An Agent-based System for Supporting Learning from Case Studies

Marta C. Rosatelli[1], John A. Self[2], and Tulio V. D. Christofoletti[3]

[1] Graduate Program in Informatics, Catholic University of Santos
R. Dr. Carvalho de Mendonça, 144, Santos-SP, 11070-906, Brazil, Tel: +55 13 32055555
rosatelli@unisantos.br

[2] Computer Based Learning Unit, University of Leeds
Leeds, LS2 9JT, UK, Tel: +44 113 2334626
jas@cbl.leeds.ac.uk

[3] Department of Informatics and Statistics, Federal University of Santa Catarina
Cx.P. 476, Florianópolis-SC, 88040-900, Brazil, Tel: +55 48 3317111
tulio@inf.ufsc.br

Abstract. A main issue in collaborative learning is providing support and monitoring both the individual learners and the group activities. In this sense, there is a variety of functions that might be accomplished by a collaborative learning support system. Some examples are: knowledge diagnosis and evaluation, group and individual feedback, student and group modelling, and so on. LeCS (Learning from Case Studies) is a collaborative case study system that provides a set of tools and accomplishes some functions that together support learners during the development of a case study solution. This paper gives an overview of LeCS, focusing on the system design and architecture. The LeCS design is based on our model of supporting the learning from case studies method in a computer-based environment and in a distance learning context. The LeCS architecture is agent-based and includes three kinds of agents.

Keywords. Artificial intelligence in education, agent-based system, collaborative learning, intelligent distance learning, learning from case studies.

Type of submission. Paper Track.

Conference topic. Artificial intelligence in education and intelligent tutoring systems.

An Agent-based System for Supporting Learning from Case Studies

Abstract. A main issue in collaborative learning is providing support and monitoring both the individual learners and the group activities. In this sense, there is a variety of functions that might be accomplished by a collaborative learning support system. Some examples are: knowledge diagnosis and evaluation, group and individual feedback, student and group modelling, and so on. LeCS (Learning from Case Studies) is a collaborative case study system that provides a set of tools and accomplishes some functions that together support learners during the development of a case study solution. This paper gives an overview of LeCS, focusing on the system design and architecture. The LeCS design is based on our model of supporting the learning from case studies method in a computer-based environment and in a distance learning context. The LeCS architecture is agent-based and includes three kinds of agents.

1 Introduction

A main issue in collaborative learning is providing support and monitoring both the individual learners and the group activities. In this sense, there is a variety of functions that might be accomplished by a collaborative learning support system. Some examples are: knowledge diagnosis and evaluation, group and individual feedback, student and group modelling, simulated students, and so on. The usual problem in most Intelligent Tutoring Systems (ITS) of generating appropriate feedback and determining the contents of this feedback is also present in this kind of systems. This is specially critical in a distance learning context.

LeCS (Learning from Case Studies) [1] is a collaborative case study system for distance learning that provides a set of tools and accomplishes some functions that together support learners during the development of a case study solution. This paper gives an overview of LeCS, focusing on the system design and architecture. The design of LeCS is based on our model [1] of supporting the learning from case studies method in a computer-based environment and in a distance learning context. The LeCS architecture is agent-based and includes three kinds of agents.

The paper is organised as follows. In the next section we outline some related work. In the section thereafter, we present LeCS: first, we give an overview about the learning from case studies method and a methodology to develop the case study solution; then, we describe LeCS and its graphical user interface with the student; next, the LeCS architecture and functions accomplished by each kind of agent are detailed. Finally, we present the conclusions and directions for future work.

2 Related Work

The work presented in this paper introduces the case study element, which is particularly novel in ITS research. Learning from case studies is well established as an educational method in the traditional classroom [2]. However, the characteristics of case studies activities have led to their relative neglect in the ITS and Artificial Intelligence in Education areas.

On the other hand, this work shares characteristics with other approaches used in these area: agent-based ITS (e.g., [3]), work on collaboration (e.g., [4]), and work on supporting the problem solving process at a distance (e.g., [5]).

