Abstract
The important characteristics of any intelligent systems are the possibilities of explanation. So, any software product which intend to be intelligent must provide some kind of explanation, i.e., explanation of some conclusions, explanation of new knowledge (theorem), etc. As Intelligent Tutoring Systems (ITSs) intend to be intelligent software, the explanation feature must be provided in ITSs. In this paper, we will briefly survey how we realized the explanation properties and features in Intelligent Tutoring System (ITS) shell called EduSof. The OBOA (Object-Oriented Abstraction) model for representing the knowledge, interaction within that knowledge and actions on that knowledge is used for the model of explanation, the transitions and the interactions features in EduSof shell.

1 Introduction
The activity of learning with computers has been studied and modeled by Artificial Intelligence (AI) researchers for almost a decade. The educational and instructional researchers tried also to build their own approach in using the computers in education process. Their current methodology for building computer-based tutoring systems requires that designers assemble these systems from basic component modules, in the best case, and completely from scratch in the worst case. This is true of most commercially available authoring tools (i.e., Director, Authorware, HyperCard, SuperCard, etc.) and even in many of the more advanced systems developed in university research labs. On the other hand, rapid development of computer technologies, introduction of computers into schools, widespread use of computers by people of different vocation, education and age, has made education to become a field of great importance to AI researchers. The intensive development of the methods of knowledge engineering and reasoning with computer enables that education by computers. The main goal of that approach is the complete individualization of the educational process.

Intelligent Tutoring Systems (ITS) are very sophisticated software systems based on cognitive science and AI technology. As research in ITS [4] continues to produce more refined systems, the gap between the research community and educational community continues to widen. The educators’ understanding and usage of these research results have been much slower than the research progress. As this theory-application gap widens, it becomes more difficult for educators to participate in ITS research and application, and as result research becomes increasingly academic and unconnected to the pragmatic aspects of teaching and learning.

In the sense of the above trends and research issues, the goal of designing the EduSof [6] system is to construct a conceptual framework for representing the objects, events, responses, reactions, and relationships involved in tutoring. Also, the aim is to build a highly usable knowledge acquisition interface for rapid prototyping and easy creation, modification, deletion, and testing of both teaching concepts and tutorial strategies. Finally, our intention is to produce a system that enables teachers to build their own lesson without any computer or AI specialist. In this paper we will at first, discuss some open problems in automation of the learning process, and than we will describe the explanation techniques used in the realization if the explanation features in EduSof system.

2 Explanation architectures in AI systems
As we said, the ITS can be defined as the intelligent, special-purpose systems with the feedback and the inference mechanisms. ITS are complex software products and represent the connection of several disciplines: education, AI, computer science, cognitive science, psychology, pedagogy, etc. We extended the common definition of the ITS model (Expert, Student, Tutor, Diagnostic and Interface module), with Explain module, to allow the explanation features in learning.

The problem of computer planning of explanations first arose when people wrote application programs containing data structures to be communicated to a user and faced the problem of selecting and organizing this material for expression in the linear medium of text. The typical solution was to write procedures that selectively followed links in the data structures, such that the path so traced provided both the content and structure of the explanation. The classic example of this approach is the query facility of MYCIN [2], an expert system for diagnosing bacterial infections. MYCIN explained why it was asking a question or
how it reached a conclusion by translating a record of its reasoning steps into English. Given the central role of pre-existing data structures in these applications, it was natural that procedures for traversing these data structures was the mechanism of choice for generating explanations. The utility of procedural traversal mechanisms derive from their suitability for selectively exploiting existing data structures. However, when used as the sole device for explanation planning, traversal mechanisms force both the content and organization of explanations to be coupled too closely to the data structures being traversed. The data structures might be missing information a student needs, such as definitions of terms and justifications of statements made. The organization of the data structures might not be the appropriate way to organize an explanation for a given student. In general, any explanation mechanism that is based solely on exploitation of content representations can't utilize communicative strategies other than those derivable from the content itself, restricting the range of explanations that can be modeled.

