THE ANNALS OF "DUNAREA DE JOS" UNIVERSITY OF GALATI
FASCICLE III, 2000 ISSN 1221-454X
ELECTROTECHNICS, ELECTRONICS, AUTOMATIC CONTROL, INFORMATICS

# INTEGRATING HYPERMEDIA OBJECTS IN AN INTELLIGENT TUTORING SYSTEM

**Emilia PECHEANU, Diana STEFANESCU, Sabin Corneliu BURAGA†**
**Adrian ISTRATE**

*Department of Computer Science and Engineering*
*University "Dunarea de Jos" of Galati–Romania*
*e-mail: Emilia.Pecheanu@ugal.ro, Diana.Stefanescu@ugal.ro*
*†Faculty of Computer Science*
*University "A.I.Cuza" of Iasi-Romania*
*e-mail: busaco@infoiasi.ro*

Abstract: The paper describes the internal architecture of an Intelligent Tutoring System, CS-Tutor. The architectural design of the tutorial system was developed in a collaborative work at the Department of Computer Science of the University of Galati and the Department of Applied Informatics of the Faculty of Computer Science of Iasi. Intelligent Tutoring Systems (ITS) are software packages which use the Artificial Intelligence techniques to aid in learning of some subject or skill. In recent years, Hypermedia has been gained the interest of many researchers working in the teaching field of study. The CS-Tutor internal architecture is based upon integrating Hypermedia Objects in an Intelligent Knowledge-Based frame.

Keywords: Pedagogical Model, Hypermedia Object, Pedagogical Unit.

## 1. INTRODUCTION

The authors of this paper have been jointly working on a project of designing the architecture of an instructional system, CS-Tutor, using both intelligent tutoring methods and hypermedia technologies. The migration of hypermedia toward intelligent hypermedia obviously meets the specific requirements of the learning process, such as student modeling, student diagnosis and learner guidance in his investigation paths.

The goal of the project developed at Department of Computer Science was to design an instructional environment that will take advantage of integrating hypermedia objects into an intelligent knowledge-based system architecture.

The CS-Tutor system can be used to teach the curricula of the Computer Science specialization (Bumbaru, 1996), in order to provide on-line extensive learning assistance for the engineering graduate and postgraduate students.
The tutorial is conceived to be a colection of related lessons (or pedagogical modules). The lessons will consist of theoretical components including different simulations (animated images, Java applets) and testing exercises. Lessons and testing exercises will be presented to the user as a sequence of Web pages.

The Web pages forming lessons can include text, examples, images/animated images, diagrams and links to related information. The testing components will perform the assessment of students' knowledge level and can turn into a tool for improving the quality of the instructional process (Razmerita, 1997).

While designing the tutorial architecture the authors used a knowledge based aproach for building the kernel components of the system: the Domain Model, the Student Model and the Pedagogical Model.

This paper was recomended for publication by S. Bumbaru

The knowledge based approach includes the following design principles :

- Represent instructional content and instructional strategies separately.
- Explicitly represent abstract pedagogical entities.
- Design at the pedagogical level, as opposed to the media level, when possible.
- Modularize the instructional content for multiple use and re-use.
- Create generic teaching strategies that can be used with different instructional content.

Designing tutoring systems in this way can grant many advantages over the traditional CAI systems design paradigm (Murray, 1988).

## 2. THE INTERNAL ARCHITECTURE OF THE TUTORING SYSTEM

An Intelligent Tutoring System (ITS) is composed of four major components: the Domain Module, the Student Module, the Pedagogical Module, and the Interface Module (Figure 1).

The Domain Module is the main core of the instructional system. It contains representations of the knowledge which is to be communicated to the student, including descriptions of the various concepts and skills of an expert in a particular domain. The Domain Module provides the source of knowledge to be presented, as well as explanations of concepts or responses to students' tasks.

The Domain Module serves also as a standard for student evaluation. The Domain Module can be thought as a component manipulating a data structure, the Expert Model, or as a dynamic vision of the domain knowledge.

