OBJECT-ORIENTED MODEL OF KNOWLEDGE FOR THE INTELLIGENT TUTORING SHELL

Ljubomir Jerinic  
Institute of Mathematics, University of Novi Sad  
Department of Computer Science  
Trg Dositeja Obradovica 4, 21000 Novi Sad, Yugoslavia  
Email: Jerinic@uns.ns.ac.yu

Vladan Devedzic  
FON - School of Business Administration, University of Belgrade  
Department of Information Systems  
Jove Ilica 154, 11000 Belgrade, Yugoslavia  
Email: edevedzi@ubbg.etf.bg.ac.yu

ABSTRACT

Rapid development of computer technologies and Artificial Intelligence (AI) methods, and introduction of computers into schools have yielded an ever-increasing use of AI methods in design of education software. The teachers in schools have become interested in taking an active role in designing educational software, and not being only passive users of these tools. However, the advancement of AI methods and techniques makes understanding of Intelligent Tutoring Systems (ITS) more difficult, so that the teachers are less and less prepared to accept these systems. As a result, the gap between the researchers in the field of ITS and the educational staff is constantly widening. This paper describes our efforts toward developing uniform data and control structures that can be used by a wide circle of authors, e.g., domain experts, teachers, curriculum developers, etc., who are involved in the building of ITS.

1. INTRODUCTION

The endeavor to design Intelligent Tutoring Systems and Intelligent Learning Environment [Fra88, BOP93, and Gre95] has opened a new research direction in the application computers for educational purposes. These systems are capable to help solving difficult problems in the process of knowledge transfer, and enable an almost full individualization of the teaching process, as well as the student's advancement in the stages of acquiring knowledge and application of the new knowledge according to their capabilities, aspirations, previous knowledge and the like. Also, ITSs stimulate the use of computers in education as a new education tool and controller of the teaching process, and opened new approaches for the use of computers in education.

On the other hand, it is generally accepted view within computer community that the 1990’s are the decade of successful application of Artificial Intelligence (AI) expertise as real world systems. High on the list of the new software technologies that are going to be applied in the marketplace are Intelligent Systems (IS) or knowledge-based systems. Certainly, what distinguishes IS from the more traditional computer programs are not only their different names. Every computer program makes use of some kind of specialized knowledge to solve the problems of its application domain and is, therefore, in some sense knowledge based. The main difference lies not in use of knowledge but above all in the form of organization within them and in their general architecture.

In IS, the paradigm of problem solving in the application domain is explicitly expressed of as a separate entity, called knowledge base, instead of being hidden in the program's code. The knowledge base is managed by the separate, clearly identifiable, control component, usually a general purpose inference engine that is capable of making decision from the knowledge stored in knowledge base, answering questions about this knowledge, and determining the consequences implied by it. This means that the knowledge incorporated into intelligent systems is independent of the program that manipulates it. That knowledge could be easily used by the users and potentially more easily modified or extended.
The intensive development of the methods of knowledge engineering and reasoning enables that education by the computers could be possible. The main goal of that approach is the complete individualization of the educational process. We assume, that the tutoring is a task of helping a student or pupil to learn particulars in a certain domain. This requires a special concern about the cognitive load that the student or pupil will bear in the learning process. The research on intelligent tutoring systems has produced the technology to do this task within the computer program. But, as the computers have entered teaching practice, teachers show an increasing interest in educational software systems. For these teachers, however, ITSs is the most difficult one in terms of grasping the total aspects [MW90]. Of course, they have a chance to read articles about ITS but what they can understand through the articles is confined to the abstract level of comprehension. Experience in using ITSs directly through a system that transperer the inner mechanisms of ITSs would be a powerful tool to help them in comprehending it.

As research in ITSs continues to produce more refined systems, the gap between the research community and educational community continues to become wider. The educators' understanding, and usage of these research results have been much slower than the research progress. As this theory-application gap becomes wider, it becomes more difficult for educators to participate in ITS research and application, and the result of the research becomes increasingly academic and unconnected to the pragmatic aspects of teaching and learning.

