A MULTI-AGENT APPROACH TO CO-OPERATIVE WORK

JC. Routier, P. Mathieu
Laboratoire d'Informatique Fondamentale de Lille
CNRS upresa 8022
Cité Scientifique
59655 Villeneuve d'Ascq cedex, France
routier@lifl.fr, mathieu@lifl.fr

Abstract This paper presents a multi-agent approach for user interface design of co-operative applications. We show here that this approach provides advantages from multi-agent researches about intelligence and modeling. It allows the designer to develop more easily the user interface of his application. Two different aspects are improved: first, the management of the interactions between components of the interface and second, the interactions between the user and the application, in particular thanks to intelligent assistants. Moreover, an agent oriented analysis of the problem highlights the different *role* and *skills* and thus eases the programmer task and the evolution capacity of the application.

To illustrate our purpose, we present a co-operative work application built with our multi-agent development framework, called MAGIQUE. The main goal of this article is not really to present a groupware application but rather to show how it can be easily developed, maintained and extended thanks to MAGIQUE and its multi-agent approach.

Keywords: co-operative work, agent based design, collaboration, interaction of humans and agents, user-interface applications design

Introduction

For many years, many researchers and industrials have worked on working group applications [Ellis et al., 1991, Baecker, 1993, Baudouin-Lafon, 1998, Derycke and Hoogstoel, 1995]. It is indeed more and more necessary to provide tools that give to physically remote people the possibility to work together, to communicate, to exchange documents and to present their information to the other members of the distributed team. Now, we must remark that the few available tools offer only some of these functions. This is probably due to the fact that the existing tools are difficult to apprehend and moreover difficult to extend, mainly because of the diversity of the new idea to be added

to the software. Simultaneously these last years, works on multi-agent model [Jennings et al., 1998], [Ferber and Gutknecht, 1998] have been put into concrete form through the realization of development platforms [Gutknecht et al., 2000], [Routier et al., 2001]. These platforms offer tools specially fitted to the modular conception of multi-skilled applications as those previously mentioned. Indeed, the properties of reactivity and proactivity of agents allow a slender and evolutive development of such applications. Moreover, the interpretation in terms of roles and skills of the capabilities of agents eases the analysis work and its decomposition.

This article presents a co-operative application based on the multi-agent platform called MAGIQUE (read [Tarpin-Bernard and David, 1999] for another example of the use of multi-agent in groupware). The objective of this application is to provide to the user the possibility to exchange slide-like information, to all its collaborators. The analysis highlights two kinds of role: the teacher or *speaker* and the students or *listeners*. The difference between these roles is determined by the fact that the *speaker* holds the unique remote control. This allows him to spread slides to listeners and listeners can claim to the remote control. At a given moment there is only one speaker, but holder of this role can change during the conference in order to allow a former listener to show his slides. Some other tools, useful to documents manipulation, are provided too.

The building of this application is based on the multi-agent system (MAS) development platform called MAGIQUE. MAGIQUE is primarily a system to build MAS distributed on a heterogeneous network. MAGIQUE uses the skill notion, and in MAGIQUE, everything is a skill [Routier et al., 2001]. Then, to be (ie. play the role of) a listener is a skill, as well as the remote control. MAGIQUE offers some important features to manage and exchange these skills, then it is very easy to give the remote control from a user to another through the skill exchange. That means that the changes in the roles of the agents is not a trivial flag placed in source code but there is an effective dynamic evolution of the agents capabilities and then of the roles. Moreover, since an agent can have as many skills as it wants, it is easy to add new functions to the system to adapt it to some new required features. For example, we will present in the following an *assistant* skill to the user agents, this was not present in the first version but had been added with no impact on what was existing, only by creating the new skills and plugging them to the agents. This assistant is a proactive and reactive "intelligent" entity that helps the user in its management of the co-operative work and then allows him to better contribute to the conference. This assistant provides also a model of the other members of the conferences since it determines their expertise fields.

The main goal of this article is not really to present a groupware application but overall to show how this, thanks to multi-agent approach and the MAGIQUE

platform, can be easily designed, maintained and extend. In a first part we will present the application and its functions, and particularly the assistant. Then after a brief presentation of MAGIQUE, we will describe the principles of the design of such an application using MAGIQUE.

