International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:12, 2007

Integrating Computational Intelligence Techniques and Assessment Agents in E-Learning Environments

Konstantinos C. Giotopoulos, Christos E. Alexakos, Grigorios N. Beligiannis and Spiridon D. Likothanassis

***Abstract*—**In this contribution an innovative platform is being presented that integrates intelligent agents and evolutionary computation techniques in legacy e-learning environments. It introduces the design and development of a scalable and interoperable integration platform supporting:

I) various assessment agents for e-learning environments,
II) a specific resource retrieval agent for the provision of additional information from Internet sources matching the needs and profile of the specific user and
III) a genetic algorithm designed to extract efficient information (classifying rules) based on the students' answering input data.

The agents are implemented in order to provide intelligent assessment services based on computational intelligence techniques such as Bayesian Networks and Genetic Algorithms.

The proposed Genetic Algorithm (GA) is used in order to extract efficient information (classifying rules) based on the students' answering input data. The idea of using a GA in order to fulfil this difficult task came from the fact that GAs have been widely used in applications including classification of unknown data.

The utilization of new and emerging technologies like web services allows integrating the provided services to any web based legacy e-learning environment.

***Keywords*—**Bayesian Networks, Computational Intelligence techniques, E-learning legacy systems, Service Oriented Integration, Intelligent Agents, Genetic Algorithms.

We thank the European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II), and particularly the Program HERAKLITOS, for funding the above work.

K. C. Giotopoulos is with the Pattern Recognition Laboratory, Department of Computer Engineering & Informatics, University of Patras, Rio 26500, Patras, Greece (corresponding author); phone: 0030-2610-997520; fax: 0030-2610-969001; e-mail: kgiotop@ceid.upatras.gr).

C. E. Alexakos is with the Pattern Recognition Laboratory, Department of Computer Engineering & Informatics, University of Patras (e-mail: alexakos@ceid.upatras.gr).

G. N. Beligiannis is with the Pattern Recognition Laboratory, Department of Computer Engineering & Informatics, University of Patras (e-mail: beligian@ceid.upatras.gr).

S. D. Likothanassis is with the Pattern Recognition Laboratory, Department of Computer Engineering & Informatics, University of Patras (e-mail: likothan@ceid.upatras.gr).

I. Introduction

THE wide adoption of e-learning environments in all levels of human education has leaded scientific research in the field of adaptive and intelligent e-learning systems to provide higher quality services towards the end users of the e-learning systems.

Adaptive and intelligent e-learning systems is more than clear that satisfy the demanding need of e-learning users for personalization. It is widely known that the most efficient products are tailored made to the needs of the client. The same applies in the e-learning market. People are eager to bye products or services that fit exactly to their personal needs and interests.

That is the main reason for the huge scientific effort in all market sectors to introduce features in products or services that satisfy the need for personalization. The same applies to the new and emerging sector of e-learning.

E-learning systems need to introduce the aforementioned features in terms of functionalities in order to meet the market trends and user requirements. This aspect comprises the key aspect for success in all e-learning environments.

Our proposal introduces a scalable and interoperable integration platform supporting:

I) various assessment agents for e-learning environments,
II) a specific resource retrieval agent for the provision of additional information from Internet sources matching the needs and profile of specific users and
III) a genetic algorithm designed to extract efficient information (classifying rules) based on the students' answering input data.

The agents are implemented in order to provide intelligent assessment services based on computational intelligence techniques such as Bayesian Networks and Genetic Algorithms.

The proposed Genetic Algorithm (GA) is used in order to extract efficient information (classifying rules) based on the students' answering input data. The idea of using a GA in order to fulfil this difficult task came from the fact that GAs have been widely used in applications including classification of unknown data.

The utilization of new technologies such as web services allows to integrate the provided services to any web based legacy e-learning environment. A first approach to the integration of computational intelligence techniques in order to assist e-learning process has been presented in [1].

The paper is structured as follows. Section II includes a brief presentation of the related work in the field of adoption of intelligent agents in e-learning systems. In section III the architecture and the functionalities of the proposed agent platform are described. Section IV presents an example of a query assessment agent based on Bayesian Networks, which is integrated in a legacy e-learning system. Section V presents the resource retrieval agent that anticipates the lack of additional information for a specific subject taking into account the special needs and interests of the specific student. Section VI presents the utilization of a Genetic Algorithm in order to extract efficient information (classifying rules) based on the students' answering input data. Finally, section VII summarizes the conclusions and suggests future applications and extensions of the proposed e-learning platform.

II. Related Work

Utilization of agents is significant for providing intelligence in e-learning environments. Thus, a lot of work has been done concerning these fields, mainly focusing on adoption of intelligent agents to integrate e-learning systems. Buraga [2] proposes an agent-oriented extensible framework based on XML family for building a hypermedia e-learning system available on the world-wide web. It deploys mobile agents that can exchange information in a flexible way via XML-based documents. The intelligent tutoring system is composed of four major components. The information processed by each component can be stored by XML documents. Some of the components have been implemented as intelligent agents.

Angehrn et al. [3] suggest the use of K-InCA to provide a personalized e-learning system. K-InCA is an agent-based system designed to assist people in adopting new behaviors. The agents within the system examine users' actions and maintain a "behavioral profile" reflecting the level of adoption of the desired behaviors. Based on the user profile, and relying on a model borrowed from change management theories, the agents provide at different stages customized guidance, including mentoring, motivation or stimulus, in order to support real learning and smooth adoption of new behaviors.

