

# Context-aware agents for developing AmI applications<sup>\*</sup>

Andrei Olaru<sup>\*</sup> Adina Magda Florea<sup>\*\*</sup>

<sup>\*</sup> *Computer Science Department, University Politehnica of Bucharest, 313 Splaiul Independentei, 060042 Bucharest, Romania (e-mail: cs@andreiolaru.ro).*

<sup>\*\*</sup> *Computer Science Department, University Politehnica of Bucharest, 313 Splaiul Independentei, 060042 Bucharest, Romania (e-mail: adina@cs.pub.ro)*

---

**Abstract:** In a layered vision of Ambient Intelligence, the intelligent services layer presents to researchers a great number of challenges, and it is especially interesting from the perspective of Artificial Intelligence and Multi-Agent Systems. While software agents have been used before for building AmI middleware, this paper proposes an agent-based architecture that is both scalability-oriented and context-aware. This research is based on two previous approaches by the authors: one dealing with context-aware information sharing in a system using self-organization mechanisms, and the other defining an implicit representation of context by means of agent hierarchies. In this paper, elements of both approaches are used to build an architecture in which agents are structured in a topology that relates to several types of context, and that can be used for context-aware information sharing and searching.

*Keywords:* Multi-agent system, ambient intelligence, context-awareness.

---

## 1. INTRODUCTION

Ambient Intelligence is one of the prominent future applications in the domain of IT (Ducatel et al. (2001)). It integrates elements from many domains: Artificial Intelligence, distributed and mobile computing, human-computer interfaces, sensor networks, and many others (Satyanarayanan (2001)). Derived from the concept of Ubiquitous Computing introduced by Weiser (1995), the term of Ambient Intelligence was coined at the beginning of the 21st century to describe *a ubiquitous electronic environment that would pro-actively, but sensibly and non-intrusively support people in their daily lives* (Ramos et al. (2008)).

It is difficult to talk about Ambient Intelligence without mentioning context-awareness. Many systems with applications in Ambient Intelligence implement context-awareness as one of their core features. One of the definitions of context is that it is *the set of environmental states and settings that either determines an application's behavior or in which an application event occurs and is interesting to the user* Chen and Kotz (2000). In previous work in the field of context-awareness there are usually two points of focus: one is the architecture for capturing context information; the other is the modeling of context information and how to reason about it. However, there is little mention of a generic way to use context information,

when context information does not only refer to physical conditions.

One other essential issue in AmI environments is scale. Many implementations proposed in the literature make considerable use of centralized components, and many applications are only demonstrated using a small number of devices. It is important to observe that in a real-scale implementation of AmI, the number of devices and users will be outstanding, much larger than the number of users and devices (including mobile ones) in the Internet today.

This work is based on two previous approaches by the authors, towards dealing with context-awareness in a manner that scales. A central element of these approaches is the use of software agents as building blocks for an AmI environment (Ramos et al. (2008); El Fallah Seghrouchni et al. (2010a)). The first approach (Olaru et al. (2010)) studied the control of the distribution of information in a large multi-agent system, by using simple and generic context measures to direct the spreading of information in terms of speed, direction and extent. The second approach (El Fallah Seghrouchni et al. (2010b)) proposed an implicit representation for context, by means of agent hierarchies, using the CLAIM agent-oriented programming language (Suna and El Fallah Seghrouchni (2004)) and inspired from the ambient calculus of Cardelli and Gordon (2000).

This paper shows how the elements in the authors' previous approaches can be used to build a new architecture: using the agent behavior developed in the AmIciTy project (Olaru and Gratie (2010)), that features scaling, local context-aware behavior and scalability, and the context-related agent hierarchies in the Ao Dai project (El Fal-

---

<sup>\*</sup> This work has been supported by CNCIS-UEFISCSU, project number PNII-IDEI 1315/2008 and by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/6/1.5/S/16.

Table 1. Features of the systems described in Section 2: the manner of representing knowledge; the use of ontologies; implementation of context-awareness; learning capabilities; consideration of security and privacy-awareness; use of mobile agents; support for scalability; flexibility of the architecture; centralized vs decentralized system.

