# DESIGN AND ANALYSIS TECHNIQUES FOR INTELLIGENT LEARNING SYSTEMS

**Ana Lilia Laureano-Cruces[*], Fernando de Arriaga[**] and Javier Ramírez-Rodríguez[***]**

[*]Universidad Autónoma Metropolitana-Azcapotzaclo Departamento de Sistemas
[**] Universidad Politécnica de Madrid
[**]Universidad Autónoma Metropolitana-Azcapotzaclo Departamento de Sistemas

[*] clc@correo.azc.uam.mx, [**] farriaga@mat.upm.es, [***] jararo@correo.azc.uam.mx

***Abstract:*** *This paper reviews several important techniques applied to the design of Intelligent Tutoring Systems using artificial intelligence. We analyze the different techniques related to the four components of a traditional architecture: 1) domain or expert module, 2) student model, 3) tutorial module and 4) user interface. We also explain the various types of control used in practice, including a control for reactive teaching systems. Aiding the acquisition of cognitive abilities is an important issue within the teaching-learning process. The student should not only control his own learning process, but also the way he/she does it. In other words, it is important the student regulates how and when to use a cognitive strategy as part of a learning domain, and this drive us to intelligent learning systems. The analysis is complemented with the description of some examples of intelligent tutoring systems and intelligent learning systems.*

***Key words:*** *interactive leraning environments, intelligent tutoring systems, teaching/learning strategies, evaluation methodologies.*

## 1. Introduction

In this paper we revise some of the existing Intelligent tutoring systems (ITS), describing their behavior on the basis of the design techniques used in their four components.
We will start by describing the type of knowledge that can be transmitted by an ITS: Procedural (related to skills), declarative (related to facts or concepts) and qualitative (related to the mental skill of simulating and reasoning based on dynamic processes).
The objective of an ITS is having the ability to adapt itself during the tutorial process. This objective is achieved by the interrelation of the four basic components of the system: 1) domain or expert module, 2) student model, 3) tutorial module and 4) user interface.
Systems of this type need to use Artificial Intelligence (AI) for: a) representing the knowledge of the dominion the system has; b) controling the tutorial process and the methods used to apply these principles.

The different design techniques that the components of an ITS. can use to achieve the aforementioned objective will be described in what follows.

## 2. The domain or expert module

This component includes the specific and detailed human knowledge obtained from human experts that have spent years studying the cognitive task (CT) that is intended to teach. Research in this field is focused on how to codify that knowledge and how to represent the expertise. There are different techniques used to approach the problem, namely: a) black box, b) glass box, and c) cognitive models. The three have advantages and disadvantages when used to build an ITS.

### 2.1. Black box expert.

In this type of representation, knowledge is compiled, *i. e.*, the answer to a question is obtained based on inputs and outputs, giving a measure of the correctness of the result, but we have no access to the details of why a given decision was made.

A classic example of this type of tutor is SOPHIE (Brown, Burton and Bell, 1975) that uses a general purpose electronic simulator. The idea was to teach students how to find failures in an electronic circuit. The tutor used its simulator to determine if the measurements the student took on several points of the circuit were correct. The simulator worked based on a set of equations, not like a human would, and therefore it was not possible for the system to give a detailed explanation of its decisions.

This kind of representation can be used in reactive learning environments, showing the student whether or not the operation executed is correct and, possibly, which is the next adequate move. This combination of black box and reactive tutor is an interesting approach to be used when building an ITS based on expert systems (not limited to black boxes), for it saves time in the implementation stage.

In the case of SOPHIE the tutor performed well with a reactive environment, although newer versions were made (Brown, Burton and de Kleer, 1982), where a causal model of circuits was incorporated, oriented toward allowing the detailed explanation of decisions.

However, an ITS should not be limited to say whether the decision is correct or incorrect, but it should also know which points are critical, or when they should be explained in more detail. Other techniques exist, like the glass box expert, which takes into account this other aspect of training systems.

### 2.2. Glass box expert

The basic methodology to build this type of expert system involves a knowledge engineer and a domain expert; the later identifies the area and the range of the problem. In this stage, key concepts of the domain are enumerated and formalized, devising a system to implement the knowledge and finally prove it and refine it iteratively.

These systems are characterized by containing a great deal of expert knowledge in an articulate way. Their construction is divided in two basic stages: the first one deals with the acquisition of knowledge (time invested in building the expert component), and the second tries to automate that knowledge.

Due to the fact that the resulting expert system contains less compiled knowledge, teaching is more feasible than with a black box expert. In this kind of expert system, the expertise of the human expert is represented in an articulate form. The former has in itself the advantage of being able to reach at deeper detailed levels during the teaching process that is the system motivation.

One of the main conclusions of the expert system GUIDON (Wenger, 1987) was that you should not tend only to the representation of knowledge in the expert system, but also to the form in which this knowledge is presented. For a tutor to act in an appropriate way, the expert module should display that knowledge in the same form that humans do, considering its restrictions.

The work done by Clancy, of paramount importance, showed that ITSs are limited if they only use expert systems for their tutorial process. This last reflection has led to the use of cognitive models.

### 2.3. Cognitive models

The objective of a cognitive model is to develop an effective simulation of the problem solution in a given domain, from the human point of view. In this technique knowledge is divided into components that bear a direct relation to the form in which humans classify and use them.

The merit of this approach is providing an expert module whose taxonomy allows a better tutorial process and a deeper communication with the student. Although there have been important advances in cognitive sciences for the last ten years, these models require a large amount of development time, due to the great number of details that are to be included.

Three basic questions have to be considered: which components from the cognitive analysis are important for the tutorial process; at what level the components should be represented and last, how the different types of knowledge, be it procedural, declarative or qualitative should be dealt with in this modeling technique.

MAKATSINÁ[1] (Laureano and de Arriaga, 1999; Laureano and de Arriaga, 2000) is an ITS in which a cognitive model is used, and whose design and analysis were made following the research of Castañeda (1993), Redding (1992) and Ryder and Redding (1993). The result of this analysis is to know the number of abilities that are integrated in a more complex one. Makatsiná centers its activity in learning the ability to solve triangular structures by the node method, in accordance with reactive philosophy (Beer, 1990).

[1] **Makatsiná** means ***tutor*** in Totonaca, a Mexican pre-Columbian language. It is the name of the ITS.

There are other ITSs where cognitive models have been used with the purpose of clarifying the teaching process (Fletcher and Harris, 1996), (Gott, 1989).

### 2. 4. Different types of knowledge

Another important aspect of this expert module is the type of knowledge it will manage, which brings along the manner in which it will be represented.

#### 2. 4. 1. Procedural knowledge

It means basically the knowledge underlying the development of a task, and is directly related to production rules, considered by the researchers as a representation that gets the essence of the human process, with its cyclic mechanism of validation – action (if - then). Such it is the case of LMS systems, that teach algebraic procedures (Sleeman and Brown, 1982); BIP teaches BASIC programming (Barr, Beard and Atkinson, 1976); MENO, Pascal programming (Woolf and McDonald, 1984); WEST teaches arithmetic on an elementary level (Burton and Brown, 1982); CAPRA also deals with programming (Fernández, 1989) and Makatsiná, that teaches the ability of solving triangular structures by the node method (Laureano, de Arriaga and Martínez, 1999).

#### 2. 4. 2. Declarative knowledge

This type of knowledge deals basically with facts that are not related to a specialized use for a particular case. For instance, the statement "*America was discovered by Christopher Columbus*", represents a concrete, isolated fact. Such is the case of the systems: SCHOLAR, teaching the geography of South America (Carbonell, 1970); WHY, for teaching meteorology (basically rain probabilities and the factors that influence them); LISP Tutor, that teaches the declarative knowledge of the LISP language (Anderson and Reiser, 1985) and GUIDON, mentioned above.

#### 2. 4. 3. Qualitative knowledge

This is the knowledge that underlies the ability of humans to simulate and reason regarding dynamic processes in a mental fashion. An example of this type of knowledge can be found in SOPHIE (Brown et al., 1975).

### 2. 5. Knowledge representation

Within the existing formalisms to represent procedural knowledge, we have: production rules and compiled experts; the type of representation chosen is closely related to the technique that will be used for error detection and for tutorial intervention, and that is why the choice is so important.

#### 2. 5. 1. Production rules

Among expert systems using production rules (Laureano, 1995), there are many variations, but they all involve a set of rules, that are matched to a work memory of facts, in order to be triggered. Part of the advantages of the tutorial process is the fact that the tutor module can make decisions based on running a simulation of the desired learning; another one is the modularity, or being able to apply them in blocks, something which is inherent to the teaching process, and finally the state of the student knowledge can be diagnosed as a set of production rules. Examples of the latter are LISP Tutor (Anderson et al., 1985) and LMS (Sleeman et al., 1982).

### 2. 5. 2. Compiled experts

These are used when, because of limitations in equipment requirements, or because it is not convenient, the production rules mode of representation is not used. In this case, the solutions to the problems or most of the calculations for the solution are compiled. This representation is known in the literature as "*compiling the expert out*" (Anderson, 1988), and it forces a design that should keep in mind file structures allowing an easy and fast access to disk. There are implementations in hardware of these databases. With this technique the capability to solve any problem posed by the student is lost; but its characteristics allow the implementation of the tutorial process from a different viewpoint.

An example of ITS using a compiled expert is Makatsiná (Laureano, et al., 1999) where it is combined with rules, basically, because of the type of tutorial process carried out (sports coach). It is applied to a cognitive task that leads to integrate abilities. One of the factors that influenced the choice of compiled expert was the constraint that a reactive system should know the following step in advance.

## 3. The Student model

This module contains all the student records at a given instant and can be used to diagnose the effects of the tutorial process. This information will be used to choose the following topic to be taught, and what tactics will be the most adequate for the training. In case of a mistake, remedial tactics will be considered.

The student model consists of two components: a) the database representing the student behavior during the tutorial process and b) the diagnosis process that handles the database.