The Student Module contains information about the student's understanding of the domain knowledge.
The Student Module dynamically builds a model of how students learn and use the diagnostic tools contained within the Pedagogical Module. This model is permanently up to date in order to extract the learner's knowledge state related to the subject taught. Through inference capability, the Student Module is able to produce an interpretation of the student's actions and to reconstruct the knowledge that led to these actions.

The Student Module can determine, from explicit representations, the incorrect interpretations of the target knowledge, so that remedial actions can be taken. Inferential (excluding student input) and interactive (student and ITS dialog) diagnostic approaches can be used to provide diagnosis of student learning efforts. The Student Module may be thought of as manipulating a data structure called the Student Model, a dynamic vision of the student's knowledge of the domain. The student model is "runnable" or "executable", so that predictions about a particular student in a particular context can be made.

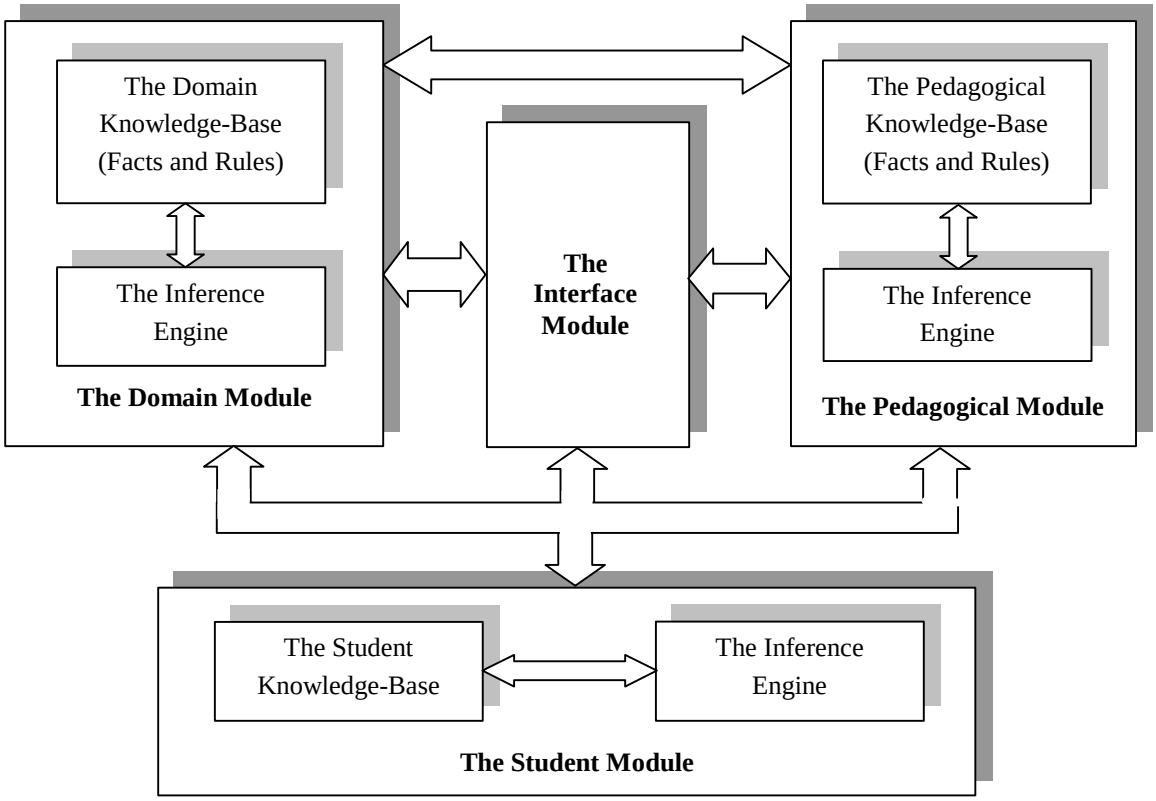


Fig. 1. The Architecture of an Intelligent Tutoring System

The Pedagogical Module contains rules or other decision making tools that allow it to judge how well the student's understanding of the subject domain (as represented by the Student Model) matches actual knowledge structure (as represented by the Expert Model). It may then generate correct forms of instruction or remediation to be given to the Interface Module.

The Interface Module presents the user with a uniform environment within which instruction, diagnosis, remediation, and user driven learning may take place.