In order to recommend useful learning material to students Andronico et al. [4] proposed a multi-agent recommendation system that suggests educational resources to students into a mobile learning platform that supports mobile learning processes. They have first extended the Learning Management System (LMS) in order to incorporate mobile technologies, allowing users to interact with the systems using mobile devices like PDAs, cellular phones etc. This extension arises the problem of designing new learning models that will be able to adapt to the changes of student's performance during the learning process. Another extension of an LMS presented in that paper regards the integration of a multi-agent recommendation system, whose aim is to collect data about the users' behavior and preferences and then suggest educational resources to them according to their profile.

Zaiane [5] suggests the use of web mining techniques to build such an agent that could recommend on-line learning activities or shortcuts in a course web site based on learners' access history to improve course material navigation as well as assist the online learning process. The automatic recommendation system takes into account profiles of on-line learners, their access history and the collective navigation patterns, and uses simple data mining techniques.

According to the service-oriented integration for e-learning systems based on web services, Pankratius [6] provide a related architecture and describes the extensions to support software agents. It focuses on using intelligent software agents for the distributed retrieval of educational content. In this architecture intelligent software agents can be used to acquire user specifications of learning content and search for matching Learning Objects on behalf of the user. The work utilizes the web services as a wrapper around Learning Objects.

III. Agent Platform

The contribution presented introduces a scalable implementation architecture that is based on an agent platform. This platform is used in order to manage the execution of the various intelligent agents for supporting legacy e-learning systems. In this section, a detailed description of the functionalities and the implementation of the agent platform architecture are presented. Furthermore, the web services technology is utilized in order to provide communication and interoperability between the proposed agent platform and e-learning legacy systems.

A. Platform Architecture

The Agent Platform Architecture that this contribution proposes is based in open and interoperable standards, since main effort has been given in the reusability perspective of the specific platform. The main idea behind the specific architecture is to use a multi-agent platform to design and develop the features (agents) that will provide the added value to legacy e-learning systems. In order to follow the international standards in the implementation of multi-agent systems, the proposed agent platform is developed utilizing Java Agent DEvelopment Framework (JADE) [7]. JADE is a java-based platform that provides the basic mechanism for the implementation of peer-to-peer agent based applications according to the FIPA [8] guidelines for the development of multi-agent systems.

The main concept is to exploit the features of the proposed agent platform and develop agents that will implement the features that are needed. The basic aspect for the architecture is the fact that the agent platform architecture (Fig. 1) communicates with the legacy e-learning system through the use of web services and more specifically by using the SOAP

protocol.

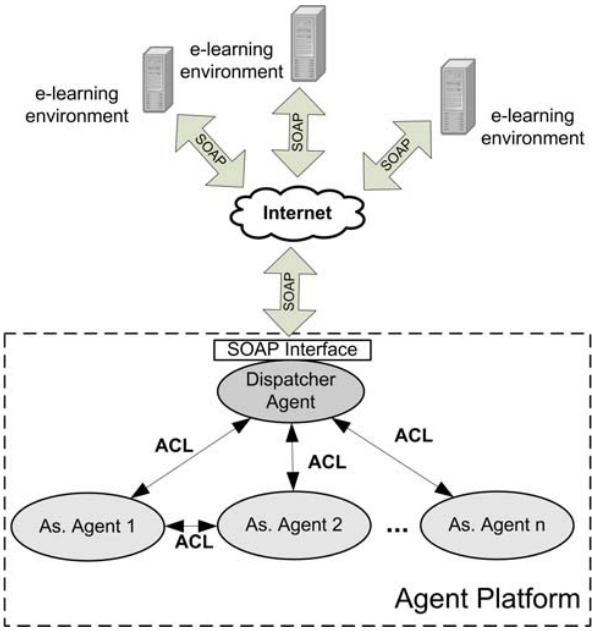

Fig. 1 Agent Platform Architecture

The requests coming from the e-learning system are being processed by the Dispatcher Agent. This specific agent is the front end of the agent platform and is responsible for identifying the requests coming from the e-learning system and for allocating them to the appropriate agent in order to be processed. Each agent is designed to implement functionalities that arise from the user requirements.

The communication within the agent platform is being realized with the ACL (Agent Communication Language) [9] that is part of the FIPA template. It is important to emphasize that the Dispatcher Agent is the most important component of the agent platform. The reason for this is its multiple behavior and usefulness. It serves as process identifier, process distributor and communication handler. Thus, it is a critical agent for the proper operation of the agent platform.

The Dispatcher Agent is "informed" by the agents with the functionalities that they provide. Its main responsibility is to recognize the incoming requests from the e-learning systems through a SOAP Interface and translate them to ACL messages in order to reroute them to the corresponding agent.

The utilization of the JADE framework allows affective communication between different agents. It is possible that an agent will require the execution of functionalities belonging to another agent. In that case, ACL messages are exchanged between the involved agents establishing interoperability features between different agents.

B. Communication with the E-learning Environment

For the interconnection of the agent platform with the legacy e-learning systems Web Services [10] technology is utilized. Web Services are referred as "software applications identified by a Uniform Resource Identifier (URI), whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and support direct interactions with other software applications using XML-based messages via Internet-based protocols". Web services are loosely coupled, communicating through XML based documents. According to the above description, a web service is given in terms of the messages it sends and receives. At the concrete level, a binding specifies transport and wire format details for one or more interfaces. An endpoint associates a network address with a binding. Finally, a service groups together endpoints that implement a common interface.

