2

Developing and Implementing a Multi-Agent System for Collaborative E-learning

Hani Mahdi and Sally S. Attia
Computer and Systems Engineering Department,
Faculty of Engineering, Ain Shams University,
Cairo, Egypt

1. Introduction

Computer-Supported Collaborative Learning (CSCL) systems have naturally evolved over the past decades as a consequence of the availability of high speed networking and the Internet. The conventional educational methodology has been influenced and different new learning methodologies have evolved accordingly. An excellent example is e-learning which has become one of the most popular teaching methods in recent years. One of its modes is the blended learning where learners can view teaching materials asynchronously from a teaching website and collaborate with their peers, while providing for necessary face-to-face explanation, discussion, and physical operation in the classroom.

Teaching Assistants (TAs) play an essential role in the learning process when the ratio of the number of students to that of the instructors is large due to the difficulty of direct contact between an instructor and his students in a given course. This is noticeable in education environments in youth societies and developing countries like Egypt. Although computer systems are used to solve many problems of handling large volumes of data as in database and data-mining systems, the usage of computer systems to solve the above problem is carried out considering one course and/or one instructor.

Parallel to the evolution of e-learning methodologies, the intelligent agent paradigm has generated such a remarkable interest in many application domains over the last two decades. It is growing to be a continuously evolving and expanding area. This chapter considers the results from developing a Multi-Agent System (MAS) in the field of e-learning education with the help of collaborative learning (learning from peers). The dynamism in e-learning can be made more powerful with the help of intelligent agents. Intelligent agents – the so called e-assistants or helper programs - can reside inside a computer and make the learning in e-learning occur dynamically to suit the need of the user. They can track the user's likes and dislikes in different areas, the level of knowledge and the learning style and accordingly recommend the best matching helpers for collaboration.

This chapter outlines the development and the implementation processes of a Multi-Agent System for Collaborative E-learning (MASCE). It considers mainly the innovation part of the system that is the Teaching Assistant Agent. This chapter deliberates one of the roles of the Teaching Assistant Agent which is the matching and the clustering of the students to find,

for example, the best helper colleague for a student (Mahdi & Attia, 2008 a, b). Collaboration between students (peers) is one of the main tasks of any serious e-learning system.
However, this is not the only task of MASCE which is designed to be used to assist the teaching and learning processes. This system considers the blended learning environment as a supplement to the face-to-face lecture where students can use the system in the lab or from home after attending the traditional lecture in the faculty. The goal is to incorporate the intelligence of the multi-agent system in a way that enables it to actively and intelligently support the educational processes, where multiple agents can interact to exchange information so that students may collaborate on how best to gain knowledge.

2. E-Learning

Over the past decade, speed in the transmission of knowledge is no longer the way it used to be. Due to this change, conventional educational methodology has also followed this trend and has gone through a huge transformation. An excellent example is e-learning which has achieved and broken through the restrictions of time and space. In recent years it has become one of the most popular teaching methods (Yang, 2000).
Digital learning, or e-learning, is defined by the American Society of Training and eDucation (ASTD) as: "E-learning is the application of digital media by the learner in the learning process, where digital media includes the Internet, corporate networks, computers, satellite broadcasts, audio tapes, videos, interactive television, and CD-ROMs, etc. The scope of e-learning applications includes online learning, computerized learning, virtual classrooms, and digital cooperation." While the ASTD definition emphasizes digital tools, the inflexible use of digital tools can by no means affect the sweeping reform of educational methods. For example, the Computer Aided Learning (CAL) used in the 1960's concentrated only on the use of computers in learning. Course content and learning activities were not significantly different from those of the age of pencil and paper. After the Internet emerged in 1995, online learning – which is also known as web-based learning (WBL) – used database and server technology to ensure that data transmission, access, and response were highly interactive and extensible. As a result, research in e-learning must include the establishment of a teaching and learning platform, development of teaching material content, and design of learning activities (Chen & Chiu, 2005).
Online e-learning uses three kinds of transmission models: asynchronous, synchronous, and blended. The two former models overcome the spatial limitations of conventional classroom learning, while the blended model proposes improvements to overcome some of the disadvantages of the first two models. Asynchronous teaching models employ a transmission method that can be divided into two parts: "Teaching content" is transmitted via a webpage or CDROM, while "teaching interaction" is transmitted via e-mail or discussion area. The synchronous model employs "virtual classroom" hardware and software, and uses the Internet to conduct videoconferencing and chat room activities to transmit "teaching content" and "teaching interaction." The aim of this model is to achieve teaching as synchronous as in a real classroom. If we compare the two models, we can see that the former offers the advantages of in-depth learning at any time or place, but has the disadvantages of inconvenient communication and lack of real-time response capability. Although the latter enables real-time interaction and a sense of participation, it has the disadvantages of time limitations, high cost, and need for high-technology supporting

equipment. As a result, in order to cut costs while offering good interactivity, most recent e-learning systems adopt the highly feasible blended model. In a typical blended model, learners can read teaching materials asynchronously from a teaching website, while providing for necessary face-to-face explanation, discussion, and physical operation in the classroom (Rosen, 2009).

2.1 E-Learning 2.0

The term e-learning 2.0 is used to refer to new ways of thinking about e-learning inspired by the emergence of Web 2.0. From an e-learning 2.0 perspective, conventional e-learning systems were based on instructional packets that were delivered to students using Internet technologies. The role of the student consisted in learning from the readings and preparing assignments. Assignments were evaluated by the teacher. In contrast, the new e-learning places increased emphasis on social learning and use of social software such as blogs, wikis, podcasts and virtual worlds such as Second Life (Rosen, 2009).

E-learning 2.0 is built around ***collaboration***. E-learning 2.0 assumes that knowledge is socially constructed. Learning takes place through conversations about content and grounded interaction about problems and actions. Advocates of social learning claim that one of the best ways to learn something is to teach it to others.

E-learning in general must not be considered as an electronic variant of classical education. That is indeed the problem. Not only the conditions of the educational offer are totally different, but also the cognitive and social behavior of humans require a completely dedicated analysis that most of the times has no precedents and thus requires a research attitude. This is the challenge of e-learning. E-learning is therefore NOT an application of technologies to human learning, in the sense that assuming to know what to apply (the technologies) and how (the pedagogy) one gathers things together and the result will be a success (people learn). On the contrary, each profound effort risks being unique in the sense that it requires specific technologies and specific pedagogical principles to be developed and applied in a trial and error fashion (Cerri, 2002).

The major obstacles for e-Learning are bound to the innovation for individuals and institutions of the asynchronous distance interactions among humans and electronic resources (documents, but also programs). However, the available technical tools are quite sophisticated and developed in many respects. Investigating the recent Intelligent Tutoring Systems or AI in Education Conferences one may notice the progress. Perhaps one can enhance the image by putting research efforts in the integration, or in the dialogue management, that is yet poor in real situations. However, it is believed that the bottleneck is more to be found on the human motivation for engaging in e-Learning practices. By "human" we include any role: learners, teachers, managers, experts, as well as combinations, i.e.: societies (classes, groups of teachers, etc.). One of the reasons for a lack of motivation in learners is the difficulty for certification of their learning when it has occurred at a distance. Another is the relative lack of friendliness of systems. The list of problems continues, yet human motivation is fundamental in order for technologies to be successfully introduced in human social practices (Cerri, 2002). Some solutions for these problems will be discussed later just after we have a background of agents.

