Supporting a Learner Community with Software Agents

Neil Taurisson and Pierre Tchounikine
Laboratoire d'Informatique de l'Université du Maine
Avenue Laennec, 72085 Le Mans Cedex 9, France
neil.taurisson@lium.univ-lemans.fr
pierre.tchounikine@lium.univ-lemans.fr

Abstract

This paper describes a multi- agent approach that aims at supporting learners involved in a collective activity. We consider pedagogical situations where students have to explicitly define the articulation of their collective work and then achieve the different tasks they have defined. Our objective is to support these students by taking some of these tasks in charge whilst making them work out such organisation features. For this purpose, we propose to consider that the group of students forms a multi- agent system and to introduce software agents that can achieve some of the tasks within this group. This conducts students to tackle the problem in terms of human and software agent coordination. We present how this approach can be conceptualized and modelled using Engeström's triangle, how task delegation can be used as a means to enable students to define the software agents' behaviour and the multi-agent implementation we propose.

Keywords

Collective learning, Activity theory, Multi- Agent System

Introduction

The general context of our work is the study of collective activities in a learning context (CALC; Betbeder & Tchounikine, 2002). The term "activity"will be used in this paper to denote both what students are asked to do, i.e. the "pedagogical activity" and in the broader sense of "students' activity". Additional precision will be made when the context is not clear enough to distinguish these meanings in learning contexts. More specifically, we tend to support distance students in defining the articulation of their collective work in a way that leads them to have a reflective analysis of their activity. For this purpose, we have developed Symba (Betbeder & Tchounikine, 2003), a Web-based environment that helps students to organize themselves. However, analyzing students' interactions about their organization and activities highlighted that a large part of these interactions are in fact not interesting, neither from the point of view of the learning objectives nor from the social point of view of making students aware of each other. It therefore appears interesting to support students by taking this part of the work in charge.

In this paper we present an approach based on introducing software agents into the group of students with the objective of (1) supporting students by taking some uninteresting parts of the work in charge whilst (2) enhancing the pedagogical objective (i.e., that students should work out organisational features) by proposing a protocol focusing on organisational features.

Section 2 presents the general context and objectives of this work and the multi-agent approach we propose, section 3 describes how we use Engeström's triangle (from the activity theory) to conceptualize the way we integrate the agents in the community, section 4 discusses how we have put this approach into practice and section 5 talks about the multi agent implementation we propose.

Context, objectives and general approach

The Symba framework

The Symba framework (Betbeder & Tchounikine, 2003) is a Web-based environment for CALCs designed to emphasize and render explicit organisation features in order that students should have a reflective activity on their organisation.

A typical example of the type of collective activities we consider is that of a group that has to (1) identify how conceptual maps can be used to describe a curriculum, (2) propose different individual points of view of their curriculum and then (3) construct collectively, from their individual productions, a conceptual map describing

their curriculum. Such activities are designed with a double pedagogical objective, making students address some feature of the curriculum domain (conceptual maps) and making students practice collective work and, in particular, work out how they have to organise themselves: decompose the work to be done into different individual and collective tasks, organise these tasks (beginning and ending dates, input, output, etc.), define what tools should be used to perform each of them: file transfer tool, mail, chat, Forum.

Symba presents two environments. The organisation environment enables students to define the set of tasks to be realized, delegate these tasks to an individual or a subgroup (eventually the entire group itself) and specify the tools and resources that are required to achieve these tasks. The activity environment then provides the tools and resources that have been asked for at the organization level.

Requirement for some additional support

In Betbeder, Taurisson & Tchounikine (2003) we presented three different levels of support that could be proposed to a CALC community with the different kinds of tools that could provide these particular levels of support:

- The "conceptual support" provided by conceptual tools (models and methodology). For instance, in Symba, the proposed conceptual tool is the task notion.
- The "task achieving support" provided by general purpose tools. For instance, Symba proposes different general tools such as a chat, a mail or a file exchange tool, general tools that support users in achieving some tasks.
- The "task handling support" provided by tools that are able to take the achievement of a given complex task in charge e.g., organize a vote.

In this paper, we consider this "task handling support". The importance of proposing this type of support was raised by an analysis of the different chat, mail and forum discussions from different experiments through a communication - acts grid (Betbeder & Tchounikine, 2002). This analysis highlights that although a great amount of communication-acts are related to organisational features (i.e., concern our core pedagogical objective), a large part of these interactions are in fact not interesting, neither from the point of view of the learning objective nor from the social point of view of making students aware of each other. A typical example is the finding of a date for a synchronous meeting, which requires centralizing each participant's availability to propose a date that seems to be suitable. For such a task, one of the students must achieve a boring and time consuming job (multiple cycles of sending mails and compiling answers) for the community benefit. Such tasks, which have been analyzed in the role conflict theory (Watt, 1993) as potentially causing breakdowns of the community, should be avoided.