According to the prospects that are supported by the web services, the main agent platform is enhanced with a SOAP server in order to provide a flexible and standard based service for accessing the agents' capabilities and functions. A client script is also installed in each of the integrated systems in order to invoke the provided web services from agent platform.

The utilization of web services is based on the capability of "easy" integration that is achieved through an intermediate adapter layer that relays commands and data between the web and the system. A big part of the implementation has been realized using technologies such as .NET framework and Common Object Model (COM) for Microsoft based applications, Enterprise JavaBeans (EJB) for java based applications and PHP classes for web based applications.

IV. Questioner Assessment Agent

The use case example presented in this section shows the utilization of the proposed platform for assessment of students in a questioner-based examination process. In order to manage the student's answers given to the questioner, a Questioner Assessment Agent is implemented. This agent uses Bayesian Networks in order to determine the sequence of questions that are given to the student according to his/her possible answers.

A. Bayesian Network for Questioner Optimization

The agent is based on Bayesian Networks' techniques in order to manage the questioners of an e-learning system. Bayesian Networks [11], [12] are compact networks of probabilities that capture the probabilistic relationship between variables, as well as historical information about their relationships. They are very effective in modelling cases where some information is already known and incoming data is uncertain or partially unavailable [13]-[15]. These networks also offer consistent semantics for representing causes and effects (and likelihoods) via an intuitive graphical representation.

According to the presented approach the questions are structured in a decision tree that denotes the relevance between two successive questions. A graphical representation of a questioner used to explain the functionality of our module is depicted in Fig. 2.

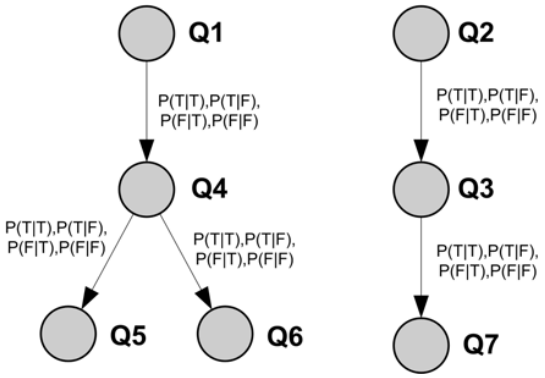

Fig. 2 Graphical Representation of a Questioner

Each of the questions in the Bayesian network is represented by a node. Each node can be in several states. Each state corresponds to a different set of probable values for each variable of the node. In our case, for example, each question is represented by a 2-state variable that can be either true (correct answer) or false (false answer). Nodes are connected to show causality with an arrow (edge) indicating the direction of influence.

The Bayesian network also contains probabilistic relationships among some of the states of the domain. These relationships are used to answer questions like the following: If the student answered correctly the fourth question, was it more likely to have answered correctly the first question or not?

The probability of any node in the Bayesian network being in one state or another without current evidence is described using a conditional probability table. Probabilities on some nodes are affected by the state of other nodes, depending on causality. Prior information about the relationships among nodes may indicate that the likelihood that a node is in a specific state is dependent on the specific state of another node. For example, prior information may show that if the student answered correctly the first question, the likelihood of answering correctly the fourth question is higher. An example of the conditional probabilities for the Bayesian network applied in this work is shown in the following table.

TABLE I
CONDITIONAL PROBABILITIES

Parent (Q1)	Child (Q4)	
	True	False
True	0.8	0.2
False	0.6	0.4

The table shows that the likelihood of giving a correct answer at the fourth question is 0.8 if a correct answer was given at the first question and 0.6 if a false answer was given at the first question respectively. Similarly the likelihood of giving a false answer at the fourth question is 0.2 if a correct answer was given at the first question and 0.4 if a false answer was given at the first question respectively. In cases where a node does not have a parent, the table has only two values that show the likelihood of giving a true or a false answer (Table II).

TABLE II
NODE WITH NO PARENT

Q1	true	False
	0.5	0.5

After storing all the essential history information stored in the conditional probability tables, Bayesian networks can be used either to help making decisions or as a way to automate a decision-making process. Someone can use Bayesian networks to perform inductive reasoning (diagnosing a cause, given an effect) and deductive reasoning (predicting an effect, given a cause). The operation of Bayesian networks is based on a well known mathematical rule, the Bayes' rule. Most simply, Bayes' rule can be expressed as follows [16]:

$$P(\omega_i / \mathbf{x}) = \frac{P(\mathbf{x} / \omega_i) \cdot P(\omega_i)}{p(\mathbf{x})}$$

where ω_i is a state of nature, $\mathbf{x}$ is the vector of the monitored characteristics, $P(\omega_i)$ is the a priori probability, $P(\mathbf{x}/\omega_i)$ is the probability density function (likelihood), $P(\omega_i/\mathbf{x})$ is the a posteriori probability and $p(\mathbf{x})$, which is known as marginal likelihood, equals $\sum_{j=1}^{n} P(x/\omega_j) \cdot P(\omega_j)$ (n is the number of different states of nature).

According to our implementation, an xml file is created which contains all the information needed for handling the questioners; such as the questions, the correct answers, a decision-threshold, the initial likelihood for each node and the values of the conditional probability table (for example 0.8, 0.6, 0.2, 0.4). The system retrieves the first question from the xml file and prompts it to the student. Based on the answer of each student and according to the Bayes' rule, the system estimates the likelihood of being correct for the next consecutive answers (based on the respective Bayesian tree structure).

