Proceedings of the Second
IASTED International Conference

IASTED

Artificial Intelligence and Applications

Editor: M.H. Hamza

September 9-12, 2002
Málaga, Spain

A Publication of The International Association of Science and Technology for Development - IASTED

ACTA Press
Anaheim • Calgary • Zurich

ISBN: 0-88986-352-0
ISSN: 1482-7913

September 9-12, 2002, Malaga, Spain

INTELLIGENT TUTORING SYSTEM FOR MATHEMATICS USING AN EVOLUTIONARY STUDENT MODEL

Sergio A. Rojas Johanna Ramirez Oscar Romero

Adaptation, Computing and Mind (ACME) Group
Faculty of Engineering
Universidad Distrital Francisco José de Caldas
Bogotá, Colombia
srojas@udistrital.edu.co, acme@atenea.udistrital.edu.co

Abstract

Intelligent Tutoring System usually identifies the cognitive structures of its learners to classify them as acceptable, erroneous or missing, and to determine the proper action for instruction. However, with just those structures the system can not typify the student's learning style which could be useful to customize the tuition according with his talents and preferences. In this paper we propose a tutoring system that builds a student model for discovering and classifying cognitive styles based in an evolutionary approach. The model has been applied in basic arithmetic teaching for primary school learners.

Key Words: Intelligent tutoring systems, student modeling.

1. Introduction

Traditional education in the classrooms have some kind of collective fashion, that is, a teacher must instruct a large group of students giving general explanations without considering individual needs nor expectations. Furthermore, the lack of time for lesson customization and the conflicts between teacher and student's styles of learning turn out less effective the education process.

Intelligent Tutoring Systems (ITS) have appeared as an attempt to solve this problem. They imitate the teacher behavior detecting specific obstacles for each student and presenting suitable information to help him. For this intent, the system must model the student cognitive structures, either if they are right or erroneous, finding not only the what (what have to be adjusted) but the how (how to present the contents for the learning to be acceptable).

In this paper we present an evolutionary approach for student modeling that adapts to the student cognitive style developing a representation of the student's learning behavior (talents and preferences such as conceptualization and application level, short memory performance, favorite didactic material). We have tested the model taking as reference a pilot group of children between 9 and 11 years old in 4th course of primary school.

Next section explains ITS basics, section 3 talk about cognitive styles, section 4 gives a detailed explanation of the evolutionary student modeling approach, experimental results are presented in section 5, and finally we outline some conclusions.

2. Tutoring Systems

The ITS main components are [1]:

- *Tutorial module*: this module contains fundamental strategies to teach lessons contents. Additionally, provides didactics to teach each topic according to curriculum.
- *Expert module*: represents the teacher's domain knowledge. This module provides information about different mechanisms for helping the student, for example showing either the right solution or the error explanation, or showing a counterexample for disproving of the current solution.
- *Student module*: this module represents the teacher information about student knowledge state and his typical method of working (student learner-type or his preferences). This information provides the fundamentals for decisions the tutorial module has to make during the learning and teaching process.

2.1. Tutoring module

In our experiments we have designed an ITS for basic mathematics, that is arithmetic operations such as addition, subtraction, and multiplication. We have developed a tutorial module based on an outer space story that guides the child over a set of worlds in which he must try out different adventures for resolving arithmetic problems as he was the hero. The tutorial includes several hypermedia pages with animations and interactive questions for the student.

2.2. Expert module

For this module we used a production rule based expert system in which rules represents knowledge about student behavior [2]. First we represented knowledge about arithmetic problem solving in a set of Generalized And Or Graph as it helped us to determinate the objectives and conclusions of a task. After that, we represented the knowledge in production rules.

When the student comes up with a wrong answer, the tutoring system attempts to simulate the student's behavior to see how he could gone wrong and therefore offering advice, which is to some extent the result of a reasoning process similar to that of the student [3].