The classical architecture of an Intelligent Tutoring System (shown in Figure 1) was considered in CS-Tutor in a different approach, in order to enable the integration of the multimedia and hypermedia elements. As a result, an object-oriented formalism was adopted in building the Interface Module of CS-Tutor system.

As a first level of decomposition, the tutorial system architecture was split in three logical levels (see Figure 2): Presentation Level (i.e., Level 1), Hypermedia Level (i.e., Level 2) and Intelligent Tutoring Level (i.e., Level 3) (Stefanescu, 1996).

The Presentation Level is devoted to describe the content of multimedia interactive objects as they will be shown to the learner on his workstation, and how each object will interact with the learner.

The Hypermedia Level permits sequencing and navigation through these interactions, taking into account the learner's input.
The Intelligent Tutoring Level uses Artificial Intelligence techniques to meet the general requirements of intelligent tutoring systems.

Levels 1 and 2 are a collection of autonomous and active objects with related services; each object is an instance of a related class. All classes and meta-classes are predefined and independent from a specific learning application.

For Level 3, three main components are highlighted: the Domain Module, the Student Module and the Pedagogical Module. Each component, as a collection of rules and facts conforming to the declarative approach, can do specific services without any ad hoc functions.

During the courseware execution, levels components interact between them by means of message sending, for services invocation. A Level component (i.e., object of the two bottom levels or a rule of Level 3) has to send a specific message, to an object for a services activation. Then the receiver replies with a report witch carries the relevant results of the service activated. Messages are generally propagated from higher levels to lower levels and reports in the reverse way.