In particular, the system described in this paper has features that are quite similar to the ones encountered in other collaborative and intelligent distance learning systems. For instance, COLER [6, 7], which is a web-based collaborative learning environment in the domain of database design using the Entity-Relationship modelling formalism. COLER also focuses on both individual and collaborative learning, and on an agreement by the group on a joint solution for a collaborative task. Similarly as LeCS, it monitors the students' participation and encourages them to discuss their differences. Finally, COLER also generates advice to the students, concerning issues such as group participation, group discussion, feedback, reflection, checking the students own discrepancies, and the ER modelling. Some of these issues are also subject of the generation of interventions in LeCS.

3 LeCS: Learning from Case Studies

3.1 The Learning from Case Studies Method

Learning from case studies [2] is typically used in the business schools to train the students in disciplines that contain open-ended problems. Such kind of problems usually present complex, realistic situations, and demand cognitive flexibility to cope with them. The case method is used when the situated nature of cognition in the learning process and/or learning in ill-structured domains is required [8].

The case method has been widely used for years in a variety of disciplines. Among them we may cite law, engineering, business, and management. The common characteristic between such disciplines is that they introduce the kinds of problem that no analytical technique or approach is suitable to solve, with no "correct" or clear-cut solution.

A central issue in learning from case studies is the case discussion. It is so important that the case method is often referred to as the process of teaching by holding discussions, as opposed to lectures or labs. The case discussion process is often described as fluid and collaborative and is intrinsically related to the instructor's role in the case method. The case study text basically furnishes raw material for the case discussion.

The case instructor role - different from the teacher in the traditional classroom - is to lead the process by which the individual students and the group explore the complexity of a case study and develop the case solution. He or she maximises the opportunities for learning by asking the appropriate questions during the discussion, rather than having a substantive knowledge of the field or case problem.

The method application in the traditional classroom consists roughly of presenting a case study that introduces a problem situation to a group of learners who are supposed to discuss the case and find a solution to it.

The Seven Steps approach [9] is a methodology used to carry out the case solution development. It proposes that the case study solution be developed step-by-step. Each step of the approach has its own goal and suggests an activity to be carried out by the learners in order to achieve such goal. It guides the case solution development, splitting it into parts.

3.2 LeCS Description

LeCS provides a set of tools and accomplishes some functions that together support the learners during the development of the case solution. The tools are a browser, a chat, a text editor, and a representational tool. The support LeCS provides consists of representing the solution path taken by the learners and making interventions concerning the following aspects of the case solution development:

- *the time that the learners spend on each step of the Seven Steps approach* [9];
- *the learners' degree of participation in the case discussion*;
- *the misunderstandings that the learners might have about the case study*, and
- *the coordination of the group work.*

LeCS was implemented in the Delphi language and has a client-server architecture. The server hosts sessions and the client interacts with sessions. A session is associated with a group of students working collaboratively on the solution of a case study. The clients run on the students' machines. The server can run on one of the student's machine or alternatively on a different machine.

Graphical User Interface: Student.

The LeCS graphical user interface with the student displays the following components shown in Fig. 1: a pull down menu, a participants list, a browser, a solution graphical representation, a text editor, a chat, and a system intervention area.

The pull-down menu includes among others: (1) a case studies library containing the set of case studies available; and (2) the forms, where the learners fill out the agreed group answer to each step question. There is a form to each step. The forms are numbered and each entry corresponds to a component sentence of the step answer.

The participants list shows all the group participants that are working on a case study. The participants who are on-line at a particular moment - logged on a certain session - have their names annotated with the green colour whereas the ones that are logged off are annotated in red. The participants can carry out tasks directly with another participant (send a message or see the information available about him or her) just by clicking on the button corresponding to his or her name (cf. Fig. 1). Also, a timer is included in this area.

