exploration. When a student is engaged in learning through exploration, the student model keeps a temporary record on the number of times the student has referred to each domain icon. The tutor model bases part of its 'unexpected change of strategy' decision mechanism on this record. This record is only kept in the short term as local variables because the information it provides will no longer be useful when the tutor model moves onto tutoring through problem solving activities.

6. Concluding Disscussion

This paper has shown the possibility with SONATA of the development and implementation of an ITS that offers multiple tutoring strategies and can switch between them in a sensible manner. SONATA is an experimental prototype and as such succumbs to many shortcomings which may, however, provide the necessary ground for any future research and development efforts to enhance it.

Short-term research and development to enhance the system performance may be undertaken in several areas. As far as SONATA's strategy selection is concerned, the system bases all its decision on its diagnosis of the student and the student's own preference is never consulted [Siemer and Angelides, 1993]. Improvement of the system can be made by identifying situations where the student should be able to express his preference. For example, at a given point at which the student's prior success rates with the different strategies are equal, rather than selecting the strategy for the next problem solving activity at random, SONATA should allow the student to choose a strategy himself. SONATA can even go further onto making a record of the student's initial preferences and take them into account in its future strategy selection decisions.

SONATA assumes that each new user is an absolute beginner. With respect to pre-modelling, the system may be made more sensitive to the student's needs if the system can ask the user a series of standard questions, so as to deduce the level of knowledge of a new student about the domain. As for student modelling, at least one more element can be added to SONATA's student model to enhance the model's usefulness as an informative agent, that of a set of mal- rules. With the incorporation of mal-rules, SONATA will then be able to diagnose and represent a student's misconcpetions as well as missing concepts, such that remedial activities can be offered accordingly. While misconeption diagnosis may not be a requirement in certain ITSs, it would be appropriate if SONATA can take into account the type of misconceptions a student has about the domain when deciding which strategy is to be used.

More works can be done to enhance the communciation channel between SONATA and its student. At the moment, the system is not able to answer arbitrary questions or hyperthetical questions from the student about the subject matter, nor can it give an explanation of a problem solution. This is due to the lack of a natural language interface since SONATA does not incorporate the underlying grammar that reflects the semantics of the domain. Further work on SONATA in the long run will eventually have to be diverted to an extensive research on natural language processing if SONATA is to be able to repond intelligently to questions and situations posed by the pupil. In the short term, future development can aim at improving the existing menu-based interface, perhaps by increasing the variety of legitimate inputs from the user and the corresponding preset output messages, so as to provide a wider channel of communication between SONATA and its users.

In terms of implementing multiple tutoring strategies in ITSs in general, future researchers should not merely rest upon developing a model which can merely accommodate a number of different strategies and being able to switch between them in a reasonable manner. In order to recognize such an ITS as a system that is educationally and psychologically valid, more solid foundation on which its strategy selection is based is necessary. The core of implementing multiple strategies in ITSs requires more specific and detailed accounts of teaching. We have to understand why, when and how human tutors switch strategies, and more fundamentally in what way do the strategies themselves and the changes in strategies affect a student's learning, which in turn bring us to the question of how does one learn. In order to understand the essence of the interaction between the tutor and the student, we must jump out of the limits set by the boundaries of existing ITS architechture to try to capture and model other aspects in the field of education, such as reasoning and learning processes [Elsom-cook, 1991].

References

1. J. R. ANDERSON (1988) The Expert Module. In *Foundations of Intelligent Tutoring Systems* (M. C. POLSON and J. J. RICHARDSON, Eds.), pp. 21–53. Lawrence Erlbaum, USA.

2. J. R. ANDERSON AND B. J. REISER (1985) The Lisp Tutor. *Byte*, 10(4)

3. M. C. ANGELIDES (1992) *Developing the Didactic Operations for Intelligent Tutoring Systems: A synthesis of artificial intelligence and hypertext.* PhD Thesis. University of London.

4. M. C. ANGELIDES AND G. I. DOUKIDIS (1990) Is There a Place in OR for Intelligent Tutoring Systems? *Journal of the Operational Research Society*, 41(6), pp. 491-503. Reprinted in *Artificial Intelligence in Operational Research* (G.I. DOUKIDIS and R.J. PAUL, Eds.), pp. 287–299. Macmillan.

5. M. C. ANGELIDES AND I. GARCIA (1993) Towards an Intelligent Knowledge Based Tutoring System for Foreign Language Learning. *Journal of Computing and Information Technology*, 1(1), pp. 15–28.

6. M. C. ANGELIDIES AND G. GIBSON (1993) PEDRO – The Spanish Tutor: A hypertext-based intelligent tutoring system for foreign language learning. *Hypermedia*, 5(3).