Questions that are considered to be answered correctly easily, according to a decision-threshold which is initially configured by the administrator, are by-passed and never prompted to the user of the e-learning system. The decision-threshold denotes the difficulty of a particular question. This is implemented by using a Black List in which all the questions to be skipped are deposited.

The benefit of this scheme is that the student does not have to spend time by answering questions that are considered to be far easy for his/her knowledge level.

B. Implementing Questioner Assessment Agent

In order to provide the questioner assessment functionalities according to the Bayesian network - based approach, introduced in this paper, a Questioner Assessment Agent is implemented. This agent is developed using the Java programming language and the JavaBayes [17] class library.

This library comprises a set of tools for the creation and manipulation of Bayesian networks. The Questioner Assessment Agent functionalities are implemented in a java class that includes the following methods:

Initialize_Questioner(xml_file): This method has as input an xml file that is structured according to the XMLBIF xml schema and is used from the JavaBayes class in order to pass to the Bayesian network the probabilities of the questions and the decisions' thresholds that are assigned from the tutor to each specific questioner. Also, there are additional information such as the questioner id and the student id that are used by the agent in order to identify the requests. Furthermore this initialization function creates the necessary log files for storing the student answers in order to be utilized from the proposed genetic algorithms. Two log files are initialized:

- The "question_reportdata" log file that stores for each question the answers of each student, the next question that is selected and the threshold of the question.
- The "student_reportdata" log file that stores a history for the questioners and the associated answers that have been executed from a particular student.

Get_Questioner_Answer(student_id,questioner_id,query_id, answer): This method is used in order to execute the Bayes inference rule and calculate the next question that will appear to the student according to the answer that he/she gave to the previous question. The student's answer and the result of the algorithm executed on Bayesian network are logged in

Get_Questioner_Result(*student_id,questioner_id*): This method is executed when the students are finished with the questioner. It processes the wrong and correct answers and, by taking accounts the questions that do not appear to him/her, it calculates the result of the questioner exam for the specific student. In addition this function is executing the Genetic Algorithm for student classification. The genetic algorithm parses the log files and according to the rules described above decides the level of classification of the student that has answered the questioner.

The next step of the implementation is to publish the functionalities of the Questioner Assessment Agent to the Dispatcher Agent. This works by adding references of the corresponding methods to the initialization function of the Dispatcher Agent.

The Questioner Assessment Agent is communicating directly with the Dispatcher Agent of the proposed agent platform through ACL messages. Using the Bayesian_Logic library the Query Assessment Agent is responsible to react to the requests coming from the Dispatcher Agent.

The main processes are:

1. Initialization of the questioner for the user.
2. Identification of the next question that the user has to answer.
3. Calculation and provision to the e-learning legacy system whether the user has passed the questioner or not.
4. Execution of a Genetic Algorithm that classifies the users (students) in categories (eg. good, intermediate, novice) and updates the respective

SOAP REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="
http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <SOAP-ENV:Body>
  <ns1:get_questioner_answer SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="
  http://bayesian_net">
   <questioner_id xsi:type="xsd:int">7</questioner_id>
   <student_id xsi:type="xsd:int">11</student_id>
   <query_id xsi:type="xsd:int">14</query_id>
   <answer xsi:type="xsd:string">a</answer>
  </ns1:get_questioner_answer>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP RESPONSE