In designing and testing tutoring strategies several problems are encountered. First, the experience of learning via ITSs is such a novelty that neither teachers nor theorists can anticipate many important issues. Second, teachers generally do not have well-articulated and mathematically or algorithmically defined theories of learning. Third, relevant instructional and cognitive theories are not operationalised to a level easily realized as the computer programs.

Also, the educational programs are different among the distinct countries. Then, the required knowledge for learning some lesson in some domain is not the same even in one country itself (the north - south problems, country - town, etc.). The teachers want to have the active role in design and eventually update and improvements of intelligent tutors.

In view of the above trends and research issues, the goal of designing the EduSof [Jer94, JD95] system was to construct a conceptual framework for representing the objects, events, responses, reactions, and relationships involved in tutoring. Also, the aim was to build a highly-useable knowledge acquisition interface for rapid prototyping and easy creation, modification, deletion, and testing of both teaching concepts and tutorial strategies. Finally, our intention was to produce a system that enables teachers to build their own lessons without any computer or AI specialist.

2. OBJECT-ORIENTED KNOWLEDGE BASES

First, we describe a unified abstraction of different knowledge representation techniques and different models of expert knowledge in Expert System (ES) knowledge bases in general. Those techniques (of a unified abstraction) are used while designing the shell for design intelligent tutoring lessons. The abstraction is derived and realized by applying object-oriented approach. The motivation for defining such an abstraction was to provide:

- a unified description and representation of different elements in ES knowledge bases;
- more general knowledge access methods for use in interactions between an ES knowledge base and the other ES modules at runtime, as well as in interactions between different modules of integrated ES building tools (ESBTs) at development time.

As a result, an object-oriented model of knowledge bases is developed and is called OBOA (Object-Oriented Abstraction). It covers both design of knowledge bases and communication between the knowledge base and other ES and ESBT modules. However, the OBOA model sets only general guidelines for developing and using of object-oriented knowledge bases. It is open for fine-tuning and adaptation to particular applications. The ultimate practical goal of developing the OBOA model was to use it as the basis for building a class library of knowledge representation and knowledge access tools and techniques. At the time of writing, the implementation of the library is incomplete, but it is sufficient for exploring the ideas of the OBOA model.

The background for developing the OBOA model was several different ideas coming from the fields of knowledge and data modeling, representation and management. Gruber and Cohen [GC87] have described a hierarchy of knowledge representation techniques and tools, starting from most primitive ones to quite complex knowledge structures. Techniques for representing knowledge about external signals in real-time ESs, described in [Dev95], were developed as a class-like representation of properties of I/O data that real time ESs exchange with external processes they monitor and/or control. Finally, some ideas from the field of object-oriented databases were also adopted in the OBOA model.

On the other hand, we also investigate the possible application in design of intelligent tutors. A tutor can be defined as a provider of additional, specialized, or individualized instruction. We are concerned with the
construction of a general framework for intelligent tutoring design, i.e., the intelligent tutoring shell. This shell can be used to create and maintain a curriculum for an individual domain. In this paper we introduce knowledge primitives for an interactive environment in which the teacher can manipulate graphical objects at different levels of abstraction, based on the OBOA model.

Building of an intelligent tutoring system requires the ability to model and reason domain knowledge, human thinking and learning processes, and the teaching process. Acquiring and encoding this large amount of knowledge is difficult and time consuming. We have been searching for efficient ways to do these knowledge engineering tasks. This paper describes our efforts toward developing uniform data and control structures that can be used by wide circle of authors, e.g., domain experts, teachers, curriculum developers, etc. who are involved during building the system. One of our goals is to build tools that will enable experts to use the computer directly, without any help of knowledge engineering.

The field of AI in education is concerned with development of artificial intelligence techniques for the study of human teaching and for the engineering of systems that facilitate human learning. That field tries to answer to the questions that are a long term in nature: How can computer systems help learning and is it possible the measurement of learning progress? The term intelligent tutoring system is frequently used concerning the engineering side of the usage of artificial intelligence techniques in making the educational programs. In ITSs the computational methods are used in support of AI activities such as planning, control, knowledge representation and acquisition, explanation, cognitive modeling and dialogue management. Also, the computational models are used to explore and evaluate alternate theories about learning.