### 3. 1. Dimensions of the space to be considered in the student model

According to Van Lehn (1988) three main spaces have to be considered in the modeling of the student: 1) bandwidth, 2) type of knowledge to be taught, and 3) differences between the student and the expert.

### 3. 1. 1 Bandwidth

It means the input of different types of information, which allow knowing what the student is doing or saying. The quantity and quality of this information are important; since it is

communicated to the diagnosis component. This component uses this information, in its inferences and beliefs, regarding the state of the user.

Van Lehn proposes the following self-contained levels of information, in decreasing order according to the information they can provide:

- Mental States Level.
- Intermediate States Level.
- Final States Level.

In the highest level are the mental states the student goes trough when he/she solves a problem; these will have their own intermediate states, which in turn will contain the final states.

A mental state could be one that reflects the mental states the user can go through during the solution of a problem. In the case of intermediate states, they are formed by the changes of state that lead to the solution of a problem, from their initial state to the final one. These states are the ones of interest for the tutorial process. The final states are, as the name implies, the final stage of the problem solution.

The more information the diagnosis module has, obviously will be better. But one cannot always have access to the intermediate states. On the other hand, there is no way of accessing the mental models from an ITS (it would be tantamount to being able to *see* what the student has in mind every moment of the development of the problem). Through questions we can reach an approximation.

The information obtained from the bandwidth is important in the sense that the algorithm to be used in the diagnosis depends on it.

### 3. 1. 2. Knowledge Type

It has to do with the different treatment techniques that exist, according to the classification of the knowledge (procedural, declarative or qualitative).

In the case of *procedural knowledge*, included in the resolution of a problem, a class of interpretation is needed that relates the student model to the problem resolution knowledge, because the interpreter has to make decisions based on local knowledge. Two types of procedural knowledge exist: a) ordered or hierarchical, b) not ordered.

When we speak of *ordered knowledge*, we know that it has implicit sub-objectives; this characteristic is intimately bound with the type of diagnosis used, *i.e. e.*, it is necessary to know the conditions that trigger another state of the problem, as well as the whole set of states and sub-objectives.

### 3. 1. 3. Differences between the student and the expert

These are represented in general by the expert module, plus a list of the elements the student does not have (*missing conceptions*) but the expert does; and those the student has and not the expert (*misconceptions*). There are different implementation techniques: a) overlay, b) error files, c) library of mistakes made, in parts.

In the *overlay* form, the knowledge of the student model is represented as a subset of the expert knowledge, which is equivalent to the expert model plus a list of the elements the student does not have.

In the *error files* form, the differences stand out bearing in mind, the misconceptions and the missing conceptions; here the student model is represented by the expert model plus a list of errors; the system diagnoses the student finding his errors in that list.

Since the errors are very important for the diagnosis, different techniques exist to find them: a) the errors are obtained starting from the work done in education on that topic; b) observing the students directly, carrying out the Cognitive Task (CT); c) if a learning theory exists on the domain, it is used to predict the errors in the development of the CT.

**Table 1** - Diagnosis Techniques proposed by Van Lehn (1988)

| *Type of Knowledge* | *Non-ordered procedural* | *Ordered procedural* | *Declarative* |
|---|---|---|---|
| *Bandwidth* | | | |
| Mental states | | Model – Tracing (1) | |
| Intermediate states | Issue - Tracing (5) | Plan Recognition (4) | Expert system (6) |
| Final states | Path Finding (2) | Decisions Tree (7) | Generate and Test (8) |
| | Condition by Induction (3) | Generate and Test (8) | |
| | | Interactive Diagnosis (9) | |

### 3. 2. Diagnosis Techniques

In the following section we will deal with the second component of the student model, namely, the *diagnosis procedure*, used on the database. According to Van Lehn's classification (1988), there are nine diagnosis techniques, that are numbered in Table 1, where a recommendation is also made for their use, taking into account two of the dimensions of the student modeling (bandwidth and type of knowledge).

- Model-Tracing.
- Path Finding.
- Condition by Induction.
- Plan Recognition.
- Issue-Tracing.
- Expert systems.
- Decision Trees.
- Generate and Test.
- Interactive Diagnosis.

Techniques 1, 4 and 5 will be approached to explain how they work in an ITS; in this case Makatsiná. For a deeper coverage of the subject, the related bibliography should be consulted.

In the *model -tracing* technique proposed by Anderson, J. and commented by Van Lehn (1988), it is assumed that all the significant student mental states are available to the diagnosis procedure. The basic idea centers around the use of a non-determinist interpreter to model the problem solution; in this way, in each step of the solution process; the interpreter can suggest a set of applicable rules. The diagnosis procedure will trigger these rules and it will obtain thus the set of all possible following states (in the case of a determinist interpreter we will only have a successor state); one of these states will be the one that corresponds to the student solution. Like all the techniques, this has its advantages and disadvantages; among the disadvantages is what to do if none of the triggered states coincides with that of the student, since this would not make sense in the case of a determinist one.

The technique of *plan recognition* needs to fulfill two conditions: 1) that the knowledge must be procedural and ordered, and 2) that all, or almost all, observable physical states of the student may be used by the diagnosis procedure. In this technique the domain is represented in a tree, where the terminal nodes are the most primitive actions, the non-terminal ones incorporate the sub-objectives and the root the final objective.

The *issue - tracing* technique (Burton et al., 1982) is based on the analysis of short episodes in the problem solution, dividing its observation in sets of micro-skills or issues that are used during that episode. This type of analysis does not explain how these issues interact, neither the role they play in the global problem solution; it is only interesting to know whether or not they are used.

In Makatsiná the objective was to achieve a reactive tutor. It was implemented with model – tracing, because we only have to teach a strategy for the solution of triangular structures. The former is due to the constriction that in reactive systems it is of prime importance to know how to act. It was combined with issue - tracing, given the characteristics of the knowledge (procedural-ordered). Finally, due to the division of the problem solution in finer stages represented by sub-tutors that will revise the use of several skills, plan recognition was implemented within those micro-worlds.

## 4. Tutorial Module

This module is concerned with everything related to the problems of curriculum development and the manner of teaching that curriculum. The curriculum refers to the selection and sequence of the teaching material. The teaching, also known as tutorial process, concerns the methods of presenting that material, according to Halff (1988). Hence, the ITSs can use different teaching techniques. Tutorial interventions generally should contain, if not all, at least some combination of the following characteristics: a) having some control over the curriculum and its sequence; b) being able to respond to questions made by the user, and c) realizing when the user needs help, and of what type.

### 4. 1. Central aspects

Central aspects that are dealt with in the tutorial module design are: the control on the selection and the sequencing, the learning style, the teaching style, and the type of the CT domain; these three intimately bound aspects are fundamental in the design of this module.

#### 4. 1. 1 Forms of selection control and sequencing in the ITSs

According to Fernández (1989) 3 control types exist, in which the different ITSs can be grouped:

1. Systems based on rules.
2. Systems based on pedagogic states automatons.
3. Systems based on planning.

In the work of Fernández (1989) it is thoroughly explained of what each type consists. A summary of the basic ideas of each type of control types is given below.

To systems with control **based on rules** belong those that are built based on an expert system. This type corresponds to the GUIDON, NEOMYCIN and HERACLES systems. We will explain how GUIDON works.

GUIDON (Wenger, 1987) was designed to fulfill the following objectives: 1) to show the pedagogic utility of the expert system MICYN database; 2) to discover the additional knowledge in an intelligent teaching system, and 3) to express the strategies for a process in terms independent of the domain.

In GUIDON (Clancy, 1982) the capabilities of tutorial dialogue are separated from those of problem solving. Training is achieved through a consultation program based on rules and dialogue capabilities.

GUIDON compares the student questions with those of the expert, and criticizes them based on the following pattern: when the user poses a hypothesis, it is compared with the conclusion to which the expert (MYCIN) arrived, drawn from the same data. The transfer of knowledge is done through dialogues in different cases. The purpose of the system is to

increase the student's knowledge, pointing out to inadequate lines of reasoning and giving suggestions about aspects the student did not consider.

There are basically two sets of rules: those that make up the knowledge base of the expert system and the rules in charge of the tutorial process, which in turn are divided in rules to build up certainty (premises) and rules to select a discourse procedure (actions).

As a system **based on pedagogic states automatons**, we will mention MENO-Tutor (Woolf et al., 1984). This system was developed with the purpose of providing an adequate framework to define and try rules belonging to the tutorial process, with the purpose of generalizing these rules. The system uses two different components for the planning and generation of the tutorial process: the component in charge of the tutorial process and the natural language generator.

The component in charge of the tutorial process consists of a set of decision units organized in three levels of planning, refining successively the tutor's actions. In the highest level (Pedagogic Level) what is decided is the frequency of interruption of the student, and the frequency with which his/her knowledge will be checked. In the second level (Strategic Level) the pedagogic decision is refined in a strategy specifying the method to be used, like maybe determining the student's skill through questions. In the third level (Tactical Level) a tactic is selected to implement the strategy.

The implementation of this component is similar to a transition network that travels from one state to another by means of an iterative routine through a path space. The paths from state to state are not fixed, since the control structure provides a set of metarules that produce changes in the paths by default.

Systems **based on planning** are born like a necessity to build teaching systems that use strategies for long courses. The planning can be static or dynamic.

Peachey and McCalla (1986) proposed to use planning techniques to create long and individualized courses dealing with wider fields. This system belongs to the **static planning** type.

Planning techniques have been used mainly in robotics (García-Alegre, Ribeiro, Gasos and Salido, 1993; Torra, 1993), and also in areas like the modeling of reasoning (Hayes-Roth, 1979), and natural language understanding (Wilensky, 1983), among others.

The planning process consists of deciding one course of action before it is carried on; and a plan is the representation of that course of action. A planner is a program that goes from an initial state and tries to reach the desired final state through the application of a set of operators on the objects that constitute the world in which it operates. That is, it constructs the sequence of operators that allows transforming the initial state in the final state. A planner incorporates a knowledge base formed by: 1) the change of state operators; 2) a database in which the final state and the objectives are characterized, and 3) an inference mechanism.