1. The slide-conference application

The aim of this co-operative work application is to provide a framework that give the possibility of a conference between geographically remote people.

Each of the members has a set of document resources ("*slides*") that he wants to spread to the others. He can do that if and only if he is the owner of the (unique) *remote control* which is associated to the application. This control allows him to browse and spread his resources. Moreover, it plays also the role of a "pointer" that the speaker can use to attract the other users attention on a particular point of the currently displayed document. This pointer can be seen in real time simultaneously by every users. Two *roles* can be distinguished in this application: the one of *speaker*, plaid by the control holder, and the one of *listener*, for the all other members. Like in a real conference (we mean where all the audience is gathered in the same place), the speaker can, at any time, give the control to one of the listeners and then the respective roles of the two users evolved dynamically. We have particularly bring attention to the similarity with real conferences, since this should ease the user to apprehend such an application. Figures 1 and 2 give a look on the application from the speaker and listener points of view[1]. It is possible to distinguish different tools on the two images: the view tool on the left side of the screens (you can notice that the dimensions are the same in the two views), the mouse pointer that can be viewed in the speaker window is represented by a large dot in the listener one (In the two screenshot figures pointers have been enlarged to highlight them) . Some other tools such as the remote control (which is "active" for the speaker and "passive" for the listener), the team management and messages windows, and the assistant (here it makes an announce in the speaker window and you can notice the two different kind of presentations, the top one with an animated character and the bottom one with a most recent messages list.) at the top of each screen are visible too.

1.1 Basic functions

In the prototype, resources are designed by URL. The advantage of such a format is clear. It allows to represent different kinds of documents (texts, images, sounds, video, etc.). Moreover, it allows to benefit of the browsing capabilities due to hyperlinks, as well as to dispose of dynamic document thanks to applet for example. Each user has his own resources and he can gather

Figure 1. A speaker side view...

Figure 2. ... and the corresponding listener side view.

and organize them in order to build a structured lecture. In order to afford the conference, the two essential tools are the *view tool* and the *remote control*.

The view tool. Each participant has a medium to view resources (in our case it is a URL browser). The speaker has the control over the interactions of this tool for all the audience. The views in the different tools are fully synchronized with his own and the WYSIWIS [Stefik et al., 1987] principle is applied here. Thus, the mouse pointer in his browser can be seen by all the listeners, so he can insist on some particular point of the slide (cf. Figures 1 and 2). In the same way, the speaker controls the size of and the scrolling in all the viewer tools.

The remote control. The second essential function is of course the remote control. It is active for the speaker and passive for the listeners. At any moment, a listener can ask for the control, and this is the speaker who has the choice to give it or not to him. Its first role, when it is active, is to allow to spread a document to all the audience or to automatically browse a slides sequence. In reality, the speaker side transmits to the members the URL address of the document that he proposes. The view tool of each listeners is responsible of the downloading of the applications and of its interpretation. The remote control provides also some tools that allow to gather and to organize documents.

1.2 Tools that Help Co-operation

Adding to the minimal functions presented above, the application offers some other tools that promote the co-operation and the awareness of the other listeners. Let us cite three of them: the audience "visualization", an assistant and a message system. We will present them briefly.

1.2.1 The Members Window.

The first tool goal is to highlight for the user the fact that he is not alone. The list of the members can be seen in a window where the current speaker is clearly identified. The need to concretize a "global awareness" is a well known problem in CSCW [Brinck and McDaniel, 1997]. The request for the control can be done from this tool. The different requests made by the different listeners, as well as their "oldness", can be viewed by all. The names of the requesters are highlighted and this effect vanishes slowly with time. Interest is twofold. First, the speaker can directly see from how long a request has been made, and he can, for example, consider that a request is too old and then is no more appropriate. Second, from the listener point of view, he can hesitate to make a request when he sees that a lot of requests have already been made (this is like the number of raised fingers in an assembly).

1.2.2 The Assistant.

A second tool is provided to allow to everyone to bring a better contribution to the co-operation. It is an *assistant* who,

obviously, must ease the use of the tool by the user. It is possible to cite three main roles for this assistant: highlighting of events, helping to participate to the co-operation, modeling the other members for future collaborations.

