



Multi-Agent Systems: a Paradigm to Design Ambient Intelligent Applications*

Amal El Fallah Seghrouchni, Adina Magda Florea and Andrei Olaru

Abstract In this paper we present a Multi-Agent System (MAS) paradigm and discuss how it can be used to design intelligent and distributed systems. The main features of this MAS, such as natural distribution of the system, inherent intelligence of its agents, and their mobility help address a large scope of distributed applications including the domain of ambient intelligence. Other features of the MAS, like multi-agent planning, context-awareness and adaptation are also very useful since they bring added value, by allowing to implement intelligent and collective behavior. The paper also presents a scenario of ambient intelligence and shows how it could be designed using the MAS paradigm.

1 Introduction to Multi-Agent Systems

A Multi-Agent System (MAS) [7] is an organization of a set of autonomous and potentially heterogeneous agents operating in a shared and dynamic environment. MAS represent (e.g. manage, model and / or simulate) physical systems (as is the case in the field of robotics) or (more often) software. The MAS keystone is the double inference mechanism that is used by the agents. Agents, unlike other design paradigms as objects or components, distinguish the level of task completion

* Original publication at <http://www.springerlink.com/content/u0372853523ux693/abstract/>

Amal El Fallah Seghrouchni
LIP6, University Pierre et Marie Curie, 4 Place Jussieu, 75005 Paris, France

Adina Magda Florea
University Politehnica of Bucharest, Splaiul Independentei 313, 060042 Bucharest, Romania

Andrei Olaru
University Politehnica of Bucharest, Splaiul Independentei 313, 060042 Bucharest, Romania (in cotutelle with LIP6, University Pierre et Marie Curie)

e-mail: amal.elfallah@lip6.fr · adina@cs.pub.ro · cs@andreiolaru.ro

or problem solving from the level of control of the problem solving. Thus, they may act, observe their actions and change their own course of action. Agents have specific properties such as autonomy (an agent controls his condition and his actions regardless of any outside intervention); reactivity (an agent senses its environment and reacts to its changes); pro-activity (an agent tends to generate and achieve goals all by itself); and sociability (an agent interacts with other agents in the system).

Within a MAS, agents interact to achieve cooperative (e.g. distributed problem solving) or competitive (e.g. coalition formation, auction) group behavior. Finally, a MAS is deployed in a environment that impacts its dynamic behavior.

The dynamism of MAS covers several aspects:

- The dynamic structure of the MAS (the organization of agents) may change over time due to openness (arrival and departure of agents) and to the evolution of functional requirements (creation / removal of agents).
- The dynamics of acquaintances between agents: links can appear (arrival or creation of agents), others may disappear (departure or removal of agents) and / or change (e.g. for mobile agents).
- The dynamic environment of the MAS: changes in the environment are perceived by agents and taken into account incrementally.

At the agent level, it is the very structure of the agent that may change over time. Beyond changes in the cognitive elements of the agent (e.g. knowledge, goals, preferences, plans, etc.), we have proposed a model of agents whom are able to absorb other agents (their sub-agents) or conversely, to dissolve into their parents.

To capture these dynamics and concepts, we proposed an agent-oriented programming language that incorporates the cognitive foundations of MAS and extends them by means of local and distant mobility. Thus, a MAS is represented by a set of hierarchies distributed on multiple networked machines, each agent being a node in the hierarchy. It consists of cognitive elements (such as goals, beliefs), of processes, and of sub-agents.

An agent can dynamically acquire the knowledge, the capabilities and the sub-agents of sub-agents that it absorbs. The migration of agents, enriched with mechanisms of absorption and dissolution, allow the dynamic reconfiguration of MAS.

2 Context-Awareness

One of the central features that makes distributed systems "intelligent" is *context awareness*. One of the definitions of context is that it is *the set of environmental states and settings that either determines an application's behaviour or in which an application event occurs and is interesting to the user* [4]. One important point in the definition above is the relevance to the user. Either an event must be relevant to the user, or the application's behaviour must change so that it becomes relevant to the user. Context awareness is that characteristic of an application that makes it change its behaviour in function of, and according to, context.