The tutor system proposed by Peachey et al. (1986) is formed by five components: 1) knowledge base of the domain, 2) student model, 3) teaching operators collection, 4) planner and 5) executor.

The planner builds its plan by means of a sequence of steps to achieve an instructional objective. Each step is an operator instance and each operator has an associated action. The executor executes the plan invoking the actions associated to each plan step. The planner creates its teaching plan simulating the effects of the operators' actions on the student model An operator is similar to a rule in a production system and it includes a set of preconditions and a set of expected effects. Using the operator, the planner simulates the call of the teaching plan steps and considers the results that may be obtained on a virtual student model. There is not really a direct connection between the expected effects of an operator and the actual effects it produces. The executor must detect and recover these deviations, using the built-in options within the plan, or even re-invoking the planner to revise a plan that was not successful.

Fernández (1989) proposes a type of **dynamic planner**. In this type of system the planning techniques are set apart from the problem solution. One characteristic of this system is that it avoids re-planning costs every time the traced path does not adjust to reality.

Some pedagogic objectives are established for each session. To achieve them, some teaching strategies related with a set of plans are chosen. Every time the system interacts with the student, it is able to detect whether there are conflicts, in which case the objectives at a local or global level are reconsidered. It contains teaching strategies based on states independent from the domain to be taught, in order to be able to use it in some other domain of structured nature. The strategies control is made in the most flexible way; since a preset form of transitions between states does not exist. Transitions are set in terms of plans and conflict solution rules.

The dynamic planner proposed by Fernández (1989) is composed by the cooperation of four modules: 1) pedagogic decision; 2) thematic decision; 3) teaching module, and 4) supervisor module.

The pedagogic decision takes care of: 1) selecting the combination of teaching strategies more adequate to get the student objectives keeping in mind information from the user; 2) developing the chosen strategies, generating progressive sub-objectives, and 3) communicating with the student on the choice of different didactic activities to be developed.

The thematic decision is the component that refines the objectives set by other modules, considering the domain that is the teaching object.

The teaching module explains or verifies the concepts selected by the thematic decision. It also deals with the communication and the update of the student model.

The supervisor module acts every time a user-system interaction takes place. To detect possible conflicts between the user objectives and those set by the tutor (current or on wait). It selects a local strategy or notifies the need to restate the strategy.

From the 80s on, a different form of confronting artificial intelligence problems appears, known as **reactive philosophy** (Brooks, 1991a; Brooks, 1991b; Laureano, 1998). One of the objectives of reactive systems is to achieve the distribution of control among the agents, to have *the quickest possible* decision taking cycle. We must emphasize that in the case of reactive agents there is no exhaustive symbolic model of the environment on which the system may reason. The model on which it makes decisions is *the real world*; hence the control of a reactive system depends on *the objectives on each agent* and their interaction with the real world.

In Makatsiná the meaning of control in the terms mentioned before does not exist. This was designed according to the reactive philosophy principles (Laureano and de Arriaga, 2000). In a general way it consists in: agents that trigger their action on the basis of the environment perception (errors) and each agent's objectives. One of the main contributions of its development is the proposal of MA (Multi-Agente) architecture, based on reactive agents (Laureano and de Arriaga, 1997; Laureano and de Arriaga, 1998).

In Makatsiná the control works by means of a hierarchy based on errors that was established according to the order of learning of different sub-abilities.

### 4. 1. 2. Learning style

It means the best way of teaching a given domain. On this question there is a great deal of theory developed within the education field by pedagogues and psychologists. Usually they start from experiences to develop techniques that in some way give better results. For instance, the ability of the engineering students to perceive in a problem the significant implicit visual aspects and use them in is resolution is well known. In this case, the use of visual methods would enhance these characteristics. At the present time you can resort to techniques that allow knowing more about visualization, and by these means being able to guide the students toward the development of this capability, that will grant them a better learning.

In Alonso and Gallego (1994), an historical review of definitions and contributions is made that concludes with the design of a questionnaire that can be applied to groups of students, and with it to know the learning style preferences. The questionnaire distinguishes among four learning styles: active, reflexive, theoretical and pragmatic. These in turn represent the experiences obtained during the learning process: to live the experience, reflection, generalization and elaboration of hypothesis, and finally application. Nevertheless, clear differences exist, based on the different areas of knowledge. We could think that the fact that different University Schools are grouped according to the way they organize the knowledge in technical, humanistic or experimental careers, means they lean toward a given type of learning; that is directly related to the kind of domain and the form of using it

in practice. To be able to know the style of learning of the students and their preferences implies designing courses that contribute better results. In the particular case of the ITSs, we could design them emphasizing such learning styles.

### 4. 1. 3. Teaching style

It means the opposite part to learning, and must be designed in such a way as to take advantage of the *learning style*. We must view the *teaching-learning process* as a communication mechanism that has to occur between two agents. Hence the tutorial process is approached as a communication process (Girard, Gauthier and Levesque, 1992), more than as a conventional teaching process.

### 4. 1. 4. Domain type

It refers to the difference existing among the different types of tutors: **expositive and procedural.** In an *expositive tutor* the emphasis is on the *factual knowledge* (it represents the facts that constitute declarative knowledge) and starting from this knowledge first order abilities are inferred that belong to the declarative knowledge class.

*Procedural tutors* carry out the teaching of *skills and procedures* that have applications in the real world; these tutors perform rather as trainers, supplying examples and showing the use of skills through problems development. Inside their content there are problems oriented toward tests and special practices.

## 4. 2. Curriculum

This problem, as already mentioned, is approached from two different outlooks: 1) to find an adequate representation for the material to be taught, 2) to select and sequence that material.

The first problem is related with the knowledge about the teaching instructions and with the expert module, and the second is related with the type of control (Section 4.1.1).

### 4. 2. 1. Selection and sequencing

With regard to *selection and sequencing*, differences exist according to the tutor type; in the *expositive* ones the problem is centered on maintaining coherence in the material presentation for a later reflection. *Procedural* tutors have besides the problem of having to order the sub-skills that integrate the skill-objective, and of having to select the exercises and examples that reflect that order.

### 4. 2. 2. Selection and sequencing in procedural tutors

Tutors of this type base their teaching on *exercises and examples.* In this case the main point is to have appropriate mechanisms for the selection and sequencing of the material. The ideal situation would be to be able to choose them according to the learning style, but,

according to Anderson (1988), there is not a precise enough and powerful enough theory of learning to support an interactive tutorial process.

Current research recommends to keep in mind the following points in relation to exercises and examples: a) *manageability*: all the exercises must have solutions, and they should be understood by a student that has covered the previous material, b) *structural transparency*: the sequence of exercises and examples must reflect the structure of the process to be taught, and should have the quality of guiding the student in the acquisition of the skill-objective, and c) *individualization*: the exercise presented to the student must cover the sub-skill(s) that the student already handles, and should be able to be easily related to the sub-skill(s) that are intended to be acquired at the moment.

Van Lhen (Halff, 1988) proposes the so called "step-theory", where he says that the curriculum must be divided according to important aspects or characteristics. These divisions represent the *theory steps*. Each lesson will be able to use only a single theory step, so that the procedure to be learned takes advantage of the division of the curriculum in steps. He also speaks about the *felicity conditions* or ability to find the domain divisions that help most the tutorial process. Such division will coincide with a step in the procedure to be learned. In relation to the same it is considered that what is needed is the help of a good *cognitive task analysis* (CTA) in order to find those critical domain breaks that help clarifying points of teaching, like those discussed by Castañeda (1993), Redding (1992) and Ryder et al. (1993).

As for the *process of selection and sequencing of the material*, it is important to design the curriculum to fulfill the following functions: 1) containing the material divided in such a way as to make its manipulation easy, based on instructional objectives; 2) sequencing the material in a manner such that its structure suits the user; 3) assuring that the objectives posed in each unit can be achieved, and 4) updating the same as a consequence of the evaluation mechanisms for the tutorial process impact on the student.

### 4. 3. Tutorial process (teaching)

This component includes the functions covering *presentation of the material*, forms of being able to answer the student questions, the conditions and the content of *the intervention of the tutorial process*.

#### 4. 3. 1. Presentation methods

The presentation methods depend on the type of domain to be taught and the objectives that are to be fulfilled during the tutorial process.

*Expositive tutors* use different forms of dialogue; *procedural tutors* oriented to the management of skills use examples and exercises with a trainer, to achieve the management of those skills.

The different forms of dialogue imply different teaching objectives (Halff, 1988). Collins proposes a guide for the selection of dialogue according to the teaching objectives. This classification coincides with the one proposed by Gagné and Merrill and is summarized in Table 2.

**Table 2:** Dialogue strategies for different instructional objectives, according to Collins (Halff, 1988)

| I*nstructional Objective*s | *Strategies* |
|---|---|
| Teach facts and concepts | Separate facts or concepts |
| Explain facts or concepts | Teach rules and relations<br>Selection strategies<br>Cheat |
| Teach skills by induction | Exercises and examples oriented to show sub-skills |

### 4. 3. 2. Tutorial Intervention

The tutorial intervention is necessary to keep the situation under control during the tutorial process development, in order to keep the user away from an inappropriate or incorrect learning, and keep him/her far from paths straying from the teaching objective. To automate this process implies to devise rules to decide whether to intervene or not, aside from formulating the intervention content.

There are basically two forms to guide a tutorial intervention, already commented as diagnosis techniques. These are: 1) following a path (*model-tracing*), and 2) following an issue (*issue-tracing*).

#### 4. 3. 2. 1. Intervention with model – tracing

In this type of intervention you have the paths the user could choose for problem solution. The behavior of both the user and the system is revised, trying to match the user's development with some of the paths that can be followed. When the match fails, the tutor intervenes supplying some advice that will allow the student to resume the right path. The disadvantage is that the tutor will intervene whenever it cannot recognize the path followed by the user, even if it is a better one

#### 4. 3. 2. 2 Intervention with issue – tracing

This form of tutorial intervention was developed by Burton et al. (1982) and implemented in WEST. Here the skills or knowledge that the student has to learn are called *issues*.