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:get_questioner_answerResponse
   soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns1="http://bayesian_net">
   <get_questioner_answerReturn xsi:type="xsd:int">16</get_questioner_answerReturn>
  </ns1:get_questioner_answerResponse>
 </soapenv:Body>
```

Fig. 3 Dispatcher Agent SOAP Request – Response example

user model.

C. E-learning Platform Integration

Using the Eclipse Web Tools Platform [18] a SOAP wrapper is generated on the Dipsatcher Agent in order to publish the functionalities of the Questioner Assessment Agent through the Internet infrastructure. The Web Service that is generated has three operations similar to the functionalities of the Questioner Assessment Agent:

- initialize_questioner().
- get_questioner_answer().
- get_questioner_result().

Additionally, the corresponding WSDL document is created, having the invocation and grounding description of the above operation. Both the WSDL document and the SOAP server code are deployed in an application server (in our case the Apache Tomcat has been used) in order to be accessed through the Internet. Fig. 3 presents an example of the SOAP messages (request and response) that are exchanged between the e-learning system and the Dispatcher Agent in order to invoke the functionality Get_Questioner_Answer of the Questioner Assessment Agent.

The three mentioned operations can be easily invoked from a legacy e-learning system with a SOAP client which can be implemented in any of the well known programming languages. A legacy e-learning system can use these functionalities and interoperate with the Questioner Assessment Agent through the Internet by adding some lines of code. The legacy e-learning system used in our application was implemented using the PHP language. Some blocks of code were added to the e-learning system in order to allow the tutor to assign probabilities and threshold values to the questions of a questioner. Also, a SOAP client is adjusted to the routines implementing the exam for the communication with the Dispatcher Agent. Furthermore, the e-learning system is capable of retrieving the results of the agent's execution and uses it in order to assess the examination process of a particular student.

V. Resources Retrieval Agent

Another problem that is usually presented in the classical e-learning environments is the lack of additional information about a particular educational subject that could not be found in the course's notes nor presentations. In order to face this problem, a second agent is proposed in this paper, specifically, the Resource Retrieval Agent. The Resource Retrieval Agent is an agent that takes advantage of the well known Google search engine in order to search the Internet for information about a specific educational subject. The agent is initiated for the following method:

Retrieve Data (search_string): This function has as input a query string that comes from the e-learning environment and includes the description of the educational subject for which a student needs more information. The next step of the agent is to send a search request to the Google search engine using a SOAP client to invoke the Google API [19]. The SOAP response is filtered from the agent and the results that can be both web-site links and file links are forwarded to the e-learning system.

In order to publish the functionalities of the Resource Retrieval Agent to the legacy e-learning environments we utilize the same methodology for creating the appropriate web services in the Questioner Assessment Agent example. Using the Eclipse Web Tools Platform a new method named retrieve_data() is added to the methods of the web service that is published from the Dispatcher Agent.

The overall process that is executed from the Resource Retrieval Agent is depicted in Fig. 4.

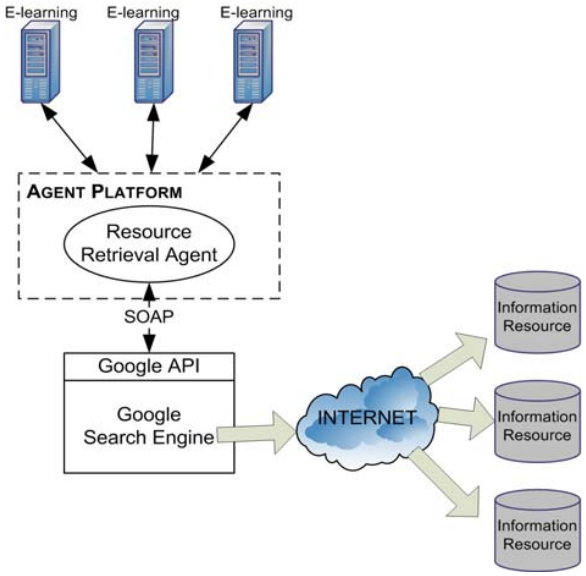

Fig. 4 Resources Retrieval Agent

VI. The Genetic Algorithm Module

A. Evolutionary Computation Techniques

Evolutionary computation techniques have been developed by attempting to imitate the mechanisms of natural selection and natural genetics. They have been around for quite some time, but have recently gained popularity, due to the advances in computing equipment, which make their implementation feasible and efficient. One of the most popular approaches to evolutionary computation is Genetic Algorithms (GAs) initially introduced by John H. Holland [20]. GAs operate on binary string structures, analogous to biological creatures. These structures are evolving in time according to the rule of survival of the fittest by using a randomized, yet structured information exchange scheme. Thus, in every generation, a new set of binary strings is created, using parts of the fittest members of the old set [21, 22, 25].