A second class of planning methods were developed based on analyses of naturally occurring explanations. These methods match abstract descriptions of the structure of explanations to a collection of propositions, sometimes called the "relevant knowledge pool," in order to organize this material. The most influential schematic generation work was that of McKeown [3]. Her TEXT program answered questions about a database, by describing what information was available in the data base, providing definitions, and making comparisons between database entities. McKeown analyzed a variety of texts by dividing them into units playing distinct functional roles, and identifying recurring patterns of these roles. The patterns were implemented on the computer as a type of recursive finite state automata called "schemata". A top-level schema was associated with each question type, and text was generated by traversing the schema. When an arc in the schema was traversed, content that filled the corresponding role was selected from the relevant knowledge pool and expressed in text. The strengths of schematic methods include the clarity with which they describe the structure of explanations, their ability to impose organizations found to be effective in naturally occurring explanations, and efficiency of execution. However, schemata tend to generate inflexible explanations, and are less appropriate for exploiting the existing structure of a relevant knowledge pool than procedural traversal techniques. Schematic mechanisms generally leave choice of the relevant knowledge pool to other mechanisms.

A third line of research views explanation as a problem of finding a sequence of actions that achieves some goal. The appropriateness of a top-down goal refinement mechanism was demonstrated by Appelt's KAMP [1]. Such a mechanism applies a library of plan operators that specify ways in which a goal can be refined into subgoals or achieved by executable actions. Planning is complete when all conjunctive branches of the plan have been expanded to actions. Appelt's planner began with a goal to get a person to do something (e.g., disassemble a pump), and applied plan operators that reasoned about intentions and beliefs in refining this goal until actual utterance acts were planned. Like schematic approaches, top-down planning is a way of imposing structure. They differ in that top-down planning records the hierarchical relationships between goals and the acts that satisfy them, selects content as well as structuring it, and does both at a finer granularity than schemata.

Recall our conclusion that schemata are insuitable for augmenting an explanation with background that a student might need to understand the primary explanation. Top-down planning operators are often abstract in that they don't specify exactly what content they will select, only what type of content. The retrieved content may contain unanticipated concepts that need to be explained as background. Hence, a data-driven mechanism that responds to the appearance of such concepts is more appropriate for modeling the background selection task.

The important characteristics of any intelligent systems are the possibilities of explanation. Several features of the possibility are of interest from the standpoint of explanation planning in ITS. First, the question did not specify exactly what kind of information should be provided in response. Second, more than a literal response is provided. The description of a car's parts, for example, does not by itself answer the question, instead it enables the student to understand the primary response. Third, the explanation uses one of a multitude of possible models of the structure and operation of a car. For example, the question could have been answered with a simple analogy ("like an ...") or a mathematical description of the problem. Fourth, the statements made in the explanation are presented in a coherent order. For example, the process description is preceded by the structural background needed to understand that description, and events are described in the order that they occur. These features suggest that different subtasks are involved in explanation planning, including selection of an informatively satisfying response, addition of other pedagogically motivated material, selection between alternate models that could form the basis for an explanation, and ordering of selected material in a coherent manner. Explanation in EduSof is based on a hybrid architecture based on the object-oriented data-driven method of explanation, as well as on the work of D. Suthes and his PEG[5].

**3 Model of knowledge in EduSof**

The domain, i.e., the knowledge that student must learn, is essentially a tree, where each node is a piece of domain knowledge and any parent-child link in the tree implies that in order for the student to be able to understand the knowledge of the parent node, he must also understand the knowledge of the child node. The representation of the knowledge domain is object-oriented. The main object classes are the Lessons and the way of their presentation. Lessons are represented by units of knowledge that can be taught, repeated, summarized, etc. They are categorized according to knowledge type, for example: text, picture, simulation, more examples, and so on. The concepts have pointers, including various types of prerequisite, part-of and related-misconception links to other concepts, forming the lesson network. They have pedagogical information such as summary, motivation, examples, tasks, etc., that point to presentation. The presentation or the array of questions or tasks, represent expository or inquisitor interactions with the student. They are composed of a task, such as a multiple-choice question, or problem solving exercise, and an environment for doing the task, such as a picture or simulation of some system. Presentation also contains the possibilities for responding to the pupil, such as a hint, congratulation, elaboration of the answer, etc. 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. Any lesson is consisted of one or more issues, modeled by the class Topic. The basic attributes of the lesson are: the name (Name), current topic (CurrentTopic), the issue or the problem introduced, defined and/or explained at the present moment), the task that is currently
solved, the level of prerequisites for the student (StudentLevel), and so on.

