# An Object-Oriented Tutoring System for Teaching Sets

H. Altay Güvenir
Department of Computer Engineering and Information Science
Bilkent University
Ankara 06533, TURKEY
guvenir@olympus.cs.bilkent.edu.tr

**Abstract**

Over the recent years several prototypes of intelligent tutoring systems for scientific subjects have been developed. Meanwhile, the object-oriented paradigm has become popular in the software engineering and artificial intelligence communities. The objective of the research presented in this paper is an application of the object-oriented paradigm to the design and implementation of an intelligent tutoring system. The domain of the system is the set theory at the secondary school level. It is shown that the inheritance hierarchy of the object-oriented paradigm is very useful in defining and organizing the components of the set theory, and in the generation of examples and questions. The issues raised in the object-oriented design of an intelligent tutoring system are discussed.

## 1. Introduction

An Intelligent Tutoring System (ITS) is usually defined as a computer program that employs Artificial Intelligence (AI) techniques in an interactive environment to help a person learn [8]. One reason that they are called "intelligent" is because they have the ability to construct examples and questions according to the needs of the student, also to solve the problems that they have asked students [1]. Besides, an ITS provides individualized instruction by means of understanding individual student's goals and beliefs. Another major characteristic of an ITS is its ability to decide what to do next and how to do it by analyzing the solution history.

In order to design a proper intelligent tutoring system, its building blocks must be thoroughly analyzed. Generally there are three basic components within an ITS [4, 7]; namely, domain module, student model, and tutoring module.

The domain (or problem solving) module is a control center that encompasses the entire domain knowledge, generates instructional content and evaluates student's performance. The student model is used to determine the student's current knowledge status, his conceptions and reasoning strategies. On the other hand, the tutorial module contains a series of specifications about what instructional material is to be presented including the presentation method and timing. This module usually employs one of two approaches for presenting the learning material: Socratic method or Coaching method [5, 11]. Recently, these methods have largely been abandoned in favor of complex environments that promote interaction and feedback. These integrated modules are tied up towards the main objective of improving student's understanding by formulating ideas, discovering relationships, drawing conclusions and realizing misconceptions.

The *object-oriented paradigm* has emerged in recent years as a new software engineering discipline [10]. Versatility and flexibility constitute proven virtues of this approach, that has been successfully applied to a wide spectrum of applications and programming environments. As a programming language paradigm, it is more structured than previous attempts at structured programming. It is also more modular and abstract than previous attempts at data abstraction and detail hiding [6]. The

challenge of the object-oriented programming is that it requires one to set aside habits and ways of thinking which have been considered standard for many years. In this paper, we are going to present the methodology that we used in the object-oriented design and implementation of an ITS.

The objective of the research presented in this paper is an application of the object-oriented programming approach to the design of an intelligent tutoring system called the OOST (Object-Oriented Set Tutor). The OOST is an ITS designed for teaching set theory. The modular and hierarchical structure of the concepts in the set theory was the motivation behind applying the object-oriented approach in the design and implementation of an ITS in this domain.

The rest of the paper is organized as follows. The next section will overview the main characteristics of the object-oriented paradigm. The third section will describe the organization of the OOST system. Finally, our conclusions about the application of the object-oriented programming to the implementation of an ITS, especially in the domain of teaching set theory, will be presented.

## 2. Object-Oriented Programming

Object-oriented programming (OOP) is a technique which has many benefits such as easy construction of complex systems, availability of inherently reusable objects, reduction in total life-cycle software cost and a much more modular structure. Actually the OOP is a self-explaining term since object, which means an entity, is the base of the OOP. All objects exist in an object universe. Execution of an object-oriented program is based on the communication between objects. Objects communicate by sending messages to each other. In order to do anything with an object's data, one must send a message to that object. A message is a request for an object to perform one of its operations. An object is not allowed to access directly the data of another object. In order to process data in an object one must define a method, which is a definition of an object's behavior [4].

The three most important attributes of the OOP are encapsulation, inheritance and polymorphism. Encapsulation is collecting the code and data together into objects. Inheritance is a mechanism that the OOP technique possesses by which data types inherit characteristics from simpler and more general types. In the OOP, polymorphism implies that a single action can be shared within an object hierarchy with the capability of each object to implement the action in a way appropriate to itself [2].