Project Name	knowledge representation	ontologies	context-awareness	learning	security - privacy	mobile agents	scalability	flexibility	centralized
iDorm Hagras et al. (2004)	-	-	-	Yes	-	-	?	-	Yes
Spatial Agents Satoh (2004)	-	-	-	Yes	Yes	Yes	?	-	Yes
EasyMeeting Chen et al. (2004)	Ont.	SOUPA	Yes	-	Yes	-	?	Yes	Yes
SodaPop Hellenschmidt (2005)	-	-	-	-	-	-	Yes	Yes	No
LAICA Cabri et al. (2005)	-	-	-	-	-	-	Yes	Yes	partial
MyCampus Sadeh et al. (2005)	CBR	Yes	Yes	Yes	Yes	-	Yes	Yes	Yes
AmbieAgents Lech and Wienhofen (2005)	CBR	Yes	Yes	-	Yes	-	Yes	-	partial
ASK-IT Spanoudakis and Moraitis (2006)	-	some	Yes	some	-	-	-	No	Yes
CAMPUS El Fallah Seghrouchni et al. (2008)	Ont.	Yes	Yes	some	-	-	Yes	Yes	No
Dalica Costantini et al. (2008)	tuples	Yes	-	-	-	-	-	-	partial

lah Seghrouchni et al. (2010b)), the result may be a multi-agent system that spreads information across context hierarchies, by only using local information and actions, thus remaining context-aware and scalable.

The next section deals with related work in the field of Ambient Intelligence, particularly with respect to context-awareness and to the use of multi-agent systems. Sections 3 and 4 present the essential features of the AmIciTy and Ao Dai projects, respectively. Section 5 is the main contribution of the paper, presenting an architecture based on two previous approaches. An example scenario is presented in Section 6. The last section draws the conclusions.

## 2. RELATED WORK

In the field of agent-based Ambient Intelligence platforms there are two main directions of development: one concerning agents oriented towards assisting the user, based on centralized repositories of knowledge (ontologies), and one concerning the coordination of agents associated to devices, and potentially their mobility, in order to resolve complex tasks that no agent can do by itself, also considering distributed control and fault tolerance.

The first approach is closer to Intelligent User Interfaces and local anticipation of user intentions, coming from the field of intelligent personal assistants. For instance, embedded agents form an AmI environment in the iDorm implementation, by Hagras et al. (2004). Agents are used here to manage the diverse equipment in a dormitory, resulting in the control of light, temperature, etc. They learn the habits of the user and rules by which to manage those parameters

EasyMeeting, by Chen et al. (2004), is an agent-based system for the management of a "smart" meeting room. It is centralized, and it manages all devices in the room by means of reasoning on appropriate action. It is based

on the Context Broker Architecture (CoBrA) and uses the SOUPA ontology.

MyCampus (Sadeh et al. (2005)) is a much more complex system, in which agents retain bases of various knowledge about their users, in what the authors call an e-Wallet. There are also agents associated to public or semi-public services (e.g. printers). The e-Wallet manages issues related to security and privacy. It represents knowledge using OWL and accesses resources as Web Services. The e-Wallet provides context-aware services to the user and learns the user's preferences. Other components of the system are the Platform Manager and the User Interaction Manager, that offer directory and authentication services in a semi-centralized way.

Other projects that use similar approaches are ASK-IT and DALICA, presented by Spanoudakis and Moraitis (2006) and Costantini et al. (2008), respectively.

The second approach to agent-based AmI platforms concerns solving different issues like user mobility, distributed control, self-organization and fault tolerance, having a more global perspective on how an AmI platform should function.

The SpacialAgents platform (Satoh (2004)) employs mobile agents to offer functionality on the user's devices. Whenever a device (used by a user), which is also an agent host, enters a place that offers certain capabilities, a Location Information Server (LIS) sends a mobile agent to execute on the device and offer the respective services. When the agent host moves away, the agent returns to the server. The architecture is scalable, but there is no orientation towards more advanced knowledge representation or context-awareness, however it remains very interesting from the point of view of mobile agents that offer capabilities to the user.