The community as a multi agent system

In order to avoid students spending time on features that are pedagogically counterproductive frameworks usually propose tools that deal with these features. However, the fact that the CALC environment should provide task - handling support to avoid community breakdowns caused by role conflict must not come into conflict with our pedagogical objective: get students to have a reflection on their organization. Therefore, articulation tasks such as the finding of a date must not be hidden by the CALC or compiled into a black box.

The approach we propose is to provide task- handling support by integrating software agents that can be delegated these specific tasks, into the comunity. This way, tasks potentially causing role conflict are handled by the environment whilst remaining explicit in the organization. Within this approach, students do not use tools, they define a task to be handled by a software agent, in a similar way that they define some other tasks to be handled by human agents. The emphasis thus remains on organisational features (defining and delegating tasks), i.e., the point we want them to work out. A tool is viewed by its user as a manner of doing something. The relation between a tool and its user is individual: what can I do with it, how can I use it? Very differently, the relation between actors is collective: what can I do with him, how can we work together?

Within our approach, the setting of a collective activity is therefore conceptualized as a multi agent system where human agents are collaborating to achieve a common goal (Dillenbourg, Baker, Blaye & O'Malley, 1996). Coherently with the distributed cognition theory (Hollan, Hutchins & Kirsh, 2000), a group of students involved

in a CALC must not be analyzed as an addition of individual agents: the group must be considered as a cognitive system.

Coordination aspects

Differently from software agent communities, the articulation of the collective work of such a human community cannot / must not be modelled and automated a priori (Suchman, 1987; Dillenbourg, Baker, Blaye & O'Malley, 1996). Moreover, we do not attempt to automate coordination like multi-agent architectures do, we want the students to work it out. The agents we intend to introduce in the community are therefore limited to articulation tasks, e.g. finding a date, organizing a vote or dealing with file exchanges. Coordination remains a human-concerned feature: the work to be achieved is organized by a community composed of the students and it is achieved by an extended community composed of the students (considered as human agents) and the software agents. However, although the software agents are not in charge of the control, their impact on the overall organisation must be studied carefully.

Guidelines for specifying and analyzing the agents' role can be found in Schmidt and Simone's work on coordination mechanisms, that they define as "a construct consisting of a coordinative protocol on the one hand and on the other hand an artefact in which the protocol is objectified" (Schmidt & Simone, 1996). Every collective activity is associated to a coordination protocol, although it can remain implicit. Schmidt and Simone's work highlights the interest of reifying some coordination aspects in an artefact in order to make them accessible and non-ambiguous to all members of the community. It points out different important features such as the fact that some articulation tasks of the protocol should be automated so as to relieve actors from always having to pay attention to the respect of the protocol, that the protocol should be flexible enough in order to allow local changes without requiring the whole community to redefine it or that the protocol cannot be defined entirely before the activity takes place, it should be possible to start with an imprecise protocol that will be defined more precisely in action.

Within our approach, the artefact reifying the protocol is the Symba organization environment, where every task can be browsed through by all the members of the community. The software agents, although limited to articulation tasks, manage however a certain part of the protocol, relieving human agents from having to permanently watch that, for example, the time schedule is respected by all. Following Schmidt and Simone's principles, the software agents must be flexible enough so that students experiencing temporary problems should not be rejected from the activity. As an example, this means that the agent in charge of collecting documents must be tolerant for latecomers and accept (to a certain extent) documents provided after the deadlines.

From a general point of view, integrating software agents to a community imposes a certain level of rigidity as software agents cannot implicitly understand the community organization as humans do: integrating software agents requires an explicit definition of the community work articulation. This could be seen as a negative point if our objective were limited to facilitating the collective work. However, as our principal goal is to encourage students to work out organization features, such explicitness in fact participates positively to the clarification of the organization.

Conceptualisation

Theoretical background: the activity theory

Software agents cannot be introduced in a human community (in particular a students' community) without analyzing the impact from the humans' point of view. We must understand how learners can perceive the agents and their impact on the work to be achieved in order to study the agents' characteristics and their status in the group / in the framework.