It is possible to think of an ES knowledge base as a logical entity composed of three related parts (components), Fig. 1.: the domain knowledge, control knowledge (i.e., a model of the ES), and the third part that we call explanatory knowledge (see the discussion below). Domain knowledge is represented by using one or more knowledge representation techniques. In the most general case, domain knowledge is a structured record of many interrelated knowledge elements. These elements describe relevant domain models and heuristic, and can vary a lot in their nature, complexity, and representation. They can be everything from simple data to instantiated knowledge primitives such as frames, rules, logical expressions, procedures, etc., and even more complex knowledge elements represented by using either simple aggregations of the knowledge primitives or conceptually different techniques based on the knowledge primitives and other (simpler) knowledge elements.

Knowledge elements of the domain knowledge are usually grouped to create meaningful collections. One possible criterion of defining such a collection (the one used in the OBOA model) is the type of knowledge element that will be included in the collection. Therefore we can speak of collections of rules, frames, etc., or collections of more complex knowledge elements. Each such collection is homogeneous regarding the type of knowledge elements it contains. Generally, domain knowledge can be composed of n collections of knowledge elements, and there can be zero or more collections of elements of a particular type.

---

1The word "collections" is used rather than sets, lists, tables, etc., in order to avoid suggesting a particular implementation technique at the analysis phase.
The class diagram in Fig. 2 illustrates the organization of the elements of the domain knowledge as homogeneous collections, K_elements. A K_elements object is a collection of objects of the K_element class. The K_element class describes a single knowledge element of the domain knowledge and is elaborated below. The double line from the D_knowledge class icon to the K_elements class icon, with the point close to the D_knowledge icon, means that the D_knowledge class uses the K_elements class. The 1 and n at the ends of the double line denote the cardinalities of the corresponding classes: the D_knowledge class can have only one instantiation for a particular knowledge base, and there can be n instantiations of the K_elements class. Similarly, for each instantiation of the K_elements class, there can be n instantiations of the K_element class. K_elements is a generic (parameterized) class, which can be instantiated to generate classes denoting collections of knowledge elements of a particular type. The instantiation relationship between the corresponding classes is denoted by a dashed line, with an arrow pointing to the generic class.

For the sake of simplicity and basing this description on commonly known concepts, it is assumed that some knowledge primitives belong to the second highest level in the organizational hierarchy of knowledge elements. Therefore the classes’ Frames, Rules, etc. are instantiations of the K_elements class, and denote homogeneous collections of frames, rules, etc., respectively. In the actual implementation, there can be several intermediate levels between the level of knowledge primitives and the highest (and conceptually the most abstract) level of knowledge elements, i.e., the level of K_elements and K_element classes.

K_element is an abstract base class with several subclasses inheriting its properties. In the Booch [Bo91] notation, the inheritance relationship among two classes is denoted by a solid line with an arrow pointing to the base class. Abstract base classes cannot have instantiations, only the objects of the Rule, Frame, Slot, If-clause, Signal, and other subclasses can exist in the knowledge base only. Again, we assume that the classes designating the knowledge primitives mentioned are derived directly from the K_element class, skipping more complex classes of knowledge elements. The OBOA model is opened for extensions by less common (or even new) knowledge element types. The K_element class defines common properties inherited by all types of knowledge elements and adding another knowledge element type requires only type-specific properties to be defined. Second, all the elements of a knowledge base are treated the same way, regardless of what their type is and how common (standard) or specific they are. This means that all the common operations on elements of the knowledge base (like updating, searching, deleting, etc.) use the interface of the K_element class, and element-specific operations are provided through polymorphism in the corresponding subclasses.