Research in the domain of context awareness has shown that there are many aspects of context. One classification of context [4] divides context into computational context – available computing and networking resources, including the cost for using them; user context – user’s profile, location, people and objects nearby, social situation; physical context – light and noise levels, temperature, traffic conditions, etc; and time context – the current time coordinate of the user and related information (like the season, for instance). Context can be further classified [5] as primary – sensed directly by sensors and specialized devices – and secondary – which is inferred from the primary context.

If many authors consider context as merely a set of sensed values [1, 6], a particularly interesting approach to context-awareness is taken by Henricksen et al [8, 9], that model context as associations between entities or between entities and attributes, where an entity can be a person, a place, a communication device, etc. These associations can be of different types: static – associations that remain fixed for the lifetime of the entity; dynamic and sensed – obtained from sensors, usually transformed afterwards, changing frequently and subject to sensing errors; dynamic and derived – information that is inferred, usually from sensed or static associations; dynamic and profiled – introduced explicitly by the user, leading to greater reliability, but also subject to staleness.

In a context-aware system, there are several layers that deal with context information. One possible organization [12] uses three layers: data acquisition, data interpretation and data utilization. However, considering that much context information is volatile (e.g. user’s location and time), a context-aware system must also feature components for the degradation of context information.

3 Mobile MAS Meets Ambient Intelligence

The agent-based paradigm is one of the paradigms that can be used for the implementation of distributed systems [7]. In the case of Ambient Intelligence, agents are particularly appropriate, because they offer features that originate from the field of Artificial Intelligence and that are vital to the needs of Ambient Intelligence [10]: proactive and reactive reasoning, autonomy, social abilities and learning. Autonomy is useful because individual devices in an Ambient Intelligence environment must be able to act on their own, without the need for user intervention or control from centralized components. Learning can serve to adapt to the user’s habits. And reasoning – as well as the capability to make plans – is what makes a system appear intelligent to the user.

The agent-oriented paradigm is also useful in modeling real-world and social systems, where optimal solutions are not needed and problems are solved by cooperation and communication, in a fully distributed fashion [10]. Currently several agent-oriented programming languages exist [2], that allow the programmer to describe an application only by specifying the behaviour of individual agents.

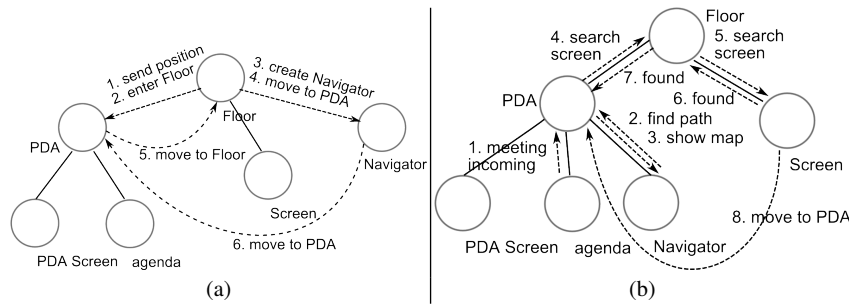


Fig. 1 Sequences of messages exchanged between agents: (a) *Floor* announces *PDA* of its new position, and instructs it to move as its child, then creates a *Navigator* that will offer services to *PDA*; (b) *Agenda* announces a new meeting, *PDA* asks a path from *Navigator*, which in turn asks for a larger screen – which is searched on the floor, and found, then *Screen* will move as a child of *PDA*.

Such an agent-oriented programming language is CLAIM, that also features a deployment platform for agents, called Sympa [11]. In CLAIM, each agent has a knowledge base, offers a to the exterior a certain number of capabilities and is capable both of reactive (by means of rules) and proactive behaviour. More importantly, the multi-agent system has a structure that is inspired from ambient calculus [3]: agents are placed in a hierarchical structure and an agent can have another agent as parent, as well as several other agents as children. Agents in CLAIM are mobile – they are able to change the host on which they are executing, and they are also able to change their place in the hierarchical structure. Moreover, when an agent moves, its children move with it automatically.

It is the hierarchical structure of the CLAIM multi-agent system that makes it especially appropriate for the implementation of an Ambient Intelligence system. That is because CLAIM makes it easier to implement context-awareness. An agent's ambient – formed by itself and all if its children can also represent a context. Agents can represent smart places, can manage smart devices, or can offer services.