3. Agents and Multi-Agent Systems

Over the last two decades, a major Computer Science research topic has been the development of tools and techniques to model, understand, and implement systems in which interaction is the main issue.

Agents, Software Agents or Intelligent Agents are intelligent in the sense that they are adaptive, independent, and possess reasoning capability. They can plan and execute tasks in cooperation with other agents in order to satisfy their goals. A Multi-Agent System (MAS) is defined as a loosely coupled network of problem solvers that work together to solve problems that are beyond the individual capabilities or knowledge of each problem solver (Agent). The increasing interest in MAS research is due to significant advantages inherent in such systems, including their ability to solve problems that may be too large for a centralized single agent, provide enhanced speed and reliability, and tolerate uncertain data and knowledge. Some of the key research issues related to problem-solving activities of agents in a multi-agent system MAS are in the areas of coordination, negotiation, and communication. With advances in Web technologies, collaborative applications are now server based and the user interface is typically a Web browser. Thus, a collaborative application can be a Web-based solution that runs on a local server that allows people communicate and work together, share information and documents, and talk in real-time over the Internet. Recently, much research has been conducted in distributed artificial intelligence and collaborative applications. Several interesting methodologies and systems have been developed in areas such as distributed multi-agent systems for decision support, Web search and information retrieval, information systems modeling, and supply chain management. In particular, intelligent agents and multi-agent systems, and a variety of applications have been built in various domains (Sugumaran, 2009).

''Agent'' has different meanings and definitions in different domains and communities. Agent refers to a computer-simulated character, which presents users with human-like properties, such as domain competence, emotions, and other personal characteristics. The characteristics can be expressed or displayed in text, graph, icons, voice, animation, multimedia, or virtual reality.

An agent is a computer system that is capable of *independent* action on behalf of its user or owner (determining what needs to be done to accomplish design objectives, rather than constantly being ordered). A multi-agent system is one that consists of a number of agents, which *interact* with one-another. In the most general case, agents will be acting on behalf of users with different goals and motivations. To successfully interact, they will require the ability to *cooperate*, *coordinate*, and *negotiate* with each other, just as people do (Wooldridge, 2002). The field of Multi-Agent Systems is influenced and inspired by many other fields such as Economics, Philosophy, Game Theory, Logic, Ecology, and Social Sciences.

We build agents in order to carry out *tasks* for us. The task must be *specified* by us. But we want to tell agents what to do *without* telling them how to do it. The *agent function* maps percept sequence to actions: f: $P^* \rightarrow A$. The *agent function* will internally be represented by the *agent program*. The agent program runs on the agent *architecture* to produce f (Wooldridge, 2002).

As part of their study of agent systems, researchers began to develop terminology for agents. Some of these terms are explained as follows (Mendez, 1999):

- **Agent Architectures** analyze agents as independent reactive/proactive entities. Agent architectures conceptualize agents as being made of perception, action, and reasoning

components. The perception component feeds the reasoning component, which governs the agents' actions, including what to perceive next.

- **Agent System Architectures** analyze agents as interacting service provider/consumer entities. System architectures facilitate agent operations and interactions under environmental constraints, and allow them to take advantage of available services and facilities.
- **Agent Frameworks** are programming tools for constructing agents. Examples of these are Voyager, Aglets, JADE, and Odyssey.
- **Agent Infrastructures** provide the conventions that agents follow to communicate and to understand each other, thereby enabling knowledge sharing. Agent Infrastructures deal with the following aspects:
    - **Ontologies**: allow agents to agree about the meaning of concepts.
    - **Communication Protocols**: describe languages for agent communication.
    - **Communication Infrastructures**: specify channels for agent communication.
    - **Interaction Protocols**: describe conventions for agent interactions.

Using Mobile Agent in E- learning

Comparing the three distributed paradigms (client-server, code on demand, mobile agent), it can be seen that mobile agent provides greater flexibility. Furthermore, mobile agent possesses the following advantages (Quah et al., 2002):

1. Mobile agent moves computation code to data rather than data to computation, the repetitive request response handshaking is eliminated, by creating an agent to handle the transaction and sending it from client to server, the intermediate results and information passing are reduced, hence the network bandwidth consumption is reduced and the efficiency is improved.
2. The agents do not require a continuous connection between machines. The client can send off an agent into the network when the network connection is healthy, then it can go offline. The network connection needs to be reestablished later only when the result was returned by agents from remote host. Hence it provides a more reliable performance with intermitted or unreliable network connection.
3. The agent operates asynchronously and autonomously, and the user doesn't need to monitor the agent as it roams in the Internet. This saves users' time, reduces communication costs, and supports a decentralized network structure.
4. With adaptive learning and automation added to agents, the agent can be tooled with AI for information retrieval and filtering.

The main problem with mobile agent is security, which still presents a prospective area of research. In an agent system with low level of security, the mobile agent may harm the host or the host may harm the mobile agent. With the above characteristics, the software language to construct mobile agent system should be object oriented, platform independent, with communication capability, and implemented with code security. At present, the languages being used for mobile agent include Java, Telescript, Tcl and IBM Aglets with Java (Quah et al., 2002).

3.1 Multi-Agent Systems

Various definitions from different disciplines have been proposed for the term multi-agent system (MAS). As seen from DAI (Distributed Artificial Intelligence), a multi-agent system is a loosely coupled network of problem-solver entities that work together to find answers

to problems that are beyond the individual capabilities or knowledge of each entity. More recently, the term multi-agent system has been given a more general meaning, and it is now used for all types of systems composed of multiple autonomous components (agents) showing the following properties (Sugumaran, 2009):

- Each agent has incomplete capabilities to solve a problem
- There is no global system control
- Data is decentralized
- Computation is asynchronous

One of the most important current factors promoting MAS development is the increasing popularity of the Internet, which provides the basis for an open environment where agents interact with each other to reach their individual or shared goals. To interact in such an environment, agents need to overcome two problems: they must be able to locate each other (since agents might appear, disappear, or move at any time); and they must be able to interact.

Multi-agent systems are composed of two or more intelligent agents. An agent has detectors, effecters and a decision-making mechanism. Using these mechanisms, a multi-agent-based simulation reproduces some phenomena inside online communities, including decision-making process of an agent, local interactions among agents and the system dynamics generated by local interactions, allowing us to observe and understand them (Yamada et al., 2004).

One of the most interesting research topics is the organizational architecture styles of multi-agent system (Kolp et al., 2006) where human organizations is used as a metaphor to suggest a set of generic styles for agent systems. Several architectural styles have been used in the development of multi-agent systems. (Shehory, 1998) describes four such organizations:

- Hierarchical multi-agent systems
- Flat multi-agent systems
- Subsumption multi-agent system
- Modular multi-agent system

3.2 Traditional Multi-Agent System Applications

Multi agent-based simulations have previously been used to analyze social systems, such as public goods problem, traffic simulation, supply chain management and Open Source Software (OSS) development. Public goods problem used the simulation based on Game Theory to explain why we need to provide the public goods. This simulation used an agent that has a simple payoff matrix given by system designers as a consumer.

Traffic simulations were used to analyze how a roadway or a parking space affects traffic flows. This simulation used an agent that has some simple rules given by system designers as a car. These rules were made from the empirical rules of designers, for example, *stop at a red light* and *reduce speed when the distance between two cars is narrow*.

Supply chain management used multi-agent-based simulations to optimize the supply chain network. Supply chain management is a new management technique to minimize total costs that are composed of inventory costs, distribution costs, and so on. This simulation used an agent as an element of a supply chain network, such as manufactures, distributors and customers (Yamada et al., 2004).