For dealing with uncertainty we use an intuitive mechanism, the triangular rules method [4]. This method assigns a certainty factor to each configuration (set of rules that represent a possible answer) and later applies inference rules to resolve the uncertainty.

2.3. Student module

This module has a diagnostic component for discovering the student model. For extracting information we follow two methods: *model tracing* (following of the observable behavior) and *reconstruction* (requires the inference of non-observable actions) [3].

In our ITS, the student model consists of a set of production rules that reflects his domain knowledge. The rules could be coarse-grained or fine-grained. When student's actions are matched with a set of rules, it reflects a model tracing approach. In the model tracing mode, the system allows the tutor to solve the exercise, step-by-step, along with the student.

The student model architecture we used is called the *Perturbation* or *Buggy Student Model* [6] which is depicted in figure 1. This model seeks out for: (i) knowledge possessed by the student that is not present in the expert knowledge; and (ii) incorrect information (misconceptions).

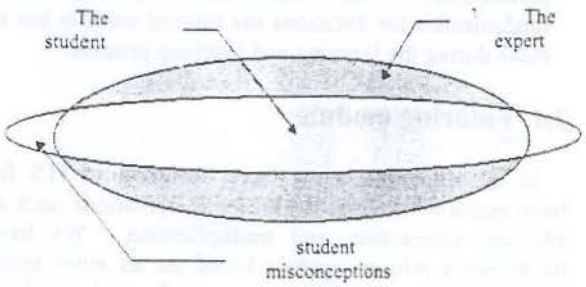

Figure 1. The buggy student model

The buggy student model extends the knowledge of the expert system by means of a bug library. We use a generative bug library that implements a learning tree. Each time the student follows a wrong procedure, the bug is searched in the tree through the boughs; if it isn't found, the bug is inserted in one leaf of the tree.

Traditional student models just represent student specific mental process used to resolve a determined task. Our research is focused in modeling the cognitive style for every student, so the individualization will be more adaptive to his needs. Our approach will be detailed described in section 4.

3. Cognitive and learning styles

A learning style is comprised of cognitive, affective and physiological features that determine how the learner perceives, acts, and responds to the learning environment. It includes the cognitive style that represents the typical way of thinking, remembering and resolving problems of a person [7].

Based on research work of Witkin [8] and Kolb [9] we have defined a 3-dimentional space to locating eight cognitive styles for arithmetic understanding (fig. 2).

Kolb proposed a theory of experimental learning that involves four principal stages: concrete experiences (CE), reflective observation (RO), abstract conceptualization (AC), and active experimentation (AE). He suggested four types of learners (divergent, assimilator, convergent, and accommodator) depending upon their position on these two dimensions. Similarly, Witkin proposed two styles for learning: analytic and holistic [10].

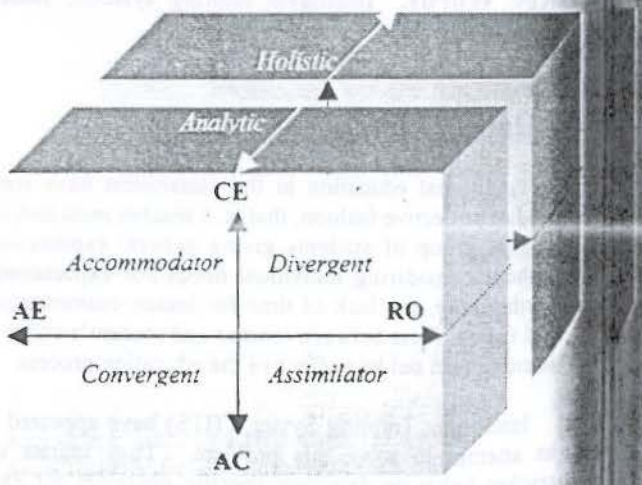

Figure 2. Spatial representation for learning and learner styles

Hence we have the following cognitive styles:

- *Accomodator Student* (AS): he needs to know what things are made for, he have an intuitive style to resolve problems, and primarily he wants to adapt his learning to situations of his own life.
- *Convergent Student* (CS): his better resource resides in the practical application of his ideas, he has to know how the things work and learns theory by contrasting with sensation experimentation.