4 A Case Study

Take for example the following scenario (also refer to Figure 1): a user has a meeting in a building that he / she does not previously know. When arriving at the right floor, the user's PDA automatically connects to a local wireless access point. A CLAIM agent executes on the user's PDA – we will call this agent *PDA*. Another agent executes on a local machine and manages the context of the building's floor – call it *Floor*. *Floor* detects the presence of the user's PDA, and instructs the *PDA* agent to move in the agent structure and become a child of *Floor*. The movement is only logical: the agents keep executing on the same machines as before.

When *PDA* enters the floor, *Floor* also spawns a new agent – called *Navigator* – and instructs it to move as a child of *PDA*. This time, the movement is not only logical: *Navigator* is a mobile agent that actually arrives on the user’s PDA and will execute there for all the time during which the user is on the floor. The *Navigator* can provide *PDA* (and inherently the user) with a map of the floor, can translate indications of the floor’s sensors (sent to *Navigator* by *Floor*, and through *PDA*) into positions on the graphical map, and can calculate paths between the offices on the floor. *Navigator* is an agent that offers to the user services that are available and only makes sense in the context of the floor.

For displaying the map, *PDA* may detect that its screen is too small too appropriately display them map, so *PDA* will proactively initiate the search for a larger screen in the nearby area. The search can have several criteria: the space in which the search will take place (the current office, a nearby office, the whole floor), the range in which to search, and the minimal size of the searched screen. Devices are searched by the capabilities they offer – in this case the *display* capability is needed. *PDA* sends the query to its parent – *Floor* – which in turn locates among its children an agent *Screen*, that manages a physical screen that fits the requirements, is located near to the user and is available. *Screen* answers the query and *PDA* asks it to move to become its child. Being a child of *PDA* also marks the fact that *Screen* is in use by the user, and *PDA* gains control over the displayed information. Agent *Screen* may either run on the actual intelligent screen, or may only manage the screen while being executed on a server. When the user moves farther from the screen, the PDA will detect that the context is no longer compatible and will free *Screen*, which will return to be a child of *Floor*.

5 The Ao Dai Project

In the Ao Dai project, we have implemented, using CLAIM, a prototype of multi-agent system that handles several aspects of context-awareness, like user’s location, available resources and user preferences. We have also implemented a scenario which is an extension of the one above. The project has been implemented by Thi Thuy Nga Nguyen, Diego Salomone Bruno and Andrei Olaru, under the supervision of Prof. Amal El Fallah Seghrouchni.

The prototype is implemented in CLAIM and executes on the Sympa platform. It features several types of agents: *Site*, which is used for ”smart” places like *Floor* and *Office*; *PDA*, which directly assists the user from his personal device; *Navigator* and *Agenda*, which offer services to the user; and *Screen*, which represents a ”smart” device with the capability of displaying information.

The prototype has been demonstrated during the 5th NII-LIP6 Workshop held on June 21-22 in Paris, France. The prototype was run on 2 machines. The *Floor* agent (of type *Site*) ran on one machine, and two *Office* agents (also of type *Site*) ran on the other machine. The floor and the two offices all featured screens of different sizes, managed by *Screen* agents (see Figure 2). During the demonstration, a *PDA* agent

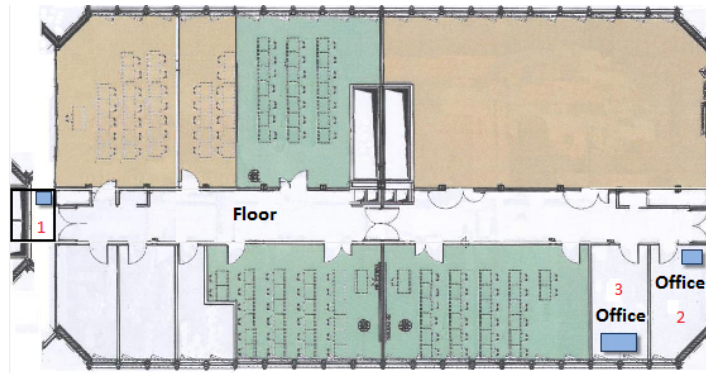


Fig. 2 The map shown by different screens in Ao Dai. There are three *Site* agents: *Floor* and two *Office* agents. Each one has a child of type *Screen*, representing the screens in the different places. The user starts on the floor (1) then moves to one office (2) and then to the other (3).