The analysis of OSS (Open Source Software) communities used multi-agent-based simulations to understand the phenomena inside of OSS communities. This simulation used an agent as an open source software developer. The agent selected an action from some

rules, which were: *creation of a project*, *participation of a project*, *abandonment of a project* and *continuance of the current collaborations*, based on simple utility function given by system designers (Yamada et al., 2004).

4. Design Issues of the Educational Agent

(Bernard and Sandberg, 1993) proposed that a student in a learning environment should be placed within the framework of surrounding entities that assist the student's access to learning resources, learning by various means, and participating in different learning activities. This idea has been extended to a situation in which educational agents are presented in a social learning environment. In this environment, a student has access to many learning resources, which can be classified into three categories: *content, community, and computational support.* The *content* includes the student's learning materials, such as books, libraries, museums, and databases. The student can participate in *communities* and communicate knowledge with fellow students, teachers, volunteers, and parents. Additionally, the student is empowered by *computational support*, such as calculators, notepads, and different kinds of computer software.

Accordingly, an educational agent is a kind of computational support, which enriches the social context in a social learning environment either by providing virtual participants to enhance the member multiplicity of communities or by supporting facilities to promote communication among real participants.

The positive influence of research on educational agents lies in its ability to reinforce the social learning environment. The research issues can be divided into *design issues* and *implementation issues*. Design issues are concerned with two main elements of defining a particular social learning model— the structure of the model and the protocols of learning activities. Structure relates to the members of the learning environment, including the numbers, roles, and characteristics of the educational agents. The protocol is a set of rules that governs the learning activity, including communication among participants.

The existing social learning model or theory, which is successful for real participants, is usually adopted in designing educational agents. A widely applied theory is Vygotsky's hypothesis, which regards cognitive development as the gradual internalization and personalization of what was originally a social activity (Vygotsky, 1978). Vygotsky also proposed a hypothetical distance, ''*zone of proximal development*'', between development through solving problems independently and potential development through solving problems under adult guidance or in collaboration with more capable peers. Other social learning theories have been recently explored and applied to educational agents, such as *responsibility sharing*, *socially distributed cognition*, *reflection and articulation*, *reciprocal tutoring*, and *learning by teaching* (Chou et al. 2003).

Through different communication media, face-to-face communication among real participants and communication between a student and an educational agent (computer software) have different benefits and constraints. Face-to-face communication is easy to produce, has a high bandwidth and is highly interactive, but cannot be reproduced, distributed, or modified. Meanwhile, the educational agent, as a kind of computer software, is easily reproduced, distributed, and modified, but is difficultly produced. It allows computer interaction, and variable bandwidth of communication. Thus, educational agents and real participants provide a student with alternative social contexts to promote learning.

Various combinations of educational agents and real participants enhance a social learning environment (Chan, 1995). Furthermore, an educational agent can not only directly enforce its perspectives on the student, but also stimulate learning and collaboration among real participants (Chou et al. 2003).

Educational agents with controllable characteristics facilitate particular learning models or learning environments. For example, Aimeur and Frasson proposed a strategy of learning by disturbing, in which an educational agent intentionally gives misleading advice at times (Aimeur & Frasson, 1996; Aimeur et al., 1997). If the system detects that the student's self-confidence is high, the agent provides a mistaken suggestion and waits for the student to explain his or her decision or provide a correct solution. Otherwise, the agent provides correct suggestions to reinforce the student's beliefs.

The strategy "strengthens the attention of the student, increases the perception of details and provides the means for arguing and memorizing" (Aimeur & Frasson, 1996). Chang and his colleagues proposed a learning environment in which a student is surrounded by four kinds of companions—collaborator, peer tutor, troublemaker, and tutee (Chang et al., 1999). These companions have different levels of knowledge, adjusted according to the knowledge of the student. Thus, during learning, the knowledge levels of these companions are adapted to fit the student's expectations of the companion and to support the planned pedagogical strategy (Chou et al. 2003).

Several investigators have suggested the multiple "agents" implementation approach to deal with the complicated tasks and various perspectives of educational agents. Unlike the agents discussed previously in this section, these "agents" are pieces of software, which autonomously deal with some tasks, interact with other software "agents", and hide inside the system. The approach divides the task of implementing educational agents into several subtasks and handles these subtasks by employing several software ''agents''. For example, the mechanism for building a student model can be implemented by constructing spy "agents" and task "agents". The spy "agents" observe and identify the student's actions; the task ''agents'' analyze the action and construct the model (Chou et al. 2003).

5. Research Status

Multi-agent methodology has recently appeared as an alternative to conceive distributed learning applications as a consequence of the evolution of multi-agent technology itself. The main reason of this is that the multi-agent methodology deals well with applications where vital issues, such as: distance, cooperation among different entities and integration of different components of software, are found (Biswas, 2001).

An empirical study done in (Thaiupathump, 1999) to evaluate the effectiveness of intelligent agents in online instruction has shown that agents can improve completion rates of e-learning courses, learner satisfaction and motivation.

Systems designed to aid human learners are usually divided into two categories (Sklar and Richards, 2006):

- *Training*, which generally refers to adult learning of job-related skills, often but not solely in military settings; and
- *Education*, which generally refers to child and adult learning in academic settings, including schools, colleges and universities.

While the terms "education" and "training" may indicate different areas within the field of human learning, from the point of view of computer systems development, they have a lot in common.

There are three main types of agents embedded in human learning environments. The first are *pedagogical agents*, personalized assistants that interact directly with a learner and explicitly guide him through the domain. Pedagogical agents are involved in the teaching component and user interface. Typically they consult the user model in order to understand the student and provide feedback that encourages the learner within his suitable "zone of proximal development".

The second type of agents are *peer learning agents*, the explicit use of agents as interactive partners in the learning process itself. These agents are built into the user interface and, as with pedagogical agents, have knowledge of the user. While these agents may have teaching capabilities, they are typically less engineered for guiding learning explicitly than pedagogical agents.

The third type of agents is *demonstrating agents*, where the agents themselves are interactive mediums for learning; for example, interactive agent-based simulations or educational robotics. These agents represent the domain knowledge and are removed from the other components of the learning system. An open area for future work is the development of systems which combine this third type of agent with one or both of the others (Sklar & Richards, 2006).

Several projects implement learning systems based on multi-agents architectures. For the sake of brevity, it is not possible to present an extended review about the application of agents in the area. Some of them work on a generic platform of agents but usually the focus is given to a specific agent type. Interesting results have been achieved by pedagogical agents regarding the student motivation and companion agents acting sometimes as mediators of the learning process. Finally, tutor agents are usually related to student modeling and educational decision taking.

6. MASCE

The complete Multi-Agent System for Collaborative E-learning (MASCE) system provides functionalities essential in the educational process, such as real-time, as well as offline data and information gathering, analysis and distribution, embedded feedback, assessment, and collaboration.

The first phase of this complete system considers mainly the development of the basic functionality of the Student and Instructor Agents. It focuses on the building of the complete functionality of the Assistant Agent. The analysis and design phase of MASCE is done using Beliefs, Desires, Intentions-Agent Based Software Development (BDI-ASPD). We find desires first from the system requirements and then find their intentions and corresponding beliefs. This idea comes from the natural approach we usually do in the real world. An agent's *beliefs* are a set of data describing the state of the environment. They are the knowledge that *intentions* use to fulfill their goals (*desires*).