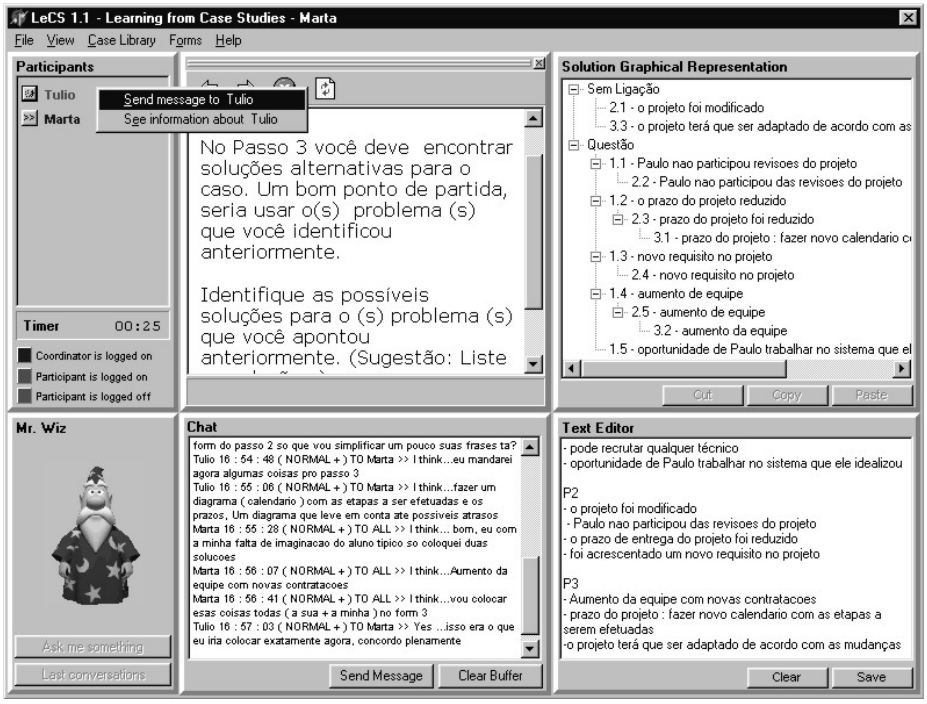

Fig. 1. LeCS graphical user interface.

The browser is used to access the web pages that display the teaching and learning materials and that guide the learners through the system use. The browser window is the only one that can be drag and dropped and customised by the user.

The solution graphical representation area displays the tree that is generated by the system during the case solution development. The representation is displayed graphically as a directory tree. The nodes of the tree are numbered with the forms correspondent numbers, which are followed by the correspondent textual answer.

The text editor is an individual space where the learners can edit their individual answers. It is used to answer individually the questions posed by the system, i.e., during the part of the solution process when individual learning takes place. The individual answers edited with this tool are supposed to be used when the learners participate in the case discussion (when collaborative learning takes place).

The chat tool is quite similar to the traditional programs of this kind and is where the case study discussion takes place. The participant can, besides writing free text, (1) express his or her emotional state; (2) direct his or her message to the whole group or to a particular participant (although this message is visible by all the group); and (3) make use of sentence openers [10, 11] to scaffold conversation and facilitate the process of reaching an agreement in the case discussion.

The intervention area includes an interface agent (a Microsoft-based agent) that can be characterised as an animated pedagogical agent [12]. All the interventions that LeCS makes are presented through this agent.

3.3 Architecture

The LeCS agent-based [13] architecture (Fig. 2) is organised in a federated system. The LeCS agents can be classified either as interface, reactive, and/or hybrid agents [14]. The agents communication is based on an Agent Communication Language [15]. The messages exchanged between the agents use the KQML (Knowledge Query and Manipulation Language) format [16]. The communications structure establishes that the communication does not happen directly between agents, but rather through a facilitator. The facilitator is a special program - implemented as an agent - that keeps the information about each agent in the system, and is responsible for routing the messages, working as a broker. In addition, two databases were implemented: in the first one, the facilitator stores all the necessary information in order to route the messages; in the second one, it logs all the exchanged messages. The LeCS architecture includes three classes of agents: interface agent, information agent, and advising agent. There is one information agent and one advising agent running during a session, but as many interface agents as there are participants logged on.