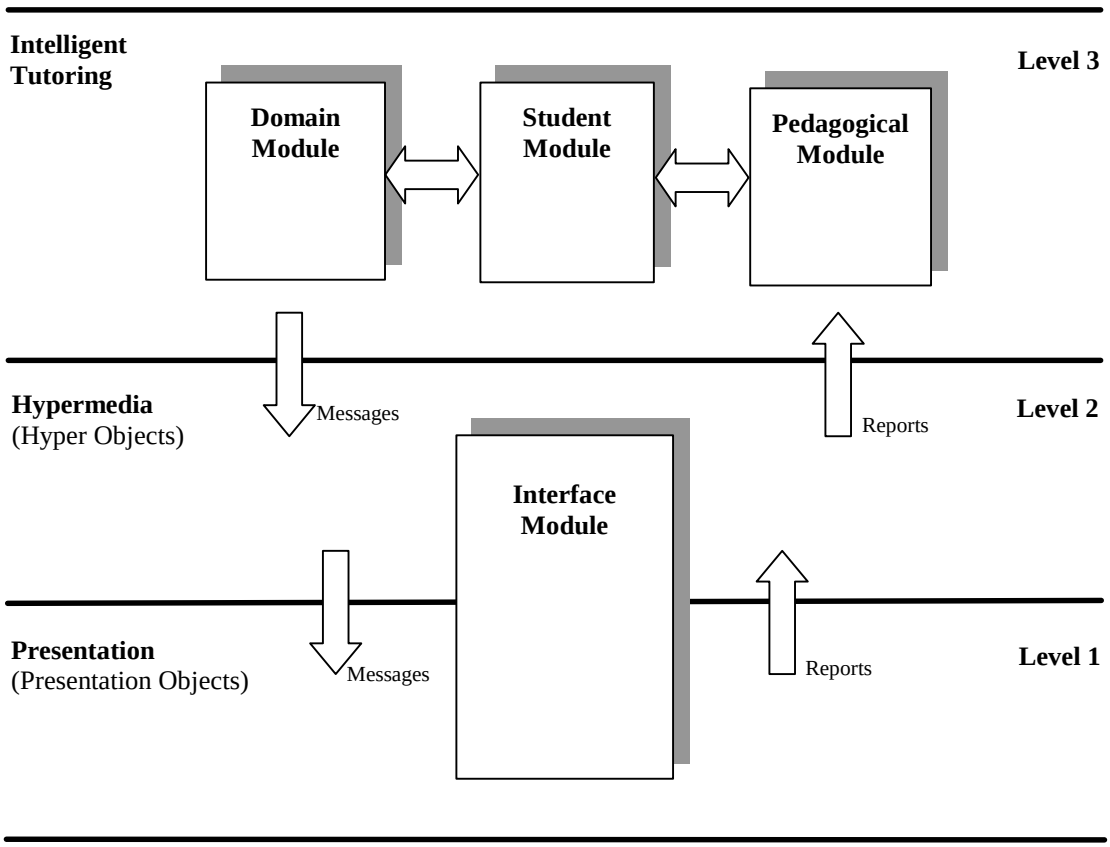


Fig. 2. Levels of the Tutoring System

## 3. THE COMPONENTS OF LEVEL 1 AND 2 OF THE TUTORIAL SYSTEM

Level 1 of the CS-Tutor system is composed by *Presentation Objects*. A Presentation Object is an elementary information to be used through the learner-computer interface.
The Presentation Level is composed by several Presentation Objects, that can be used or re-used during the learner-computer interactions.
The services provided by a Presentation Object can be described as follows: in reply to an invocation from higher levels, the Presentation Object restitutes the presentation contents back to the calling Hyper Object for a further analysis.

Level 2 of the is tutoring system consists of a collection of *Hyper Objects*. The purpose of an Hyper Object is mainly to decide what will be the next learner-computer interaction, taking into account the current results of the interaction and the pedagogical goal. Like in hypermedia systems, Hyper Objects are structured as nodes of a network (see Figure 3). Four kinds of Hyper Objects can be distinguished in Level 2:

- Learning Unit: it is an Hyper Object of an output kind, giving information about what is to be taught. It may be an example, an hint or an explanation about a domain concept.

- Solicitation Unit: it is an Hyper Object of an input kind, devoted to assess the learner's comprehension about what is already taught; it may be a simple question, an exercise, or a problem to solve. Author's expected responses, wich are objects too in our architecture, are attached to a Solicitation Unit. A Solicitation Unit refers to several Presentation Objects from Level 1.

- Pedagogical Module: it is an Hyper Object consisting of a collection of relevant and organized Learning Units and/or Solicitation Units; a Pedagogical Module is an object specialized in teaching a specific domain concept.

- Link Object, wich connect two Hyper Objects; a Link Object is activated when the attached conditions are true; Link Objects in Level 2 are considered as static links in the tutoring system

Hyper Objects are entities wich are involved when results of input data come back from the Presentation Objects. Hyper Objects can provide two main services:

- Analysis of input data, made by the current Solicitation Unit, in comparison with expected response objects. These latter may be either stored beforehand by the author or calculated dynamically by Level 3.
- Decision making, based upon user behavior:
  - Local decision: what will be the next Learning Unit or Solicitation Unit following the current one in the current Pedagogical Module. This kind of decision is made by the current Solicitation Unit as looking for the Link Object which activation conditions are true.

  - Global decision: when a Pedagogical Module is finished, the next Pedagogical Module is chosen. Which one is chosen depends upon the learner's behavior to all solicitations in the current Pedagogical Module. This decision is made by the current Pedagogical Module as looking for the Link Object which activation conditions are verified.

A teaching domain concept is either factual (i.e., factual knowledge) or procedural (i.e., exercise, problem to solve). The section of courseware which is centered around a domain concept is the Pedagogical Module.
A Pedagogical Module is represented by an Hyper Object in level 2 entailing both statistics about the learner's behavior toward the module and a method to make global decision. A Pedagogical Module is also represented by some knowledge in Level 3.

From the learner's point of view, a courseware may be seen as a sequence of Pedagogical Modules. A Pedagogical Module is seen as a sequence of Pedagogical Units and Solicitation Units.
A Pedagogical Unit is a learner-computer interaction, which may be either an explanation (i.e., Learning-page), an example shown to a learner, and a Solicitation Unit can be an exercise or a problem to solve the related concept. From the system's point of view, a Pedagogical Unit is seen as a vertical section of the architecture.