Highlighting Events The distributed nature of the application implies that it is necessary to emphasize some events that would have been more easily detected in a real conference. As an example we can cite the incoming/outcoming of members, the change of the speaker, the sent messages (cf. figure 3)... The contextualization, according to the mentioned events, of the graphical interface of the assistant helps the user by leading him to the concerned tool. Thus this function contributes to ease the use of the application.

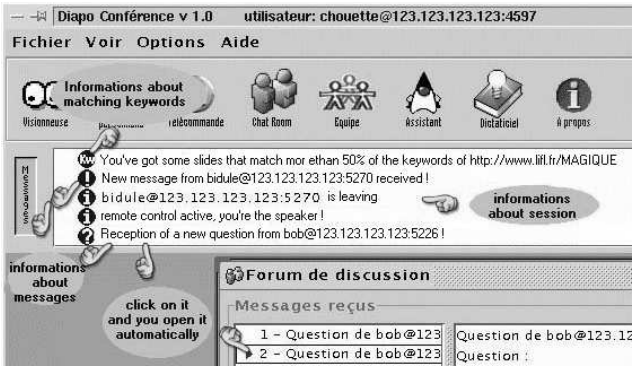

Figure 3. Events mentioned by the assistant.

Helping Co-operation "reactions", one of them provides an help to the co-operation emergence. Indeed, during a slide-conference, the assistant will help the listener user to bring its tribute to the conference. Each time a new slides is proposed by the speaker, the assistant warns the listener if it has in its resource base, a document which is correlated to the currently diffused one. Then the assistant proposes to the user to preview this (or these) selected document(s) so that he can judge by himself of the pertinence to propose it. Then the listener can ask the control to become the speaker, but it can also simply ask to his assistant to transmit this document to the current speaker assistant. Thus the speaker assistant warns its user that a listener has proposed a document and this one can chose to spread it or not. In the current version, the decision process of the assistant is based on keywords that must have been given in the resources. The assistant explores the user resource base and wakes up each time there is sufficient matching keywords. The correlation criteria can be changed by the user.

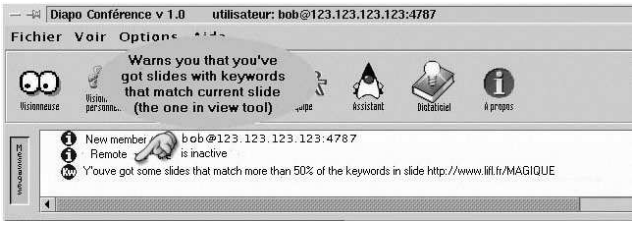

Figure 4. The assistant warns me that I have correlated resources.

Modeling other members can have another use and be exploited differently by the assistant user. It is indeed possible, for every proposed resource, to note the keywords and thus to build step by step an image of the expertise domains of the speaker, but also of the domains of interest of the listeners. Thus from conference to conference, a model of the members can be done. This knowledge can then be used to facilitate the creation of new conferences: experts can be invited to bring their contribution and interested listeners to attend it. This information could be used outside of the conference context too, when the user needs an information on a particular domain, then he can ask to its assistant if a skilled person is known.

1.2.3 The message system.

At last, a message system allows the listener to ask question to the speaker (and only to him). This last can broadcast a message (an announce or the answer to a question) to all the audience.

2. Realization

We have briefly presented in the previous section the offered functions of this slide conference application. The problem of the well fitted tools that can be used to develop such an application arises. We claim that the use of a multi-agent development framework, and as an example of the MAGIQUE platform, eases the realization of such applications. The advantages of a multi-agent development environment like MAGIQUE is that it provides all the functions for the distribution and all the mechanisms for the communications between agents. This allows the programmer to get rid of all these problems and thus to concentrate on the purely applicative problems. In this case, it consists of the delegation to the agents of the management of the different parts of the application, thus several agents can be used for one client.

We will present *quickly* the MAGIQUE multi-agent platform before to show how it can be used to develop the conference application.