Another interesting design detail in OBOA is the fact that certain types (classes) of knowledge elements, denoting part of more complex elements, are at the same level of hierarchy as the corresponding aggregate elements. For example If-clauses can be treated as "stand alone" knowledge elements, and rules. Such a design allows a particular If-clause object to be shared by several Rule objects, which is more flexible and more efficient, and can also simplify the rule compilation process [Dev95]. The classes in OBOA are designed by filling-in a standard template for class properties [Boo91].

Most elements of the K_element class have obvious meanings - a K_element object lets other objects (communicating with it) use its public interface functions to set and get its name, get its size, update it, etc. It should be stressed that the OBOA model strictly applies encapsulation, as a general and extremely important principle of object-oriented design: most attributes (fields) of classes and objects used in this design cannot be accessed directly. Client objects must use the server object’s public interface functions. Public operations of the K_element class are only declared here, and are actually implemented in the derived classes using polymorphism. Thus it is possible to, for example, retrieve an arbitrary knowledge element from the knowledge base referring to the element as a K_element object and using the Get_K_El function. If the element is a rule, due to polymorphism, the code executed then will be the code of the Get_K_El function implemented in the Rule class. Each K_element object belongs to a collection of knowledge elements, which is specified by using the Collection_Ptr field, and uses public interface functions of the corresponding K_elements object.

To illustrate the design of a particular knowledge element type in the OBOA model, Fig. 3 shows the class diagram of
the Rule class. It is assumed that only AND operator is used to connect the clauses in both the If-part and the Then-part of a rule. The main parts of such a rule are collections of If-clauses and Then-clauses, represented by the If-classes and Then-clauses classes. An object of an If-clauses (Then-clauses) class is a collection of one or more If-clause (Then-clause) objects. Since in, many rule-based ESs rules are often divided in sets of rules or groups of rules to provide more efficient focusing of the inference process. Each Rule objects in the OBOA model can belong to one or more rule-collection objects, which are instanitiation of the Rules class.

3. OBOA REPRESENTATION OF EduSof

In this section the usage of the OBOA model in the knowledge representation of the Expert module of EduSof is briefly presented. That knowledge representation is concerning some particular and concrete type of knowledge elements that are necessary for the realization of the Expert module in any ITS shell. One of the most important elements of knowledge for designing the intelligent tutoring lesson is the model of Lesson, and this is basic class needed for modeling the learning process. Any lesson is consisted of one or more issues, modeled by the class Topic in the OBOA model. We assume that the student must learn these issues, during mastering that lesson. The basic attributes of the lesson are: the name (Name), current topic (CurrentTopic, the issue or the problem introduced, defined and/or explained in the present moment), the task that is currently solved, the level of prerequisites for the student (StudentLevel), and so on.

The class Lesson, used for describing that elementary knowledge is inherited from the more general class Frame, and in the OBOA model that class is defined with:

Name: Lesson
Visibility: Exported
Cardinality: n
Base classes: Frame
Derived classes: FirstLesson, LastLesson, HardLesson, EasyLesson, ...
Generic parameters: -: only for generic class
Public Interface
Operations: SetTopic, GetTopic, UpdateTopic, GetTopicCollection, ...
Implementation
Uses: Topic, Goal, ...
Fields: Name, InformationPtr, ImagePtr, LessonPtr, SimulationPtr, Explanation, Hints, QuestionPtr [ ], TaskPtr [ ], ...
Persistence: Static

In this section the usage of the OBOA model in the knowledge representation of the Expert module of EduSof is briefly presented. That knowledge representation is concerning some particular and concrete type of knowledge elements that are necessary for the realization of the Expert module in any ITS shell. One of the most important elements of knowledge for designing the intelligent tutoring lesson is the model of Lesson, and this is basic class needed for modeling the learning process. Any lesson is consisted of one or more issues, modeled by the class Topic in the OBOA model. We assume that the student must learn these issues, during mastering that lesson. The basic attributes of the lesson are: the name (Name), current topic (CurrentTopic, the issue or the problem introduced, defined and/or explained in the present moment), the task that is currently solved, the level of prerequisites for the student (StudentLevel), and so on.