The proposed MASCE system considers two types of users; namely students and instructors. Each of these users has a corresponding agent. These are Student Agent and Instructor Agent. Each student is equipped with a Student Agent, which helps the learning process of the student. It manages the student's personal profile and also tracks the student

actions during learning process and updates his profile accordingly. On the other hand, the Instructor Agent provides teaching materials, assesses the progress and participation of different students through quizzes, and manages the progress of the course.
The innovation in the proposed system is the introduction of the Assistant Agent which is initialized as soon as any of the users starts to use the system. It plays a centric role in the proposed system. Thus, the proposed MASCE consists of these three types of agents; the Student, Instructor, and Assistant Agents.

6.1 Student Agent

Each student has the corresponding Student Manager Agent that helps the learning process of the student. It acquires the student's preferences and profile. During the learning process, as the student enrolls in new courses, a dedicated student agent for each course is created.

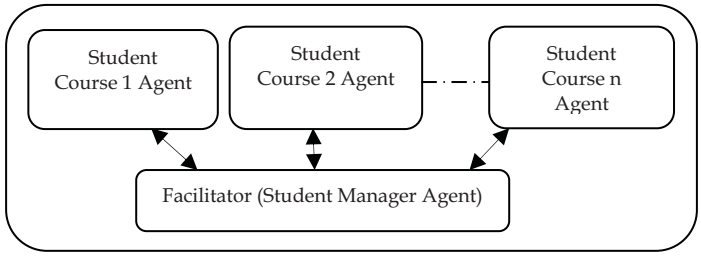

Fig. 1. Student Agent

It tracks the student actions in that course. Accordingly, the tracking mechanism updates the student's profile and preferences. All the student agents of different courses of the same student are under the control of the Student Manager Agent as shown in Figure 1.

6.2 Instructor Agent

The Instructor Manager Agent or simply the Instructor Agent assists the teaching process while interacting with the instructor. It is assigned for each instructor. For each course that is taught by the instructor a dedicated instructor agent is created. It provides teaching materials when requested by Assistant Agent for distributing to students' agents, assesses the progress and participation of different students through quizzes, and manages the progress of the course. All the instructor agents of different courses of the same instructor are under the control of the Instructor Manager Agent as shown in Figure 2.

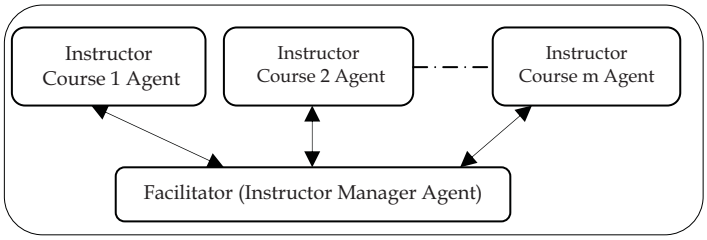

Fig. 2. Instructor Agent

6.3 Assistant Agent

The innovation in the proposed system is the introduction of the Assistant Agent shown in Figure 3 which is initialized as soon as any of the users starts to use the system. The Assistant Manager Agent or simply the Assistant Agent plays a centric role in the proposed system. For each course, a dedicated Assistant Course Agent is created. It has a collaboration mechanism which will be used for "match-making" and "community-building" to help increase collaboration between peers in a certain course. It also gives hints to the instructor of the course to help in the teaching process such as statistics of the results of quizzes and summaries of students' profiles to help in the final grading. It acts a mediator (facilitator) between Student Agents and Instructor Agent of a specific course. After receiving the preferences (goals) of the instructor and the students, it will run autonomously and self-dependently. All the Assistant Agents of different courses are under the control of the Assistant Manager Agent.

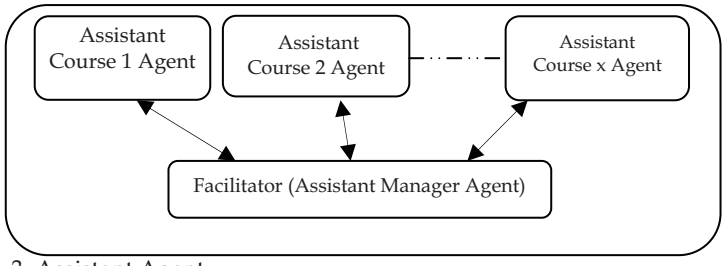

Fig. 3. Assistant Agent

The course material is going to be structured in a hierarchical form where the course is divided into chapters and each chapter is divided into sections which in turn are divided into subsections and so on until we reach the leaves (concepts which cannot be divided any further).

For each of these leaves the following will be provided:

1. Teaching materials
2. Quizzes to test student's knowledge level
3. Students' notes (blogs)
4. Discussion Forums
5. Questions asked by students requiring help

The student can review all the teaching materials provided and add notes to his blogs if he wants to. He will take quizzes after each module to test his understanding (knowledge level)

in that part so as to update his profile. If the student asks a question in a particular section, the Assistant Agent (Match Maker) will try to find the best potential helper for this question who is currently available online, willing and able to provide help. The Assistant Agent uses the students' models in this match making process.

7. Parameters for Modeling

Two classes of parameters are considered for modeling students; namely static and dynamic parameters. Static parameters are collected from the student himself through a questionnaire given to the student when he first uses the system including:

1. Help willingness
2. Initial availability
3. Preferences such as cognitive style, maximum numbers of concurrent discussions
4. Initial belief of the student knowledge level through a simple quiz given to the student to classify him as either: novice, beginner, intermediate or advanced
5. Weighted importance of various attributes: such as if he requires help quickly from any available willing helper, or he would rather wait to be matched with the helper with the best knowledge level in the concept he is asking about

The other parameters needed are dynamic; they are updated dynamically as the student interacts with the system and more new information is collected. Old information may be *outdated* or even *wrong*. For example, after each help session between two students, an evaluation form is presented to each of them to evaluate his colleague. These peer evaluations along with the collected information about the student by the tracking system such as rate of his responses, are all used to update the helpfulness parameter. Different Computational Intelligence (CI) techniques can be used for modeling see for example (Hagras et al., 2007).

The Assistant Agent also groups students together according to their similar preferences and complementing abilities into "buddies groups" to solve group projects or assignments. Thus a community of learners may be built up. So in addition to the one-to-one relationships that grow through the use of MASCE, learners can be arranged into groups with similar concerns who support each other on specific issues. This might be a problem solving session with one or more learners assisting each other with similar problems, or it might be a longer term study group where each member contributes to mutual understanding of the subject matter in some way. Interaction at the one-to-one and the group level is designed to bring learners closer together, and contribute to the evolution of a community of learners anxious to support each other in their learning. Thus the second goal of MASCE which is the promotion of collaboration and knowledge sharing can be fulfilled.

Using this principle of the ready, willing and able helper greatly increases the likelihood of benefit among participants: Helpers are willing, and are not receiving help requests for which they have too little time or knowledge; the helpee is more likely to receive the kind of help he/she requires, and will appreciate the positive attitude of the helper. In return, the helpee may be more keen to assist the helper should an occasion arise where he is in a position to do so (students can add others to a 'friends list', indicating to their agent their particular willingness to help that person in the future). Thus the kind of community of learners that is built in MASCE is designed to benefit everybody which is the third goal of MASCE. This is clear in the case of the helpee, who receives personalized assistance when

facing a problem. In addition, formulating their questions may help some learners to solve their problem for themselves, thus providing them with an additional learning strategy they can apply in the future. Furthermore, it is not only the helpee who benefit from the interaction; the helper also can improve his own understanding of the topic through peer tutoring (learning by teaching).