2.1 MAS : Magique

MAGIQUE means "hierarchical multi-agent" ("Multi-AGent hiérarchIQUE" in French, see `http://www.lifl.fr/SMAC`, MAGIQUE topic). MAGIQUE proposes a hierarchical organization of multi-agent systems (MAS). This structure provides a mechanism to delegate the exploitation of the skills of the other agents, and thus eases the programming task. MAGIQUE has been put into concrete form through an API JAVA that allows to develop hierarchical distributed MAS. Thus MAGIQUE is a framework that frees the programmer from the distribution and agent communication problems. MAGIQUE can be used to do different kinds of multi-agent applications: auction system with buyer agents with different strategies and one seller agent, exploration of a unknown territory by several agents, multi-player games with one referee agent and several player agents, etc.

2.2 Agents, Skills and Hierarchies

In MAGIQUE, an agent is an entity gifted with some skills. These skills give to the agent the possibility to play a *role* in a multi-agent application. The skills of an agent can dynamically evolved (through exchange with other agents) while the agent lives. This implies that the roles that it can play in a MAS, can evolve too. In fact, an agent is dynamically build from an atomic "empty" agent through the learning of its skills. ¿From the programming point of view, a skill can be seen as a software component that groups a coherent set of functions. The skills can thus be developed independently from any agent. Once the skills are created, the building of a MAGIQUE agent is simply made by a "plugging" like mechanism of these skills (this the agent education phase). Thus it is easy to add new functions to a MAS application.

All the agents are grouped into a MAS which is organized according to a hierarchical structure that constitutes the default acquaintance links between the agents. Thus a hierarchical link means that there is bi-directional communication channel between the two agents. Thus when two agents from the same hierarchy communicate, the path, followed by the message they exchange, maps to the hierarchy. The organization of the agents can evolved dynamically if a re-organization of the MAS structure is required during the use. This hierarchical organization provides an easy mechanism of delegation between agents inside a given MAS. An agent has a task to achieve, this task requires the exploitation of some skills that the agent can or cannot own. In the both cases, but the "invocation" of the skills by the agent will be done in the same way in MAGIQUE. That the realization of some subtask will be delegated to some other agent will be hidden and taken into account by the hierarchy. The consequence is that, an agent that must delegate some skill realization, does not need to explicitly known the agent who will perform it, the organization has the

responsibility to find it for him. It is a main difference with the "distributed objects" approach: the receiver of a message is not necessarily named and "who does what" is not necessarily known, the MAS takes this in charge. Thus, interactions between skills is automatically performed, when one skill S_1 needs to interact with another skill S_2, whether S_2 is or not known by the agent that has "invoked" S_1 is transparently managed by MAGIQUE. Even if the set of skills known by the agent evolved, this evolution is taken into account.

One advantage of this delegation is from the programming point of view. It is not required to explicitly know the (name of the) agents that will be present in the application. The references are located at the skill level, and it suffices for the programmer to know that a skill will be present in the MAS, without necessarily knowing the skilled agent. A consequence is, that when designing a MAS, the important point is not really the agents themselves but the skills. Therefore it is more easy to reuse the code. Moreover, the MAS application is more robust since the skilled agent not being necessarily defined, it is possible that for a given skill, this agent changes with time (if an agent disappear from MAS, or becomes unavailable or overloaded for example).

2.3 The realization

To design a MAS, we need to determinate the implied agents and the skills of each, then to define the organization (or structure) of the MAS and the possible interactions.

2.3.1 Analysis. In a first step, the required roles must be clearly identified as well as the skills corresponding to these roles. The definition of a multi-agent methodology based on the skill/role/interaction principles is the purpose of one of our current project. Nevertheless, a methodology like the one presented in [Wooldridge et al., 2000] can be used.

The roles. We have already clearly identified two of these roles for our conference application: the *speaker* and the *listener* roles. However there exists a third roles that we have not yet explicitly named. Indeed, it is necessary for the conference to happen, that the members can join (and leave) it. therefore an entity is required to make the link between all the participants. We will name "*co-ordinator*" this role. But it can be seen as the *environment*, classical in MAS, since it constitutes the support for the interactions and since it gives a shape to the co-presence of the members of the conference.

The skills. For the *co-ordinator* role, the corresponding skills are rather limited. It suffices to provide to the other agents an entry point to the conference and to maintain the coherence in case of departure (particularly if the speaker is leaving the session, it is necessary to find a new speaker and to give

him the remote control). The skills are quite the same for the *speaker* and *listener* roles. They correspond to the functions described in the first part of the article. To describe them quickly: the *view skill* is used to download, to interpret and to display the resources, to manage the pointer and the mouse event on the windows; the *members window* allows to manage the control request as well as its exchange; the *message system* contributes to allow the exchange between the members with the respect of the status of *speaker* and *listener*, thus this skill must be sensible to the role context; the*assistant* provides the functions to help the co-operation and to model the other members as mentioned above.