The activity theory is a general framework for studying different forms of human activity as development processes (Kuutti, 1996). Issued from the soviet cultural historical research tradition, it has recently come back to the front of the scene as a theory that provides conceptual tools to describe and analyze an activity as a social development process. Within our context, the activity theory is particularly interesting in that it postulates that an activity has to be analyzed both as an individual process and a social process.

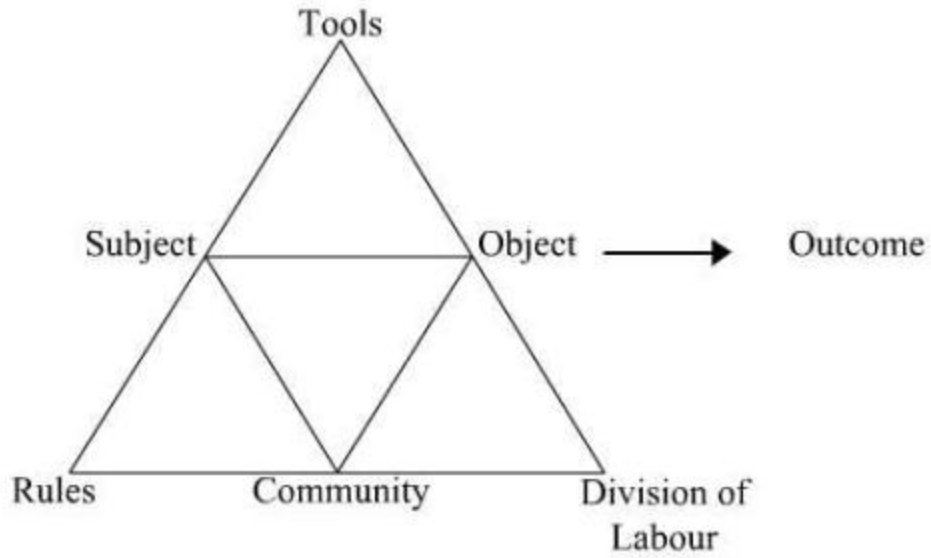

Figure 1: The Engeström activity triangle (Engeström, 1987)

Within this general context, Engeström (1987) proposed a model (Figure 1) emphasizing that an activity is oriented by an object shared by its community. What motivates the existence of an activity is the willingness of its subject (which can be a collective subject, eventually the entire community) to transform this object into a product. What makes this model attractive for analyzing an activity is that it is a good structure to isolate elements of different natures constituting the context of an activity and to highlight mediation relationships between elements. As an example, the subject is related to the community by a set of rules, the relationship between the subject and the object is mediated by a set of tools (tool is meant here at a conceptual level: it can be a document or, within our context, a chat.), the community is related to the object in the context of a given division of labour, etc. (Figure 2). As highlighted by Lewis (2000), focusing on triads of the Engeström model is a way to understand the impact of one element of the activity on another by the mediation of a third. For our analysis of the integration of software agents to the students' community, the triads from the down part of the Engeström triangle are particularly interesting as they focus on the collective side of the activity.

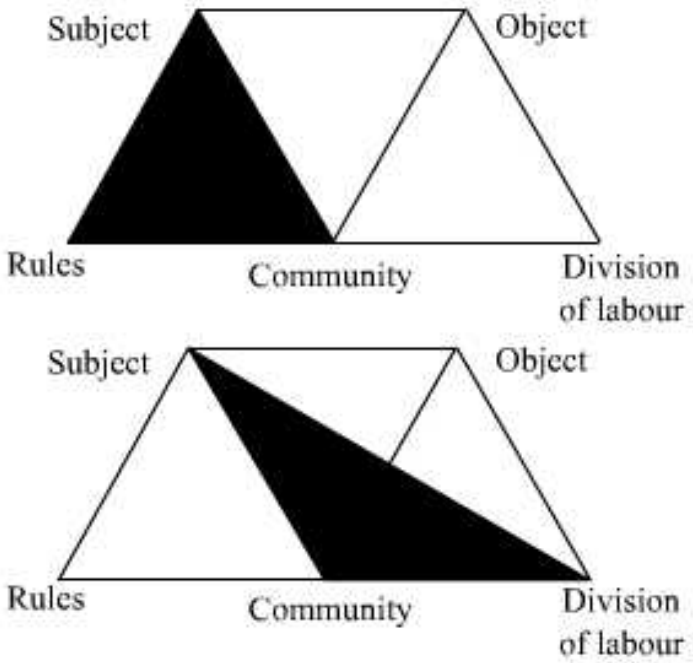

Figure 2: Focusing on triads of the Engeström triangle

Within our research we state as a working hypothesis that, although the activity theory is a descriptive framework for analysis, its notions can be used in a more prescriptive way to design computer-based

frameworks. Following this hypothesis, we use the Engeström triangle as a conceptual framework to both (1) analyze the integration of software agents in a CALC and (2) define the protocol followed by students to describe their (and the agents') activity, task modelling and task delegation.