7. M. L. BEGEMAN AND J. CONKLIN (1988) The Right Tool for the Right Job. *Byte*, 13(10), pp. 255-267.

8. J. J. BROWN, R. R. BURTON AND A. G. BELL (1975) SOPHIE: A step towards learning environment. *International Journal of Man-Machine Studies*, Vol 7.

9. M. T. ELSOM-COOK (1991) Dialogue and Teaching Styles. In *Teaching Knowledge and Information Technology* (GOODYEAR, P.). Ablex Publishing Coporation.

10. I. P. GOLDSTEINS AND B. CARR (1977) The Computer as Coach: An athletic paradigm for intellectual education. *Proceedings of the International ACM Conference*, Seattle, Washington D.C.

11. I. P. GOLDSTEIN (1982) WUMPUS. In *Handbook of Artificial Intelligence*, Vol 2 (A. BARR and F. A. FEIGENBAUM). Addison-Wesley, USA.

12. H. M. HALFF (1988) Curriculum and Instruction in Automated Tutors. In *Foundations of Intelligent Tutoring Systems* (M. C. POLSON and J. J. RICHARDSON, Eds.), pp. 79–108. Lawrence Erlbaum, USA.

13. J. D. HOLLAN, M. D. WILLIAMS AND A. L. STEVENS (1981) An Overview of STEAMER: An advanced computer-assisted instruction system for propulsion engineering. *Behaviour Research Methods and Instrumentation*, Vol 13.

14. S. HOLLAND AND M. T. ELSOM-COOK (1990) Architecture of a Knowledge Based Music Tutor. In *Guided Discovery Tutoring* (ELSOM-COOK, M.T.), pp.70–93. Paul Chapman Publishing.

15. G. MARCHIONONI AND B. SHNEIDERMAN (1988) Finding Facts vs. Browsing Knowledge in Hypertext Systems. *IEEE Computer*, January, pp. 70-80.

16. J. NIELSEN (1990a) *Hypertext and Hypermedia.* Academic Press, USA.

17. J. NIELSEN (1990b) The Art of Navigating through Hypertext. *Communications of the Association of Computing Machinery*, 33(3), pp. 297-321.

18. B. SHNEIDERMAN, AND G. KEARSLEY (1989) *HYPERTEXT-HANDS-ON!: An Introduction to a New Way of Organising and Accessing Information.* Addison-Wesley, USA.

19. J. SIEMER AND M. C. ANGELIDES (1993) Towards a Model For Remedial Operations in Intelligent Tutoring Systems. In *Opportunity and Risks of Artificial Intelligence Systems* (M.C. ANGELIDES and J. SIEMER, Eds.), Proceedings of the Artificial Intelligence Stream, 35th Annual Operational Research Society Conference, University of York, September, pp. 4–32.

20. A. F. SMEATON (1991) Retrieving information from hypertext: Issues and problems. *European Journal of Information Systems*, 1(4), pp. 239–247.

21. A. STEVENS, A. COLLINS AND S. E. GOLDIN (1982) Misconception in Students Understanding. In *Intelligent Tutoring Systems* (D. SLEEMAN and J. S. BROWN, Eds.), pp. 13–24. Academic Press, USA.

22. K. VANLEHN (1988) Student Modelling. In *Foundations of Intelligent Tutoring Systems* (M. C. POLSON and J. J. RICHARDSON, Eds.), pp. 55–78. Lawrence Erlbaum, USA.

Received: September, 1993
Accepted: May, 1994

Contact address:
Marios Angelides
Information Systems Department
London School of Economics and Political Science
Houghton Street
London WC2A 2AE
G. Britain

Dr. MARIOS ANGELIDES is a Lecturer in the Information Systems Department at the London School of Economics. He holds a B.Sc. degree in Computing and a Ph.D. in Information Systems both from the London School of Economics. He has six years of experience in researching in the area of Intelligent Tutoring Systems in which he completed his Ph.D. He has authored and co-authored 12 journal papers in Intelligent Tutoring Systems. In addition he has published another six articles in other areas of Artificial Intelligence, one of which is in the area of Artificial Intelligence and Simulation. He is the co-author of the book, Lisp: From Foundations to Applications published in 1988. He is Vice-Chairman of IFIP's (International Federation for Information Processing) Working Group 9.5: Social Implications of Artificial Intelligence Systems.

Ms. AMELIA TONG is a full-time research student reading towards the degree of Ph.D. (Econ) in Information Systems in the Information Systems Department at the London School of Economics. Her research area is that of Intelligent Tutoring Systems. She holds the degrees of the B.Sc. in Computing, and the M.Sc. in Analysis, Design, and Management of Information Systems, both from the London School of Economics.