The characteristics of the OOP discussed above allow the programs to be more structured, modular, and extensible while being able to handle high levels of complexity. In the implementation of the OOST, encapsulation helped us to organize the code of the program in a very structured manner. Due to the inheritance feature of the object-oriented approach, the domain knowledge of the OOST was organized in a very natural way, since all components of the set theory can be defined by an inheritance hierarchy. This organization is used also by the tutorial module and the student model. The tutorial module simply sends messages to the appropriate objects encoding the corresponding topics and subtopics of the set theory. The tutorial module can also send messages to the domain module to construct appropriate examples and questions, or to evaluate the response of the student for a given question. The student model maintains the state of the student's current knowledge in the same hierarchy. The details of the organization of the OOST are given in the following section.

## 3. Organization of OOST

The OOST is an intelligent tutoring system that aims at teaching set theory at secondary school level. The system is implemented in Turbo Pascal 7 which is an object-oriented extension of the Pascal programming language. The OOST runs on DOS and MS-DOS operating systems on all models of IBM PC compatibles and IBM PS/2 systems.

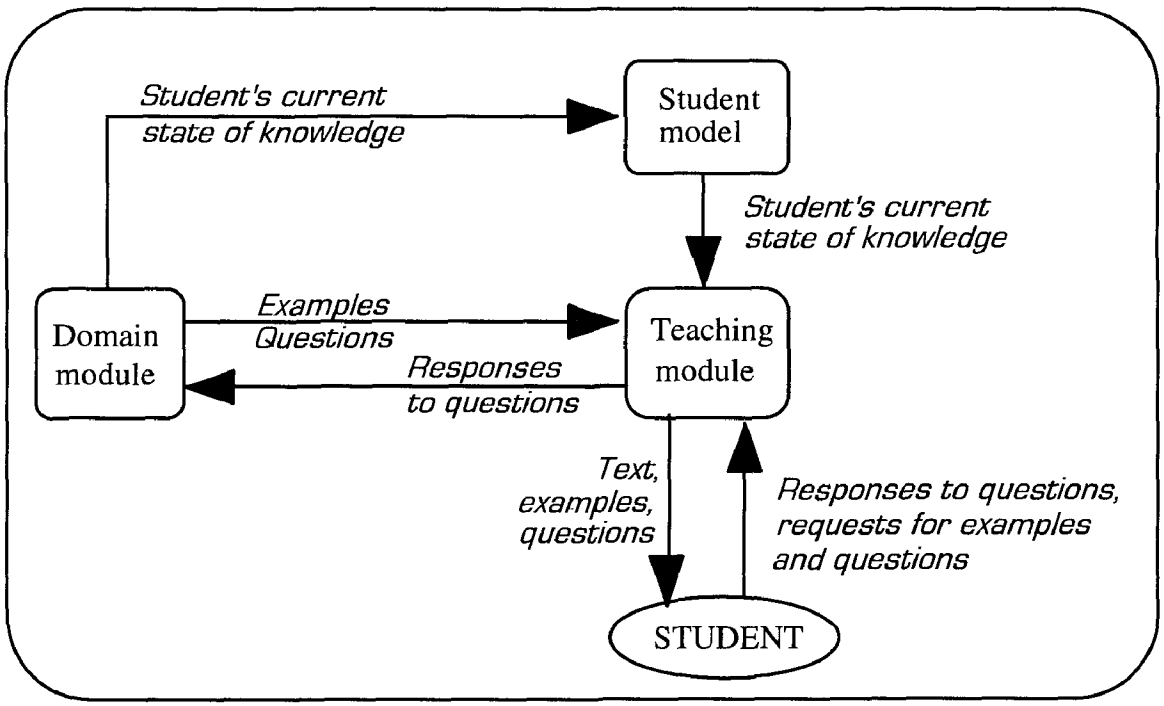


Figure 1. The organization of OOST.

Similar to most other ITS's, the OOST comprises three main modules: domain module, teaching module, student model. The organization of the OOST is illustrated in Figure 1. The implementation of these components in the object-oriented paradigm will be explained in detail in the following sections.