- *Divergent Student* (DS): he perceives information through a concrete testing, he tends to be very imaginative and emotive; he learns listening to ideas and sharing them with others.

- *Assimilator Student* (SS): he wants to know what experts think about, he perceives abstract information and process it in a reflexive way; he shows interest in abstract concepts.

- *Analytic Student* (ANS): he is an unbiased and a very reflexive student, he is inclined to developing of right solution strategies; he likes visual material.

- *Holistic Student* (HS): he is a very impulsive learner who has a good short memory performance; he likes to participate in groups of people and show preferences for acoustic material.

For student customization, the tutoring module of our ITS has different sets of hypermedia material arranged accordingly with the styles showed before.

4. An evolutionary approach for student modeling

Using a reinforcement machine learning technique known as Classifier Systems which is based on a Genetic Algorithm (GA) [12], we proposed an approach for adaptive student modeling. Next we describe briefly the classifier system, and then we explain how it has been designed and applied.

4.1. Learning Classifier Systems (LCS)

Classifier systems were proposed by Holland [11] as an evolutionary technique for machine learning. Their architecture is showed in figure 3. They have three subsystems: rule and message system, credit distribution system and genetic algorithm.

The rule and message system is a special kind of production rule system with the following rule syntax:

`<classifier>::=<condition>:<message>`

where `<condition>` and `<message>` are strings of a ternary alphabet {0, 1, #} whit '#' is a wildcard that matches with a 0 or 1. The `<condition>` portion of the classifiers corresponds to the sensors of the system. Similarly, the `<message>` portion is the actuator that can activate internal or external actions.

The credit distribution system is intended to distribute the reward received by the system from the environment because its actions. Since the whole system has a pool of classifiers for attending to different stimuli, there must be an apportionment of credit system to distribute the total system earnings between all classifiers. It is an algorithm that simulates an economy where the privilege to trade information is bought and sold by classifiers. This service economy contains two main components: an auction (in which each classifier bets for sending a message) and a clearinghouse (in which the house pays to those good acting classifiers).

The genetic algorithm works searching for new classifiers that adapts better to the environment, using an evolutionary algorithm that follows genetic operators (selection, crossover, mutation) over a number of epochs. The fitness measure for each classifier is related with the gains or strength it can obtain over its life.

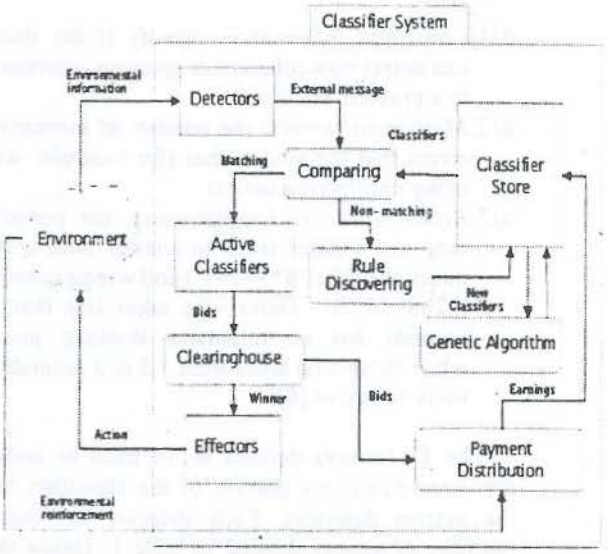

Figure 3. LCS basic architecture

4.2. Classifier coding scheme for adaptive student model

We have designed an LCS in the student module of our ITS for identification of the student cognitive style. Depending on it the system will be able to design a pedagogical strategy for each individual and will report a student diagnosis to the teacher with respect to a specific task; thus the customization level of teaching is more adaptive.