The class Lesson, used for describing that elementary knowledge is inherited from the more general class Frame, and in the OBOA model that class is defined with:

Name: Lesson
Visibility: Exported
Cardinality: n
Base classes: Frame
Derived classes: FirstLesson, LastLesson, HardLesson, EasyLesson, ...
Generic parameters: -: only for generic class
Public Interface
Operations: SetTopic, GetTopic, UpdateTopic, GetTopicCollection, ...
Implementation
Uses: Topic, Goal, ...
Fields: Name, InformationPtr, ImagePtr, LessonPtr, SimulationPtr, Explanation, Hints, QuestionPtr [ ], TaskPtr [ ], ...
Persistence: Static

In this section the usage of the OBOA model in the knowledge representation of the Expert module of EduSof is briefly presented. That knowledge representation is concerning some particular and concrete type of knowledge elements that are necessary for the realization of the Expert module in any ITS shell. One of the most important elements of knowledge for designing the intelligent tutoring lesson is the model of Lesson, and this is basic class needed for modeling the learning process. Any lesson is consisted of one or more issues, modeled by the class Topic in the OBOA model. We assume that the student must learn these issues, during mastering that lesson. The basic attributes of the lesson are: the name (Name), current topic (CurrentTopic, the issue or the problem introduced, defined and/or explained in the present moment), the task that is currently solved, the level of prerequisites for the student (StudentLevel), and so on.

The class Lesson, used for describing that elementary knowledge is inherited from the more general class Frame, and in the OBOA model that class is defined with:

Name: Lesson
Visibility: Exported
Cardinality: n
Base classes: Frame
Derived classes: FirstLesson, LastLesson, HardLesson, EasyLesson, ...
Generic parameters: -: only for generic class
Public Interface
Operations: SetTopic, GetTopic, UpdateTopic, GetTopicCollection, ...
Implementation
Uses: Topic, Goal, ...
Fields: Name, InformationPtr, ImagePtr, LessonPtr, SimulationPtr, Explanation, Hints, QuestionPtr [ ], TaskPtr [ ], ...
Persistence: Static

In this section the usage of the OBOA model in the knowledge representation of the Expert module of EduSof is briefly presented. That knowledge representation is concerning some particular and concrete type of knowledge elements that are necessary for the realization of the Expert module in any ITS shell. One of the most important elements of knowledge for designing the intelligent tutoring lesson is the model of Lesson, and this is basic class needed for modeling the learning process. Any lesson is consisted of one or more issues, modeled by the class Topic in the OBOA model. We assume that the student must learn these issues, during mastering that lesson. The basic attributes of the lesson are: the name (Name), current topic (CurrentTopic, the issue or the problem introduced, defined and/or explained in the present moment), the task that is currently solved, the level of prerequisites for the student (StudentLevel), and so on.

The class Lesson, used for describing that elementary knowledge is inherited from the more general class Frame, and in the OBOA model that class is defined with:

Name: Lesson
Visibility: Exported
Cardinality: n
Base classes: Frame
Derived classes: FirstLesson, LastLesson, HardLesson, EasyLesson, ...
Generic parameters: -: only for generic class
Public Interface
Operations: SetTopic, GetTopic, UpdateTopic, GetTopicCollection, ...
Implementation
Uses: Topic, Goal, ...
Fields: Name, InformationPtr, ImagePtr, LessonPtr, SimulationPtr, Explanation, Hints, QuestionPtr [ ], TaskPtr [ ], ...
Persistence: Static

In this section the usage of the OBOA model in the knowledge representation of the Expert module of EduSof is briefly presented. That knowledge representation is concerning some particular and concrete type of knowledge elements that are necessary for the realization of the Expert module in any ITS shell. One of the most important elements of knowledge for designing the intelligent tutoring lesson is the model of Lesson, and this is basic class needed for modeling the learning process. Any lesson is consisted of one or more issues, modeled by the class Topic in the OBOA model. We assume that the student must learn these issues, during mastering that lesson. The basic attributes of the lesson are: the name (Name), current topic (CurrentTopic, the issue or the problem introduced, defined and/or explained in the present moment), the task that is currently solved, the level of prerequisites for the student (StudentLevel), and so on.