Each issue represents an *articulated mini-theory*; that is, a piece of the expert seen as a glass box characterized by two procedures.

The *first procedure* plays the role of an observer of the student development. Its mission is to obtain evidence to judge whether or not the student uses that particular concept or skill. This procedure is called *issue recognizer*, and its observations are used to build a model of the student development.

The *second procedure* knows how to use the information contained in the student model to decide whether or not the student masters the issue. This procedure is called *issues evaluator*.

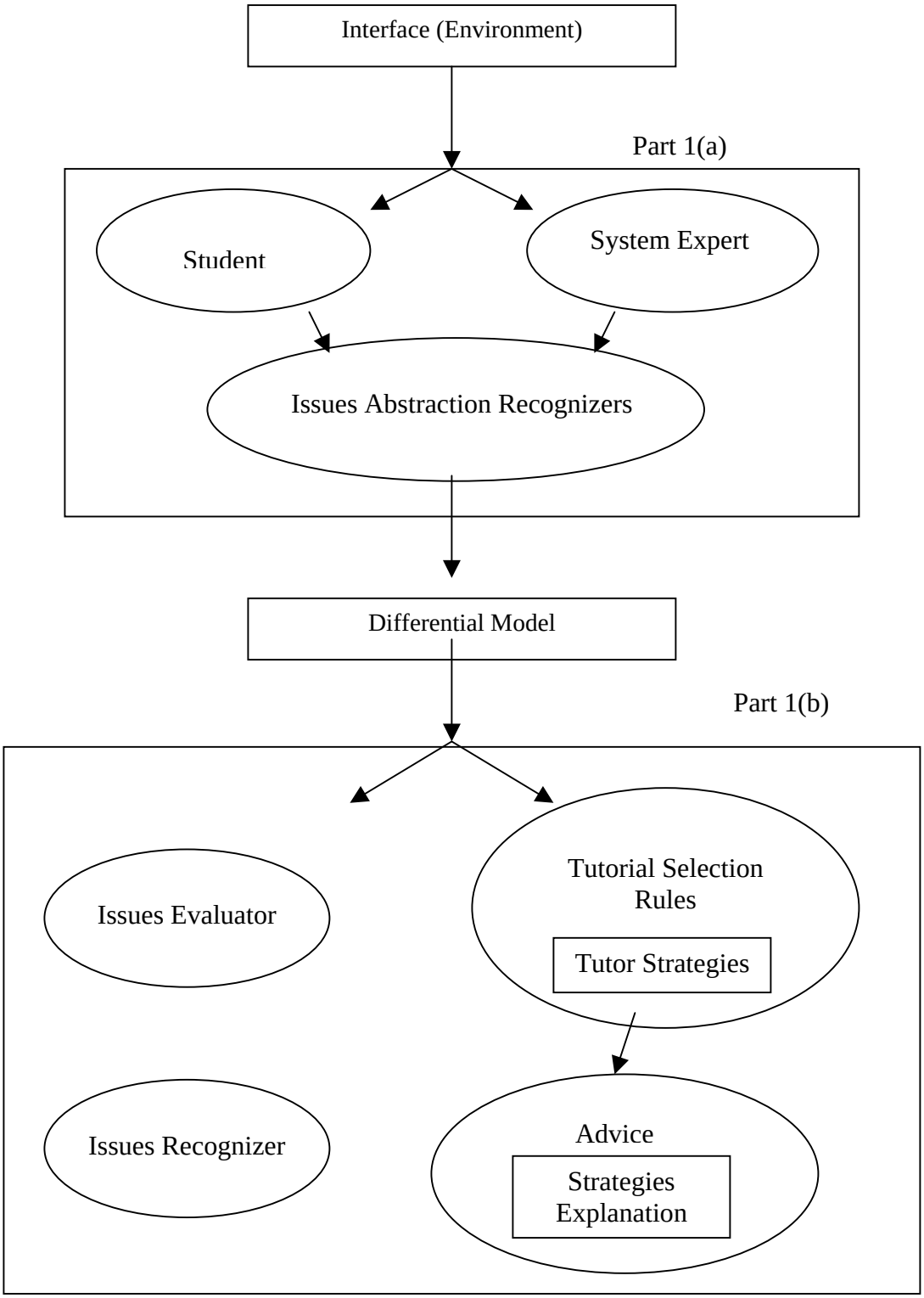
In the specific case of WEST, a trainer type of tutor was used. To know the progress of the student all the model evaluators were executed. When the student executes a wrong play, this entails a weakness or lack of skill, since it was not good. Then his/her weakness is compared with the skills (issues) necessary to make a better play, and in that way it is found why he/she did not made that particular play.

Figure 1, composed by Part 1(a) and Part 1(b) presents the process of tutorial modeling using *issues and examples*. In Part 1(a) the construction process of a model of student development during the solution of a series of problems is shown. In the special case of WEST they are plays, since learning is carried out in a game context. Each time the student makes a play, the *issues recognizer* elaborates the abstraction of the important aspects in the development of the student and in the system's expert in the same environment, using the same issues recognizers for both. The two abstractions are compared to obtain a differential model that allows finding out the issues that the student does not master.

Once it has been identified in what issue the student fails, the trainer produces an explanation of that issue, followed by an example of use. What is pursued with this type of intervention is for the student to receive the information when he is more receptive to process it, in other words, as soon as he/she makes the failure.

It is important to point out that without the expert it would not be possible to evaluate the degree of mastery of an issue (skills or knowledge) the student has, or if he/she has not used it for lack of adequate experience.

**Figure 1**. Tutorial Model with issues and examples, by Burton and Brown (1982)



Part 1(b) it represents the highest level of abstraction in the tutorial intervention of the *trainer* type. When the student carries out a play not considered as optimal (compared to the expert), the trainer uses the i*ssue evaluator* to create a *list of issues* in which the student is wrong; on the other hand, it has the list of the expert's best plays. Then the trainer invokes the *issues recognizer* to determine which issues are registered as best plays. From those two lists, the trainer selects an issue from the list of *issues that the student failed* and an expert's good play that contains this issue to show it. If there are no common issues in the two lists, it is inferred that the student's problem is outside the tutor's reach and the trainer says nothing. Based on other tutorial principles the decision to interrupt or not the student is made, but, if the trainer decides so, the issue and the example are passed on to the student; it is expected that this action generate a feedback in the student.

An ITS can use either one of these two techniques for the tutorial intervention, or combine them, which is frequently the case. An example is Makatsiná (Laureano and de Arriaga, 1998) that uses the issue - tracing technique for the tutorial intervention, and this is combined with a trainer type tutor that detects the skills the student uses in a wrong way. The latter is based on the mechanism proposed by Burton et al. (1982).

## 5 Interfaces

The interface is a very important element in the architecture of the ITS, since it covers several activities in the system's global performance**:**

- It is the communication bridge between the student and the system.
- It is the only physical means capable of capturing the student's development.
- It represents the means through which the tutor (system) will carry out the interventions.
- In accordance to the teaching domain, the interface potential must be exploited to the maximum, using the most appropriate means (video, audio, etc.) for the best understanding of concepts or abilities management.

(The last two activities have a didactic tool focus).

### 5. 1 Help

This help is basically the one given when the student is solving some problem. ITSs have different help types:

#### 5. 1. 1 System Help

Most ITS interfaces have this help type, useful when the student requests it or when mistakes are made. For instance, MACSYMA (Genesereth, 1978) builds a plan explaining the actions that may have led to the error and proposes a list of probable misconceptions causing the user to make that mistake.

### 5. 1. 2 Assistants

They are mechanisms allowing developing a part or the whole task. This help type allows the student to concentrate in those parts where he/she finds difficulty, leaving the parts he/she masters to the system. In Brown & Foss' Algebra Land or Anderson's Algebra Tutor (Burton, 1988) the students have this option, which allows them to see how the operations are carried out.

### 5. 1. 3 Power Tools

These include the capture of decisions and actions of the student in structures for their further observation. This type of mechanism helps the student to think over the activities developed during the problem solution. Algebra Land and Anderson and Swarecki's Geometry Tutor (Burton, 1988) have this kind of help. In the special case of Algebra Tutor, each decision of the student is captured in a tree structure allowing him or her to visualize the search space.

### 5. 1. 4 Reactive

In systems of this kind, there are mechanisms that respond immediately to the student's action, in the specific situation context, viz. SOPHIE - I (Brown et al., 1975). An example of reactive environments is those that react to the request of evaluating the hypotheses related to the measurements the student takes. They will not tell him or her they are wrong, since the logic can be correct according to the measurements the student made. In case the proposed hypothesis is not consistent, the system confronts the student with examples having the same characteristics (in a correct form), and that he/she is overlooking. A characteristic of these systems is that they allow the students to articulate their hypotheses in a form opposite to those they are using.

### 5. 1. 5 Modeling

In this kind of help the system develops the task while the student is observing. It allows the student to observe how an expert would behave in the task development. To achieve that, it is necessary for the expert's model to articulate its decisions, and that they coincide with the selection strategies for them. The articulate expert in SOPHIE - II (Brown et al., 1982) is a good example, since it allows the student to create a failure in the circuit and nevertheless make it work. The student will carry out the measurements up to where it is possible for him/her and the expert will explain later each measurement and the reason why it has been made, based on its strategies and the system qualitative analysis.

### 5. 1. 6 Trainers

This tutor has been developed in order to follow the student closely in the task realization and to interrupt him/her when their realization is not optimal or they made a mistake, providing suggestions in those cases. It is worth mentioning that to implement this type of mechanism, it is necessary to know *the expert way*. Already classic examples are WUSOR and WEST (Burton et al., 1982); both tutors are interrupted when the student makes a bad

move, and they give advice. They create the student's model comparing his/her development with that of the expert. With WEST it was shown that, although you cannot trace all the student does, the models of a non optimal development can be recognized.
There is also the possibility of combining several help types like in the case of Makatsiná (Laureano and de Arriaga, 2002; Laureano et al., 1999) where several help types are managed: there are power tools (all the actions developed by the student are saved, allowing him to observe at any time the decision points), modeling (there is always the option of having the expert to execute part of or the whole task) and trainer (approach used to develop the tutorial model).