The LCS captures some features of the student cognitive style through a diagnosis evaluation using the detectors. This detection is based on the following aspects:

d1: *Exercising level*: the percentage of finalized exercises the student made in each session.
d2: *Real circumstances*: obtained results when the student resolve problems situated in a real life context.
d3: *Help employment*: the percentage of help topics demand versus total available.
d4: *Procedure following*: specify if the student has followed the properly strategy to solve an arithmetic problem.
d5: *Visiting level*: the percentage of hypertext nodes the student visits additionally to the tutoring classes.

d6: *Tools used*: stores the number of tools the student has used.

d7: *Conceptualization level*: the percentage of conceptual questions right-answered.

d8: *Navigation style*: specify the student preference for supervised or self-organized contents instruction.

d9: *Short memory performance*: captures the persistence or weakness of volatile memory.

d10: *Interference level*: indicates if the student correctly identifies forms in the presence of distracting elements, using and embedded figure test.

d11: *Consistent information*: specify if the student can detect non-coherent or spurious information in a problem statement.

d12: *Mechanical errors*: the number of memorizing errors that the student has (for example, while using multiplying tables).

d13: *Reflexiveness vs Impulsiveness*: the period of time the student takes to answer both correct questions (like "8*3=24?") and wrong questions ("7-8=16?"). Those who takes less than 1.3 seconds has an impulsive thinking process while those who takes from 1.3 to 3 seconds are more reflexive [8].

The 13 features defined above must be codified in the `<condition>` portion of the classifier; they are the system detectors. Each detector has the length (number of genes) showed in table 1. Hence the total length of classifier's `<condition>` is 41 bits.

Table 1. Condition coding scheme for a classifier

Detector	Bits	Description
d1	4	codes a percentage from 1 to 10 (binary 0000 to 1010)
d2	4	a percentage
d3	4	a percentage
d4	4	codes the number of error per problem, from 1 to 10.
d5	4	a percentage
d6	4	codes the number of tools used. The student up to 15 tools available (binary 0000 to 1111)
d7	4	a percentage
d8	1	it is a Boolean value
d9	1	a Boolean value
d10	1	a Boolean value
d11	4	a percentage
d12	4	a percentage
d13	2	two Boolean values
Total	41	total length of chromosome

Similarly, each classifier indicates an action to perform in its `<message>` portion. For the ITS we are interested in knowing the student cognitive style. Thus the message portion can be a 3-lenght bit string as showed in table 2.

Table 2. Message coding scheme for a classifier

Cognitive style	Code
convergent analytic student	000
convergent holistic student	001
divergent analytic student	010
divergent holistic student	011
accommodator analytic student	100
accommodator holistic student	101
assimilator analytic student	110
assimilator holistic student	111

4.3. Classifier strength in the ITS

Holland proposed the bucket brigade algorithm as the apportionment of credit method in the LCS. The main idea is that each classifier maintains a gain quantity (strength) that evaluates its performance in comparison with the rest of classifier population. The strength is computed as showed in (1).

$$S(t+1)=S(t)-B_i S(t)-Ct_{tax}S(t)+R(t)-Tax_{bid}B_i S(t) \quad (1)$$

where:

$S(t+1)$	strength in time $t+1$.
$S(t)$	currently classifier strength.
$B_i\ S(t)$	$=(C_{bid}+BidRatio)S(t)+N$; the bet amount.
C_{bid}	bet rate, a value between 0 and 1.
BidRatio	classifier specificity level. The more wildcards the classifier has in his `<condition>`, the more general it is.
N	normally Gaussian noise.
C_{tax}	tax rate for those classifiers who never take part of an auction. A value among 0 and 1.
R(t)	reinforcement signal given by the environment because the system action.
Tax_{bid}	a tax for those classifiers who always bet but never win.