The class Lesson, used for describing that elementary knowledge is inherited from the more general class Frame, and in the OBOA model that class is defined with:

Name: Lesson
Visibility: Exported
Cardinality: n
Base classes: Frame
Derived classes: FirstLesson, LastLesson, HardLesson, EasyLesson, ...
Generic parameters: -: only for generic class
Public Interface
Operations: SetTopic, GetTopic, UpdateTopic, GetTopicCollection, ...
Implementation
Uses: Topic, Goal, ...
Fields: Name, InformationPtr, ImagePtr, LessonPtr, SimulationPtr, Explanation, Hints, QuestionPtr [ ], TaskPtr [ ], ...
Persistence: Static

In this section the usage of the OBOA model in the knowledge representation of the Expert module of EduSof is briefly presented. That knowledge representation is concerning some particular and concrete type of knowledge elements that are necessary for the realization of the Expert module in any ITS shell. One of the most important elements of knowledge for designing the intelligent tutoring lesson is the model of Lesson, and this is basic class needed for modeling the learning process. Any lesson is consisted of one or more issues, modeled by the class Topic in the OBOA model. We assume that the student must learn these issues, during mastering that lesson. The basic attributes of the lesson are: the name (Name), current topic (CurrentTopic, the issue or the problem introduced, defined and/or explained in the present moment), the task that is currently solved, the level of prerequisites for the student (StudentLevel), and so on.

The class Lesson, used for describing that elementary knowledge is inherited from the more general class Frame, and in the OBOA model that class is defined with:

Name: Lesson
Visibility: Exported
Cardinality: n
Base classes: Frame
Derived classes: FirstLesson, LastLesson, HardLesson, EasyLesson, ...
Generic parameters: -: only for generic class
Public Interface
Operations: SetTopic, GetTopic, UpdateTopic, GetTopicCollection, ...
Implementation
Uses: Topic, Goal, ...
Fields: Name, InformationPtr, ImagePtr, LessonPtr, SimulationPtr, Explanation, Hints, QuestionPtr [ ], TaskPtr [ ], ...
Persistence: Static

In this section the usage of the OBOA model in the knowledge representation of the Expert module of EduSof is briefly presented. That knowledge representation is concerning some particular and concrete type of knowledge elements that are necessary for the realization of the Expert module in any ITS shell. One of the most important elements of knowledge for designing the intelligent tutoring lesson is the model of Lesson, and this is basic class needed for modeling the learning process. Any lesson is consisted of one or more issues, modeled by the class Topic in the OBOA model. We assume that the student must learn these issues, during mastering that lesson. The basic attributes of the lesson are: the name (Name), current topic (CurrentTopic, the issue or the problem introduced, defined and/or explained in the present moment), the task that is currently solved, the level of prerequisites for the student (StudentLevel), and so on.

The class Lesson, used for describing that elementary knowledge is inherited from the more general class Frame, and in the OBOA model that class is defined with:

Name: Lesson
Visibility: Exported
Cardinality: n
Base classes: Frame
Derived classes: FirstLesson, LastLesson, HardLesson, EasyLesson, ...
Generic parameters: -: only for generic class
Public Interface
Operations: SetTopic, GetTopic, UpdateTopic, GetTopicCollection, ...
Implementation
Uses: Topic, Goal, ...
Fields: Name, InformationPtr, ImagePtr, LessonPtr, SimulationPtr, Explanation, Hints, QuestionPtr [ ], TaskPtr [ ], ...
Persistence: Static

In this section the usage of the OBOA model in the knowledge representation of the Expert module of EduSof is briefly presented. That knowledge representation is concerning some particular and concrete type of knowledge elements that are necessary for the realization of the Expert module in any ITS shell. One of the most important elements of knowledge for designing the intelligent tutoring lesson is the model of Lesson, and this is basic class needed for modeling the learning process. Any lesson is consisted of one or more issues, modeled by the class Topic in the OBOA model. We assume that the student must learn these issues, during mastering that lesson. The basic attributes of the lesson are: the name (Name), current topic (CurrentTopic, the issue or the problem introduced, defined and/or explained in the present moment), the task that is currently solved, the level of prerequisites for the student (StudentLevel), and so on.

