A multiagent architecture for a web-based adaptive educational system

J. G. Boticario **E. Gaudioso** *
Dpto. de Inteligencia Artificial
Universidad Nacional de Educación a Distancia
Senda del Rey, 9
Madrid, Spain
e-mail: {jgb,elena}@dia.uned.es

Abstract

The widespread use of the Web in distance learning, the particular nature of the distance learning student and the dispersion of the relevant information sources increase the importance of developing interactive systems, which adapt to the information and communication needs of each student. We have developed a multiagent decision system which adapts to the user's needs. To best accomplish this, we have chosen heterogeneous agents which combine the solutions learned with different bias corresponding to different machine learning methods (C5.0, Naive Bayes, Progol, Backpropagation, Tilde and Autoclass).

Introduction

In order to solve the problems characteristic of Distance Learning (DL) Education on the Web we have constructed a multiagent architecture, which is intended to be adaptable to the user's needs (Boticario & Gaudioso 1999) and which is based on a combination of techniques applied in intelligent tutoring systems (ITS) (Weber & Specht 1997), adaptive hypermedia programs (AH) (Brusilovsky 1996) and learning apprentice systems (Dent *et al.* 1992). It falls into the category of so-called *Web-based Adaptive Educational Systems* (Brusilovsky 1998). The learning apprentice approach is intended to expand the initial knowledge base in order to reduce the effort required in user decision-making (adaptive hypermedia).

Considering the advantages of collaborative learning, the multitude of separate tasks involved, the unpredictability of the result and the dispersion of the resources, we have opted for a multiagent architecture; specifically, we have chosen a multiagent decision system. The language used in the communication between agents is KQML (Finin *et al.* 1994) and the content of the messages is represented in KIF (Genesereth & Fikes 1992). The concepts of the messages depend on the ontologies used, in this case, educational ontologies (Chen & Mizoguchi 1999).

*PhD grant from UNED

System architecture

The system provides: a distributed system architecture, a communication protocol used by the agents, a distributed learning algorithm, a conflict resolution mechanism and a distributed decision-making algorithm.

The system is implemented in terms of a multiagent decision approach and is organized as follows. Two main components are involved: the user interaction and the adaptive module (see figure 1). The first is implemented by the *interface agent* and is in charge of the organized presentation of the different types of material, this presentation being designed to achieve the highest *usability*. This module provides a single, integrated response to the user.

The adaptive module is composed of the following types of agents: user model agent, user modeling agent, material agent, pedagogical agent, contact agent, service identification agent, service model agent, service modeling agent and coordinator agent. The first four provide the basic functionality of an ITS. The next four are useful for identifying, by *collaborative filtering*, the system services that are of interest to users with similar profiles. Finally, the coordinator agent broadcasts a problem instance description (user's request) between the agents involved in that problem. This agent is also in charge of constructing a single response to the interface agent.

In more detail, the adaptive module consists of a dynamic multiagent system (agents join or leave the system) with weak coupling (they may, or may not, participate in each task). We have chosen heterogeneous agents in order to combine the solutions learned with different bias corresponding to different generalization methods: C5.0, Naive Bayes, Progol, Backpropagation, Tilde and Autoclass for clustering. To implement this combination, a special type of agent has been introduced: the advisor agent. Several advisor agents learn the competence range of each of the other agents. A task is only distributed to those agents that have been proved to be competent in that task. In this manner, we aim to establish a gradual process of agent specialization with concrete architectures for specific problems.