Interface Agent.
The interface agent (cf. Fig. 1) can be characterised as an animated pedagogical agent [12]. It resides on the participant machine and all the system interventions are presented through it. A resources database contains the agent address, the name by which it is known, and its network mapping. A history database is implemented to log everything the interface agent does, including the communications with the user and the other agents. The information agent and the advising agent also have these same kinds of databases.

In addition, the interface agent stores information about the individual users: what is typed in the text editor, the number of contributions in the chat, the current step he or she is working on, the answer to each step question, and the time spent on each step (these last two functions are accomplished just by the interface agent of the group coordinator). Based on this information, the interface agent generates the interventions about timing and participation.

Timing Intervention.
The timing interventions consist of warning the learners when they exceed the time limit established for on-line collaborative work. The time limit is a function of both the total time estimated for the solution of a given case study and the time to respond to the particular step they are working on. The interface agent stores the time limit for the case studies that are modelled in LeCS. In order to generate a timing intervention it monitors the time spent by the learners in each step. An example of a timing intervention is “You are taking too long to complete this step”.

Participation Intervention.
The interface agent is also responsible for identifying and intervening regarding a low degree of participation of the individual learners in the case solution development. This is denoted by a small percentage of contributions during the case discussion. To

accomplish this, the interface agent monitors the learners' contributions in the chat per step. If a learner remains silent for more than 50% of the estimated time for working on a particular step the interface agent generates an intervention inviting the learner to participate. An example of a participation intervention is "Would you like to contribute to the discussion?".

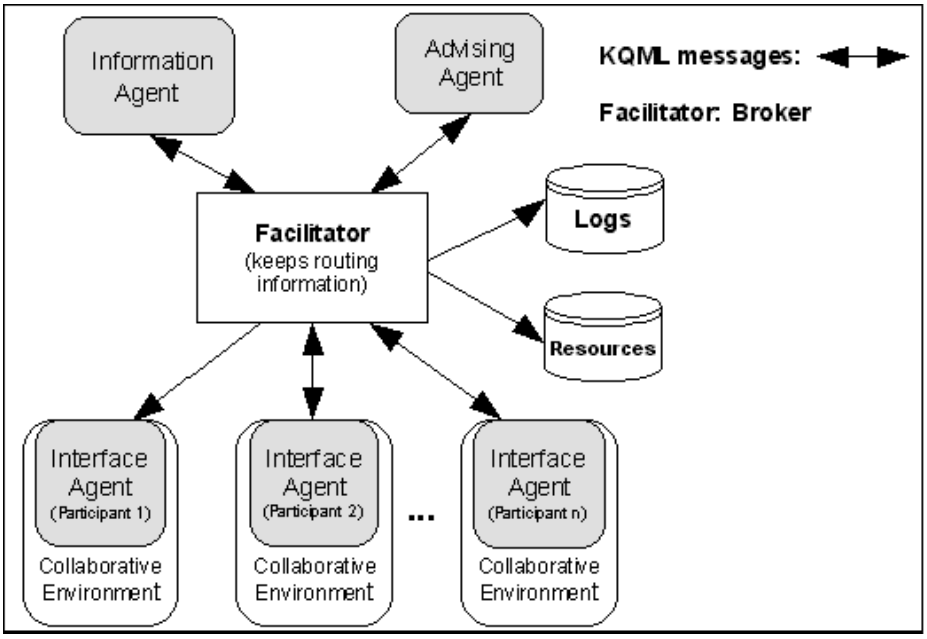

Fig. 2. The LeCS agent-based architecture.

Information agent.
This agent stores information that is divided into two different categories: didactic material and knowledge bases. Didactic material consists of HTML pages, images, and text. The knowledge bases refer to the domain and the pedagogical knowledge: the interventions about case-specific utterances and the representation of the case solution developed by the group of students. The information agent also stores the chat interactions. Both the interface and the advising agents access the information agent.