The LAICA project (Cabri et al. (2005)) brings good arguments for relying on agents in the implementation

of AmI. It considers various types of agents, some that may be very simple, but still act in an agent-like fashion. The authors, also having experience in the field of self-organization, state a very important idea: there is no need for the individual components to be "intelligent", but it is the whole environment that, by means of coordination, collaboration and organization, must be perceived by the user as intelligent. However, here the middleware itself is not agent-oriented and is not distributed.

The AmbieAgents infrastructure (Lech and Wienhofen (2005)) is proposed as a scalable solution for mobile, context-aware information services. Context Agents manage context information, considering privacy issues; Content Agents receive anonymized context information and execute queries in order to receive information that is relevant in the given context; Recommender Agents use more advanced reasoning and ontologies in order to perform more specific queries. The structure of the agents is fixed and their roles are set.

The CAMPUS framework (El Fallah Seghrouchni et al. (2008)) considers issues like different types of contexts and decentralized control. It uses separate layers for different parts of an AmI system: *context provisioning* is close to the hardware, providing information on device resources and location, as well as handling service discovery for services available at the current location; *communication and coordination* manages loading and unloading agents, directory services, ACL messaging and semantic mediation, by using the Campus ontology; *ambient services* form the upper layer, that agents can use in order to offer other services in turn. The architecture is distributed, having only few centralized components, like the directory service and the ontology.

We have summarized some features that are relevant to our work, as they are manifested by the systems that we have reviewed above, in Table 1. It is easy to observe that different agent systems consider different aspects of Ambient Intelligence and adopt different approaches to their implementation – for instance regarding centralization of the system. It is also worth noting that few of the systems address only the problem of the middleware, and many of them are trying to propose a complete architecture, from the sensing level to the user interface.

In our work, we are trying to focus only on one layer of an Ambient Intelligence environment, and use agents for what they are good at: reasoning, autonomy, proactivity. We assume that the information can be provided by the layers below, and that interfacing with the user can be done in the layer above – we believe that applying a layered structure is a better way to deal with the design of such a complex system as a flexible, generic Ambient Intelligence environment.

Ever since the first works on context-awareness for pervasive computing Dey et al. (1999), certain infrastructures for the processing of context information have been proposed (Hong and Landay (2001); Harter et al. (2002); Lech and Wienhofen (2005); Henriksen and Indulska (2006); Baldauf et al. (2007); Feng et al. (2004)), containing several layers: sensors, processing, storage and management, and application. This type of infrastructures are useful when the context information comes from the environ-

ment and refers to environmental conditions like location, temperature, light or weather. However, physical context is only one aspect of context. Moreover, these infrastructures are usually centralized, using context servers that are queried to obtain relevant or useful context information.

### 3. A MAS FOR INFORMATION SHARING

We see an Ambient Intelligence environment like a large number of devices that serve the needs of their respective users. The devices are mostly going to deal with information: delivering relevant information to interested users, aggregating, filtering and reasoning about information. The problem that is asked is: given a certain piece of information, how to deliver that piece of information to the interested users – the users to which that information is relevant? This is a problem that is addressed to the intelligent services layer of an AmI system (the layer below the intelligent interfaces and above network and interoperability, as defined by El Fallah Seghrouchni (2008)) and can be solved by a middleware, between the lower layers of the system and the applications that need the information. We have named this middleware AmIciTy:Mi, as a part of the AmIciTy Ambient Intelligence system that we are in the process of building (see Olaru et al. (2010) and Olaru and Gratie (2010)).

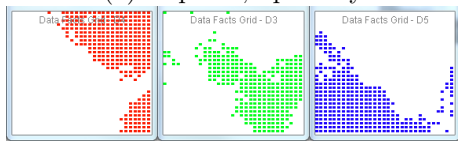
The design of the MAS that is presented in this section started from the following idea: at realistic scale, an AmI system will have to deal with a very large number of users and an even greater number of devices that communicate between each other. The implementation is based on two elements: the use of agents and the application of mechanisms of self-organization. In this project a structure for individual agents was defined, that will allow them, by means of large numbers and intense interaction, to fulfill the global desired goal – the context-aware sharing of information – and this by means of limited knowledge and reasoning, and local behavior and communication.