Studying the impact of the integration of software agents into a CALC community

In order to understand the possible impacts of the software agents on the learners' activity we focus on the down part of the Engeström triads, that are oriented at the collective side of the activity. For example, we analyze the impact of a software agent (considered as a subject) on the community in terms of imposition of rules or modification of the division of labour. From these triads we can anticipate features such as described below:

- Defining an agent's task should make rules and division of labour explicit. Defining a task to be handled by a software agent requires specifying what rules it will have to follow and how its work has to be articulated with others. We can see here that the integration of agents will have an impact on the subject-community-rules triad and on the subject-community-division of labour triad, as rules and division of labour will have to be explicit in order to enable their understanding by the software agents.
- Defining an agent's task should help learners to clarify their shared object. Focusing on the subject-object-community triad, we can anticipate that the expression of rules and division of labour will require clarifying the object.

The agent activity itself should encourage that changes to the rules and division of labour are explicit. Software agents do only what learners tell them to do. This has once again an influence on the subject-community-rules and subject-community-division of labour triads in that changes to rules or division of labour need to be explicited. This will lead learners to have a reflection on the organization all along the achievement of the activity.

From a general point of view, although in our case the control remains the students' responsibility, integrating software agents to the community imposes a certain level of rigidity. This rigidity is caused by the fact that software agents cannot understand implicitly a community organization as a human would: integrating software agents requires an explicit definition of the community work articulation. This could be seen as a negative point if our objective were just to facilitate the collective work. However, as our principal goal is to encourage students to work out organization, such explicitness participates positively, in fact, in the clarification of the organisation.

Task modelling and task delegation

Students define through Symba, in an integrated way, both the tasks they will achieve themselves and the ones that will be achieved by software agents. As stated previously, the way these tasks are defined must incite students to focus on organization features. For this purpose, we have modelled the task notion in a way that denotes Engeström's triangle notions.

For the tasks they achieve themselves, the students have to define the objective (in natural language), the nature (individual or collective), the subject (an individual, a sub-group, or all the group), the beginning and ending dates (rules), the tools (the general-purpose tools that will be accessible at the activity-level to achieve this task, e.g. a Chat or browser), and the resources and productions (file names). Tasks delegated to software agents are described similarly with some additional slots to be filled according to the task specificities. As an example, figure 3 presents a date search task delegated to Nicolas; the additional slots are the meeting duration, the meeting participants and the meeting enclosing dates. The system dynamically generates a natural language description of the task in order to avoid possible misunderstandings of the interface.

This approach allows students to specify the agent's behaviour through a task delegation semantic. This semantic (inspired from Engeström's notions) is conceptually simple for the students and makes them work out the competences that correspond to our pedagogical objective: defining the software agents' tasks requires having a clear idea of the software agent's role and how it will be articulated with the other agents' tasks.