We implement the bucket brigade algorithm using the parameters showed in table 3.

Table 3. LCS parameters used in the ITS

Variable	Value	Meaning
C_{bid}	0.1	10% of the classifier strength
BidRatio	0-82	For each matching gene adds 2 points to BidRatio
S_i	500	Initial strength for all classifiers
n	1000	Population size
C_{tax}	0.034	Accordingly with median classifier lifetime (20 generations)
Tax_{bid}	0.001	1/n
e	20	Frequency iterations for GA execution
Rp	10%	Offspring rate
Pm	50%	Mutation prob. in the GA
Pc	100%	Crossover prob. in the GA

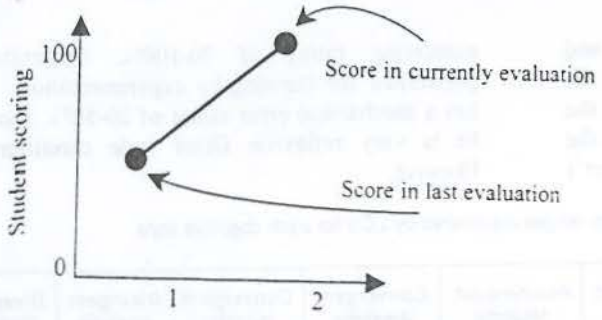

Figure 4. LCS reinforcement based on student scoring

4.4. LCS reinforcement mechanism

The reinforcement received by the LCS from environment is related with the academic scoring of the student in different time intervals. The expected behavior is that of the student growing its rating as he interacts with the ITS which is each time more customized. If this happened we can conclude the LCS is making a good classification (it must receive a positive reinforcement), and thus the ITS are presenting material according to student's learning style facilitating his learning process. If not, the LCS is wrong (it must receive a negative reinforcement), and the ITS must correct its strategy.

Therefore we designed our reinforcement measure as a function depending on the slope of a straight line drew between the last evaluation point and the current student evaluation (see fig. 4). The reinforcement signal in time t is given by (2):

$$R(t)=200*m(F_p(t))+100 \qquad (2)$$

where $m(F_p(t))$ is the slope of the scoring function at time t. If this slope is negative, it means that the learner rating is going down, so the classifiers are doing a bad identification of his cognitive style; hence, they must be penalized with a negative gain. In the other hand if the slope is positive, the students' rating is rising, so the classifiers are doing a good job adapting didactic contents and their strengths must be rewarded with positive gains. Finally, if the slope is zero (the scoring remains constant) the system receives just a little reward to encourage better adaptation.

5. Experimental results

In order to find out LCS performance into the diagnostic component of the ITS student module, first we test it in a pilot group with 20 students. Figure 5 shows the results. The curve is the average of the number of times for LCS activation (which comprise sensing data and action triggering). A learning period consists of an experimental session with 8 students for each cognitive style until it is correctly identified. For gathering data, each student presents a quiz with 20 scenarios that represents 13 learning features. So in order to obtain a perfect classification, the LCS requires 20 activations (each one matching a single question). It can be seen how this number decreases with the number of times the students interact with the ITS.

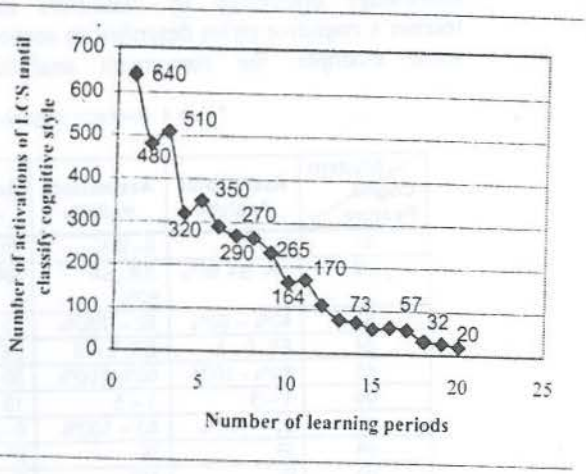

Figure 5. Performance measure of the LCS

The next step was to apply the trained LCS to the whole group (178 individuals). The aim of this stage was to giving the LCS with more diverse reinforcement signals so it can enhance the acquired knowledge. After two weeks of interactions with students, we measure the strength distribution across classifiers pool. We found the results showed in figure 6.