One can observe that Level 2 (Hypermedia) and Level 3 (Intelligent Tutoring) of the system have some common services, such as the analysis of input data and making local or global decision, but implemented in different ways. An Hyper Object provides these services only if the expected learner's response is received and the static link is foreseen by the author. In this case, the current Hyper Object has the ability to make decision without involving the Intelligent Tutoring Level, which is only informed of the student's response, in order to update the Student Model. Otherwise, Level 3 (i.e., the Intelligent Tutoring Level) is the only means to deal with dynamic and unexpected responses (i.e., dynamic link). Thus, when the Hyper Object discovers, after analysis, that the response is unexpected, it will forward the learner response to the Intelligent Tutoring Level for diagnosis and decision making.

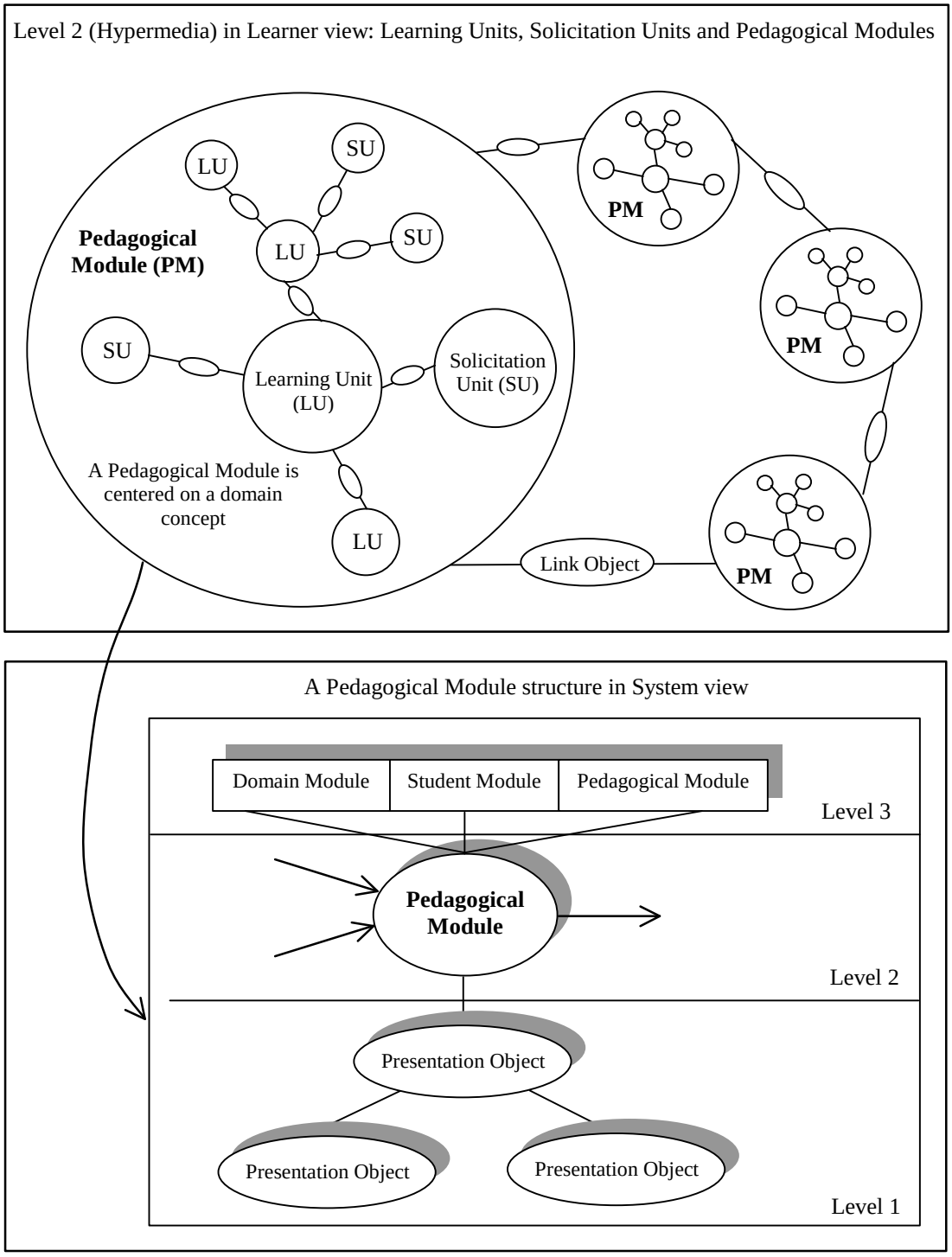


Fig. 3. The Learner and System views of the courseware

## 4. LEVEL 3 OF THE TUTORIAL SYSTEM

Level 3 of the CS-Tutor was designed in a special architecture, significantly different from the classical architecture of an Intelligent Tutoring System (ITS). The three components of an ITS: Domain Module, Student Module and Pedagogical Module were reunited in an unique intelligent layer of the instructional system.