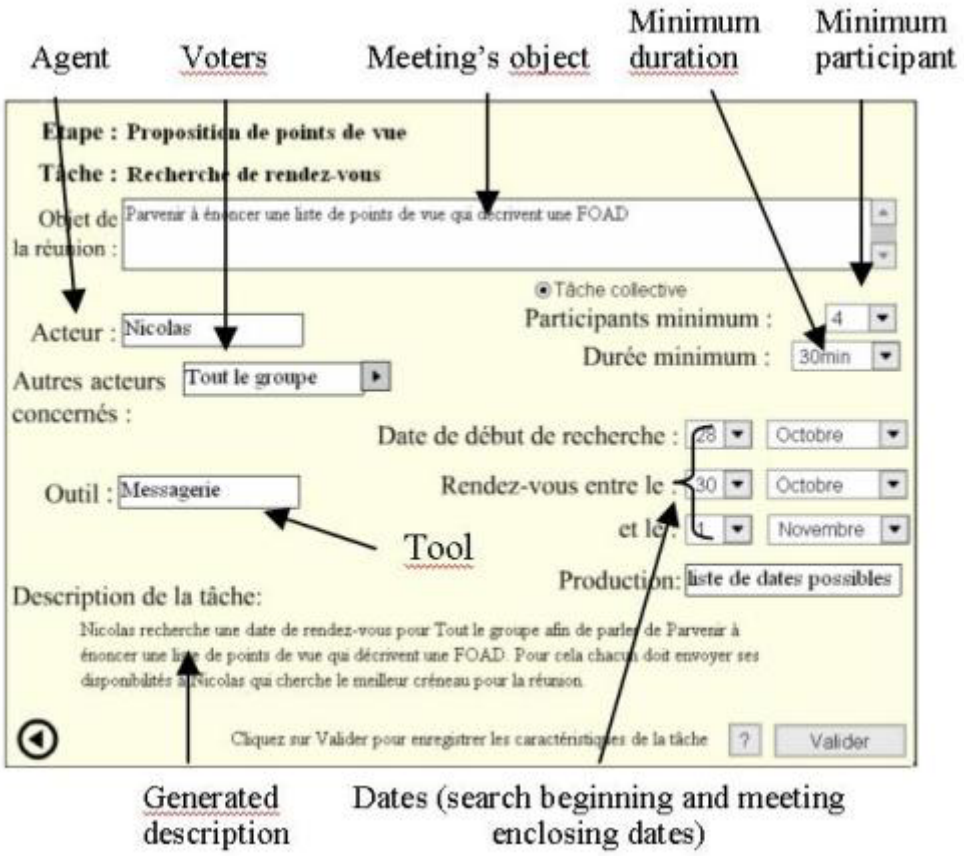

Figure 3: Nicolas's task definition

More generally, the need for precision required for agents has a border effect on the organization activity in general as it leads students to have the same precision when they come to defining their own tasks.

It can be noticed that delegating tasks to software agents is a way for human agents to tailor their environment: when defining the software agents, the students modify the functionalities of the "system", i.e., how the Symba environment can support their activity (this goes further than simply modifying some interface features such as "preferences"). This is an innovative approach to the notion of tailorable systems as defined by Morsh (Morsh, 1997), i.e., integrating a way to tailor the system as one of its functionalities. When providing users with the possibility of customizing their environment at a functional level, a classical problem is that of the programming language in which these users can express their desires. Programming requires the respect of a formal grammar, and programming language grammars requires skills that cannot be expected from students; this is one of the principal causes of un-usability of tailorable systems (Morsh, 1997). The task delegation semantic we propose is an approach that moves the problem from programming through a computer-science semantic to expressing the organisation of an activity (Betbeder, Taurisson & Tchounikine, 2003).

Putting Into Practice

Definition of the software agents' roles

The approach presented in this paper has been experimented in the context of a collective activity that takes place in the fifth university year of a computer science curriculum. The activity consists in collective and individual phases on the theme of conceptual maps, engaging students in producing individual and collective documents and exchanging them, planning synchronous meetings or voting for a decision to be taken (asynchronously and synchronously).

In order to define what software agents should be defined we have used the "role conflict" theory defined by Watt (1993), that provides a scientific basis for the empirical conclusions we have drawn from our analysis of previous experiments (Betbeder & Tchounikine, 2002). As explained before, Watt stated that, in a groupware

environment, an actor may have to handle a task that requires additional work without any (or sufficient) direct benefit: the realization of a task will benefit to the community but the member who has to handle it perceives his role as being less attractive than the others. This situation can lead to a community's breakdown as the one whose role seemed less attractive will not engage himself in the task, even though this may cause the common goal not to be achieved.

When concerning a group of humans, potential role conflicts must be avoided by getting different roles to be equivalent. However, in a hybrid groupware mixing human and software agents such as Symba, there is a benefit in that software agents handle tasks that can lead to a role conflict: this will make the use of the software agents more attractive for humans and it will reduce the risk of a community breakdown.

The software agents that have been implemented and introduced in the 2002 experiment have therefore been specified from the analysis of two previous experiments that corresponded to similar pedagogical activities (Betbeder & Tchounikine, 2002). This analysis allowed us to identify three tasks potentially causing role conflicts by requiring an unbalanced communication scheme, from which we specified and implemented three agents. Nicolas can find a date for a meeting when a group prepares a synchronous meeting. He sends a mail to each member of the community to ask them their availabilities, synthesizes the answers and proposes a date (eventually iteratively). Chick can organise a vote when the group must decide something collectively either as a top-level task (i.e., one of the tasks explicitly scheduled by the students) realized asynchronously by Mail or "live" during a synchronous meeting and achieved by Chat. Colin can handle document flow of the data produced or to be used by the group. He asks the members of the group to supply their documents and distributes them across the community (making a synthesis of individual works or getting documents to be downloaded from a Web site).