8. MASCE System Requirements

The Multi-Agent System for Collaborative E-learning (MASCE) is intended to be used to assist teaching and learning process and also to encourage collaboration learning among peers. This system shall be used in a blended learning environment as a supplement to the face-to-face lecture where students can use the system in the lab or from home after attending the traditional lecture in the faculty. Our objective is to incorporate the intelligence of the multi-agent system in a way that enables it to actively and intelligently support the educational processes, where multiple agents can interact to exchange information so that students may collaborate on how best to gain knowledge.
The system consists of the following:

1. Multi-agent software system consisting of three types of agents: Instructor Agent, Student Agents and Assistant Agent.
2. Two types of users (instructors and students) interacting with each other and interacting with collaboration services provided: forums, wikis, blogs, chat rooms, e-mails and also interacting with databases in the system. Typical web browser will be used to access the system such as Internet Explorer.

The student who shall use the system should have the following characteristics:

1. Accept technology, used software tools (discussion forums, chat rooms, etc) and possible problems associated with it.
2. Be prepared for open, honest and fruitful learning.
3. Accept different cultural backgrounds, different skills and different workloads among his buddies group.
4. Troubleshoot problems, report progress to instructor.
5. Meet deadlines.

The instructor who shall use the system should have the following characteristics:

1. Be familiar and comfortable with the used software tools (discussion forums, chat rooms, etc) and possible problems associated with it.
2. Dedicate some of his/her time to be online so as to interact with the students, answer questions posted by students and correct any misconceptions they may have.
3. Troubleshoot problems reported from students.
4. Design courses in a hierarchal form so as to facilitate retrieval of teaching materials, asking questions and testing understanding level of students through quizzes.
5. Motivate the students to use the system by rewarding them (with extra marks for example) for their positive content postings in order to encourage them to make effort when using the system. They can take turns in reading the postings during the week and summarizing them and making a presentation during the next lecture about the important postings.

9. MASCE System Analysis

The Brief External Use Case treats the System as a black box, and shows how the entities outside of the system interact with the system. The reason they are called *external* Use Cases is that they can show how external entities in the environment interact with the system and how the external entities *use* the system to get something done.

Usually, Goals can be extracted from *what* the system is trying to achieve and generally remain constantly throughout the entire analysis and design process. They can be decomposed into finer and finer-grained goals. We can find a global goal for a system and a set of intentions to achieve the global goal. We implement these intentions that in turn become the sub-goals of other intentions, and so on and on. That gives us an idea that we can decompose goals from top to bottom, step by step. During the Brief External Use Case process, we identify the services of the system being developed from an external point of view; we do not describe the internal workings, components, or design of the system. Through this task, we determine what services our system should provide. The Brief External Use Case captures *who* (actor) does *what* (interaction) with the system, for what *purpose* (goal), without dealing with the system internals.

The system for which we have written the requirements is a Multi-Agent System for Collaborative E-Learning (MASCE) whose overall goal is to enhance the learning and teaching process and to facilitate collaborative e-learning among peers. This system shall be used in a blended learning environment as a supplement to the face-to-face lecture to help students:

1. Review lecture slides, audio, video and view recorded instructor's notes.
2. Exchange views, have discussions and receive help from best-helper matches.
3. Most importantly collaborate with their peers within buddies group recommended by the agents in the system.

One key component that is missing in today's multi-agent systems in education is to enable the system to utilize and analyze the observed behavior collected from individual agents and subsequently adapt to such behavior. *Our objective is to incorporate the intelligence of the multi-agent system in a way that enables it to actively and intelligently support the educational processes,* where multiple agents can interact to exchange information so that the students may collaborate together on how best to gain knowledge. The Brief external use case for MASCE is given in Table 1.

Enhance the collaborative e-learning process

1. Instructor logs to the system and is given a user name and a password for administering particular course(s).
2. Instructor adds teaching materials (slides, audio, video of lectures and notes) to the system.
3. Student logs to the system and is given a user name and a password for authentication in a specific course.
4. Student is presented with a questionnaire asking about his general information, preferences and learning styles.
5. Student is given a quiz to evaluate his background of the course to be taken.
6. Student can view all available teaching materials: lecture slides, audio and video of the lectures, instructor's notes written on the Tablet PC and wikis.
7. Student can view others' blogs and write his own.
8. Student can write postings on the discussion forums.
9. Whenever the student needs help in a particular section of the course, he can post a question on the discussion forum.

10. The system (MASCE) finds the best potential helper to assist the student in that particular part.
11. After each help session, each of the helper and helpee are provided with an evaluation form to evaluate his peer.
12. The system (MASCE) gathers the students into buddies groups for solving group assignments and projects in that course.
13. Members of the same group are approached frequently and evaluate each other at the end of the assignment or project.
14. The system keeps track of the number of times each item of the teaching materials is viewed by the student.
15. The system keeps track of the materials added by the student to his blogs (notes, papers, etc….).
16. The system keeps track of the number of times the student accepts help requests.
17. The system keeps track of the number of times the student neglects help requests.
18. The system keeps track of the number of the student's postings on discussion forums.
19. The system keeps track of the number of the questions asked by students.
20. This monitoring information is used to update student model.
21. The instructor can log on to the system at predefined times so as to have synchronous interaction with his students in the discussion forums.
22. While instructor is online, he can correct any misconceptions that the students may have.
23. The system sends the questions posted by the students for the instructor to answer them when he is online.
24. The system dispatches instructor's responses to students who asked these questions through private emails.
25. If there are common questions, the answers are posted on the discussion forums for the benefit of all the students.
26. Instructor can also check the progress of the groups formed to solve assignments or do projects together and which have been recommended by the system using clustering and match making algorithms.
27. Instructor posts quizzes on the system after each module to evaluate the understanding of the students.
28. The system collects the students' answers to quizzes, grades them and performs statistics.
29. At the end of the course, the system sends a summary of the student profile and progress during the course for the instructor to help in the final grading.
30. At the end of course, the student is provided with a questionnaire to evaluate his learning experience.

Table 1. External Brief Use Case

The External Brief Use Case of MASCE in Table 1 has the overall goal of enhancing the collaborative e-learning process. After analyzing this main goal, we can divide it into several sub-goals. The headings of the sub-goals which are identified from the above processes are the following:

- Add teaching materials
- Collect initial information about student
- Present teaching materials and collaboration tools to student
- Help a student having a question in a particular part of the course to collaborate with peers
- Group students to solve assignments and perform group projects
- Monitor student behavior
- Assist instructor in interacting with students
- Assist instructor in evaluating students' understanding and in final grading
- Get student evaluation of his learning experience

10. MASCE System Design

From this point, we shift from the analysis to the design phase. During agent-based design, there is an emphasis on defining software agents and how they collaborate to fulfill the requirements. The design phase is the most difficult part in the software engineering. It involves elements of both science and art. In this approach, we will utilize some agent patterns suggested by (Einhorn, 2002). There are three of them: agent identification patterns, agent creational patterns, and agent goal assignment patterns.

The *agent identification pattern* tries to identify common agents in agent software systems, which includes three sub patterns; namely manager agent pattern, service agent pattern, and broker agent pattern. The manager agent pattern suggests creating a manager agent between the interacted internal and external agents. Hence, the main duty of the manager agent is to communicate the external agent with multiple internal agents. Not only it can locate the proper internal agents based on its interaction with the external agent, but also removes the need for the external agent to know detailed information about internal agents. In large software systems analysts can create several manager agents to bridge the external and internal agents. If a system has multiple manager agents, analysts can build a special manager agent, named delegation agent, above these manager agents.