This layer becomes Level 3 of the Cs-Tutor instructional system.

Level 3 of the of CS-Tutor system was then divided in two major parts: the *Local Knowledge-Base* (LKB), describing the behavior of the learning during a transaction (answers, timing, learner's choices) and the *Global Knowledg-Base* (GKB), defining the transmission of information, that is to say, the author's model.

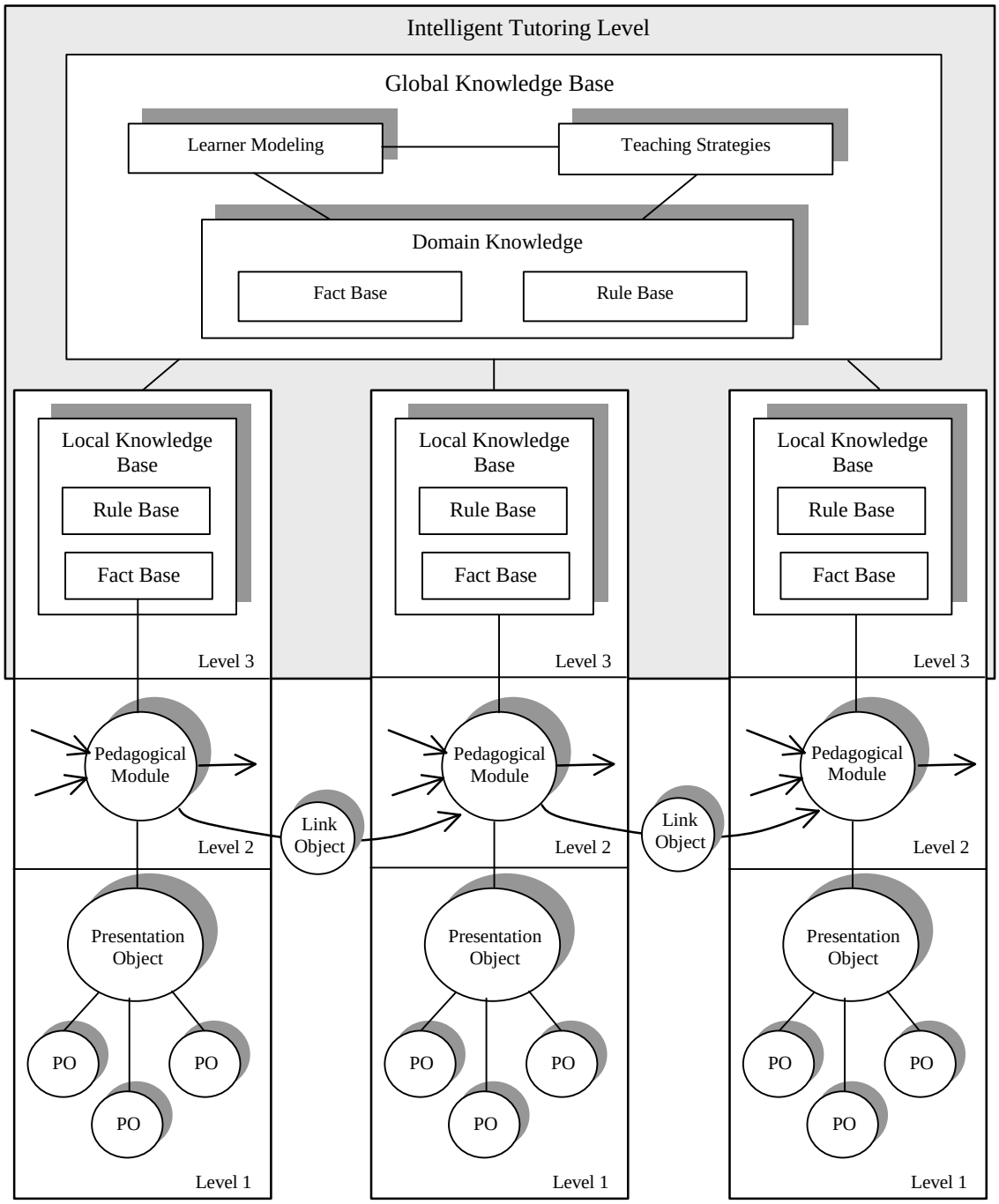


Fig. 4. The Local and Global Knowledge Base of the tutoring system

Several Local-Knowledge-Bases were associated with a Pedagogical Module, each of them defining the behavior for the instructional system, accordling with the targeted pedagogical goal (see Figure 4).

Local Knowledge-Base (LKB) should include the knowledge which is specific to a Pedagogical Module: domain facts pre-stored by the author or set dynamically by the domain expertise when an exercise is made up. The facts are used to select rules from the Global Domain Knowledge-Base and to activate them in order to solve the specific exercise or problem tackled in that Pedagogical Module. LKB may include results of interactions such as the student's response, delay of time of response, type of response. These latter facts are instantiated when the response comes back from the Hyper Object. LKB may include specific rules too, represented as a temporary working-memory.

When the Pedagogical Module is deactivated, some of its knowledge (e.g., student's response) are included in the Global Knowledge-Base (GKB), in the sense to be exploited by the following Pedagogical Module for diagnosing or making decision. Global Knowledge-Base (GKB) includes

rules and facts, about the domain knowledge, the student model and the didactic strategies, witch are shared by all pedagogical units. GKB contains:

- Predefined learning paths which define both synchronized stages in the presentation of knowledge and control of these stage which adapts the system according to the student's behavior;
- Dynamic paths, that provide a remedy for learning problems which cannot be resolved by predefined learning paths.

## 5. THE NEED FOR A DOMAIN CONCEPTUAL STRUCTURE

One of the problems with Hypermedia for educational applications is that the learner needs to have a good *conceptual map* of the domain being taught, in order to effectively use the system.

The conceptual map should implement the hierarchy of concepts of the domain knowledge. The domain knowledge concepts can be represented in a semantic net architecture, consisting of conceptual nodes and relations between nodes (see figure 5).

In our framework the creator of the hypermedia courseware needs to carry out the following steps:

- Create a concept hierarchy for the domain being taught. This would involve analysis of the domain to determine these concepts.
- Develop hypermedia material corresponding to that concept. The hypermedia architecture uses typed links such as those found in semantic networks. Types include, is-a (ISA), which represents class definition; a-kind-of (AKO), which represents membership of and inheritance from superclasses; has-a (HA), relating an object to an attribute or property and part-of which is used to show how an object is composed from smaller parts. A hypermedia system built using such link types provides a simple knowledge base about the domain and such a knowledge base may be used to reason new links automatically by utilising automatic reasoning algorithms.

Implementing various domains in the proposed system would require some effort from the domain author in that it is necessary to think about the domain carefully in order to produce the correct class nodes and correct links types.