### 3.1. Domain Module

The domain module of the OOST encompasses the entire domain knowledge, namely set theory. It can generate instructional content and evaluate the student's responses. Although the definitions of concepts are static, some of the instructional content, such as examples and questions, are generated dynamically so as to satisfy the requirements of the tutorial module. The need to generate problems with different features arises because an ITS is supposed to be able to distinguish among a large number of diagnoses. Problem pre-storing for all possible misconceptions is not feasible even for elementary domains [3, 9]. Nondeterminism incorporated in the domain module guarantees that different but meaningful examples and reasonable questions will be presented to the student at every invocation.

In the OOST the domain knowledge is organized and stored in the form of an object hierarchy. The core of the set theory is the concept of set which is also the starting point for introducing the domain to the computer. For a computer program teaching sets to be intelligent, first of all, it must know what a set is and how it can be generated. Therefore, an object called "setobj" is defined. Each instance of the setobj class has a universe and a value. Also associated with setobj class, several methods are defined to initialize the universe, to find the cardinality of the set value, to delete/add an element to the value, and to make a copy of itself. The setobj represents the concept of a basic set. In order to generate a set which satisfies the requirements determined by the tutorial module, many additional parameters are also needed. For example, the tutorial module may require that the cardinality of the set be in a given range, want to fix the universe, or require that some elements be included/excluded. Therefore, a new object "ex_set" is defined as a subclass of setobj. The definition of the ex-set is given in Figure 2. The "make" method of the ex_set subclass is responsible for intelligently and nondeterministically generating a set

```
ex_set = object (setobj)
     min, max: byte {range of cardinality}
     incl, excl: setobj; {elements to be included and excluded}
constructor default; {sets default values}
procedure initMin(newMin: Byte); {sets minimum cardinality}
procedure initMax(newMax: Byte); {sets maximum cardinality}
procedure initIncl(newIncl: setobj); {sets the set to be included}
procedure initExcl(newExcl: setobj); {sets the set to be excluded}
procedure copy(origset: ex_set); {copies from origset}
procedure make; {makes a set satisfying the requirements}
function equal(otherset: ex_set): boolean {equal test}
end; {ex_set}
```

Figure 2. The implementation of ex_set.

which satisfies the requirements determined by the tutorial module and set by the messages initMin, initMax, initIncl and initExcl.

Two of the main tasks of the domain module are to generate meaningful examples and reasonable questions intelligently yet complying with the constraints determined by the tutorial module. Two class hierarchies are defined to represent the relationships between the subtopics of the set theory, one for examples and the other for questions. The root of the example tree is the object "example", which defines standard parameters and features of all examples independent of the subtopic. The tree representing the class hierarchy for examples is shown in Figure 3. Ex_sets object is to generate basic examples of sets, while ex_relation and ex_operation are to construct examples about set relations and set operations, respectively.

Having established the fundamentals for generating examples, the generation of questions is done in a very similar way. The questions can be obtained by modifying some parts of the examples. Therefore, the OOST uses example objects in order to generate questions. Again, the relationships between the subtopics of the set theory are organized in a class hierarchy. Here the classes contain information required for generating questions. The tree representing the class hierarchy for questions is illustrated in Figure 4.

The questions are generated in the form of multiple-choice type. In order to generate questions on the membership relation, the choices are kept as elements where in remaining topics the choices are sets. Therefore, three subclasses are defined upon "question" subclass; "B_question", "S_question" and "E_question". The B_question object generates basic questions about sets, such as whether a given sequence is a set. The S_question object can construct questions whose choices are sets, while the E_question object forms questions whose possible choices are elements.

Experience has shown that students have a tendency to confuse the operations union ($\cup$), intersection ($\cap$) and set difference (\) [12]. Furthermore, A \ B and B \ A are also often confused. Therefore, this information is used to generate the possible choices of questions.