The service agent pattern guides us to create a service agent based on certain kind of service. For instance, analysts could create a database access agent for a database access service. So, other agents can access database through this database access agent. The broker agent pattern suggests if several different agents provide similar services, they can register their services with the broker agent. Then they can communicate with the broker agent when they want to use those services. After the broker agent has the request, it will choose the proper service agent to serve the requester.

The second category pattern is the *agent creational pattern*. It suggests creating an agent while a system needs, such as the long-lived agent. Whenever the system is running, the long-lived agent is needed. It is going to handle the start and shutdown processes for the system.

The third category pattern is the *goal assignment pattern*, which suggests us remain the coupling low between agents when analysts assign goals to agents. An agent, with vastly different goals, can be treated as an overly complex agent. It would be better to decompose it into several smaller agents.

The Internal Use Case concerns interactions among elements inside the system. The reason we call such Use Cases *internal* Use Cases is because they can show how entities interact to the system internally and how the entities *use* each other to get things done. The purpose of this step is to identify the *intentions* (plans). Based on the external use cases, we decompose the scenario of each use case into details. That will help us find *intentions* for the goals and discover more roles that are not easy to be seen from the external point of view. In order to create the brief internal use cases, we read the external use cases and decompose their scenarios while keeping the rest of the parts unchanged. In our research, a role includes a particular goal or set of goals and a set of intentions. It will be mapped to an agent who is responsible for satisfying those goals. An example of the internal Use Case is shown in Table 2 for Add Teaching Materials sub-goal.

Use Case ID	UC-0101
Use Case Name	**Add Teaching Materials**
Goal	Add teaching materials to system
Primary Actor	Instructor
Stakeholders	Instructor and students
Pre Conditions	1. Instructor has slides of the lectures together with his notes recorded on a Tablet PC after each lecture 2. Audio and video of lectures are recorded 3. Courses are designed in a hierarchal form to facilitate retrieval of teaching materials
Post Conditions	Teaching materials are available on the system for use by the students.
Main Success Scenario	1. When instructor uses the system for the first time, Login Service is launched. 2. Instructor first determines that he is an instructor, a login form is displayed and then instructor inputs a user name and a password for administering particular course(s). 3. Login service checks if the user name and password matches login rules. 4. The user name and password are stored in the system for authentication later on. 5. Instructor launches Teaching Materials Service which displays an input form. 6. Instructor uses Teaching material Service to add (slides, audio, video of lectures and notes) each in its place in the course hierarchy.
Extensions	3a. User name or password chosen do not match rules 3a.1 Instructor is asked to enter other user name or password 3a.2 Instructor chooses other user name or password until they are accepted

Table 2. Add Teaching Materials Internal Use Case

11. Clustering and Matching

Clustering algorithms can be classified into (Jian et al. 1999):

- Exclusive Clustering
- Overlapping Clustering
- Hierarchical Clustering
- Probabilistic Clustering

In case of *Exclusive Clustering*, data are grouped in an exclusive way, so that if a certain datum belongs to a definite cluster then it could not be included in another cluster. On the contrary the second type, the *Overlapping Clustering*, uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership. In this case, data will be associated to an appropriate membership value.

On the other hand, a *Hierarchical Clustering* algorithm is based on the union between the two nearest clusters. The beginning condition is realized by setting every datum as a cluster. After a few iterations it reaches the final clusters wanted. Finally, the last kind of clustering,

Probabilistic Clustering uses a completely probabilistic approach. In MASCE, *Hierarchical Clustering* is used for clustering and matching between students.

11.1 Hierarchical Clustering Algorithms

Given a set of N items to be clustered, and an N*N distance (or similarity) matrix, the basic process of hierarchical clustering as explained in (Johnson, 1967) is this:

- *Step 1:* Start by assigning each item to a cluster, so that if you have N items, you now have N clusters, each containing just one item. Let the distances (similarities) between the clusters the same as the distances (similarities) between the items they contain.
- *Step 2:* Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one cluster less.
- *Step 3:* Compute distances (similarities) between the new cluster and each of the old clusters.
- Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

Step 3 can be done in different ways, which is what distinguishes *single-linkage* from *complete-linkage* and *average-linkage* clustering. In *single-linkage* clustering (also called the *connectedness* or *minimum* method); we consider the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster. If the data consist of similarities, we consider the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster (Berkhin, 2002).

In *complete-linkage* clustering (also called the *diameter* or *maximum* method), we consider the distance between one cluster and another cluster to be equal to the greatest distance from any member of one cluster to any member of the other cluster. In *average-linkage* clustering, we consider the distance between one cluster and another cluster to be equal to the average distance from any member of one cluster to any member of the other cluster.

11.2 Steps for Clustering

First we get the parameters of students then we get the weights with which we will multiply the parameters to calculate the function on which clustering is based. Then we will get the type of the cluster we will deal with: if the instructor wants to cluster students with the highest calculated function together or if the instructor wants to cluster some high students with some low students or if instructor wants to put maximum number for each cluster. Sometimes the number of students in each group must be equal to the maximum number specified from the instructor, and in other times instructor may not need the maximum number of the students to be in each group.

Needed Parameters

Instructor will specify the parameters we will use to cluster like: if he wants to use the last year degree and how are the responses of each user and numbers of answers in that point and the interesting fields of each user or the rating of question of each student in this point and so on. After getting student parameters, we will normalize these parameters. We will get these parameters from the collected database of each user using queries.

Calculation of Weights

Instructor will also specify the parameters which he/she wants to have the larger weights or smaller weights like if the instructor wants to put large weights on the previous year degree

or something else. After getting student names and their parameters and weights, we will then multiply these parameters with their selected weights to get the function which would be clustered.

Available Data

For testing our methodology for matching and finding the proper helper, we used www.ee2008.com as a source for historical data which is a simple forum for students in Electrical Engineering Department of Ain Shams University, Cairo, Egypt. We analyzed 726 posts during 25 days covering: 5 Courses, 20 Chapters and 61 Students.

12. Time-Matching between Students as an Example

Time-Matching between Students can be used as a measure of the availability. The availability will be used as an example of a parameter that is set initially from the user (student) and then will be modified dynamically according to the interaction of the student with the system. Time-matching process is to find a time that suits two or more students. Let's start with studying two students logging time (plotting the results of the most recent posts of the two users). When we plot the same number of posts for two users this means that the areas surrounded with their patterns is the same as shown in Figure 4.

Matching a third student with them is to match the resultant intersection line of the two students with the third student line and so on. This makes the intersection area of the whole students' lines gets smaller and smaller, agreeing with the fact that it gets harder to arrange for an appointment when the number of persons increase.

It is easy to recognize the maxima of intersection from the graph but we need to find an algorithm that finds the maximum of the intersection (or the time period where it is located) numerically. Locate the recent N posts for each user over hours of the day (or giving the recent post higher weights). The period of intersection we get is the best period during which the two students can meet.