### 5. 2 Multimedia and cognition

The interface is so important that we could think of it in relation with the senses that detect its performance when interacting with some system.

#### 5. 2. 1 Multimedia as a didactic tool

Rickel (1989) comments that people retain about 25% of what they listen, 45% of what they listen and observe, and 70% of what they listen, observe and develop. Those data lay down a strong argument to develop interfaces that include text, sounds and graphics, aside from having the capability of generating an interaction with the student as if he/she were part of the system.

An ITS type using interfaces is the one mentioned by Rickel (1989). The system authors consider the graphic description used in causal relationships and in topologies as very important. Some concepts can be presented using graphics, facilitating the student's understanding. On the other hand, *to manage an interface based on ideograms allows a deeper connection between the student and the system*.

#### 5. 2. 2 Learning by association (cognition)

It is worth mentioning that multimedia use has a direct relationship with learning by association, as mentioned in Laureano (1993). Memorization by association can be mechanized; this reflection leads us directly to the correct use of multimedia, in order to mechanize procedural knowledge.

The learning by association method described in the year 55 BC by Cicero and extended by Godden and Baddeley shows that memory does not depend only on a formalized set of tricks to remember elements from different positions, but it is rather based on something deeper, depending on the learning context. From the former reflection is derived the importance of modeling interfaces with a deep *fidelity of display*, especially in beginners.

However, the fact that association plays an important role in the manner of linking our personal mental schemes in a coherent whole does not imply that this is the best way to supply information and knowledge. Indirect evidence against associative methods is the success, for over 500 years, of the conventional book and the development of narration

along more than a thousand years. On the other hand, they have the following disadvantages: a) an immediate answer to the student does not exist, b) the students are passive entities; c) exercises of differing complexities, according to the ability of the students, are not generated.

Those disadvantages become advantages in the case of ITSs; hence, multimedia systems, for the time being, are no more than a set of experimental techniques, useful in specific cases that are set apart from the mainstream of conventional educational thought.

Summarizing, in an ITS interface the information input is meant to be sound, but it must as well allow flexibility for the input of synthesized information arising from the observation of the student development. As for the output information, it should be rich, that is, *spelled in full*, using the power of the multimedia system to take advantage of all of the student's senses, *without falling into an excess*.

As an example of an interface fulfilling the previous criteria, we can mention the one in Makatsiná (Laureano et al., 2002; Laureano et al., 1999) where we particularly stressed the development of an interface controlled through ideograms. The tutorial interventions and the student's interaction are through graphs. The former was because the teaching domain is considered *ad hoc* to be managed by ideograms.

On the blackboard, you explain with colored pieces of chalk and a continued writing along the lines forming the triangular structure, to show the sense of the equilibrium forces, whose colors are also different, according to the type of stress (tension or compression).

## 6. Multiagent Architectures: a New Outlook

### 6.1. Background

The 90s were of great importance for computer science, because multiagent (MA) architectures were established as a novel analysis and design method for computerized applications. Those architectures appeared in the late 70s, bringing along a new viewpoint to analyze the same problems that were formerly dealt with using artificial intelligence.

These architectures have a huge potential to improve the general theory and practice of modeling, design and implementation of software systems. In this way an association of agents is configured as a set of entities capable of: 1) generating their own objectives, based on their perception of the situation of the operation environment; 2) using knowledge, by reasoning and dominion methods, for problem solving, decision making or carrying out given cognitive tasks, either in an autonomous or cooperative fashion.

Such models pose a solution to complex real-world problems, providing reusable and scalable software, capable of fulfilling objectives through autonomous agents placed in dynamic environments filled with uncertainties, which may cause interactions between agents, or between themselves and the environment, with a likelihood of learning from

their own experience. In this manner, we get flexible software structures that can be executed in a single computer or in several of them in parallel.

The various kinds of agents can be classified using different criteria, thus: 1) according to their mobility, as static or mobile, depending on whether they remain fixed in a single computer, or migrate to others through networks; 2) considering their structure, as deliberative or reactive agents, the first kind having a more complex structure, which gives the agent a mental state characterized by beliefs, desires and intentions, allowing it to take basic decisions regarding its own operation, like accepting or rejecting the goals the system may offer; 3) with regard to ideal attributes the agent could possess, among those that stand out: intelligence, autonomy, learning ability and cooperation with each other. 4) with respect to their activity, the so-called personal agents, dedicated to managing their own work log, or obtaining information, shopping, etc., in the Internet, planning visits and the like, and last: 5) hybrid agents, which belong to two or more of the above described categories.

In the former context, Nwana (1996) identifies seven main types of agents: 1) personal collaborator agents, 2) agents in charge of an interface, 3) mobile agents within given networks, 4) searching, classifying, and summarizing information agents (mainly in the Internet), 5) reactive agents, 6) intelligent (cognitive) agents, and 7) hybrid agents.

There are applications combining two or more of these categories, which, according to Nwana (1996) are called Heterogeneous Agent Systems. An explanation of their characteristics follows:

*Collaborator agents* are autonomous and cooperate with others to carry out tasks for their users, i.e., “negotiate” in order to reach together a solution on some matter.

*Interface agents* are autonomous and learn what they need to attain their objectives. These agents are a kind of “personal assistant”, collaborating with the user in a task environment (Maes, 1991).

*Mobile agents* are software agents for computing processes, able to scan a WAN (for instance, the WWW), interact with external servers, collect information or fulfill a task and return to the emitter site with the corresponding results. These agents are also autonomous and cooperative; their task being very similar to that of (Internet) Information Agents; the difference is that mobile ones actually migrate from one computer to another, and those for consulting the Internet do not.

*Internet information agents* handle, process and filter information coming from anywhere on the Net. They are not usually mobile agents; therefore, they have to rely on “browsers” that take on the job of the search and primary classification of the information they need, on which the Internet agents will later act. These kinds of agents are defined by *what they do*, as opposed to collaborator or interface agents, which are defined by *what they are* (Nwana, 1996).

*Deliberative or cognitive agents* are those possessing a "mental state" that includes an internal symbolic model of the world the way they will perceive it, and whose decisions will be made based on that symbolic model and their mental state (Wooldridge and Jennings, 1995).

*Reactive* a*gents* are a special kind of agent, not having internal symbolic models, or else having ones that are extremely reduced. Their actions are based on a stimulus-reaction cycle, in response to events developing in the real world. In this line of thinking it is intended for agents not to be intelligent in their behavior before a given stimulus, but from the set of all their elementary behaviors a global one is to result, that should be considered intelligent (Laureano et al., 2000; Laureano de Arriaga and García-Alegre, 2001).

*Hybrid* a*gent* is one that is built on two or more of these design philosophies (collaborator-mobile, for instance). *Heterogeneous agent* sy*stems* consist of a system integrated by two or more agent types.

One of the chief objectives of AI is trying to develop selection-action mechanisms that can be used by autonomous agents to determine which actions are to be developed by intelligent systems.

From the beginnings of AI, the research, according to the deliberant thought paradigm, is based on the fact that an intelligent task can be implemented by a reasoning process operating on an internal symbolic model. This representation is used by the reasoning process (called *planning* in robots) to determine the sequence of action that will accomplish the objective. This line of thought has been successful in some fields, but the results have not been as expected when we deal with a system operating in dynamic and complex environments; hence, distributed artificial intelligence (DAI) has seen significant advances for the last 15 years. Within this branch of research, agents and MA architectures made their appearance.

There are two important aspects in the design of agents with this type of architecture, no matter which classification is used; they are: 1) the reactive aspect and 2) the deliberative aspect.

### 6.2. Deliberative Aspect

In traditional DAI, systems consist of a group of agents, each one being an expert system by itself, that has a knowledge base that allows the fulfilling of his task, and once accomplished, to communicate its results to the other agents. Traditional systems are actually seen like problem solvers, where agents are intentionality based mechanisms that have explicit objectives and plans enabling them to attain the final objective. The existing problems have to do with cooperation, where several agents must coordinate their activities, and eventually solve the conflicts that may arise.

### 6.3. Reactive Aspect

The other approach requires that agents should not be intelligent in their basic behaviors, but rather that an intelligent behavior arise from their overall behavior. These agents do not have internal models of their environment; their action is based on a stimulus-reaction cycle, according to events presented in the current state of the environment they are inserted in, and hence they have no explicit objectives (aside from maybe a global one) or planning mechanisms, and even then, they can solve problems considered to be complex. This school of thought is less representative and more experimental as compared to the deliberative one, the latter being more formal; however, the use of these systems is being extended to other fields, like simulation, games (Ferber and Drogoul, 1992) and Intelligent Tutoring Systems (ITSs) (Laureano et al., 1998; Laureano, 2000).

One key point is the fact that the agents are relatively simple and interact with other agents in a rather basic way. However, complex behaviors may arise when the overall assembly of agents and environment is considered.

Maes (1993) states three key points related to these agents: 1) emergent functionality, implemented on the basis of the interaction dynamics of these agents with the environment; 2) task division: reactive agents are seen as a set of modules, operating in an autonomous way and responsible for specific tasks. Communication between them is kept to a minimum, if it exists at all, and will be done at very low levels. There is no overall model at all within each agent, nevertheless, the overall behavior is fulfilled; 3) reactive agents tend to operate on representations close to unprocessed data.

Because of all of the above characteristics, agents are considered as a joining framework for the different subdivisions of artificial intelligence necessary to design and build intelligent entities.

### 6.4. MA architectures and ILSs

According to Estevez (2002), we tackle deep problems when we ask ourselves: 1) Are students really learning?; 2) What is the degree of applicability of their acquired knowledge within the environment for which they are being prepared?; 3) Do they have enough knowledge to continue studying in colleges and universities?; 4) And due to the multidisciplinary interaction required by the current pace of scientific and technological development, When does the acquired knowledge becomes obsolete?

And there are still additional questions we could ask regarding any kind of education, particularly higher education: 5) What is the real expert’s behavior regarding the learning domain?; 6) How can we achieve the migration from a learner to a robust expert in a shorter term than currently possible?