The purpose of the system is the following: a large number of agents is given, that have a location in space and cover a certain rectangular area. The agents can only communicate locally, with their neighbors. A user of the system can insert a piece of data into the system, that contains indications of the domains to which it is of interest, of its importance, and of its validity. The purpose of the system is to distribute the said piece of knowledge among the agents in the system, so that other users – in other locations – that are interested in the data will be able to know it.

To provide context-awareness (i.e. providing the user with the relevant information, in the current context) for information sharing, four simple and generic aspects of context-awareness have been proposed: first, space is implicitly considered, because of the structure of the system, that relies on local behavior and communication; second, temporal context is implemented as a period of validity for each piece of information; third, each piece of information is related to certain domains of interest; last, each piece of information carries a direct indication of its relevance (estimated by the source).



(a) step 130, Specialty



(b) step 271, Distribution of new facts

Fig. 1. In each panel, a view on the multi-agent system is presented: each cell in the grid corresponds to an agent and shows its specialty or if it *knows* contains a certain piece of information: (a) specialty of agents, aggregated for all domains of interest (first panel) and for each of the domains *A*, *B* and *C* (next three panels); (b) distribution of facts, 140 steps later, for facts with Specialties related to domains *A*, *B* and *C*.

Context compatibility can be seen as *proximity* between the two contexts – the context of the new information and the context of agent – in terms of the aspects enumerated above. A more detailed description of these aspects of context-awareness, together with their influence on how information is shared and spread through the system is presented below:

**Local behavior and interaction** – leads to inherent location awareness. New information will first reach the agents in the area where the information was created (e.g. where the event took place). In function of the other aspects of context-awareness, the information will only stay in the area or will spread further. Also, all other measures equal, agents will give less relevance to information related to a farther location.

**Time persistence** – shows for how long is the information relevant. When its validity expires, the agents start discarding the piece of information.

**Specialty** – shows how the information relates to some *domains of interest*. In time, agents form their own notion of specialty in function of the information that they have. New information is considered more relevant if it is more similar to the agent's specialty, and agents share relevant information first, and they share it with agents that are more likely to consider it relevant. This influences the direction in which information is spread.

**Pressure** – shows how important it is for the information to spread quickly. Pressure translates into higher relevance and the agent will treat the information with higher priority. Also, the higher the pressure, the more neighbors the agent will send the information to. This way, pressure controls how quickly the information spreads.

The relevance of a certain piece of information for a certain agent is calculated by aggregating the last three measures above, which are associated with the information, and also by comparing the specialty of the information to the specialty of the agent.

Figure 1 shows some results that have been obtained, that related especially to the direction in which information

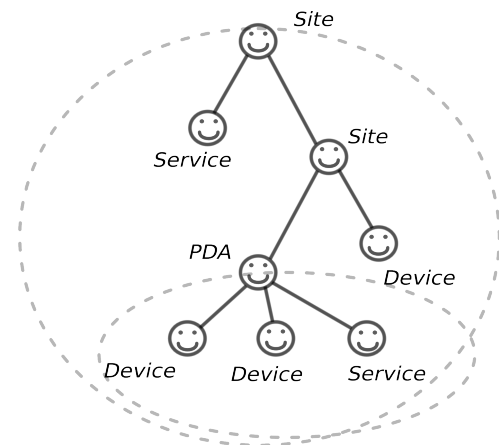


Fig. 2. Example of an abstract logical hierarchy for the Ao Dai project: The root *Site* offers a *Service* and also contains another *Site* that contains a *Device*. The user is inside the second site, and its *PDA* can offer him the capabilities of two *Devices* and a *Service*.