We distinguish two phases: application of learned knowledge and adaptation to the individual user

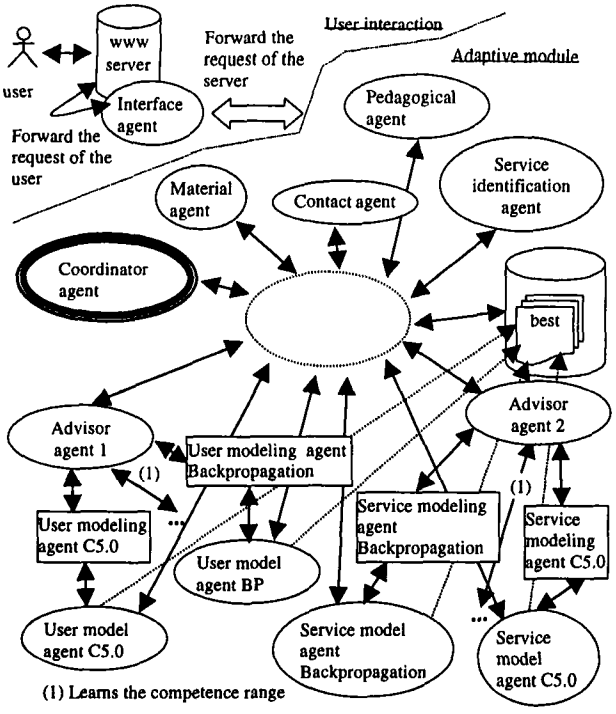

Figure 1: System's general architecture

through learning tasks. In the first, the interface agent makes a request to the coordinator. Depending on the request, this agent asks the advisors for the agents competent in the tasks involved in that request (the intention here is to improve on other approaches that distribute the same task to every agent in the system (Giráldez, Elkan, & Borrajo 1999)). In the second phase, when the interaction with the user has finished, with the available training examples (those that have been collected), the advisor agent learns the competence of each agent involved in the adaptation task: the model agent and the service agent. The task of learning the user model and that of learning the service model are implemented using the different generalization paradigms that constitute the various modeling options: service modeling agent and user modeling agent. The service identification agent selects the services that interest a significant number of users through *clustering* techniques.

Interaction with the system

The system is designed to provide new information, according to the user's needs (in terms of dynamically generated web pages which resemble web portals in many cases), and new communication channels of interest to him/her (news, shared workspaces, contact with students with similar characteristics ...), during each session.

The user is aware of the dynamic aspect of the interaction through: different presentations of the same information (project, hide, annotate), changes in the order in which interface elements appear, changes in the level of interaction (in accordance with the inferred user skill level) and different information descriptions.

The user interface has two different working areas when the advice given by the system cannot be integrated in the information presented or when the advice information is contextual and has no effect on the information shown. The system keeps two different interaction traces, one for each area in the interface. When the interface agent makes a request to the coordinator agent, the latter agent divides the previous traces and stores them.

These traces are used as training examples in order for the advisor agents to learn the degree of competence of the agents involved in the adaptation task (the user and service model agent), once the interaction with the user has finished. The following of a link recommended by the system is considered a positive training example since in this case the advice given by the system coincides with the option chosen by the user.

Once the advisor agent has picked a certain agent (a user and service model agent) as the most competent to carry out a given task, the latter writes the data included in its model to the database, thus ensuring that at any moment in time, the system database contains the best model built up to that moment. In this way, any agent needing some data from these models can consult the database and if the sought-after attribute has already been learned, it does not need to make any more queries since it can simply read the value from the database; otherwise, it must first ask the corresponding advisor agent which agent is the most competent for the task at hand and then ask the modeling agent picked by this advisor agent for the value of the attribute. This use of the database enables a personalized response to the user to be given more efficiently. In this design, the consistency between new values being introduced in the database and values already obtained from the database is maintained. To do so, an update-message mechanism is established between the agents that modify the data and those that read it.

The system therefore carries out two fundamental learning tasks: one when the different modeling agents infer a certain value of their corresponding models; another when the advisor agents learn the degree of competence of the corresponding modeling agents, once the interaction of the system with the user has finished.

Another agent with learning tasks is the service identification agent. It selects the services that interest a significant number of users through *clustering* techniques; the intention here is for the system to learn the characteristics determining which services may be of interest for a given user.