Facing such perspective, we pretend to enrich the didactic design with cognitive sciences, in order to introduce these kinds of processes into teaching and learning through the design and use of cognitive strategies. Therefore, this proposal is based on: 1) A multi-nodal

perspective (based on different theoretical sources); 2) A holistic and complete approach (knowledge, abilities, attitudes, values, …); 3) Prioritizing the use of cognitive strategies as means to activate the mental processes needed for the learning process; 4) Using the cognitive ability components, including the expert's own mental models.

The viewpoint of analysis and design of intelligent learning systems (ILSs) has evolved recently, due to the influence of MA architectures and the constructivist approach in the development of virtual laboratories, of which simulation is an important part. Altogether it has meant a greater flexibility in the way to interrelate the four classic components (tutor module, student model, expert module and interface) enriching them with artificial intelligence techniques, not only in the tutorial process and the representation of knowledge, but including the way to interact with the users, which gives a more attractive interaction.

Les, Cumming and Finch (1999) comment in what way agents can be applied in education: 1) agents acting as trainers or counselors, providing individual explanations; 2) agents with different personalities, offering different outlooks, 3) agents assisting in the navigation and other incidental tasks, directed according to learning objectives, 4) interfaces formed by agents capable to adapt to the user, based on his interests (Veitl, Petta, Spour and Obermaier, 1999), and 5) agents interacting with other agents and/or users, to enhance collaborative learning.

On the other side, the constructivist approach and its relation to the directed approach has to be mentioned (Urretavizcaya, 2001). The later is the one that has been used in the development of ILSs, where an *ad hoc* teaching-learning process is assembled for a given dominion, under a set of pedagogic strategies based on the experience of one or several teachers. In the constructivist approach, the formulation is based on the premise that the best way to learn is dedicating oneself to construct some entity consciously. Here the student is in control of the teaching activity by constructing his own learning session. In this last approach the agents can interact with the user, providing advice for the development of a given activity. (Lester, Callaway Grégoire, Stelling, Towns and Zettlemoyer, 2001).

In the use of agents two main types are outstanding in the development of ILSs: 1) systems developed using MA architectures (with reactive, cognitive and/or hybrid agents) that strive to direct to a small or large extent the learning process, and 2) systems emphasizing the development of friendlier interfaces during a teaching-learning session using pedagogic agents.

### 6.4.1. Deliberative-cognitive agents based systems

The work of Girard et al. (1992) describes a very interesting MA architecture, implemented by means of an ILS that we could define as traditional, on account of the modules included, except for the fact that the complete system, including the student module and the tutor module, are considered as agents. The motivation of this architecture comes from considering the human teaching activity as a mainly opportunistic activity;

hence, the ILS developing mechanism is considered an opportunistic planning process: the ILS is definitively formed by a tutor, planning, curriculum (or dominion) and micro-world module.

The system components appear with layered semantics, obtained from a decomposition of the problem (to teach, in the current case). This decomposition will allow the agents, their responsibilities and their interactions to be known.

Thus, system agents are divided into: human and programmed. The human ones are: the student and the teacher, and the programmed ones: the planner, the tutor and the micro - world. The curriculum is not part of the active components, since it does not influence the behavior of an ILS during the teaching process. The curriculum is only the object of communication which receives the interactions of the teacher and planning agents.

Finally, the learning process will be implemented on the basis of two policies: 1) planning for conjecturing, forecasting and provoking events pertaining to teaching, and 2) evaluation for the observation, analysis and understanding of teaching events.

On the other side, the responsibility of each agent, based on those two policies, has an internal and external behavior: 1) viewed from the inside, each agent must conjecture the events produced by teaching, and analyze and understand other events, produced by teaching in an evaluation stage; 2) viewed from the outside, each agent must provoke the teaching events, in order to plan and observe the evaluation.

The intention of this work is twofold: Because of the impossibility of considering the changes before the system comes back from the inference about its symbolic world, we first attempt a task distribution approach, and then we take into account the partial representation in each one of the agents based on their assigned tasks. The system contemplates decomposition in a rougher grain than does reactive philosophy.

In the work of Néhémie (1992) a MA form is proposed to model the student in the different teaching systems, among which the ILSs stand out. His study is based on a blackboard type of architecture, implemented as a MA system. His main contribution is that the student module is seen as an open system that analyzes information coming from the overall system. This module is an agent modeling the student, based on the analysis of the incoming information from two sources: 1) the inputs: their objective is to make sure that the student model be updated regularly, including recent events that happened during the session. In the blackboard architecture context, this flow is accomplished verifying the reports from several activities and interactions carried on the blackboard by different agents; 2) the outputs produced by this agent during the learning process are a synthetic model of the student with a summary of his/her trends. This model is permanently available to the other agents, and is kept updated.

Vassileva, Geer, McCalla and Deters (1999) developed a MA distributed and collaborative help environment, in which humans and *ad-hoc* resources are found, according to the request made (question or learning). This type of environment motivates the users to help

each other; in this way, help is extensible both in depth and in breadth. An important characteristic of their system is that no difference exists between human and software agents. This approach pretends to make the ILSs more flexible, since each component (tutor, expert, and diagnosis) is enriched with the collective human agents.

Keeling (1999) proposed a methodology to build educational agents where machine learning methods and knowledge acquisition methods are combined, as well as intelligent tutoring process concepts and other tools from educational fields. Such methodology intends to reduce the amount of work implied in knowledge engineering, as well as the time required to conform a dominion expert. An assumption of this research is that the methodology can be used to: 1) build agents capable of including intelligent aspects of educational software, and 2) build independent agents capable of assisting both teachers and students.

Canut, Gouarderes and Sanchis (1999) devised a novel agent model to design ILSs. Their work shows an evolution in design from intelligent agents with three parameters: 1) mobility (the agent's degree of freedom), 2) autonomy (degree of autonomy given to an agent, in terms of interrelations and authority), and 3) intelligence (degree on which preferences, reasoning and learning are displayed).

Within these parameter's limits an agent can learn and adapt itself to his environment (in terms of available inputs and objectives). These three properties are studied as properties of a MA system. On the basis of the aforementioned approach, and starting from the above quoted model, the authors propose an enriched model, where the components of an ILS are turned into actors, an actor being an intelligent agent with the following aspects: reactive, capable to learn, adaptive and cognitive. The actor architecture includes four modules (perception, action, control and cognition), in turn distributed in three layers: reactive, control and cognitive.

The agent architecture proposed by El Alami, de Arriaga and Ugena (1999) is fully modular. The agent has a mental state module, a communication module containing a natural language processor to allow free messages for communication among agents, a specific knowledge module which, depending on the action complexity, can be split in different layers: reactive, tactical and strategic. In the reactive level, the answer to certain questions or the actions in response to the appearance of certain stimuli is produced in real time with a poor or null cognitive complexity. In the tactical level the answer produced or actions taken are the consequence of running an algorithm (tactics); the answer or actions taken are usually related to short term decisions; the cognitive complexity of this process is medium or average. In the strategic level the answer produced or actions taken are related to long term decisions (strategies) and the process has greater or enough cognitive complexity. The architecture has also a hierarchical control system: a specific control module at each level with transfers among them.

The operation cycle for this architecture consists of: situation analysis (obtaining the next sub-goals), planning and execution (vigilance and control). It is carried on by: 1) a mental state module with deliberative capacity that allows the agent to modify even its own state,

based on the external or internal situation; 2) a learning module, important for the development of its own cognitive capacities; 3) a perception module (necessary if the agent has its own sensors) and a communication module that receives messages from other agents and also determines the identity and analyzes the message type.

In the case of ILSs we need agents modeling human behavior in learning situations; hence the cognitive aspect of an agent resides in its ability to learn, discovering new facts that improve its knowledge in order to make its use suitable to a higher degree.

### 6.4.2. Reactive agents based systems

Since they do not tackle task planning and complete modeling of the dominion, Reactive agents present an apparently more limited functionality than cognitive agents. They constitute an approximation to be borne in mind for implementing intelligent learning systems. Above all, we must state that the need to coordinate their behavior or the arbitration of these agents themselves motivates establishing modules in charge of their coordination, in the form of elementary controllers or planners, giving them a hybrid characteristic. We can not talk definitively about purely cognitive or reactive agents; we are actually dealing with agents emphasizing more or less planning, modeling and control tasks (reactive and deliberative aspects).

On the other hand, it is necessary to account for the new so called “student-centered” approaches, which rather than being mere attempts to improve student learning patterns, have brought a strong commitment from several universities, like Roskilde, Ahlborg, Maastricht, Eindhoven, for over twenty-five years. These approaches are in fact different approximations, among which teaching by problems and teaching by projects stand out.

In such an approach the student assumes a directing role and is responsible for his own learning; lectures are relegated to a secondary resource, and may even disappear completely (de Arriaga et al., 2002). Training and coaching tasks instead increase their relevance, acquiring an important role. The main characteristic of these tasks is that they allow doing the students’ follow-up and tutoring functions without a large planning and modeling load, and are thus easily assumed by agents of this kind.

Laureano and de Arriaga (1998 and 2000) present a MA architecture that does not need a student model, nor any kind of control in the pedagogic module, which in terms of reactive philosophy implies not having preset objectives nor an exhaustive representation of the environment. The system actions are based on its observations, following the reactive philosophy created by Brooks.

The advantages provided by this new architecture are: 1) better response time, 2) decomposition of the ability to learn in sub-abilities, whose sub-experts deal with errors in a specific way, 3) teaching of the integrated ability is embedded in the teaching-learning interaction cycle. This permits a reaction to the lack or misuse de any of the sub-abilities, and 4) the advantages of growth based on agents; where all or some of the system agents may be needed to implement the teaching of another ability. All this opens the possibility

of an exhaustive diagnostic of errors in charge of the sub-experts, (Laureano, de Arriaga and Ramirez, 2002; Laureano, El Alami, de Arriaga and Ugena, 2003).