Case-Specific Intervention.
The interventions that LeCS makes about the learners' case-specific utterances are based on what is modelled in the domain knowledge base concerning the case study. In order to represent this knowledge we have adapted the idea of constraint-based modelling [17]. Constraint-based modelling (CBM) proposes that a domain be represented as constraints on correct solutions. As such, it is appropriate to the identification of case-specific utterance patterns that characterise a learner misunderstanding the case study contents.

In CBM [17] the unit of knowledge is called a state constraint and is defined as an ordered pair <*Cr*, *Cs*>, where (1) *Cr* is the relevance condition that identifies the class

of problem states for which the constraint is relevant, and (2) *Cs* is the satisfaction condition that identifies the class of relevant states in which the constraint is satisfied. For our purposes, *Cr* is a pattern that indicates that a particular sentence is a case-specific utterance and *Cs* is the correct case-specific utterance. The semantics of a constraint is: if the properties *Cr* hold, then the properties *Cs* have to hold also (or else something is wrong). In LeCS when "something is wrong" it means that there is a learner misunderstanding about this particular sentence. As a consequence the system should initiate an intervention.

In order to initiate the intervention LeCS checks for any violations of what is modelled. Hence, if the satisfaction pattern of a relevant constraint is not satisfied, then that state violates the constraint. In LeCS the problem states are the input sentences, that is, the sentences that compose the group answers to each step question, which are entered by the group coordinator in the forms. The intervention consists of stating the correct sentence.

Graphical User Interface: Case Instructor.
The LeCS graphical user interface for the case instructor allows modelling a case study to be worked on with the system. The case editor is used to edit the case study text. The constraints editor is used to edit the set of constraints for the case study. Thus, prior to run time, besides editing the case study text, the case instructor edits the constraints regarding each particular case study using the forms provided by the constraints editor (see Fig. 3). The system interventions concerning case-specific utterance patterns are generated based on these constraints. It is important to note that in the current implementation the order in which the words are used by the learners does not make any difference to the system response. Also, the constraints editor allows the case instructor to edit as many constraints as he or she deems necessary or appropriate for a given case study. In addition, it allows the modelling of more than one word to each constraint, through the use of the logic operators *and* and *or*, when the case instructor intends to contemplate different words that the students might use.

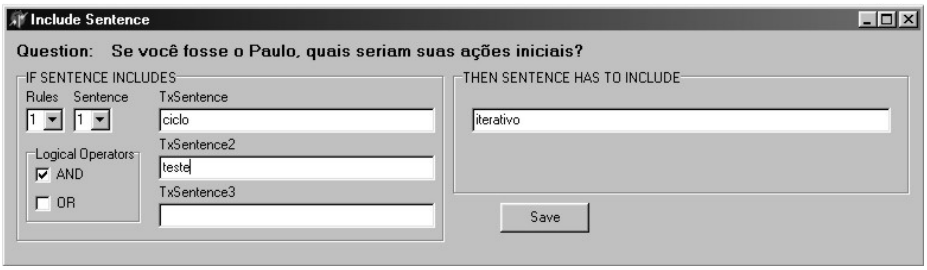

Fig. 3. LeCS graphical user interface for the case instructor.

Advising Agent.
The advising agent has engines to reason about the user actions, and to recognise situations in which some support is needed. It executes an algorithm to generate the solution tree representation with the information provided by the interface agent, and returns this representation to this agent. It identifies the case-specific utterance patterns that denotes a learners' misunderstanding of the case study with the information provided by the information agent. When applicable, it generates an

intervention about the misunderstanding, and sends the request of an intervention to the interface agent. It also generates and requests interventions to the interface agent concerning the coordination of the group work and the jumping of a step.

Solution Development Representation.
LeCS dynamically generates a knowledge representation, a tree data structure [18] that represents the case solution development according to the Seven Steps approach [9]. The tree represents the solution path taken by the group to solve the case study and is referred to as the solution tree. The solution tree is presented to the learners on the user interface. This means that what is displayed represents the case solution developed so far by the group (cf. Fig. 1). The objective of this representation is to help the learners choose the best envisaged solution. All the group participants see the same representation.