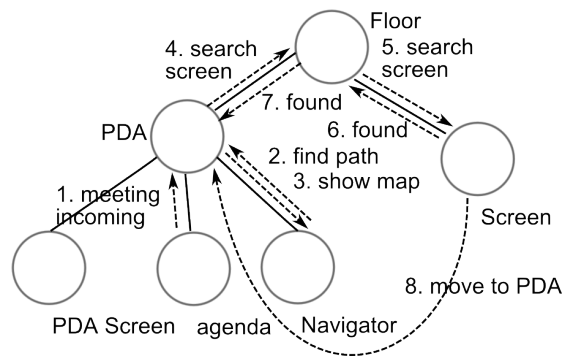


Fig. 3. The user (and its PDA) enters the floor of a building. It uses a navigation service, an agenda service and its screen device. On the floor there is also a screen device. The figure presents the sequences of messages exchanged between agents: *Agenda* announces a new meeting, *PDA* asks a path from *Navigator*, which in turn requires a larger screen – which is searched on the floor, and found.

spreads: it is easy to see the direction of the spreading is strongly influenced by the preexisting specialty of agents. The other context measures also influence the spread: higher pressure makes information spread considerably faster; after the persistence of information expires, it quickly disappears from the system; and, of course, the first agents that get to know a piece of information are the ones closer to its source.

#### 4. THE AO DAI PROJECT

This section presents a second approach to the implementation of Ambient Intelligence. If AmIciTy:Mi deals with the scalability of future AmI systems, the Ao Dai project – Agent-Oriented Design for Ambient Intelligence – studies in more detail the connection between agents and context-awareness, in which context is represented in a more advanced manner than in AmIciTy:Mi .

The purpose of the Ao Dai project (presented in El Fallah Seghrouchni et al. (2010b)) was to demonstrate how

an agent-based AmI system can assist the user in a simple scenario, based on the awareness of several types of contexts.

The Ao Dai project has been implemented in CLAIM. CLAIM, developed by Suna and El Fallah Seghrouchni (2004), is an agent-oriented programming language that is based on explicit declaration of agent's characteristics: knowledge, goals, messages, capabilities. All these components are defined using first-order predicate logic and the programmer can program agents by working only at this higher level – the Java-based Sympa platform manages the creation, execution and migration of agents.

What is more important about CLAIM is that, on the one hand, it offers strong mobility of agents (which is made seamless and effortless by the Sympa platform), and on the other hand, it allows agents to be placed in a logical hierarchy (that can span across different machines). The idea of the Ao Dai architecture is to map the hierarchy of agents to a hierarchy of contexts, considering physical and computational contexts, as well as user's preferences.

An example of such an hierarchy is presented in Figure 2: the hierarchy of agents reflects the hierarchical structure of spatial and computational context. Figure 3 presents the interaction between the Ao Dai agents: the agents communicate based on the local context. For instance, searching for devices / services with certain capabilities is done first in the agent's subtree of agents, then the agents queries its parent, which in turn searches in a larger context. This procedure has two advantages: first, the communication is decentralized, and the system is able to scale; second, the first results will be found in a context that is closer to the user.

## 5. A NEW APPROACH TO CONTEXT-AWARENESS

AmIciTy:Mi offers a behavior for agents that allows them to obtain a useful global behavior by means of only local knowledge and actions. Ao Dai offers an architecture that allows for implicit context-awareness in a multi-agent system. Combining the two approaches is the natural choice. But: how would this combination look, and what would it do?

The result that is proposed in this paper is a middleware, as the two projects were themselves. In this middleware, there are one or more agents associated to each device, and one agent associated with each service. The middleware is formed only of the agents that compose it – no other components are needed. The devices access the middleware by means of the agent(s) that exist on them. It is the devices that provide information (perceptions) to the agents, in a uniform representation – which is provided by the interoperability layer of the AmI system – and it is the agents that offer to the applications the potentially relevant information – applications that execute on the devices; the information is then used by the applications, or presented to the users by means of the intelligent interfaces. Figure 4 depicts this process.

Unlike in AmIciTy:Mi, the new middleware does not use a simple topology: it will use a topology based on the ideas in Ao Dai, where agents were placed in a hierarchy. It will, however, use the same behavior for agents and the same

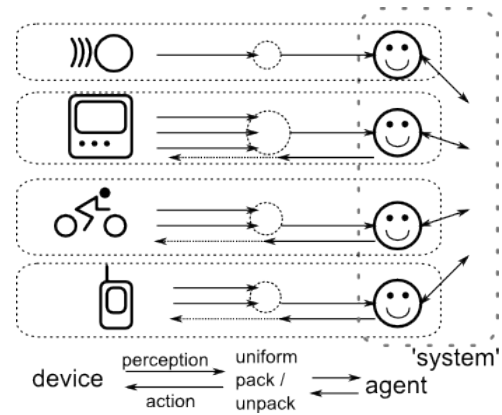


Fig. 4. The structure of the middleware, as seen from the perspective of the devices. The "system", i.e. the middleware, is actually formed of the agents that compose it. The agents are executing on the devices, using their communication hardware.

policies for the spreading of information, but within the new topology.