Other characteristics of the aforementioned system are a) a subject following technique is used for tutorial intervention, combined with a trainer type tutor; b) abilities used in an incorrect manner by the student are detected; this is achieved by the mechanism proposed by Burton et al. (1982). In this architecture, the agents represent the different sub -abilities conforming together the ability to be taught. Each agent is divided in two sub -agents, one of them realizing the monitoring (diagnostic) process, which is exclusive and based on the obtained evidence; an input is generated for the next sub-agent, in charge of the tutorial process.

### 6.4.3. Intelligent learning systems with pedagogic agents

Rickel and Johnson (1997) developed an autonomous agent with pedagogic purposes within a virtual learning environment. Virtual environments are useful for training, especially where life is at stake, like in air combat, or in very complex manufacturing processes.

The dominion represented by the environment is related to training of using and repairing complex machinery. The agent in mind is called *Steve* and is capable of verifying and handling dynamic virtual environments. It can also adopt different forms, like a humanoid shape, or that of hands doing pointing tasks.

*Steve* uses a series of intelligent capacities during his interactions with the student and the environment, allowing him to undertake revision or plan execution actions, explanations and a monitoring process on the student development.

A virtual autonomous agent can be invaluable when students do not recognize their actions as inadequate or simply non optimum, in which case a virtual agent may intervene with adequate advice. Some other times they can face unfamiliar situations, and due to the insufficient knowledge to meet the challenge, they could profit from having somebody to guide them, solve their doubts or show them the procedure. Another important aspect is that they can simulate the effect of losing personnel, allowing the student to train in multi-person tasks without needing other humans.

*Steve* dwells in the virtual environment and constantly records the state of the environment; which it controls periodically through virtual actions. Its objective is to help in learning the development of procedural tasks; like the operation and maintenance of complex devices. All of these tutorial skills are integrated in a single agent.

*Steve* consists of two main components: 1) the cognitive one, implemented in SOAR (Laird, Newel and Rosenbloom, 1987), which handles high level cognitive processing, and 2) one that handles the detection engine. The cognitive component interprets the state of the virtual world, carries plans through to accomplish objectives and makes decisions relative to actions. The detection engine component is the interface from *Steve* to the

virtual world, allowing the cognitive component to perceive the state of the virtual world and produce changes in it. In this case, we could consider *Steve*'s cognitive component as an ILS, capable of deciding what to say, when and how to say it.

In the work of Lester and Stone (1997), emphasis is placed on the confidence, based in their credibility, that these pedagogic agents must inspire the student. The latter arrives from two strengths: the visual quality of the agent and the behavior sequences engine that manages them. That management is made on the basis of the evolution of the interactions with the user.

An important point in this implementation is the ability to produce different *ad-hoc* behaviors according to pedagogic motives, and the way the behavior administrator rewards or penalizes them, recording their development and comparing them with *ad-hoc* behaviors for given situations. Once a behavior is chosen, it becomes an autonomous agent, capable of interacting with the student. This work presents an evolution described as follows.

Lester *et. al.* (2001) describe three projects using pedagogic agents that explore different aspects. The first of them describes the role agents can play in the design of a constructivist-centered environment. This approach has received increased attention in recent years, due to its emphasis on the active role the users assume when learning new concepts and procedures. Due to the inherent complexity in learning problem solving, the role the agents play in providing advice and explanations becomes interesting from the point of view of the interaction. Agents in this type of environment must possess contextualizing, continuity and temporality.
In the second project, the critical aspect these agents may exhibit when coordinating gestures, locomotion and velocity in real time is studied, as well as using objects of the environment to get their explanations. Agents of this kind must be able to move in learning environments; pointing to objects and referring to them in a concrete way, in order to provide advice on problem solving. Here are considered the physical properties of the world they live in, as well as a behavior planner according to the current occasion.

The third project examines the role these agents can have in a three dimensional environment, using an AVATAR framework. In this case, the former aspects are brought together, adding two other fundamental elements: 1) an AVATAR and 2) an error handler directly controlled by the AVATAR, consisting of: a) an error detector, b) an error classifier and c) an error corrector. Learning in these environments consists of directly manipulating the objects of the virtual environment trough the AVATAR.

The benefits contributed to the educational environment by using these pedagogic agents are: 1) they take care of student progress, and convince him/her that they are in it together, 2) they are sensitive to student progress, and therefore capable of intervening when the student loses interest or is frustrated, 3) can be emotional and enthuse the user on different levels, similar to a human, and 4) an agent with a rich and interesting personality can simply make learning more fun.

**7. Conclusions**

On the general level, in the development of any system involving artificial intelligence techniques, two main problems arise: 1) emotional and social aspects are not considered, and 2) as a consequence of the former issue, and by its own nature, these systems are working with incomplete information.

We intend to solve some of this problems by the techniques explained throughout this paper, but we are still far from being able to perceive human emotions. Recent research attempts to solve those two problems using cognitive psychology tools and methods to deal with incomplete information, such as fuzzy sets, or reactive philosophy itself.

For instance, to accomplish evaluation within ILSs, up to now, only concrete aspects, product of the teaching-learning interaction are borne in mind. However, we must insist that the techniques that permit us to implement the evaluation in ILSs come from fields like education and cognitive psychology.

Most of the time a teaching-learning plan is developed, we neglect some aspects, like the level of competence with which the student comes to learn a given dominion, and even important aspects such as the motivational, social and emotional ones. In disregarding aspects such as these, we are invalidating the actual evaluation of the teaching-learning plan devised, by taking into account only cognitive aspects of the dominion.

If users are not motivated or have emotional problems, they will fail, even if we offer them the best of teaching-learning plans to them. It would be very interesting to introduce an evaluation of affective, motivational and social skills in systems of this type (Castañeda *et al.*, 1999), beside cognitive ones, to improve the learning capabilities.

On the other hand, we find it advisable to make clear that developments produced in Artificial Intelligence techniques would allow us to tackle a good part of the problems mentioned, if only we had available the adequate operating models from cognitive psychology and educational sciences. It is therefore necessary to include as much as possible the findings of these sciences, and integrate them into the developments of computer science and software engineering.

The authors are currently working in the development and implementation of a ITS that will incorporate a type of evaluation based on fuzzy sets (Laureano et al. 2002). It will furthermore include affective, motivational and social aspects, taking into account the integral evaluation and instruction model proposed by Castañeda *et al.* (1999). This model intends to develop three skills: 1) the knowledge of a domain, 2) a proficiency in academic-cognitive skills, and 3) the self determination of learners the former framed in a MA architecture designed for an ITS (Laureano et al., 2000; Laureano et al., 1998).

**References**

Alonso, C. and D. Gallego. 1994. *Los Estilos de Aprendizaje*. Ediciones Mensajero. Bilbao (Spain).

Anderson, J. R. and B.J. Reiser. 1985. The Lisp Tutor. *BYTE*, Vol.10, pp. 159-175.

Anderson, J. 1988. The Expert Module (Chapter II). In *Foundations of Intelligent Tutoring Systems*. Eds. Martha C. Polson and J. Jeffrey Richardson Lea. Hove & London.

Barr, A., M. Beard and R.C. Atkinson. 1976. The Computer as a Tutorial Laboratory: The Stanford BIP Project. *International Journal of Man Machine Studies*. Vol. 8, pp. 567-596.

Beer, R. D. 1990. *Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology*. Academic Press, Inc.

Brown, J. S., R.R. Burton and A. G. Bell. 1975. SOPHIE: a Step Towards a Reactive Learning Environment. Int. *J. Man Machine Studies*. Vol. 7, pp. 675-696.

Brown, J. S., R.R. Burton and J. de Kleer. 1982. Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In D. Sleeman & J.S. Brown (Eds.). *Intelligent Tutoring Systems*, pp. 27-282. New York: Academic Press.

Brooks, R. $1991_a$ Intelligence Without Representation. *Artificial Intelligence*, No. 47. pp. 139-159.

Brooks, R. $1991_b$ .Intelligence Without Reason. *Memo no.1293*, MIT Artificial Intelligence Laboratory. April.

Burton, R. and J. Brown. 1982. An Investigation of Computer Coaching for Informal Learning Activities. In D. Sleeman & J. S. Brown (Eds.) *Intelligent Tutoring Systems*, pp. 79-98. New York Academic Press.

Burton, R. 1988. The Environment Module of Intelligent Tutoring Systems (Chapter V). In *Foundations of Intelligent Tutoring Systems*. Eds. Martha C. Polson and J. Jeffrey Richardson. Lea. Hove & London.

Canut, M., Gouarderes, F., y Sanchis, E. 1999. The Systemion: A New Agent Model to Design Intelligent Tutoring System. *Artificial Intelligence in Education*, pp 54 - 63. Eds. S.P. Lajoie and M. Vivet. IOS Press.

Castañeda, S. 1993. Procesos Cognitivos y Educación Médica. Facultad de Medicina-UNAM. *Serie Seminarios No.1*

Castañeda, S. 1999. Enseñanza y aprendizaje estratégicos. Modelo integral de evaluación e instrucción. In *Revista Latina de Pensamiento y Lenguaje*, pp. 251-278. Monographic Issue 2B, from Volume 4. ISSN: 0188-9842. Issue dedicated to the diffusion of technological advances on Cognition, Education and Evaluation.

Carbonell, J. 1970. AI in CAI: An Artificial Intelligence Approach to Computer Assisted Instruction. *IEEE transactions on Man-Machine Systems*, MMS. 11 (4) pp. 190-202, December.

Clancy, W.J. 1982. GUIDON. In the *Handbook of Artificial Intelligence*. Ed. A. Barr, E.A. Fegenbaum, Los Altos, California. pp. 267-278.

de Arriaga F., El Alami M. 2002. El Aprendizaje Centrado en el Estudiante: Aprendizaje por Problemas, in *Psicología Educativa*, ed: Sandra Castañeda, México.

El Alami M., de Arriaga F., Ugena A. 1999. SIMUL: An Intelligent Simulation Environment for Decisión Making Learning. in: *Project Studies*, (eds.) H. Olessen, J. Jensen. Roskilde University Press.