Human / software agents' communication

Human / software agents' communication must not impose features that are pedagogically counterproductive or that can discourage students from using the agents.

13:45:49 [Chick] Hello everybody! To ask me to organise a vote, send me a personal message (by clicking on my name in front of one of my message or by beginning the message with /TO Chick) saying: "organise a vote"
{Note: public message; it appears in every participant's interface}
13:46:01 [Jack]>[Chick] organise a vote
{Note: private message is. It appears in Jack's and Chick's interfaces}
13:46:05 [Chick] Jack asks me to organise a vote. We have to define the voting arrangements. I'll ask publicly some questions, Jack will answer them by sending me a personal message. Will the vote be a consensus or a referendum? (/TO Chick consensus/referendum)
13:46:32 [Jack]>[Chick] referendum
13:46:34 [Chick] The vote will be a referendum. Will the ballots be public or private? (/TO Chick public/private)
13:46:57 [Jack]>[Chick] private
13:46:59 [Chick] The ballots will be private. What motion will be voted? (/TO Chick the motion)
13:47:43 [Jack]>[Chick] We all agree on the synthesis
13:47:44 [Chick] I am going to organise a referendum on the motion: "We all agree on the synthesis". The ballots will be private. Do I start the vote? (/TO Chick yes/no)
13:48:02 [Jack]>[Chick] yes
13:48:05 [Chick]>[Jack] Jack asked me to organise a vote. It is a referendum with private ballots. The voted motion is: "We all agree on the synthesis". What is your position (if you don't answer within two minutes thirty, you'll be considered as an abstentionist)? (/TO Chick pro/con)
13:48:29 [Jack]>[Chick] pro
13:48:34 [Chick] Vote results: the motion: "We all agree on the synthesis" is adopted with 3 pros and 0 con. There were 0 abstention.
{Note: the results are sent in a public message}
(…)

Figure 4: Chick interacts in a chat (Snapshot from the 2002 experiment: the agent Chick interacts with the student Jack; the text is translated from French)

In our context (distant students supported by a Web-based environment), communication tools (i.e., chats, electronic messages and forums) are the principal vector for the perception of the community and its members. Therefore, similar to the students, the software agents have been implemented with capabilities to communicate

by mail (when asynchronous) and chat (for synchronous, cf. figure 4). For instance, Chick (the vote organiser) uses Mail when used asynchronously and Chat when used synchronously. Student / software agent and student / student communication are therefore homogeneous.

First results of the study

We have implemented the three agents described to study their integration in the community in an ecological experiment: groups of students from the fifth year of a computer science university curriculum engaged in an activity through the Symba framework.

The first results of the experiment highlight that the students are not bewildered by the introduction of these software agents and they in fact use them. A certain number of difficulties were raised from surface misunderstanding of the interface, but defining agents' tasks through the "activity triangle" grammar seems easy for the students.

However, the experiment highlighted that students require the agents to be glass boxes much more than human agents would. As an example, although Nicolas's protocol to find a date is similar to the one observed from students in the preceding experiments, it seems that students feel a loss of control after they have answered Nicolas's mail: they require an extended awareness of the software agents' behaviour (it seems natural to send a mail to another student and wait for his answer, but an agent should answer immediately or, at least, explain what he is doing).

A scalable implementation for a MAS mixing human and software agents

In most multi-agent systems (MAS) where human and software agents are interacting, software agents are lower level of granularity agents than human. The particularity of our approach is that human and software agents are at the same level of granularity: software agents handle tasks that are similar to the human's tasks and interact at the task level in a human environment with human agents as peer.

As we have modelled the community as a multi- agent system mixing human and software agents, we want to keep a structural correspondence between the model and the implementation. Therefore, we have developed a multi-agent system in which both human and software agents are represented. Learners (human agents) are represented in the MAS through "proxy agents". Proxy agents are a bridge between the outside world and the multi-agent system (see figure 5). For example, when a learner sends a message to a software agent through an email tool, the email is intercepted by the proxy agent representing this learner in the MAS. The proxy agent then sends a message to the software agent using the multi-agent framework messaging primitives. In the same way, when a software agent needs to send a message to a learner, it sends a message to the proxy agent in the MAS, and the proxy agent then transcribes this message in the "real world" messaging tool sending it to the learner. In other words, the MAS models the overall activity of the human and software agents.