Ao Dai uses a hierarchical topology, where agents are placed in one or more trees of agents. This is because CLAIM offers the possibility of working easily with agents that are placed in hierarchies, and it is very easy to move whole sub-hierarchies of agents. But Ao Dai considers only two types of context: *location* and *computational resources*. These types of context are inherently hierarchical: places are part of larger places, and computational resources cover certain spaces. But there are also other types of context: *temporal context*, *social context*, and *activity* as a context.

These last three types of context also have a hierarchical aspect: time can be iteratively divided in intervals of time; activities are formed of sub-activities; and social groups may also have hierarchies. Moreover, activities take place in a certain interval of time, and concern certain users, which, by participating in the same activity, form a group.

It is clear that, in order to support more types of context, more complex relations between agents than in Ao Dai are necessary. We introduce the following set of relations, and some new types of agents, that are also depicted in Figure 5:

- for spatial context – places – the *Place* agent (already existing in Ao Dai as *Site*) and the *is-in* relation. This relation can exist between subordinate *Place* agents, or between *Place* agents and other types of agents – like *Users*, *Devices*, *Services* or *Activities*. For instance, when a *Service* agent is a child of a *Place* agent, it means that it is a context-aware service that is offered within that place (and is useful only there, as it is specifically configured for that place). *Place* agents execute on machines that are connected to the network access points in those respective spaces.
- for computational context – devices and services – the *Device* and *Service* agents (already existing in Ao Dai) and several relations: *executes-on* for services, *controlled-by* and *is-in* for devices. In the case of devices, we must already break the structure of hierarchy, and introduce multiple parents for *Device*