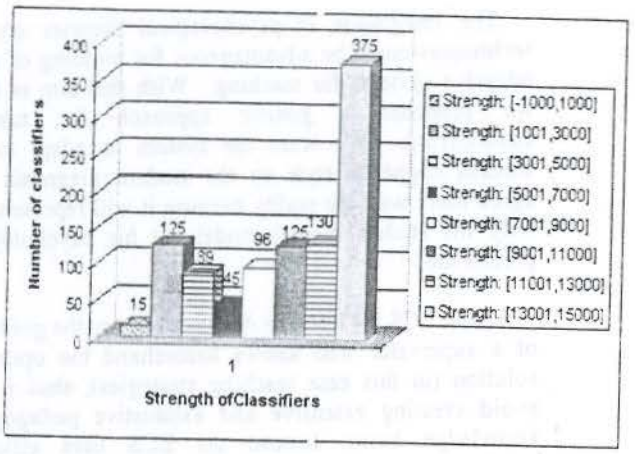

Figure 6. Strength distribution after 15,000 iterations (2 weeks)

It can be seen how the bigger amount of classifiers (375) has the highest gains (16,000). These are the classifiers who best identify the cognitive style of the student. Besides, the most used classifiers (those with strength between 7,500 and 16,000) represent almost 50% of total population. Also the worst classifiers represent only a small fraction (1.5%) of the population; this is due to the GA action that encourages good classifiers and punish up bad ones.

Then we decoded some of the best classifiers to find out the level of knowledge they achieved for the eight cognitive styles we have determined. Results are shown in table 4. A classifier representing, for example, a *Convergent analytic student*, will show the following information: exercising level between 70% and 100%; help employment of 30–60%; conceptualization level

over 70%; good short memory performance; and mechanical errors level of 20-50%. This means that the knowledge discovered by classifiers matches the learner's cognitive styles described in section 3. In the same example, the convergent analytic learner's exercising rating of 70-100%, demonstrates his preference for learning by experimentation. Also, he has a mechanical error rating of 20-50%, showing that he is very reflexive. Other style classifiers behave likewise.

Table 4. Average cognitive features ranges discovered by LCS for each cognitive style

Student / Cognit. Feature	Assimilator Analytic	Assimilator Holistic	Accomodat Analytic	Acommodat. Holistic	Convergent Analytic	Convergent Holistic	Divergent Analytic	Divergent Holistic
d1	0–30%	0–30%	70–100%	70–100%	70–100%	70–100%	30–70%	30–70%
d2	SR, SA 80%	SR, SA 80%	SR 80%	SR 80%	SR, SA 80%	SR, SA 70%	SR 60%	SR 60%
d3	40% – 60%	60 – 100%	20 – 30%	20 – 50%	30 – 60%	40 – 60%	40 – 60%	40 – 60%
d4	EE: 0 –3	EE: 3 – 6	EE: 0 –2	EE: 4–5	EE: 0 - 1	EE: 2	EE: 0- 2	EE: 4 – 6
d5	60% - 100%	60%-100%	30 – 60%	30 – 60%	70 – 100%	70 – 90%	30 – 60%	30 – 60%
d6	1 – 5	1 – 5	10 – 15	10 – 15%	5 – 12	5 – 10	1 – 7	1 – 7
d7	80 – 100%	80 – 100%	0 – 30%	0 –30%	70 – 90%	70 – 90%	50 – 80%	50 – 80%
d8	IP	IS	IP	IS	IP	IS	IP	IS
d9	R	NR	R	NR	R	NR	R	NR
d10	DI	ND	DI	ND	DI	ND	DI	ND
d11	60 – 90%	30 – 60%	70- 100%	0 – 45%	60 – 80%	60 – 90%	50 – 90%	20 – 60%
d12	30 – 50%	0 – 20%	20 – 30%	0 – 20%	20 – 50%	20 – 30%	20 – 40%	0 – 30%
d13	EC: +1 s. EE: +1 s.	EC: -1 s. EE: ±1 s.	EC: +1 s. EE: +1 s.	EC: +1 s. EE: +1 s.	EC: -1 s. EE: ±1 s.	EC: +1 s. EE: +1 s.	EC: -1 s. EE: ±1 s.	EC: +1 s. EE: +1 s.