In order to generate this representation the connections between the tree nodes (the arcs) have to be obtained. To accomplish this the system has to make a link between a node in a given level to its parent node in the previous level through some kind of reasoning. A set of if-then rules define our heuristics to make these connections [19]. Such rules were derived from an empirical study [20] and an algorithm implements those rules.

In this algorithm the tree nodes are labelled by the sentences and the arcs are labelled by the rules. Inputs are the outcomes of each step, namely the sentences that compose the learner's joint answers to each step question, which are entered in the forms. The root of the tree corresponds to the question or dilemma posed by the case study. The levels of the tree represent each of the Seven Steps [9]. Each node, in a given level, refers to a component sentence of the learners' joint answer to that step question. The algorithm has a procedure to compare the answers of two subsequent levels (steps). This comparison aims to find out which sentence in the previous level is related to the sentence that is being analysed in the current level.

The procedure to compare the sentences verifies if a set of keywords in a sentence of a given level is also included in a sentence of the previous level. This set of keywords is defined in every step when the group coordinator fills out the form with the group answer to the step question. Thus, the set of keywords of a level *n* sentence is compared with the words of a level *n-1* sentence, word by word, until either one of the keywords is found, or the set of keywords is finished. In the former case (that is, if a keyword is found) the link between the nodes is made. The expansion of the nodes in a given level then is represented precisely when the nodes in that level are linked with the parent node in the previous level.

Next, provided that the interface agent informs the advising agent about which step the learners are currently working on, one level is added to the tree, and the procedure to compare the sentences is executed again. The final state of the tree represents all (ideally) alternative solutions that can be generated through the Seven Steps approach.

Group Coordination Intervention.
LeCS coordinates the group through the Seven Steps. In order to accomplish this, when starting a session the group should define a coordinator: a member of the group who is responsible for filling out the forms with the group's joint answers to the step questions. In this way the contributions gleaned from the various participants regarding the step answers are integrated into the joint answers. This means that the

forms are disabled to all the other members of the group except the coordinator and the coordination function is accomplished by LeCS based on the group coordinator actions. He or she in a certain sense defines to the system the pace at which the group proceeds in the solution development. Thus, once the coordinator fills out the forms of a particular step question and then moves on to the next step the system "knows" which step the group should be working on. If any of the learners, for instance, accesses a web page different from the one referring to that step, the system will prompt him or her a notification, warning that the group is working on a different step. An example of an intervention about the group coordination is "The group is working on step x". At any time during the solution process the group can decide to change its coordinator.

Missing Step Intervention.
LeCS does not constrain the learners by requiring them to complete all the steps of the approach. We believe this behaviour is avoided by the solution tree representation included in the graphical user interface and the forms that the group is supposed to fill out with their answers in every step. However, if the learners still jump a step, LeCS intervenes notifying the group. An example of a intervention about a missing step is "You have just jumped step x".

4. Conclusion and Future Work

In this paper we have presented LeCS, an agent-based system for supporting learning from case studies. We provided a theoretical background about the learning from case studies method and gave an overview of the system, describing its graphical user interface, agent-based architecture, and agents' functions.

As future work we plan to test LeCS with pairs of subjects in an experiment along the same lines as the empirical study described in [20]. In LeCS next version we intend to make the adjustments indicated by these experimental results and to tackle issues such as the recognition of many solution paths and student modelling.