### 3.2. Tutorial Module

The teaching module is the control center of OOST which decides what to do next and how to do it. Thus, the teaching strategy is embedded in this module. The control strategy of an ITS is a very important design issue which is mainly made up of four effective factors. These factors are: (1) the system: the hard-coded strategy determined by the designer and implementer of the system, (2) the teacher: the strategy determined by the teacher for a particular class or session, (3) the student: the strategy that the individual student adopts for his/her needs, and (4) the student model: the presumed current state of student's knowledge about the domain. In certain systems the control belongs entirely to either the system, teacher or the student. The Logo program is a good example of a student controlled

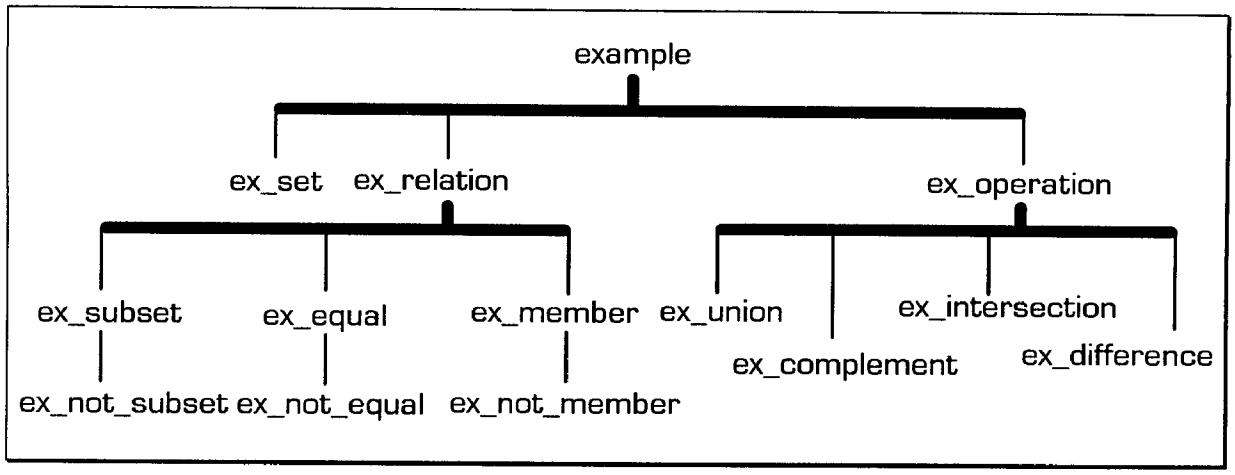


Figure 3. Class hierarchy for generating examples.

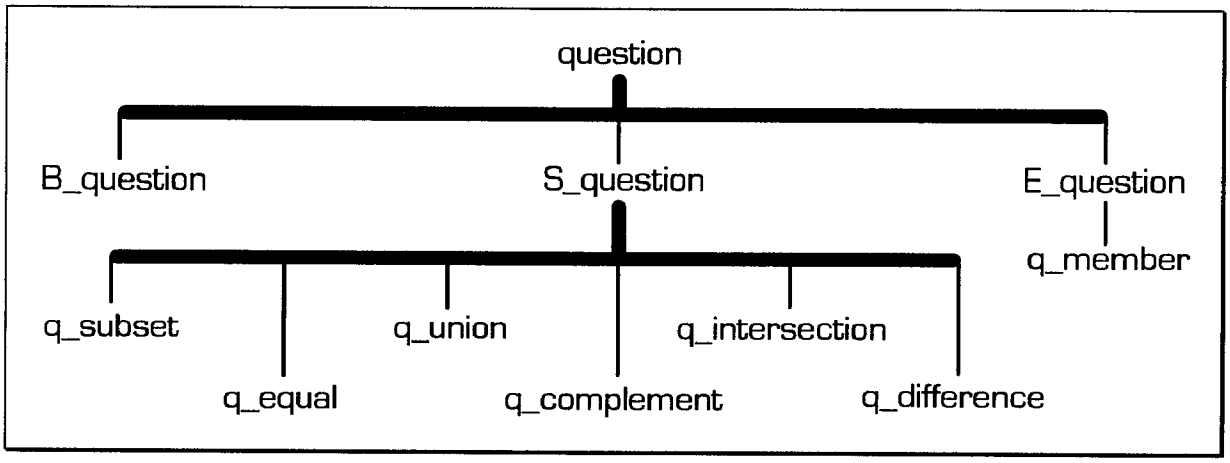


Figure 4. Class hierarchy for generating questions.

system. The advantage of such systems is that the user is "free" to explore the subjects on his own and thus learns well. However, in many cases the user can get stuck or lost and spend a lot of time losing his motivation. The student can affect the control of the system unintentionally through the student model. A program under the total control of student model may get stuck at a particular topic if student cannot score a passing grade on that topic.