Last, for the two roles, we still have to treat the case of the skill that manages the *remote control*. This skill must provide the possibility: for the *speaker*: to lead the conference (document presentation, pointer management and the visualization area); and for both roles: to manage the resources and to organize document sequences. The case of this skill is particularly interesting, since it creates the difference between the two main roles. The dynamic evolution of roles during a session (from *speaker* to *listener* and conversely) implies a dynamic evolution of the skill known by agents. It is indeed necessary to give to a new speaker the functions to lead a conference, and to take them out from the former speaker.

2.3.2 The choice of the MAS organization. As it has been mentioned, the MAGIQUE multi-agent framework has been used to develop the prototype. Each member of the conference is defined by an agent that represents him, one of this agent will have the *speaker* role and the others will be the *listeners* (see figure 5). We have decided to create an additional particular agent to play the *co-ordinate* role (even if this role could have plaid by some other agent, the initiator of the conference for example, or could even dynamically change).

We have explained that MAS in MAGIQUE are hierarchically organized. In this case, we have chosen to locate the *co-ordinate* agent at the root of the hierarchy and all the other agents at the lower level. Let us note that it would have been possible to locate the *speaker* agent at the root, this would have implied a reorganization of the hierarchy when roles change. This can be done with MAGIQUE. The dynamical evolution of the roles *speaker*$\longleftrightarrow$*listener* is easily managed with MAGIQUE. Indeed, this platform offers the advantage to allow the dynamic evolution of the skills of the agents, through exchange. In MAGIQUE, the agents learn and forget effectively their skills. The exchanges of skills are effective, independently of any assumption on the code of the skills and of the distribution of the agents over the network. The *speaker* agent will then be able, as the microphone is physically given during a real conference, to give the remote control skill to the new *speaker* and therefore to lose it (see

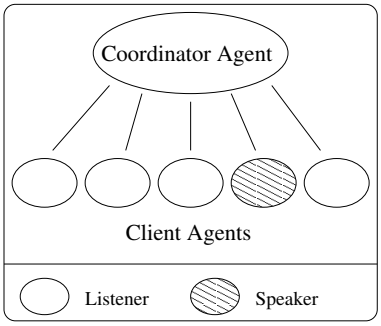

Figure 5. Hierarchical organization of the MAS

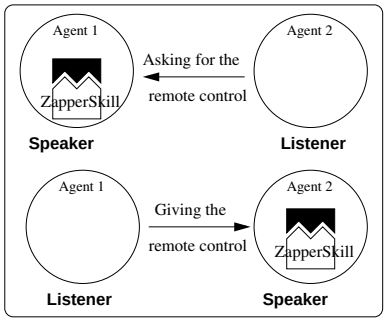

Figure 6. Dynamical evolution of roles through skill exchange

figure 6). And it is not here a simple modification in the value of an attribute but of a real concrete alteration of the agent and of the role it plays, and therefore in a way of its right over the application.

2.3.3 The programming. To realize this application with MAGIQUE, it "suffices" to develop the different skills mentioned previously. Each represents a component. Once the interface of the skill has been defined , it must be implemented by JAVA objects, the sole difference that must taken into account is the need to do interactions with the agents and the different other skills. We have already seen previously that this is easily done thanks to the delegation mechanism and primitive for delegation are provided.

The independence between skills provides independence between the different components. The possibility with MAGIQUE to make unnamed requests allows to have a more functional view of the different aspects of the applications. The roles/skills interact without needing to have an explicit knowledge of each other. This eases development and possibility of evolutions. Once you have the skills, the agent building is very easy. You just have to "teach" to the atomic agent the required skills one by one. Then the code is something like this:

Then the MAGIQUE layer manages the connection of agents and the communications due to interactions between agents. These communications do not explicitly appear in the code since they are implied by the skill invocations and interactions and thus taken in charge by MAGIQUE.