The class Lesson, used for describing that elementary knowledge is inherited from the more general class Frame, and in the OBOA model that class is defined with:

Name: Lesson
Visibility: Exported
Cardinality: n
Base classes: Frame
Derived classes: FirstLesson, LastLesson, HardLesson, EasyLesson, ...
Generic parameters: -: only for generic class
Public Interface
Operations: SetTopic, GetTopic, UpdateTopic, GetTopicCollection, ...
Implementation
Uses: Topic, Goal, ...
Fields: Name, InformationPtr, ImagePtr, LessonPtr, SimulationPtr, Explanation, Hints, QuestionPtr [ ], TaskPtr [ ], ...
Persistence: Static
Method GiveFeedback;
InferFrom Rules, Frame, Student, Psychology
Begin
  AcceptAnswer;
  ElaborateResponse;
  Find(Psychology.Type, CurrentStudent.Level);
  (* find the model of student and he/her psychology type *)
  If Congratulate = “ON” Then
      GiveCongratulate(Psychology.Type, CurrentStudent.Level);
  If Elaborate = “ON” Then
      GiveElaborate(Psychology.Type, CurrentStudent.Level);
  If Explain = “ON” Then
      GiveExplain(Psychology.Type, CurrentStudent.Level);
  If MoreInfo = “ON” Then
      GiveMoreInfo(Psychology.Type, CurrentStudent.Level);
  ....
  (* according to current student, he/her psychology type, parameterizing semantic net of frames and inherit from frame and rule class *)
End.

On the similar way the other classes and methods [Jer94, JD95] are defined in the OBOA model, which are needed in the creation of intelligent tutoring lessons.

6. DISCUSSION AND CONCLUSIONS

The purpose of the proposed OBOA model of knowledge bases and knowledge base management is to make a basis for applying the ideas of object-oriented software design methodology to ES knowledge organization, representation, and access. It covers all important aspects of knowledge bases, like their contents, knowledge representation techniques, using, updating, extending and maintaining the knowledge, etc. However, it is important to stress again that the OBOA model should be regarded as an open framework for developing ES knowledge bases, rather than as a closed set of design rules and organizational hierarchies. The presented method for building a knowledge base suitable for the educational purpose is currently under development and the realization in the present form. In the current development state we test usage of EduSof in some summer schools in various subjects (chemistry, history, geography etc.). With introduced approach in building the intelligent tutors we realize the one of the goals we set down before the start of the realizing of the above concepts. The teacher's independents in design of the intelligent tutoring lessons were almost 90% after an hour of explanation how to use the EduSof shell. It is supposed that they have some knowledge of using the computers, a little knowledge in computer graphics, and some help of professional programmers' in making the simulation's procedures.

We will also try to realize the EduSof concept in the object-oriented approach in design the knowledge base. Also, the above principles will be incorporated in the other parts of intelligent tutoring systems, i.e., in the Student, Tutor, Diagnostic and Interface modules.

REFERENCES


Prof. Takushi Tanaka
Program Co-Chair, IEA/AIE-96
Fukuoka Institute of Technology
3-30-1, Wajiro-Higashi, Higashi-ku
Fukuoka 811-02, Japan

Dear Prof. Tanaka,

Please find enclosed the final version of my paper PAPER NUMBER IEA-96 - 152, which has been accepted for the IEA/AIE-96.

Due to an extreme inefficiency of our mailing service, I hope that this letter will come to you on time. I would also like to kindly ask you to acknowledge this letter upon receiving it.

Sorry if I am boring you, but as the wire transfer from any Yugoslav bank to Japan is still impossible, and my credit cards are still frozen (even that UN sanction against my country is over), could I pay registration fees when I arrived to Fukuoka?

Thank you for your consideration, and looking forward to meeting you at the conference.

Best regards,

Ljubomir Jerinic