The programs under total system control are packet programs that cannot be modified by teacher. These programs behave more like electronic books. The programs under total teacher control are basically instructional programs that are completely written by actual teachers. Although the teacher can design the program as he likes, the teaching strategy is built into the program and cannot be modified for different classes and sessions. Also dealing with the issues of user interface and programming details is not what teachers would like to concentrate on.

The best control strategy is, obviously, a combination of all four factors, inheriting their advantages. The sharing of the control of power in these systems will form an environment that is both open to self exploration by the student and can provide guidance and assistance when needed with structured learning.

```
Topic = object
   TextFileName: fNameStr;
   NumberOfExamples: integer;
   numberOfQuestions: integer;
   studentModel: integer;
   satisfactory: boolean;
constructor initialize (dataFile: fNameStr; n, m:
integer);
procedure Teach; virtual;
procedure ShowText; virtual;
procedure GiveExamples; virtual;
procedure AskQuestions; virtual;
procedure UpdateStudentModel;
procedure CheckSatisfactory;
end; {topic}
```

Figure 5. Implementation of "topic" object.

In the ITS described here all four of the factors share the control of the system. Also a fifth factor, nondeterminism, is included. Nondeterminism is required to generate different examples and questions at each invocation. The priorities of those factors in the OOST system are as follows: system, student, teacher, nondeterminism, in decreasing order. The system depending on the current state of the execution may permit the student to ask for an example or ask a question. If the student does not interrupt, the next step determined by the teacher is executed. During the execution of the next step, nondeterminism is used as appropriate.

The OOST system is designed to assist the teacher and decrease his work load. The systems strategy (hard-coded) is defined as follows:

1. If the student has interrupted to initiate a dialogue then respond to him/her otherwise pass the control to the tutorial strategy.
2. To teach a topic (e.g., intersection of two sets) follow the following steps:
   2.1. Show related text
   2.2. Give examples
   2.3. Ask questions, evaluate student's performance
3. After teaching a topic, consult the tutorial strategy to decide about the next topic.

The relations of a topic are represented as a class hierarchy in the object-oriented paradigm as well, where each subclass corresponds to a subtopic of set theory. Each subclass has the same class parameters but different methods for each topic. The methods in the subclasses refer to the example and question objects defined in the domain module. The definition of the topic object is given in Figure 5.

When the student has completed all the topics in the teaching module, he is given a final examination. The final examination covers only the operations on sets. However, the questions in the final exam are more complex, where the student is to compute and enter the value of an expression which contains two or three operations; they are not multiple-choice questions.

The strategy given above comprises the portion of the teaching strategy that is determined by the system. The rest of it is determined by the teacher. In the OOST the teacher determined parameters include the list and elements of universes, the text to be presented and the number of examples and questions for each topic, minimum success level (threshold) to pass a topic. In generating examples and questions, the OOST system will use sets in increasing difficulty levels. A set with more than 2 elements can be considered to be easier to understand than the empty set. However, the range of possible cardinalities for generic sets is determined by the teacher, e.g minimum 3 and maximum 7 elements.

### 3.3. Student Model

The function of the student model is to determine the student's current state of knowledge, his/her misconceptions and reasoning strategies. The OOST system uses this module for determining the current knowledge state of the student. The student's misconceptions are determined in the final examination phase.

The program treats each topic independently. For a tutorial session to be concluded the question section must be successfully completed. Therefore, a threshold level must be exceeded, at each node of the set theory tree, for the entire tutorial to be successful. If the level of the student is not sufficient then the last tutorial about a node is repeated until the threshold is passed.

Another important role that the student model has is in the final examination given at the end of the session. It is known that the most confusing concepts within set theory are the operations and most novice learners seem to confuse these operations. Therefore, after teaching all operations in set theory the system conducts a final test by asking mixed question on all operations. This combined final examination is different from the previous tests that contained only a single operation. If during this final stage an operation is determined to be confused then the tutorial about that topic is repeated.