The issues that student must learn (class Topic) are realized with separate type of knowledge elements in the EduSoF system. Any topic could be introduced with text information (Information), graphically (the pictures and/or the diagram - Image), and/or with the simulation of some event (Simulation). Also, the additional or further explanation (Explanation) for that theme, or some suggestions (Hints) could be defined. The class Topic in EduSoF is made for specifying and defining the issues or notions needed for lesson creation.

Abstract class TQ served for description of the common elements for the two comparable and connected classes, one for the definition of tasks or assignments (class Task) and the other class for the realization of questions or problems (class Question). That class has the fields WhyPtr, HowPtr, WhatPtr, etc. With that fields the lesson creator made their explanations for the task and/or question. The instances of that class are given to the student during the learning process. The definition of the class TQ is:

- Name: TQ
- Base Classes: Frame
- Derived classes: Task, Question, ...
- Generic parameters: Answer, SolutionType, ...
- Interface Operations: SetInfo, GetInfo, UpdateInfo, ...
- Implementation

The type of knowledge - explanations generated by the system or required from the user is realized by object-oriented model of data-driven mechanisms. EduSoF differs between several kinds of explanations (those presented to end-users - EndUserExplanation, those presented to ITS developers - DeveloperExplanation, those required from students when checking their knowledge - StudentExplanation, those concerned with explaining the system's functioning - SystemExplanation, those explaining various concepts or topics - ConceptExplanation and TopicExplanation, etc.). In generating explanations, EduSoF can use knowledge from various kinds of elements (rules, frames, knowledge chunks, etc.). Also, in EduSoF we differs various types of explanation: WhyExplanation, HowExplanation and WhatExplanation for realization of Why, How and What student's requests. The corresponding Explanation class is designed as follows:

- Name: Explanation
- Visibility: Exported
- Cardinality: n;
- Base class: Frame
- Derived classes: WhatExpl, TQExpl, TopicExpl, WhyExpl, HowExpl ...
- Interface
- Operations: SetExp, GetExp, UpdateExp, DeleteExp ...
- Implementation
- Uses: Rule, Frame, K_chunk, Goal, Topic ...
- Fields: CanText, TopicColPtr [], RuleColPtr [] ...
- Persistency: Static/Dynamic
- The corresponding demos and lunches for the WhyExpl, HowExpl, TQExpl, TopicExpl, WhatExpl are realized to be fired when students or pupils ask the system to explain what, why or how, for some part of knowledge, or to explain the appropriate knowledge primitives TaskQuestion, Topic, Concept, etc. During lesson creation the teacher could provide different type of explanation, according to different knowledge primitives. It means we distinguished explanation for TaskQuestion knowledge primitives, and between them the What, How and/or Why explanations. The same explanations are provided for other knowledge primitives: concepts, lessons, topics etc. The type of explanation What, Why and How, and their methods DemonExpl or LunchExplain fired that explanation on students/pupils requested made by clicking Why, What or How button on the screen during learning some part of the lesson. Method DemonExpl fired their parameter before that topic is learned, and Lunch Explain fired their parameter after that topic is learned.

5 Summary

The presented method for building a knowledge base with explanation facilitates, suitable for the educational purpose is currently under development and the realization in the present form. In sum, explanation part of EduSoF is capable of playing the role of a "research resource" that helps a student find information he/she needs, chooses between alternatives when they are available, adds related information that motivates, enriches, and enables the student's understanding of the primary material selected, and orders all of this for a coherent presentation in a multimedia environment. It is no accident that there is a direct correspondence between the components of that hybrid architecture and its potential contributions as an educational resource, for the hybrid architecture was designed to clearly separate these different types of planning knowledge and match them to appropriate mechanisms. In testing of the present approach we concluded following:

- the teachers show the great interest in design of all kinds of described explanations and they use them very frequently, while designing the lessons, using EduSoF.
- the users, in almost every moment when they don't understand something in concept or task, use the corresponding demos and lunches to explore their explanation.
- these explanations helped the students to resolve some further tasks or questions based on them, solve their problems in particular questions or tasks in 20% more cases.

References