GAs process a binary coding of the parameter space and work on it. This coding (which is an essential part of the GA

design procedure) results in formation of binary strings. Having decided on the binary coding to be used, an initial set of strings (a population) is created at random. Next, a set of operators is applied to this initial population to generate successive generations that hopefully will improve over time. An objective function (usually referred to as fitness function in GA terminology) serves as a measure of goodness of a string, and is a functional of the function that we are trying to optimize. Several operators have been proposed at times; however, three simple operators perform well in a wide variety of search and optimization problems. These are the Selection operator, the Crossover operator and the Mutation operator [23, 24, 26].

Since this approach is significantly different from others, GAs have been successfully applied to search and optimization problems where other approaches failed [28]. GAs have proven to be well suited to optimization of specific non-linear multivariable systems and are being used in a variety of real – world applications including scheduling [29, 30], resource allocation [29, 31], training of ANNs [29, 32, 33, 34] and selecting rules for fuzzy systems [29, 34].

GA techniques are also included in the agent platform presented in this contribution. The GA Module of the proposed platform will be better described in the following sections.

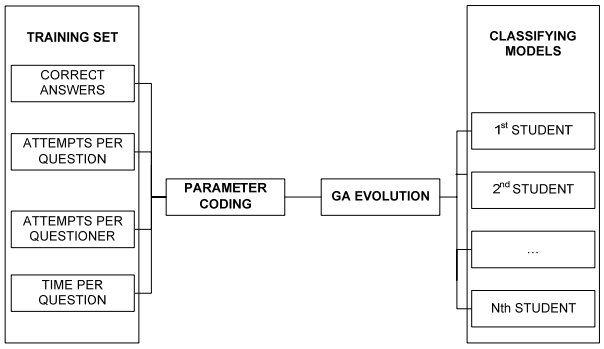

Fig. 5 The Process Model used in the GA Module

B. The Structure and Operation of the GA module

1. Modeling Students' Performance

The proposed module uses a GA in order to extract efficient information (classifying rules) based on the students' answering input data. The idea of using a GA in order to fulfil this difficult task came from the fact that GAs have been widely used in applications including classification of unknown data [29, 35].

2. Extracting Classifying Models

The process model used in the GA module is shown in Fig. 5. The system initially, gathers the following information:

1. The percentage of correct answers in the student's last attempt to answer the questioner.
2. The mean time for all answers a student has attempted to answer.
3. How many times a student has attempted to answer the questioner.
4. The mean number of times a student has attempted to answer each question of the questioner.

TABLE III
CLASSIFYING RULES USED IN THE GA MODULE

1st Criterion: Percentage of correct answers in the student's last attempt to answer the questioner.

Value: A	Value: B	Value: C
>85%	>65% and <=85%	>50% and <=65%

2nd Criterion: Mean time for all answers a student has attempted to answer

Value: A	Value: B	Value: C
<=1 minute	>1 and <=2 minutes	>2 minutes

3rd Criterion: Number of times a student has attempted to answer the questioner

Value: A	Value: B	Value: C
>=1 and <4	>=4 and <7	>=7

4th Criterion: Mean numbers of times a student has attempted to answer each question of the questioner

Value: A	Value: B	Value: C
>=1 and <4	>=4 and <7	>=7

Next, the system codes the gathered information in order to create the initial population of the GA and starts the GA evolution procedure. After a few generations, the GA provides many different binary strings each corresponding to one of the different students that have answered the questioner. Each binary string constitutes an efficient classifier for each respective student. Each such classifier comprises a binary string matching all the features that characterize a specific student. In this way, every classifier can describe the knowledge and answering performance of each student and all together can constitute a system for extracting models, according to each student's performance data. After that, the classification procedure is taking place. Each model resulted by the proposed module is compared with the classification rules given by the tutor. The classification rules used in the presented work, which are based on four major criteria, are shown in Table III.