In order to model the confusion of the student the system maintains a count representing the knowledge level of the student on the four fundamental operations of the set theory. As the questions are asked one by one, the system determines the operations used within that question and decreases the relative counts by one if the answer is wrong. If the wrong answer provides any hints about the misconceptions of the student, such as confusion between two operations like intersection and difference, then the confused operations' count are decreased by one each. For example, let A = {cat, lion, bee, tiger}, B = {lion, hawk, bee}, and C = {eagle, horse}. The student is asked to determine the value of ((A $\cap$ B) $\cup$ C). The correct value is {lion, bee, eagle, horse}. Let us suppose the student entered the set {cat, tiger, eagle, horse}. In this case, the program assumes that the student has confused the operations $\cap$ and \, since the set entered by the student is actually the value of the expression ((A \ B) $\cup$ C). The scores of $\cap$ and \ are decremented by one. However, if the program cannot determine any possible misconception, then the operations in the question are assumed to be misunderstood.

This tracking procedure is repeated for each question and at the end of the final exam the topics that require re-teaching are selected starting from the most negative count up to the last element below threshold level. Because this approach is just a heuristic criterion [12], it allows the system to determine the misconceptions of the student up to a certain extent.

## 4. Conclusions

The OOST system forms a good example of applying suitable programming techniques to subjects that possess a special internal structure. Set theory, due to its hierarchical and well-organized structure, is easily applicable to an ITS system especially using the OOP approach. Therefore, the contents of this work provides a well-established system in the field of intelligent tutoring that employ the OOP. Although the OOP has been used for various software engineering applications, it is a new approach in the field of ITS.

The main objectives of this implementation were: (1) to design a system that is able to generate the instructional content like questions and examples on its own in a nondeterministic manner, (2) to distribute the control of the tutorial session to the student, the teacher, the student model and even to add some degree of nondeterminism, (3) to design a modular system that could be easily understood, enhanced, extended and modified, (4) to demonstrate the applicability of the OOP to certain topics within the academic curricula that exhibit hierarchical structure and inheritance properties.

The objectives listed above have been met by the implemented system although there are certain possible future extensions and weak points of the system. However, all of these handicaps can be overcome with the advantages of the OOP such as modular structure and easy extension. Future enhancements that could be performed upon this study are: (1) improvement within the expert module by incorporating the bug catalogue, (2) improving the student model simultaneously with the expert module, (3) improving the capabilities of the user interface by including features like pull-down menus, graphical representations and attractive screen designs.

The attributes of set theory discussed in the introduction section such as well-defined rules and operations, suitability to easy question and example generation and the fact that set concept forms a basis of the entire theory, proved to assist the object-oriented design and implementation phases of the ITS.

## Acknowledgments

I thank Emel Cankat who worked in the implementation of the programs, and David Davenport for his valuable comments on the manuscript.

## References

[1] Clancey W.J., Intelligent tutoring systems: a tutorial survey. In Current Issues in Expert Systems (Edited by A. van Lamsweerde and P. Dufour), Academic Press, London (1987).
[2] Cox Brad J., Object-Oriented programming: an Evolutionary Approach, Addison-Wesley, (1986).
[3] Duchastel P., ICAI systems: issues in computer tutoring, Computers Educ. 13, 95-100 (1989).
[4] Fischetti E. and Gisolfi A., From Computer-Aided Instruction to Intelligent Tutoring Systems, Educational Technology 7-17 (1990).
[5] Kearsley G., Artificial Intelligence and Instruction, Addison-Wesley Publishing Company, Massachusetts (1987).
[6] LeClaire B., Object-oriented programming: an overview of key O-O concepts, OR/MS TODAY 20-24 (1991).
[7] Nwana H. S., Intelligent Tutoring Systems: An Overview, Artificial Intelligence Review, 4, 251-277 (1990).
[8] O'Shea T. and Self J., Learning and teaching with computers: artificial intelligence in education, Harvester Press, Sussex (1983).
[9] Shevchenko I. I., and Nakayama K., Subtraction problem generating: two approaches, Computers Educ. 17, 243-248 (1991).
[10] Stefik M., and Bobrow D.G., Object-Oriented Programming: Themes and Variations, AI Magazine, 6, 40-62 (1986).
[11] Wenger E., Artificial Intelligence and Tutoring Systems, Morgan Kaufmann Publishers, Inc., (1987).
[12] Personal communication with Peter Fraser, the author of the book "Mathematics - Book 1, T.E.D. Ankara College, Secondary School Publications".