2.4 Evolution et adaptivity

The skill notion eases the evolution of the application and its adaptation to different contexts. Thus it is possible to easily extend the offered functions

through the creation of new skills that you just have to "teach" to the agents. Thus it is possible to consider an incremental building of the applications. For example, a skill allowing to transport the voice could thus have been added to ease the information exchange, and this would have no consequence over the rest of the application. The adaptation to different supports or different kind of resources can also be easily obtained. For example, if one of the listeners want to use the application on a personal assistant, he can adapt the visualization skill by "teaching" to his agent the correct view tool skill instead of the default one.

To give one last example to illustrate that such an application developed with MAGIQUE is open, it is easy to consider to adapt the assistant for each of the users. For example, different strategies to determinate the correlation of the resources can be considered, or even different criteria to model the other members of the conference.

3. Conclusion

Through the presentation of an application for distributed conference, we have shown how the use of a multi-agent framework like MAGIQUE eases the development of a co-operative applications and its user interface. The use of such a platform allows to get rid of the problems due to the distribution and to the communications, thanks to the "normal" interactions of agents. Moreover, the incremental building of the agents in MAGIQUE, through dynamical skill learning, eases the programming and the evolutivity of the application.

The prototype application provides the support for co-operation and tools to help it. These are tools essential like the view tool and the remote control. This last is the central tool since its possession defines the role of each member and its exchange implies the dynamic evolution of the roles. Some additional tools are provided also. They are essential too: the team management, the capability to organize resources, or the message system. Last, the assistant allows a better exploitation of the resources when they are correlated to the current one. This assistant allows to model the co-members of the conferences and thus eases future co-operation.

This prototype, and a tutorial, can be downloaded at:

`http://www.lifl.fr/SMAC` topic MAGIQUE

Acknowledgments

We would like to thank to the anonymous referees for their helpful comments. Mostly of them have asked for more details on such-and-such point, and we are sorry if we do not have fulfilled all their requirements, but this was impossible in the limited given number of pages.

References

[Baecker, 1993] Baecker, R., editor (1993). *Readings in groupware and computer supported cooperative work*. Morgan Kaufman.

[Baudouin-Lafon, 1998] Baudouin-Lafon, M., editor (1998). *Computer-Supported Cooperative Work*. John Wiley Sons Ltd.

[Brinck and McDaniel, 1997] Brinck, T. and McDaniel, S. (1997). Chi 97 workshop on awareness in collaborative systems. In *Proc. of CHI'97: Human Factors in Computing Systems*, (position papers available via `http://www.usabilityfirst.com/groupware/awareness/`).

[Derycke and Hoogstoel, 1995] Derycke, A. and Hoogstoel, F. (1995). Le travail coopératif assisté par ordinateur : quels enjeux pour les concepteurs. In *Actes du XIIIe Congrès INFORSID (INFormatique des ORganisations et Systèmes d'Information et de Décision)*, Grenoble.

[Ellis et al., 1991] Ellis, C., Gibbs, S., and Rein, G. (1991). Groupware: some issues and experiences. *Communications of the ACM*, 34(1):38–58.

[Ferber and Gutknecht, 1998] Ferber, J. and Gutknecht, O. (1998). A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings of ICMAS'98*.

[Gutknecht et al., 2000] Gutknecht, O., J.Ferber, and Michel, F. (2000). Madkit: une plateforme multi-agent générique. Technical report, FIPA.

[Jennings et al., 1998] Jennings, N., Sycara, K., and Woolridge, M. (1998). A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems.*, 1:275–306.

[Routier et al., 2001] Routier, J., Mathieu, P., and Secq, Y. (2001). Dynamic skill learning: A support to agent evolution. In *Proceedings of the AISB'01 Symposium*, pages 25–32.

[Stefik et al., 1987] Stefik, M., Bobrow, D., Foster, G., Lanning, S., and Tatar, D. (1987). Wysiwis revised: Early experiences with multi-user interfaces. *ACM Transactions on Office Information Systems*, 5(2):147–167.

[Tarpin-Bernard and David, 1999] Tarpin-Bernard, F. and David, B. (1999). Amf : un modèle d'architecture multi-agents multi-facettes. *Techniques et Sciences Informatiques, Hermès*, 18(5):555–586.

[Wooldridge et al., 2000] Wooldridge, M., Jennings, N., and Kinny, D. (2000). The gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*.