References

1. Rosatelli, M. C., Self, J. A., Thiry, M.: LeCS: A Collaborative Case Study System. In: Frasson, C., Gauthier, G., VanLenh, K. (eds.): Intelligent Tutoring Systems. Springer-Verlag, Berlin (2000) 242-251
2. Christensen, C.R., Hansen, A.J.: Teaching with Cases at the Harvard Business School. In: Christensen, C.R., Hansen, A.J. (eds.): Teaching and the Case Method: Text, Cases, and Readings. Harvard Business School, Boston, MA (1987) 16-49
3. Shaw, E., Ganeshan, R., Johnson, W.L., Millar, D.: Building a Case for Agent-Assisted Learning as a Catalyst for Curriculum Reform in Medical Education. In: Lajoie, S.P, Vivet, M. (eds.): Artificial Intelligence in Education. IOS Press, Amsterdam (1999) 509-516
4. Mühlenbrock, M., Hoppe, H.U.: A Collaboration Monitor for Shared Workspaces. In: Moore, J.D., Redfield, C.L. Johnson, W.L. (eds.): Artificial Intelligence for Education. IOS Press, Amsterdam (2001) 154-165

5. Greer, J., McCalla, G., Cooke, J., Collins, J., Kumar, V., Bishop, A., Vassileva, J.: The Intelligent Helpdesk: Supporting Peer-Help in a University Course. In: Goettl, B.P., Halff, H.M., Redfield, C.L., Shute, V.J. (eds.): Intelligent Tutoring Systems. Springer-Verlag, Berlin (1998) 494-503
6. Constantino-González, M.A., Suthers, D.D.: A Coached Collaborative Learning Environment For Entity-Relationship Modeling. In: Frasson, C., Gauthier, G., VanLenh, K. (eds.): Intelligent Tutoring Systems. Springer-Verlag, Berlin (2000) 324-333
7. Constantino-González, M.A., Suthers, D.D., Icaza, J.I.: Designing and Evaluating a Collaboration Coach: Knowledge and Reasoning. In: Moore, J.D., Redfield, C.L. Johnson, W.L. (eds.): Artificial Intelligence for Education. IOS Press, Amsterdam (2001) 176-187
8. Shulman, L.S.: Toward a Pedagogy of Cases. In: Shulman, J.H. (ed.): Case Methods in Teacher Education. Teachers College Press, Columbia University, New York, NY (1992) 1-30
9. Easton, G.: Learning from Case Studies. Prentice Hall, London (1982)
10. McManus, M.M., Aiken, R.M.: Monitoring Computer-Based Collaborative Problem Solving. Journal of Artificial Intelligence in Education 6 (4) (1995) 307-336
11. Robertson, J., Good, J., Pain, H.: BetterBlether: The Design and Evaluation of a Discussion Tool for Education. International Journal of Artificial Intelligence in Education 9 (1998) 219-236
12. Lester, J.C., Converse, S.A., Stone, B.A., Kahler, S.E., Barlow, S.T: Animated Pedagogical Agents and Problem-Solving Effectiveness: A Large-Scale Empirical Evaluation. In: du Boulay, B., Mizoguchi, R. (eds.): Artificial Intelligence in Education. IOS Press, Amsterdam (1997) 23-30
13. Franklin, S., Graesser, A.: Is It an Agent or Just a Program? A Taxonomy for Autonomous Agents. In: Müller, J., Wooldridge, M.J., Jennings, N.R. (eds.): Intelligent Agents III. Springer-Verlag, Berlin (1997) 21- 35
14. Nwana, H.S.: Software Agents: An Overview. The Knowledge Engineering Review 11 (3) (1996) 205-244
15. Genesereth, M.R., Ketchpel, S.P: Software Agents. Communications of the ACM 147 (1994) 48-53
16. Labrou, Y., Finin, T.: A Proposal for a New KQML Specification. Technical Report CS-97-03, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County (1997)
17. Mitrovic, A., Ohlsson, S.: Evaluation of a Constraint-Based Tutor for a Database Language. International Journal of Artificial Intelligence in Education 10 (1999) 238-256
18. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Englewood Cliffs, NJ (1995)
19. Rosatelli, M.C., Self, J.A.: A Collaborative Case Study System for Distance Learning. International Journal of Artificial Intelligence in Education 13 (2002) to appear
20. Rosatelli, M.C., Self, J.A.: Supporting Distance Learning from Case Studies. In: Lajoie, S.P., Vivet, M. (eds.): Artificial Intelligence in Education. IOS Press, Amsterdam (1999) 457-564