As stated above, when the user makes a request, the interface agent sends it on to the coordinator agent. In order to respond to the former agent's request, the latter agent builds an instance of the page requested by the user in its knowledge base. As well as the attributes which define the page (page title, author ...), this

page instance contains an attribute to store the set of recommendations that the system has inferred for that particular user: next recommended link, newsgroups of interest, complementary material which could be consulted or any other action which may be advisable (to go to a particular workgroup, share a particular document ...). The values of this attribute are obtained from the replies of the other agents to the coordinator agent's requests.

Finally, the coordinator agent sends the interface agent the page instance described above; the latter agent then builds the page to be presented to the user in accordance with his/her preferences and needs, and with a view to obtaining maximum usability and access speed. The actions and criteria which this agent follows to build the page are defined in its corresponding knowledge base.

It should be stressed that the dynamic construction of web pages following the recommendations inferred according to a user model (that the system learns and builds) is not the only dynamic aspect of the system; all the tasks currently carried out by the system (curriculum sequencing, adaptive navigation support, adaptive presentation and adaptive collaboration support) are modeled in the knowledge base of the agent responsible for executing them as if they were other entities of the system. In this way, an agent can modify the definition of the task which it executes if necessary, in order to better adapt itself to the needs of the student. The other tasks which we plan to implement (intelligent analysis of student solutions, interactive problem solving support and example-based problem solving) are treated similarly.

In order to clarify some of the basic processes triggered when the user interacts with the system, below we present the communication sequence initiated when the student requests a specific resource via a link to an HTML page. We suppose that the user has previously connected to the system so that some data concerning him/her is available to it.

In the example considered, a request by the student for a page of the practical exercise module triggers the execution of the adaptive collaboration support task (carried out by the contact agent) and the adaptive navigation support task (carried out by the pedagogical agent), leading the system to propose both new links related to the requested page (adaptive navigation support), as well as contact with other students who have completed the exercise in question and consultation of related newsgroups (adaptive collaboration support). In this example we assume that the material agent has not found any recommendation to give to the student.

In order to be able to identify the connecting user, the elements of each area of the interface mentioned above and any other additional information needed to build a personalized response (apart from the name of the requested page), certain parameters are added to links to system-built pages. In the example at hand, the link which the student requests is:

`<href=''webdl?advice+computed+id-usr+clas.html''>`

The link attribute `advice` indicates that the link requested has been recommended by the system. Elements that have the attribute `computed` are not yet constructed and have to be constructed on the fly by the adaptive module. An existing element doesn't have this attribute and the adaptive module can return it immediately, in parallel with the construction of the corresponding advice.

Figure 2 shows the transfer of messages taking place when the student requests a specific exercise page. In this example, the contact agent needs to know which is the user's preferred contact medium and therefore asks the model agent chosen by the advisor agent; this agent infers the value of this attribute (in the example, it is inferred by applying C5.0), stores it in the database and then replies to the contact agent that requested it.