Conventions:
SR=Real situation; **SA**=Abstract situation; **EE**=Error per exercise; **IP**=Customized instruction; **IS**=Sequential instruction; **R**=good short memory; **NR**=bad short memory; **DI**=detects inconsistencies; **ND**=no detects inconsistencies; **EC**=correct schemes; **EE**=wrong schemes.

6. Conclusions

The integration of psychological theories and AI techniques could be advantageous for building of more adaptive systems for teaching. With this aim in mind we proposed a genetic approach for tutoring customizing. We want the system to adapt to the student cognitive style so the student diagnosis will agree more with the reality because it will represent not only the student mental model but his psychological profile too.

Using LCS for learning does not require the guidance of a supervisor who knows beforehand the optimum solution (in this case teaching strategies); thus it can avoid creating extensive and exhaustive pedagogical knowledge bases. Instead the LCS uses efficient reinforcement learning, behaving like an autonomous agent able to experiment and take decisions in arbitrary environments, adapting itself with an evolutionary strategy.

In our future work we expect to continue working in the field of student modeling using AI techniques like LCS; we are interested in using other approaches for adjusting the model to more appropriate values; for example, some variables like *conceptualization level* could be evaluated in a real scale, not in a crisp one (binary, yes or no). Hence, perhaps a Fuzzy Classifier Systems could improve the adaptability performance of the system.

7. Acknowledgements

This research project has been partially supported by the Research Center of Universidad Distrital.

8. References

[1] J.R. Carbonell, *AI in CAI: An Artificial Intelligence Approach to Computer – Assisted Instruction* (NJ: Addison-Wesley, 1970).
[2] E. Rich, *Artificial Intelligence* (MacGraw-Hilll, 1994).
[3] E. Wenger, *Artificial Intelligence and Tutoring Systems. Computational and Cognitive Approaches Communication of Knowledge* (NY: Morgan Kaufmann Publishers, 1990).
[4] J. Self, *Computers in Education* (NY: Prentice-Hall, 1987).
[5] J. Giarratano, *Expert Systems: fundamentals and programming* (NY: Thomson Publishers, 1999).
[6] J.R. Anderson, *Skill Acquisition: compilation of weak method problem solutions* (Carnegie-Mellon University, 1985).
[7] W.J. Clancey, "Situated action: A neuropsychological interpretation response to Vera and Simon", *Cognitive Science*, 1993.
[8] H.A. Witkin and D.R. Goodenough, *Cognitive Styles: Essence and Origins* (NY: International Universities Press, 1981).
[9] D.A. Kolb, *Experiential Learning* (NJ: Prentice-Hall, 1984).
[10] R. Dunn and K. Dunn, *The Complete Guide to the Learning Strategies Inservice System* (Boston: Allyn & Bacon, 1999).
[11] J.H. Holland, *Induction, Processes of Inference, Learning and Discovery* (Mich:Addison-Wesley, 1953).
[12] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Massachusetts: MIT Press, 1986).