In order each student to be classified; his/her model is compared with the rules presented above. The result of this comparison is a proposal to the tutor to classify a student to a specific knowledge level. The relational matrix of the above classification rules used for the final classification of each student is shown in Table IV.

TABLE IV
THE RELATIONAL MATRIX OF THE CLASSIFICATION RULES

	1st Criterion	2nd Criterion	3rd Criterion	4th Criterion	Final Classification
1	A	A	A	A	Good
2	A	A	A	B	Good
3	A	A	A	C	Good
4	A	A	B	A	Good
5	A	A	B	B	Good
6	A	A	B	C	Good
7	A	A	C	A	Good
8	A	A	C	B	Good
9	A	A	C	C	Good
10	A	B	A	A	Good
11	A	B	A	B	Good
12	A	B	A	C	Good
13	A	B	B	A	Good
14	A	B	B	B	Good
15	A	B	B	C	Good
16	A	B	C	A	Good
17	A	B	C	B	Good
18	A	B	C	C	Good
19	A	C	A	A	Intermediate
20	A	C	A	B	Intermediate
21	A	C	A	C	Intermediate
22	A	C	B	A	Intermediate
23	A	C	B	B	Intermediate
24	A	C	B	C	Intermediate
25	A	C	C	A	Intermediate
26	A	C	C	B	Intermediate
27	A	C	C	C	Intermediate
28	B	A	A	A	Good
29	B	A	A	B	Good
30	B	A	A	C	Good
31	B	A	B	A	Good
32	B	A	B	B	Good
33	B	A	B	C	Good
34	B	A	C	A	Intermediate
35	B	A	C	B	Intermediate
36	B	A	C	C	Intermediate
37	B	B	A	A	Intermediate
38	B	B	A	B	Intermediate
39	B	B	A	C	Intermediate
40	B	B	B	A	Intermediate
41	B	B	B	B	Intermediate
42	B	B	B	C	Intermediate
43	B	B	C	A	Intermediate
44	B	B	C	B	Intermediate
45	B	B	C	C	Intermediate
46	B	C	A	A	Novice
47	B	C	A	B	Novice
48	B	C	A	C	Novice
49	B	C	B	A	Novice
50	B	C	B	B	Novice
51	B	C	B	C	Novice
52	B	C	C	A	Novice
53	B	C	C	B	Novice
54	B	C	C	C	Novice
55	C	A	A	A	Intermediate
56	C	A	A	B	Intermediate
57	C	A	A	C	Intermediate
58	C	A	B	A	Intermediate
59	C	A	B	B	Intermediate
60	C	A	B	C	Intermediate
61	C	A	C	A	Intermediate
62	C	A	C	B	Intermediate
63	C	A	C	C	Intermediate
64	C	B	A	A	Novice
65	C	B	A	B	Novice
66	C	B	A	C	Novice
67	C	B	B	A	Novice
68	C	B	B	B	Novice
69	C	B	B	C	Novice
70	C	B	C	A	Novice
71	C	B	C	B	Novice
72	C	B	C	C	Novice
73	C	C	A	A	Novice
74	C	C	A	B	Novice
75	C	C	A	C	Novice
76	C	C	B	A	Novice
77	C	C	B	B	Novice
78	C	C	B	C	Novice
79	C	C	C	A	Novice
80	C	C	C	B	Novice
81	C	C	C	C	Novice

3. Variables and Values

The definition of the set of parameters that play major role in the modeling and classification procedure, for each of the entities in the process model that represent students' performance data (Fig. 5), was based mostly on our expert tutor. In the GA module, there are only input and output parameters (no intermediate ones). So, the parameters used by the GA module are the following:

Input parameters: percentage of correct answers in the student's last attempt to answer the questioner, mean time for all answers a student has attempted to answer, number of times a student has attempted to answer the questioner, mean number of times a student has attempted to answer each question of the questioner.

Final output parameter: final classification. It is the only final output parameter with three possible values: Good, Intermediate and Novice.

The domain of each parameter has been determined by the aid of the tutor, the statistical analysis of input data and the literature.

4. Architecture and Design

In this section, the basic structure and operation of the GA module is described. Specifically, the coding parameter unit, the GA evolution structure and the models extractor unit are presented.

The coding parameter unit operates as follows: The students' performance input data, which comprises of four input parameters, is coded into a binary string. This binary string is in fact the genome of the GA that will be evolved. Each parameter is coded using 31 bits. As a result the whole genome length equals 124 bits (Fig. 6). A set of such genomes constitutes the GA's population.

$$\underbrace{\underbrace{1010...0011}_{31} \quad ... \quad \underbrace{1100...1100}_{31}}_{4}$$

Fig. 6 The structure of the genome used in the evolution procedure

The structure of the GA used in the GA module is presented in Fig. 7.

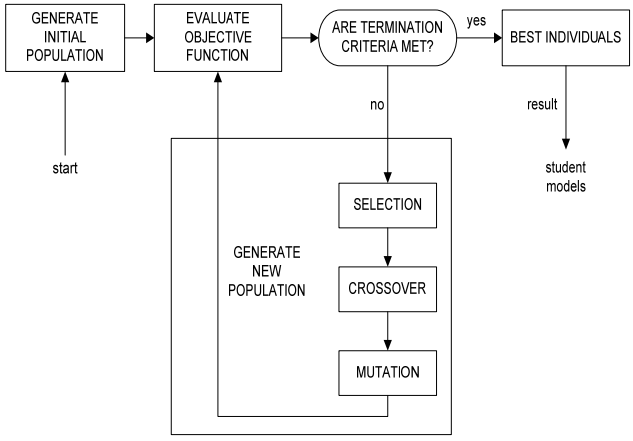

Fig. 7 General structure of the GA used

The initial population is generated using a set of random binary strings taken from the coding parameter unit. The objective function, estimates the goodness (fitness) of each individual (binary string). The fitness of each individual is: the number of cases that the individual has the same value, for specific input parameters, with instances (students' performance data) belonging to a specific student and is calculated as shown in Fig. 8.