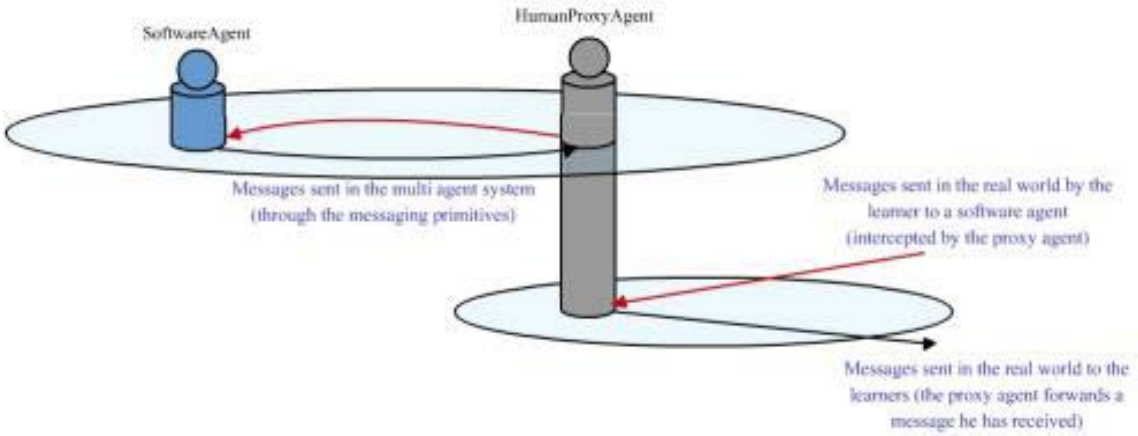

Figure 5: Proxying messages from and to the "real world"

From a multi-agent perspective, our approach is coherent with the "agentification of services" principle developed by Ferber (Aalaadin model, Ferber & Gutknecht, 1998). Within this model, the mechanisms that are

connected with the multi-agent principles such as the "distribution mechanism" (distributing a multi-agent over different computers) is modelled as a specific agent intercepting messages to forward them to another site. This approach is very powerful as the implementation features (in this case, foreign messaging) is the agent's responsibility, the system only has to make sure that an agent exists to provide the service.

We have therefore used the Madkit framework (the Multi Agent Development Kit developed by Ferber on the basis of the Aalaadin model) to implement the framework services. This "agentification of services" approach allows us to conceptualize a multi-agent system in a recursive way: an agent can be conceptualized itself as a multi-agent system of lower level agents. We also use this principle to implement general services for the agent's community such as a timer agent that can be asked to send a message at a particular time.

From an implementation point of view (cf. figure 6) an agent consists of a multi-agent system composed of: 1) a façade agent, 2) a task manager agent, 3) a rule scheduler agent and 4) a dialogue agent.

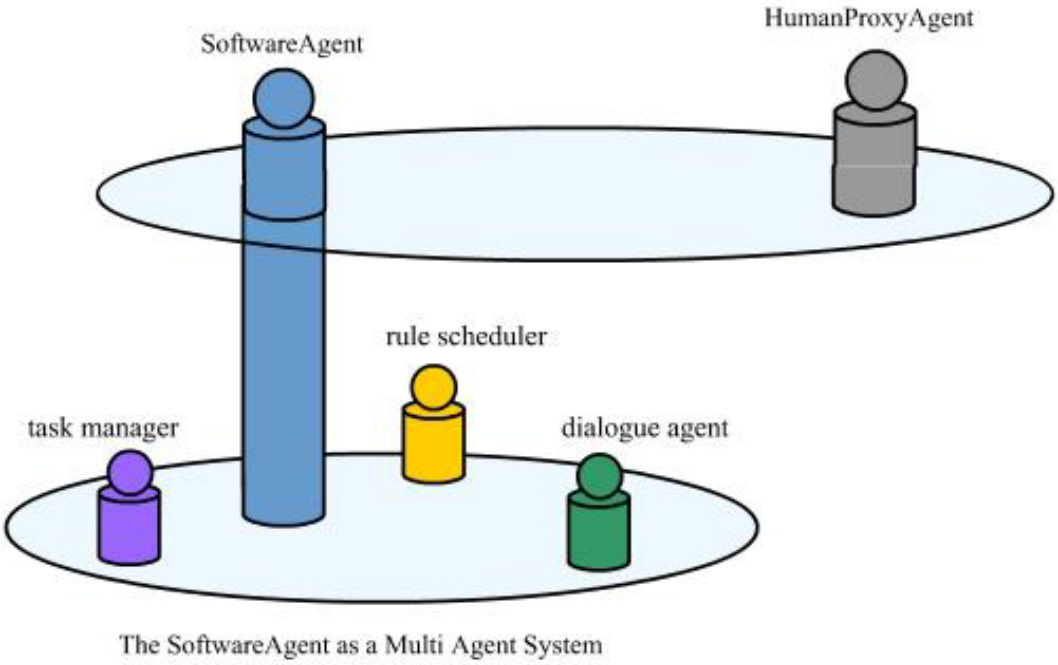

Figure 6: The software agent : a Multi-Agent System composed of lower level agents