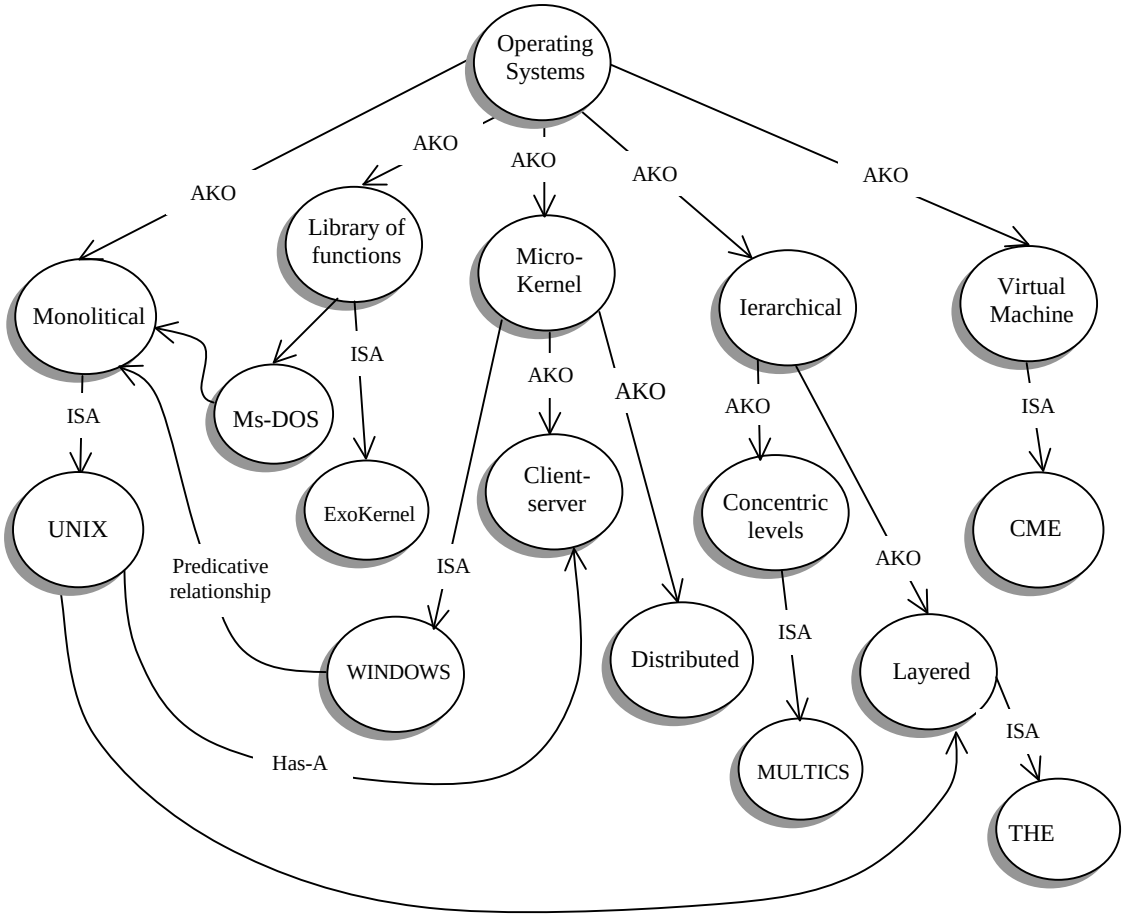


Fig. 5. A semantic net of concepts for the Operating Systems discipline.

## 6. CONCLUSIONS

The framework for the instructional system that we designed will join the Artificial Intelligence

technologies with the object oriented capabilities in storing and managing structured data. This can lead to multiple advantages in exploiting, maintaining and updating the computer-aided instructional system, offering a higher degree of generality in implementing and using the CS-Tutor system.

On the other hand, the knowledge-based aproach, that we used for designing the top level of the system, can provide more dynamic learner-computer interactions, and can adapt the sytem behavior to the needs of each individual learner.

## REFERENCES

Bumbaru, S.; Stinga, O.; Pecheanu, E.; Dumitriu, L. and Tudorie, C. (1996). A CBTS built upon Oracle RDBMS, *Proceedings of The 3-rd East-West Congress on Engineering Education*, Gdynia, Poland

Murray, T. (1988), Authoring Knowledge Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design, *Journal of the Learning Sciences*, Prentice Hall, New York.

ElHani O. and Gouarderes G. (1992). Standardized Architecture for Integrated Open Courseware, *Lecture Notes in Computer Science*, 602, pp. 198-211

Razmerita, L.; Stefanescu, D.; Bumbaru, S. and Istrate, A. (1997), Student testing and assessment in Computer Based Training Systems, *Proceedings of The 4-th International Conference Computer Aided Engineering Education*, Krakow, Poland, pg. 223-231

Stefanescu, D., Bumbaru, S. and Ariton, V. (1996), Hybrid Systems for Assisted Learning, Ptocedings of *The 9-th Symposium on Modelling, Simulation and Identification Systems, SIMSIS '96*, pg. 372-381