entered the floor, becoming a child of the *Floor* agent. A *Navigator* was created and sent to *PDA*. When the time of the meeting approached, *Agenda* announced *PDA*, which asked *Navigator* to find the path to the right office. *PDA* also searched for a larger screen, and found one near to the user, and automatically used it to display the map and the path. When the user – together with the *PDA* – moved to an office, the screen was freed and *PDA* with all children (*Agenda* and *Navigator*) moved to the other machine. There, the user explicitly requires a large screen, and *PDA* finds an appropriate one in the next room, and announces the user. The user then moves to the other office and *PDA* and its children all move to become children of the agent managing that office.

6 Conclusion

In this paper we showed how the MAS paradigm can be used to design intelligent and distributed systems. Hence, we have discussed some features of MAS such as natural distribution of MAS, inherent intelligence of the agents, and how mobile agents can help to address a large scope of applications in the domain of pervasive computing.

Other features of MAS like multi-agent planning, collective learning and adaptation deserve to be mentioned here. These additional features are very important since they bring added value by allowing intelligent collective behavior (the shift from single agent to multi-agent approach).

From our research perspective, the new challenges for MAS are related to their scalability and the balance to be found between the real time reaction of agents and the intelligent processing of the information gathered from the environment in this kind of applications. Another issue which is central, in particular in the context of

pervasive computing, is the relation to users, which become a part of the system. Our future work at a short term is to provide a computational model to predict the users' intentions.

7 Acknowledgements

We would like to thank Thi Thuy Nga Nguyen and Diego Salomone Bruno for their work within Ao Dai project as Master degree training.

References

1. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing* **2**(4), 263–277 (2007)
2. Bordini, R.H., Braubach, L., Dastani, M., Fallah-Seghrouchni, A.E., Gómez-Sanz, J.J., Leite, J., O'Hare, G.M.P., Pokahr, A., Ricci, A.: A survey of programming languages and platforms for multi-agent systems. *Informatica (Slovenia)* **30**(1), 33–44 (2006)
3. Cardelli, L., Gordon, A.D.: Mobile ambients. *Theor. Comput. Sci.* **240**(1), 177–213 (2000)
4. Chen, G., Kotz, D.: A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College (2000)
5. Dey, A., Abowd, G.: Towards a better understanding of context and context-awareness. CHI 2000 workshop on the what, who, where, when, and how of context-awareness pp. 304–307 (2000)
6. Feng, L., Apers, P.M.G., Jonker, W.: Towards context-aware data management for ambient intelligence. In: F. Galindo, M. Takizawa, R. Traummüller (eds.) *Proceedings of DEXA 2004, 15th International Conference on Database and Expert Systems Applications, Zaragoza, Spain, August 30 - September 3, Lecture Notes in Computer Science*, vol. 3180, pp. 422–431. Springer (2004)
7. Ferber, J.: *Multi-agent systems: an introduction to distributed artificial intelligence*. Addison-Wesley (1999)
8. Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing* **2**(1), 37–64 (2006)
9. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. *Lecture notes in computer science* pp. 167–180 (2002). URL <http://www.springerlink.com/content/jbxd2fd5ga045p8w/>
10. Ramos, C., Augusto, J., Shapiro, D.: Ambient intelligence - the next step for artificial intelligence. *IEEE Intelligent Systems* **23**(2), 15–18 (2008)
11. Suna, A., El Fallah Seghrouchni, A.: Programming mobile intelligent agents: An operational semantics. *Web Intelligence and Agent Systems* **5**(1), 47–67 (2004)
12. Viterbo, J., Mazuel, L., Charif, Y., Endler, M., Sabouret, N., Breitman, K., El Fallah Seghrouchni, A., Briot, J.: *Ambient intelligence: Management of distributed and heterogeneous context knowledge*. CRC Studies in Informatics Series. Chapman & Hall pp. 1–44 (2008)