To illustrate the powerfulness of this principle we can focus on the dialogue agent. Currently, the dialogue agent is implemented using a simple state automaton. The façade agent receives a message from some other agent, it then forwards it to the dialogue agent that processes it (using the automaton? and sends the result back to the façade agent that deals with it (sending a response to the original message sender or acting on the environment). We are currently working on replacing the actual state automaton dialogue agent by a more sophisticated dialogue agent. For the façade agent, nothing would be changed, it would still delegate the processing of the message and react in function of the processing result.

Conclusions

We have proposed an innovative approach that consists in introducing software agents within a group of students in a manner that supports students by taking in some uninteresting parts of the work in charge whilst enhancing the pedagogical objective of making students work out organisational features. We have presented how this can be put into practice through a task delegation principle, a possible approach of the modelling of these tasks inspired from the Engeström triangle and an implementation based on the "agentification of services" approach.

The first experiments demonstrate that this approach is realistic and influences the overall students' activity in a way that corresponds to our pedagogical objectives.

Different features pointed out by the experiments are under analysis, in particular enhancing the perception by the students of the software agents' activity and combining different agents' activity, e.g.: collecting documents

(Colin), placing them on a web site (Colin), organising a vote to know if these documents are ok (Chick) or re-collecting documents if not.

References

Betbeder, M.-L., Taurisson, N., & Tchounikine, P. (2003). An approach of tailorability within a collective activity support framework. In U. Hoppe, F. Verdejo, & J. Kay (Eds.), *Artificial Intelligence in Education. Shaping the Future of Learning through Intelligent Technologies* (pp 383-385), Netherlands: IOS Press.

Betbeder, M.-L., & Tchounikine, P. (2003). Symba: a tailorable framework to support collective activites in a learning context. *Paper presented at the 9th International Workshop on Groupware*, September 28 - October 2, 2003, Autrans, France.

Betbeder, M.-L., & Tchounikine, P. (2002). Une expérience d'activité collective médiatisée via le Web dans une FOAD. *Paper presented at the Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie*, November 13-15, 2002, Lyon, France.

Dillenbourg, P., Baker, M., Blaye, A. & O'Malley, C. (1996). The evolution of research on collaborative learning. In E. Spada & P. Reiman (Eds), *Learning in Human and Machines: Toward an interdisciplinary learning science* (pp 189-211), London: Pergamon.

Engeström, Y. (1987) *Learning by Expanding. An activity-theoretical approach to development research*, Helsinki: Orienta-konsultit.

Ferber, J., & Gutknecht, O. (1998). A meta-model for the analysis and design of organizations in multi-agent systems. *Paper presented at the Third International Conference on Multi-Agent Systems (ICMAS98)*, July 4-7, 1998, Paris, France.

Hollan, J., Hutchins, E., & Kirsh, D. (2000). Distributed cognition: Toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction, 7*, 174-196.

Kuutti, K. (1996). Activity Theory as a Potential Framework for Human-Computer Interaction Research. In B. A. Nardi (Ed.), *Context and Consciousness* (pp 17-44), Cambridge, MA: MIT Press.

Lewis, R. (2000). Human Activities in Learning Societies. *Paper presented at the 8th International Conference on Computer in Education / International Conference on Computer-Assisted Instruction*, November 21-24, 2000, Taipei, Taiwan.

Morsh, A. (1997). *Method and tools for tailoring of object-oriented application: an evolving artefact approach*, Unpublished dissertation, University of Oslo.

Schmidt, K., & Simone, C. (1996). Coordination Mechanisms: Toward a Conceptual Foundation of CSCW Systems design. *Computer Supported Cooperative Work: The journal of collaboration computing, 5* (2-3), 155-200.

Suchman, L. A. (1987). *Plans and situated action: the problem of human-machine interaction*, Cambridge, England: Cambridge University Press.

Watt, S. (1993). Role Conflict in Groupware. *Paper presented at the first international conference on Intelligent Cooperative Information Systems*, May 12-14, 1993, Rotterdam, Netherlands.