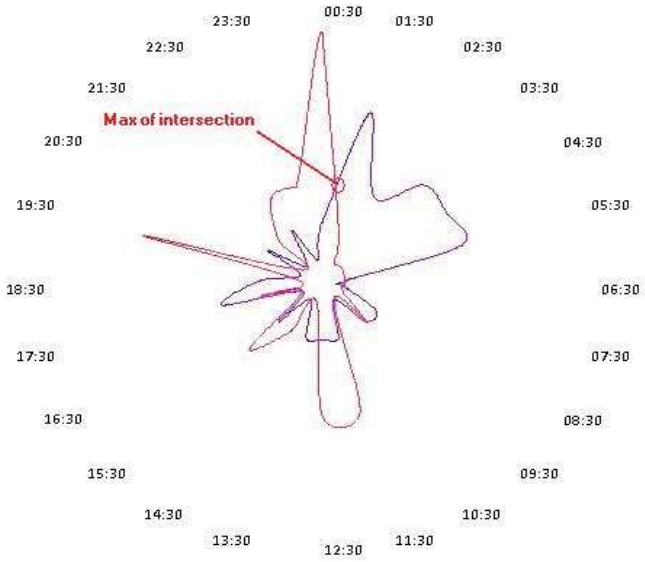

Fig. 4. Time Matching between Two Students

Applying *cross-correlation* calculation over the day will give a value representing how often they can meet all over the day time. This can be very helpful if we are going to make teams out of students as we need to group students who have a higher cross correlation value over the whole day time together. It can also be used to introduce "how often can you meet someone" value, if the student is going to choose one of many other students who can help him.

STUDENT	dony	ehab	eking	eng	esraa	fahum	fatta7	flower
dony	1.75	0.6667	1.0714	0.875	0.6667	0.5	1.4231	1
ehab	0.6667	2.4375	1.4286	0.8125	1.4444	0.3333	1.7628	0.25
eking	1.0714	1.4286	2.0204	1.5	1.381	1.0714	1.2747	0.2857
eng	0.875	0.8125	1.5	3.8125	1.5833	1.25	1.1154	0.25
esraa	0.6667	1.4444	1.381	1.5833	3.3333	0.6667	1.5128	0
fahum	0.5	0.3333	1.0714	1.25	0.6667	1.75	0.0769	0
fatta7	1.4231	1.7628	1.2747	1.1154	1.5128	0.0769	3.3432	1
flower	1	0.25	0.2857	0.25	0	0	1	1

Table 3. Cross Correlation between Students' Logging Times

We can expect that the maximum correlation for a user with another one to occur when the two users log to the system with the same timings manner i.e. calculating correlation between a student and himself gives a maximum correlation which can be shown in Table 3.

Example of finding the best helpee for Chapter 3 in the course "Optimal Control" is shown in Figure 5.

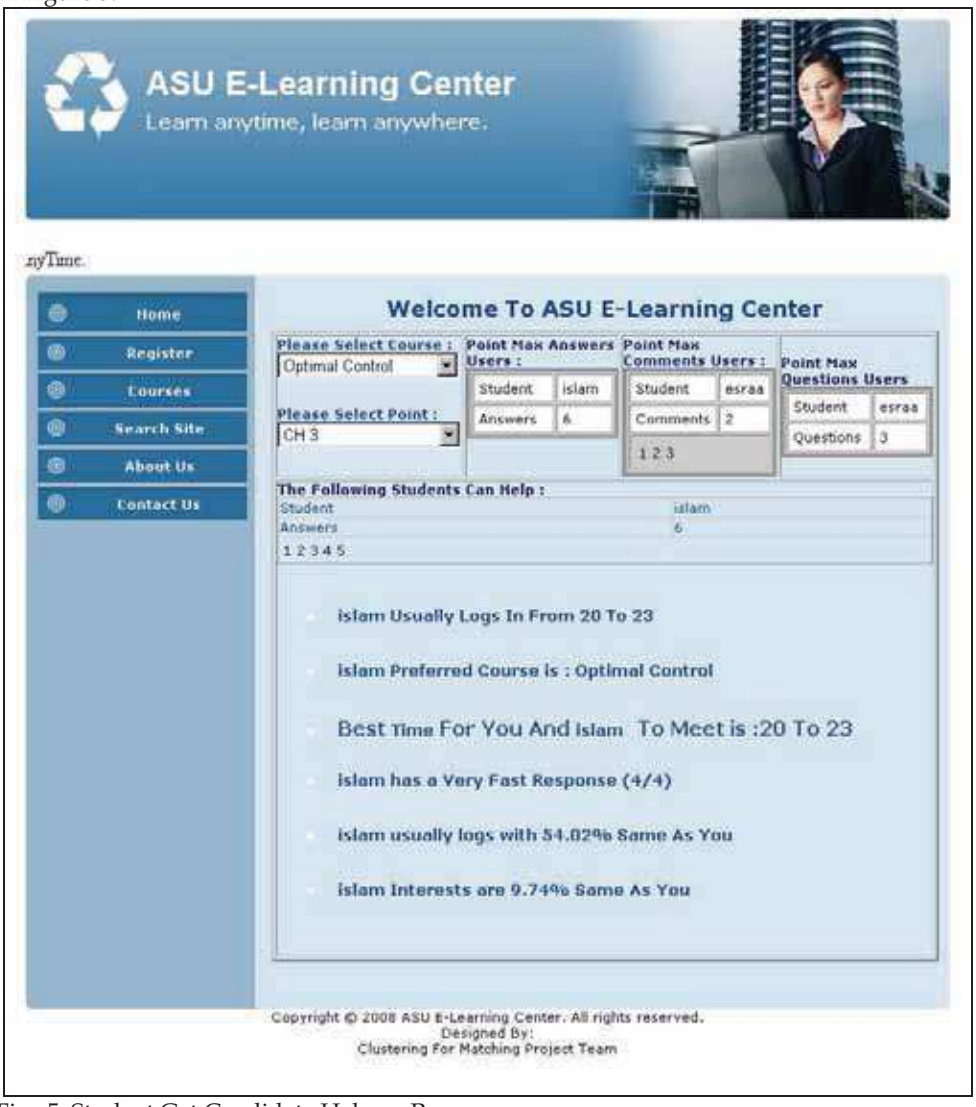

Fig. 5. Student Get Candidate Helpers Page

13. Conclusion

This chapter outlined the development and the implementation processes of a Multi-Agent System for Collaborative E-learning (MASCE). In MASCE, intelligent agents are employed to build a computer-aided collaborative learning and teaching environment. Clustering and match-making algorithms are used to build communities for facilitating learning from peers.

MASCE considers two types of users; namely students and instructors. Each of these users has a corresponding agent. These are Student Agent and Instructor Agent.
The innovation in the proposed system is the introduction of the Teaching Assistant Agent which is initialized as soon as any of the users starts to use the system. It plays a centric role in the proposed system. It can also track the user's likes and dislikes in different areas, his activities and then can nominee a peer for the user to get help.
The agents in MASCE are designed using a modified BDI (Beliefs, Desires and Intentions) architecture. Desires are determined first from the system requirements and then its intention and corresponding belief are found. This idea comes from the natural approach we usually do in the real world. Two kinds of use cases were used during designing of the system. External use cases are used for discovering the functions or services that the system should provide. Internal use cases are used for identifying plans (intentions), goals (desires), and their beliefs from the system services discovered from the external use case.
Hierarchical clustering algorithms are used for matching to find the best candidate helper for a peer according to the parameters collected by the system either from the user himself through a questionnaire or through user interaction with the system. The data used is the questions and answers posted to a simple forum for students in Electrical Engineering Department of Ain Shams University, Cairo, Egypt. 726 posts were analyzed during 25 days.
Due to time limits, the matching and grouping processes are done for the time being using the Teaching Assistant Agent. However, in the future work we want to build separate Student and Instructor Agent to be responsible about performing services in the client machines.
This can be done on two phases. In the first phase all agents will be executed on the server's side. In the next phase, the Student Agent will be located on the student's PC working as a client program. The same also goes for the Instructor Agent which will be located on the instructor's machine and they will communicate with the Assistant Agent residing on the server to provide different services for the users of the system (both students and instructors). Any announcements, updates or answers for questions will appear to the user as a pop up while working on his computer instead of having to log on to the site to check every now and then if someone answered his question for example.