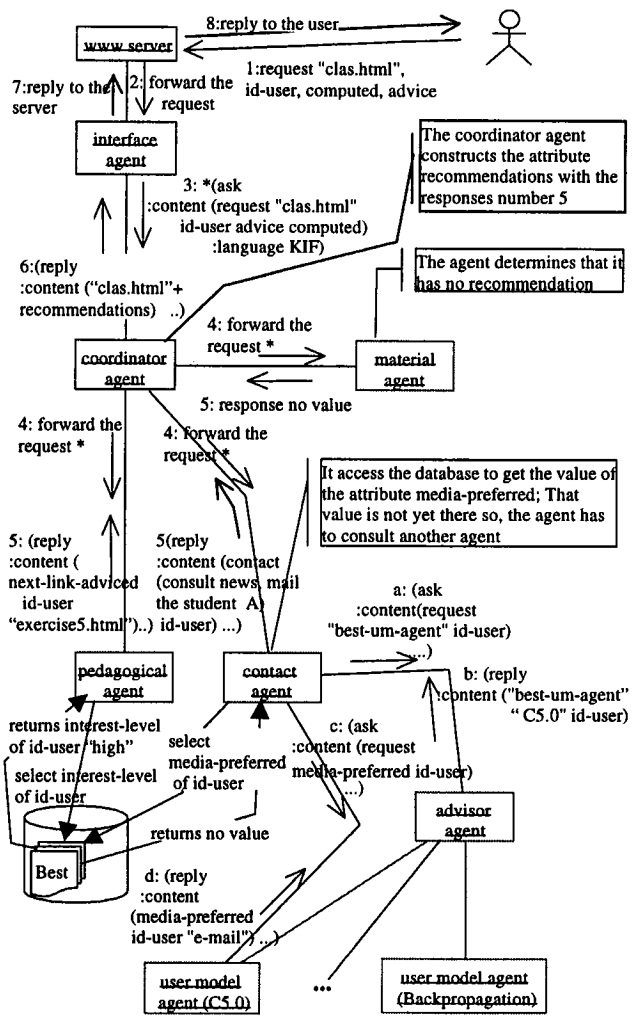

Figure 2: Flow of messages produced in an interaction of the user with the system (follow the sequence numbers)

State of development of the project

Currently, the basic architecture of the system has been implemented and it has been applied to the access to the course material of the machine-learning course of the Computer Science degree at the Spanish National University for Distance Education.

The system tasks which are planned are: curriculum sequencing, intelligent analysis of student solutions, interactive problem solving support, example-based problem solving, adaptive presentation, adaptive collaboration support and adaptive navigation support. Those which have been implemented are: curriculum sequencing (of the course material), adaptive collaboration support (of the course practical exercises), adaptive presentation and adaptive navigation support (for every page constructed by the system).

At present, we have only experimented with potential users; these experiments have validated the proposed architecture in adaptation tasks previously described. However, we expect to complete a comparison of the responses of the system in the different tasks in the course of the current academic term.

Acknowledgements

The authors would like to acknowledge the helpful comments of Simon Pickin, arising in the course of his language revision of this article. We also thank the entire Artificial Intelligence Department for providing support for this project. We would like to express our deep gratitude to professor Tom Mitchell at Carnegie Mellon University for providing THEO for research purposes.

References

Boticario, J. G., and Gaudioso, E. 1999. Towards personalized distance learning on the web. In Mira, J., and Sánchez-Andrés, J., eds., *Foundations and Tools for Neural Modeling*, number 1607 in Lecture Notes in Computer Science. Springer Verlag. 740–749.

Brusilovsky, P. 1996. Methods and techniques of adaptive hypermedia. In *User Modeling and User-Adapted Interaction*, 87–129. Kluwer academic publishers.

Brusilovsky, P. 1998. Adaptative educational systems on the world-wide-web: A review of available technologies. In *Proceedings of Workshop WWW-Based Tutoring at Fourth International Conference on ITS (ITS'98)*. San Antonio, TX: Mit Press.

Chen, W., and Mizoguchi, R. 1999. Communication ontology for learner model agent in multi-agent architecture. In *Proceedings of the International Conference on Artificial Intelligence in Education (AI-ED99)*.

Dent, L.; Boticario, J. G.; McDermott, J.; Mitchell, T. M.; and Zabowski, D. T. 1992. A personal learning apprentice. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 96–103. San Jose, CA: Mit Press.

Finin, T.; Fritzson, R.; McKay, D.; and McEntire, R. 1994. Kqml as an agent communication language. In Press, A., ed., *Proceedings of the Third International Conference on Information and Knowledge Management*, 64–69.

Genesereth, M., and Fikes, R. 1992. Knowledge interchange format, version 3.0 reference manual. Technical Report KSL-92-86, Knowledge Systems Laboratory.

Giráldez, J. I.; Elkan, C.; and Borrajo, D. 1999. A distributed solution to the pte problem. In Gini, G. C., and Katritzky, A. R., eds., *Predictive Toxicology of Chemicals: Experiences and Impact of AI tools, Papers from the 1999 AAAI Spring Symposium, TR SS-99-01*, 82–85. AAAI Press.

Weber, G., and Specht, M. 1997. User modeling and adaptive navigation support in www-based tutoring systems. In *Proceedings of the Sixth International Conference on User Modeling*, 289–300.