El Alami M., de Arriaga F., Ugena A., 2003. "El Aprendizaje Centrado en el Estudiante: Aprendizaje por Proyectos, in: *Psicología Educativa*, ed: Sandra Castañeda, México.

Estévez-Nénninger E.H. 2002. *Enseñar a Aprender: estrategias cognitivas*. Colección Maestros y Enseñanza. (Ed.) Piados. México-Buenos Aires-Barcelona.

Ferber,J. and A. Drogoul. 1992. Using Reactive Multi-Agent Systems Simulation and Problem Solving. In *Distributed Artificial Intlligence Theory and Praxis.– Computer and Information Science*, Vol. 5. Eds. Nicholas M. Avouris and Les Gasser, Kluwmer Academic Publishers.

Fernández, I. 1989. *Estrategias de Enseñanza en un Sistema Inteligente de Enseñanza Asistida por Ordenador*. Ph. D. Thesis (Third Cycle) Universidad del País Vasco, San Sebastián (Spain).

Fletcher, B. and Harris, S. 1996. Development of a Virtual Environment Based Training System for ROV Pilots. IN *Proceedings 'Oceans'96 MTS/IEEE*.

García-Alegre, M., A. Ribeiro, J. Gasos and J. Salido. 1993. Optimization of fuzzy behavior-based robots navigation in partially known industrial environments. (Eds.) Reza Langari, John Yen and John Painter. *Proceedings of the Third International Conference on Industrial Fuzzy Control & Intelligent Systems. IFIS '93*, Houston, Texas. USA.

Genesereth, M. 1978. *An Automated User Consultant for MACSYMA*. Ph. D. Thesis, Harvard University.

Girard, J., G. Gauthier and S. Levesque. 1992. Une architecture multiagent. In *Proceedings of the Second International Conference, ITS'92*. Lecture Notes in Computer Science No. 608. Springer-Verlag, pp. 172- 182.

Gott, S.P. 1989. Apprenticeship instruction for real - world tasks: The coordination of procedures, mental models and strategies. In E.Z. Rothkopf (Ed.), In *Research in Education*, Vol. 15, (pp. 97-169). Washington, D.C. American Educational Research Association.

Halff, H. 1988. Curriculum and Instruction in Automated Tutors (Chapter IV). In *Foundations of Intelligent Tutoring Systems.* Eds. Martha C. Polson and J. Jeffrey Richardson. Lea. Hove & London.

Hayes-Roth, B. 1997. Introduction. *Proceedings of the First International Conference on Autonomous Agents*, pp. 5-8. ACM Press. Marina del Rey, California. February.

Laureano, A. 1993. Multimedios y Cognición. *Perfiles Educativos*. No. 62, pp. 38-41. Centro de Investigaciones y Servicios Educativos. Universidad Nacional Autónoma de México.1993.

Laureano, A. 1995. Herramientas para la Representación del Conocimiento en Sistemas con Inteligencia Artificial. *EN LINEA* Universidad Autónoma Metrolitana. Vol. 1 No. 5, pp. 14-22. Mayo. Azcapotzalco (Mexico City).

Keeling, H. 1999. A Metohodology for Building Intelligent Educational Agents. *Artificial Intelligence in Education*, pp 46 - 53. Eds. S.P. Lajoie and M. Vivet. IOS Press.

Laureano, A. and F. de Arriaga. 1997. Un Sistema Multi Agente Reactivo en los Sistemas de Enseñanza Inteligentes. *Informática y Automática*. Vol.30. No. 4, pp. 71-84. (Spain). 1997.

Laureano, A. 1998. Los Sistemas Reactivos: Un Nuevo Acercamiento a la Inteligencia Artificial Distribuida. *NOVATICA*. No. 132, pp. 51-55. 1998.

Laureano, A. and F. de Arriaga. 2002. A Reactive Multi-Agent Intelligent Tutoring System. In *Proceedings XV Congreso Nacional y I Congreso Internacional de Informática y Computación de la ANIEI*, Vol I, pp.149-160. Guadalajara, Jalisco, MEXICO. Octubre

Laureano, A. and F. de Arriaga. 1998. Multi-Agent Architecture for Intelligent Tutoring Systems. In *Interactive Learning Environments*. Vol.6, No. 3, pp. 225-250.

Laureano, A. and F. de Arriaga. 1999. El Análisis Cognitivo de Tareas: Una herramienta para modelar la conducta de los Sistemas de Enseñanza Inteligentes. *Revista Latina de Pensamiento y Lenguaje*, pp. 315-335. Monographic Issue 2B, from Volume 4. ISSN: 0188-9842. Issue dedicated to the diffusion of technological advances on Cognition, Education and Evaluation.

Laureano, A., S. Arriaga and M. Matínez. 1999. *Paquete computacional Makatsiná: sistema tutorial inteligente para la resolución de estructuras triangulares por el método de los nodos*. Research Report No. 423. Departamento de Sistemas - Universidad Autónoma Metropolitana – Azcapotzalco (Mexico City).

Laureano, A. and F. de Arriaga. 2000. Reactive Agent Design for Intelligent Tutoring Systems. In *Cybernetics and Systems (an International Journal).* Vol. 31, pp. 1-47. ISSN: 0196-9722. (Ed). Taylor & Francis.

Laureano, A. 2000. *Interacción Dinámica en Sistemas de Enseñanza Inteligentes.* Ph. D. Thesis, Instituto de Investigaciones Biomédicas, Universidad Nacional Autónoma de México. (Mexico City).

Laureano, A., F. de Arriaga and M. García-Alegre. 2001. Cognitive Task Analysis: a proposal to model reactive behavior. In *Journal of Experimental & Theoretical Artificial Intelligence.* 13 (2001) pp. 227-239. ISSN: 0952-13X.

Laureano, A., F. de Arriaga and J. Ramírez. 2002. Los agentes reactivos y la lógica borrosa: herramientas para modelar el entorno de los sistemas de enseñanza inteligentes. In Proceedings of the *Conferencia Iberoamericana en Sistemas, Cibernética e Informática.* Invited session: 'Intelligent Tutoring Systems for Training). Volume I, pp. 356-361.Orlando, Florida, July 19-21.

Les, J., Cumming, G., y Finch S. 1999. Agent systems for diversity in human learning. *Artificial Intelligence in Education*, pp. 13 -20 Eds. S.P. Lajoie and M. Vivet. IOS Press.

Lester, J., Callaway, Ch., Grégoire, J., Stelling, G., Towns, S., y Zettlemoyer, L. 2001. Animated Pedagogical Agents in Knowdledge-Based Learning Environments. *Smart Machines in Education*. The MIT Press.

Lester, J.C. y B.A. Stone. 1997. Increasing Believability in Animated Pedagogical Agents. *Memorias Autonomous Agents 97*, pp. 16-21. Marina del Rey California USA. ISBN: 0-89791-877-0/97/02.

Maes, P. 1991. Situated Agents can have goals. In *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pp. 49-70. MIT Press, Cambridge, Massachussets. USA.

Maes, P., 1993, Situated Agents Can Have Goals. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pp. 49-70. The MIT Press Cambidge, Massachusetts London, England.

Néhémie, P. 1992. A systemic approach for student modelling in a multi-agent aided learning environment. Une architecture multiagent. *Proceedings Second International Conference, ITS'92. Lectures Notes in Computer Science No. 608*. Springer-Verlag, pp. 475-482.

Nwana, H. 1996. Software Agents: an overview. *The Knowledge Engineering Review*, Vol. 11:3, pp. 205-244.

Peachy, D. and G. McCalla. 1986. Using planning techniques in intelligent tutoring systems. In *International Journal of Man-Machine Studies*, Vol. 24, pp. 77-98.

Redding, R. E. 1992. *A standard procedure for conducting cognitive task analysis. ERIC.* Documentation Reproduction Service .No. DE 340-847.

Rickel, J. y L. Johnson. 1997. Integrating Pedagogical Capabilities in a Virtual Environment Agent. *Autonomous Agents 97*, pp. 30-38. Marina del Rey California USA. ISBN: 0-89791-877-0/97/02.

Laird, J.E., Newel, A. and Rosenbloom, P.S. 1987. SOAR: an architecture for general intelligence. *Artificial Intelligence* 33(1):1-64.

Ryder, J. M. and R. Redding. 1993. Integrating Cognitive Task Analysis into Instructional Systems Development. *Educ.* 75-96.

Sleeman, D. and J.S. Brown. 1982. *Intelligent Tutoring Systems: An Overview*. Academic Press, pp. 1-11.

Torra, C. 1993. From Geometric Motion Planning to Neural Motor Control in Robotics. *AICOM*. Vol 6, No. 1, pp.3-17. March.

Urretavizcaya, M. 2001. Sistemas Inteligentes en el ámbito de la Educación. Revista *Inteligencia Artificial*. No. 12 (2001), pp. 5 -12. ESPAÑA.

Vassileva, J., Geer, J., McCalla, G., y Deters, R. 1999. A Multi-Agent Design of a Peer-Help Environment. *Artificial Intelligence in Education*, pp 38 - 45. Eds. S.P. Lajoie and M. Vivet. IOS Press.

Van Lhen, K. 1998. Student Modeling (Chapter III***)***. In *Foundations of Intelligent Tutoring Systems*. Eds. Martha C. Polson and J. Jeffrey Richardson Lea. Hove & London.

Veitl, M., P. Petta, R. Spour and K. Obermaier. 1999. Autonomous agents in user interfaces. In *Cybernetics and Systems: An International Journal*, 30: 169-177.

Wenger, E. 1987. *Artificial intelligence and tutoring systems: Computational approaches to the communication of knowledge*. Los Altos, CA: Morgan Kaufmann.

Wilensky, R. 1983. *Planning and Understanding: A computational approach to human reasoning*. Addison Wesley.

Woodridge, M., and N. Jennings. 1995.Intelligent Agents: theory and practice. *The Knowledge Engineering Review*, 10(2), pp. 115-152.

Woolf B. and D. McDonald. 1984. Building a Computer Tutor: Design Issues. *Computer IEEE*, pp. 61-73, September.