14. References

Aimeur, E. & Frasson, C. (1996). Analyzing a new learning strategy according to different knowledge levels, *Computer and Education*, 27(2), 115–127, 1996.

Aimeur, E. ; Dufort, H.; Leibu, D. & Frasson, C. (1997). Some Justifications for the Learning by Disturbing Strategy. *Proceedings of AI-ED 97 World Conference on Artificial Intelligence in Education*, Kobe, Japan, 119–126, 1997

Berkhin, P. (2002). *Survey of Clustering Data Mining Techniques*, Accrue Software

Bernard, Y., & Sandberg, J. (1993). Open learning environment: what support should they offer? Proceeding of International Conference on Computers in Education, Taipei, Taiwan, 156–161.

Biswas, G.; Katzlberger, T.; Bransford, J. & Schwartz, D. (2001). Extending Intelligent Leaning Environments with Teachable Agents to enhance Learning, *In: Artificial*

Intelligence in Education, Moore, J. D; Redfield C. I. & Johnson. W. L. (Ed.), San Antonio, TX: IOS Press, Amsterdam, Netherlands

Cerri, S. A. (2002). Human and Artificial Agent's Conversations on the GRID, *Proceedings of 1st LEGE-WG International Workshop on Educational Models for GRID Based Services,* 2002, Switzerland

Chang, L. J.; Wang, J. C.; Hsu, B. Y. & Chan, T. W. (1999). Four Applications of Student Modeling—My Animal Companions. *Proceedings of the Third Global Chinese Conference on Computers in Education,* Macau, 366–370, 1999.

Chen, S. Y. & Chiu, M. L. (2005). Building an Agent-Based System for e-Learning in Digital Design," *Computer-Aided Design Applications*, Vol. 2, 2005, pp 469-476

Chou, C. Y.; Chan, T. W. & Lin, C. J. (2003). Redefining the Learning Companion: the Past, Present, and Future of Educational Agents. *Computers & Education*, Vol. 40, 2003, pp. 255–269

Einhorn, J. M. & Jo, C. H. (2003). A Use-Case Based BDI Agent Software Development Process, *Proceedings of the 2nd International Workshop on Agent-Oriented Methodologies - OOPSLA-2003*, Anaheim, CA, USA, Oct. 26- 30, 2003.

Hagras, H.; Doctor, F.; Callaghan, V. & Lopez, A. (2007). An Incremental Adaptive Life Long Learning Approach for Type-2 Fuzzy Embedded Agents in Ambient Intelligent Environments. *IEEE Transactions on Fuzzy Systems*, Vol. 15, No. 1, February 2007

Jain, A. K.; Murty, M. N. & Flynn, P. J. (1999). Data Clustering: A Review. *ACM Computing Surveys*, Vol. 31, No. 3, September 1999

Johnson, S. C. (1967). Hierarchical Clustering Schemes. *Psychometrika*, Vol. 32, No. 3, Springer- New York, 1967

Kolp, M., Giorgini, P., & Mylopoulos, J. (2006). Multi-Agent Architectures as Organizational Structures. *Autonomous Agents and Multi-Agent Systems*, Vol. 13, No. 1, Springer, 2006

Mahdi, H. & Attia, S. (2008 a). Finding Candidate Helpers in Collaborative E-Learning using Rough Sets, *Proceedings of International Workshop on Collaborative E-Learning Using Computational Grid held in collocation with CISIM 2008, The 7th Computer Information Systems and Industrial Management Applications*, pp. 287-292, Ostrava, Czech Republic, June 26 –28, 2008

Mahdi, H. & Attia, S. (2008 b). Small Dataset Size Clustering in MASCE, *Proceedings of the 2008 International Conference on Frontiers in Education: Computer Science and Computer Engineering*, Monte Carlo Resort, Las Vegas, Nevada, USA July 14-17, 2008

Mendez, R. A. (1999). Towards a Standardization of Multi-Agent System Frameworks, www.acm.org/crossroads/xrds5-4/multiagent.html. Last Accessed April 15, 2009. Quah, J. T. S.; Leow, W. C. H. & Chen, Y. M. (2002). Mobile Agent Assisted E-Learning, Proceedings of *ICITA,* 2002

Rosen, A. (2009). *E-Learning 2.0: Proven Practices and Emerging Technologies to Achieve Results*, AMACOM, 2009

Sklar, E. & Richards, D. (2006). The Use of Agents in Human Learning Systems, *Proceedings of the Fifth International Joint Conference on Autonomous agents and Multi-Agent Systems*, Japan, 2006

Shehory, O. (1998). *Architectural Properties of Multi-Agent Systems*. Technical Report CMU-RI-TR-98-28, the Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

Sugumaran, V. (2009). *Distributed Artificial Intelligence, Agent Technology, and Collaborative Applications,* IGI Global, 2009

Thaiupathump, C.; Bourne, J. & Campbell, J. (1999). Intelligent Agents for Online Learning. Journal of Asynchronous Learning Networks JALN, Vol. 3, Issue 2, November 1999

Vygotsky, L. (1978). Mind in Society: The Development of Higher Psychological Processes, Cambridge: Harvard University Press, 1978.

Wooldridge, M. (2002). *Introduction to Multi-Agent Systems,* John Wiley & Sons; 1st ed., 2002. Lecture slides available online: http://www.csc.liv.ac.uk/~mjw/pubs/imas/ Last Accessed April 15, 2009

Yamada, K.; Nakakoji, K. & Ueda, K. (2004). A Multi-agent Systems Approach to Analyze Online Community Activities, *Proceedings of the Fourth International Conference on the Advanced Mechatronics (ICAM'04), 2004*

Yang, C. & Ho, H. (2005). A Shareable e-Learning Platform Using Data Grid Technology, *Proceedings of IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05),* pp. 592-595, 2005

E-learning

Edited by Maja Jakobovic

ISBN 978-953-7619-95-4
Hard cover, 312 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

E-Learning is a vast and complex research topic that poses many challenges in every aspect: educational and pedagogical strategies and techniques and the tools for achieving them; usability, accessibility and user interface design; knowledge sharing and collaborative environments; technologies, architectures, and protocols; user activity monitoring, assessment and evaluation; experiences, case studies and more. This book's authors come from all over the world; their ideas, studies, findings and experiences are a valuable contribution to enriching our knowledge in the field of eLearning. The book is divided into three sections. The first covers architectures and environments for eLearning, while the second part presents research on user interaction and technologies for building usable eLearning environments, which are the basis for realizing educational and pedagogical aims, and the final last part illustrates applications, laboratories, and experiences.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Hani Mahdi and Sally S. Attia (2010). Developing and Implementing a Multi-Agent System for Collaborative E-learning, E-learning, Maja Jakobovic (Ed.), ISBN: 978-953-7619-95-4, InTech, Available from: http://www.intechopen.com/books/e-learning/developing-and-implementing-a-multi-agent-system-for-collaborative-e-learning

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821