```
for each students' performance data in the input set
belonging to a specific student do
  genome_fitness = 0;
   for each genome do
      if the value of the first gene is the same with the
      value of the first input parameter then
         genome_fitness = genome_fitness + 1;
      if the value of the second gene is the same with
      the value of the second input parameter then
         genome_fitness = genome_fitness + 1;
      if the value of the third gene is the same with the
      value of the third input parameter then
         genome_fitness = genome_fitness + 1;
      if the value of the fourth gene is the same with the
      value of the fourth input parameter then
         genome_fitness = genome_fitness + 1;…
   return genome_fitness
   end
end
```

Fig. 8 Genomes' fitness computation procedure for a student

This procedure is repeated for each different student. The termination criterion is the number of generations, that is, the evolution procedure is terminated when a specific number of generations (in our case 5) are completed. While the termination criterion is not met, the whole evolution procedure is taking place for the current population, that is, selection, crossover and mutation. When the termination criterion is met, the GA evolution procedure is terminated and results to many classifying models, one for each different student.

5. Implementation Issues

As stated before, the type of GA used in the GA module is the classic simple GA [23, 24]. The representation used for the genomes of the genetic population is the classic binary string. As far as the reproduction operator is concerned, the classic biased roulette wheel selection is used. The crossover operator used is uniform crossover (with crossover probability equal to 0.9), while the mutation operator is the flip mutator (with mutation probability equal to 0.001. Except of that, the size of the population is set to 50 while the GA uses linear scaling and elitism [21, 22, 27].

The GA module is implemented using the C++ Library of Genetic Algorithms GAlib [36] and especially the GASimpleGA class for the implementation of the GA (non-overlapping populations) and the GABin2DecGenome class for the binary string genomes (an implementation of the traditional method for converting binary strings to decimal values). All the experiments were carried out on an Intel Pentium IV 2.7GHz PC with 256 MB RAM.

VII. Conclusion

The proposed agent platform provides an integrated approach towards achieving the utilization of various assessment and retrieval agents and computational intelligence techniques such as genetic algorithms in legacy e-learning environments. Our proposal enhances significantly the overall system in terms of flexibility and efficiency while it

introduces a high degree of agents and e-learning platforms interoperability utilizing web services technology. The presented multi-agent platform can support various intelligent agents that provide assessment services and information retrieval based on computational intelligence techniques such as Bayesian Networks and Genetic Algorithms.

The specific contribution establishes the basis for a continuous scientific work in the field of e-learning systems and more specifically in intelligent and adaptive e-learning environments. The utilization of artificial and computational intelligence techniques can be extended and provide even more sophisticated intelligent services as components of e-learning systems.

More specifically, the utilization of a multi-agent platform with more sophisticated agents that extend the intelligent and adaptive services to the learners and tutors is in progress. The use of Genetic Algorithms in order to identify more suited threshold values to the questions, a more sophisticated Genetic Algorithm for the updating procedure of the user model and, of course, designing more sophisticated agents that facilitate not only the process of the exam, but also the learning content, will be the main issues of our future work.

REFERENCES

[1] K. C. Giotopoulos, C. E. Alexakos, G. N. Beligiannis and S. D. Likothanassis, "Computational Intelligence Techniques and Agents' *Technology in E-learning Environments", International Journal of Information Technology,* Volume 2, Number 2, 2005,ISSN:1305-2403, pp. 147-156.

[2] S. Buraga, "Developing Agent-Oriented E-Learning Systems", *in Proceedings of The 14th International Conference on Control Systems And Computer Science – vol. II*, I. Dumitrache and C. Buiu, Eds, Politehnica Press, Bucharest, 2003.

[3] A. Angehrn, T. Nabeth, L. Razmerita, and C. Roda., "K-inca: Using artificial agents for helping people to learn new behaviors", *in Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT 2001)*, Madison USA, August 2001, pp. 225-226.

[4] A. Andronico, A. Carbonaro, G. Casadei, L. Colazzo, A. Molinari, and M. Ronchetti, "Integrating a multi-agent recommendation system into a Mobile Learning Management System", *in Proceedings of Artificial Intelligence in Mobile System 2003 (AIMS2003)*, October 12, Seattle, USA.

[5] O. R. Zaiane, "Building a Recommender Agent for e-Learning Systems", *in Proceedings of the International Conference on Computers in Education*, Auckland, New Zealand, December 2002, pp. 55-59.

[6] V. Pankratius, O. Sandel, W. Stucky, "Retrieving Content with Agents in Web Service E-Learning Systems", Symposium on Professional Practice in AI, IFIP WG12.5 – *in Proceedings of the First IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI)*, Toulouse, France, August 2004.

[7] F. Bellifemine, G Caire, A. Poggi and G. Rimassa, "JADE A White Paper", *Telecom Italia EXP Magazine*, Volume 3, Number 3 September 2003

[8] Foundation of Intelligent Physical Agents (FIPA), http://www.fipa.org/

[9] FIPA ACL Message Structure Specification, FIPA Standard, http://www.fipa.org/repository/aclspecs.html

[10] W3C - Web Services Activity http://www.w3.org/2002/ws/

[11] F. V. Jensen, *An Introduction to Bayesian Networks*, Springer Verlag, New York, 1996.

[12] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA, 1988.

[13] J. Martin and K. VanLehn, "Student assessment using bayesian nets", *International Journal of Human-Computer Studies*, vol. 42, pp. 575-591, 1995.

[14] Cristina Conati, Abigail Gertner and Kurt Van Lehn, "Using Bayesian Networks to Manage Uncertainty in Student Modelling", *User Modelling and User-Adapted Interaction*, vol. 12, pp. 371-417, Kluwer Academic Publishers, Printed in the Netherlands, 2002.

[15] C. Conati, A. S. Gertner, K. Van Lehn, and M. J. Druzdel, "On-line student modelling for coached problem solving using Bayesian networks", *in Proceedings of the Sixth International Conference on User Modelling*, A. Jameson, C. Paris and C. Tasso, Eds, Vienna, New York, Springer, 1997, pp. 231-242.

[16] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification*, Wiley-Interscience, 2nd edition, October 2000.

[17] JavaBayes, Bayesian Networks in Java, http://www.cs.cmu.edu/~javabayes/

[18] Eclipse Web Tools Platform (WTP), http://www.eclipse.org/webtools/

[19] Google API, http://www.google.com/apis/index.html

[20] J. H. Holland (1975), Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, University of Michigan Press (second edition: MIT Press, 1992).

[21] D. E. Goldberg (1989), Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, Mass.

[22] Z. Michalewicz (1999), Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin.

[23] M. Mitchell (1996), An Introduction to Genetic Algorithms, The MIT Press, Cambridge, Massachusetts, London, England.

[24] M. D. Vose (1998), The Simple Genetic Algorithm: Foundations and Theory, MIT Press.

[25] L. D. Whitley (1993), Foundations of Genetic Algorithms 2, Morgan Kaufmann.

[26] L. D. Whitley and M. D. Vose (1995), Foundations of Genetic Algorithms 3, Morgan Kaufmann.

[27] D. B. Fogel (1995), Evolutionary Computation: Toward a New Philosophy of Machine Learning, IEEE Press.

[28] Z. Michalewicz, D. B. Fogel (2000), How to Solve It: Modern Heuristics, Springer-Verlag, Berlin, Heidelberg.

[29] T. Back, D.B. Fogel, and Z. Michalewicz (1997), Handbook of Evolutionary Computation, Bristol, UK: Institute of Physics, and New York, NY: Oxford University Press.

[30] M. Gen, R. Cheng (1997), Genetic Algorithms and Engineering Design, John Wiley & Sons, Ltd.

[31] Xiang Wu, Bayan S. Sharif and Oliver R. Hinton, 'An Improved Resource Allocation Scheme for Plane Cover Multiple Access Using Genetic Algorithm', IEEE Transactions on Evolutionary Computation, Vol. 9, No. 1, February 2005.

[32] J. D. Schaffer, D. Whitley, and L. J. Eshelman, 'Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art', International Workshop on Combinations of Genetic Algorithms and Neural Networks, Baltimore, Maryland, June 6, 1992, pp. 1-37.

[33] M. Annunziato, M. Lucchetti, S. Pizzuti: 'Adaptive Systems and Evolutionary Neural Networks: a Survey', EUNITE2002, Albufeira, Portugal, Sept. 2002.

[34] E. Georgopoulos , S. Likothanassis and A. Adamopoulos, 'Evolving Artificial Neural Networks using Genetic Algorithms', International Conference on Artificial Neural Networks and Intelligent Systems (NNW 2000), Prague, Chez Republic, July 9-12, 2000.

[35] O. Cordon, H. Herrera, and M. Lozano, 'A classified review on the combination fuzzy logic-genetic algorithms bibliography', Tech. Report 95129, URL:http://decsai.ugr.s/herrera/flga.html, Department of Computer Science and AI, Universidad de Granada, Granada, Spain, 1995.

[36] GAlib - A C++ Library of Genetic Algorithm Components, Matthew Wall, Massachusetts Institute of Technology (MIT). Available: http://lancet.mit.edu/ga/.