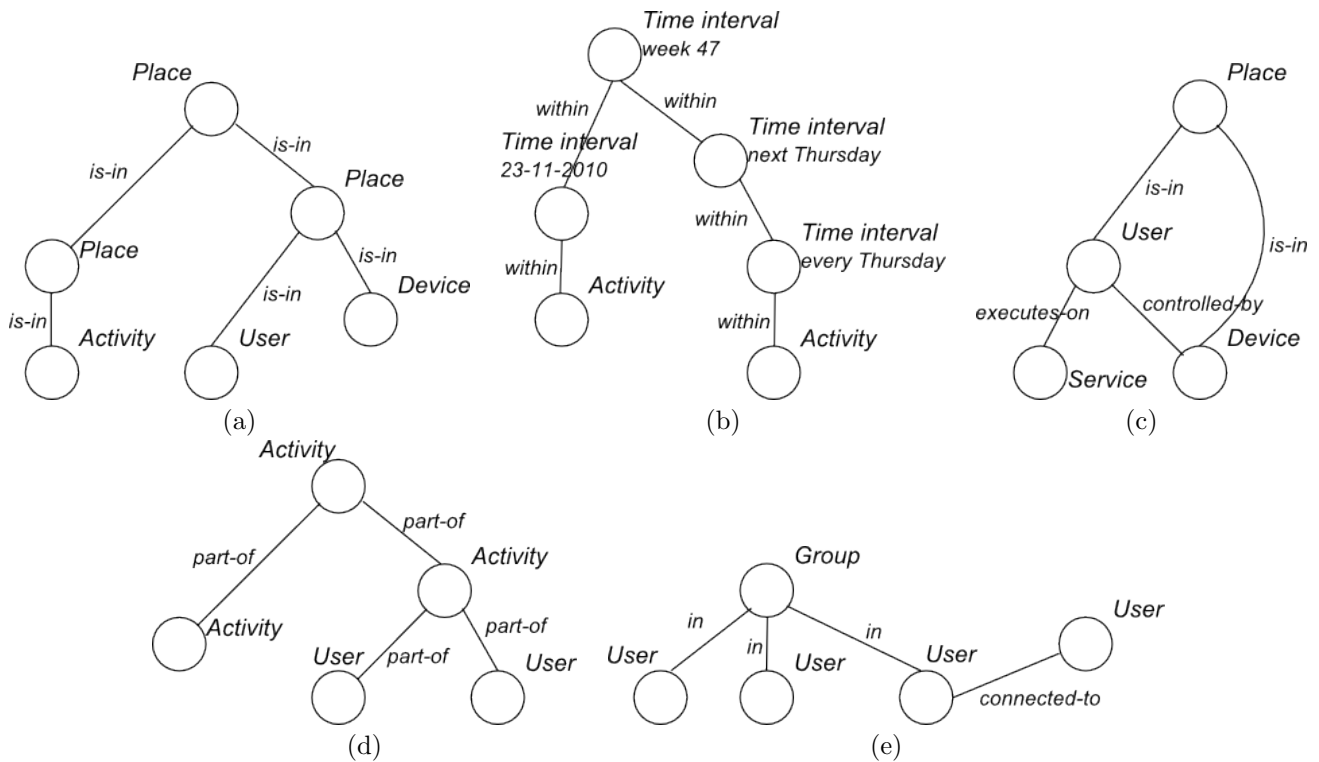


Fig. 5. Possible relations between agents, relative to the different types of context: spatial (a), temporal (b), computational (c), activity (d) and social (e).

agents. When a device is not in use, it will only have one parent, a *Place* or another *Device*, and the relation will be *is-in*. However, when used, it will become a logical child of the *User* agent, by means of the relation *controlled-by*, while keeping the link with its location (in the case the device does not move – like a presentation screen).

- for temporal context – time and time intervals – the *Time interval* agent and the *within* relation. The *within* relation designates time intervals contained by larger intervals, or *Activities* that take place in a certain interval of time. Because time is continuous, a time interval (that is relevant in relation with some activity) may span across multiple higher-level intervals. Similarly, an activity or event may happen at regular intervals of time. For these types of complex relations, the activity’s time interval will have as a “main” parent (the parent of the agent) the interval which contains the present moment. The *Time interval* agents execute on the same machines where the corresponding *Activity* agents are executing.
- for activity context, the *Activity* agent and the *part-of* relation. The *Activity* agents execute on a machine that is related to the organization of the activity, or to the user that coordinates the activity – for activities where multiple users take part. For the user’s personal activities, the *Activity* agent is a child of the *User* agent, linked by a *part-of* relation.
- finally, for social context, the *User* agent is used to represent users of the system. The *User* agent executes on the user’s PDA, or on any device that the user is currently using. For representing relationships between users of the system, two relations are used: the *in* relation shows that the user is part of a

larger group of users – managed by a *Group* agent. The *connected-to* relation shows that two users are connected, without them being part of a common group, taking part in a common activity or being in the same place. While the *in* relation is hierarchical, the *connected-to* relation is not.

The result of using the elements described above is both an architecture for a context-aware middleware, and also a way to model context. The resulting structure contains different hierarchies – corresponding to different types of context – that are intertwined. One may ask: Why did we need hierarchies after all? First, because hierarchies are easier to manage, even if now there are multiple, intertwined hierarchies. Second, because, thanks to CLAIM, agents in the sub-tree of an agent can easily move together with their parent, and this remains true now: for instance, if an activity changes place (e.g. moves to a different room), when the *Activity* agent changes parent, the *User* agents that are *part-of* the activity, and the *Services* and *Devices* that the users use, move with them.

The topology presented above has an essential advantage: the links between agents represent common context. When an agent needs a certain piece of information, or needs a device / service that offers a certain capability, it will first search for it in its sub-tree of agents, then in its “main” parent, then in the other connections that it has. This manner of interaction assures that the search is first done in the context closest to the agent. When information is disseminated (as in AmIciTy:Mi), the agent will send it to its connections that are potentially interested in that piece of information. However, this time its connections are not only related to location in space or to abstract domains of interest, but they are also related to common context.

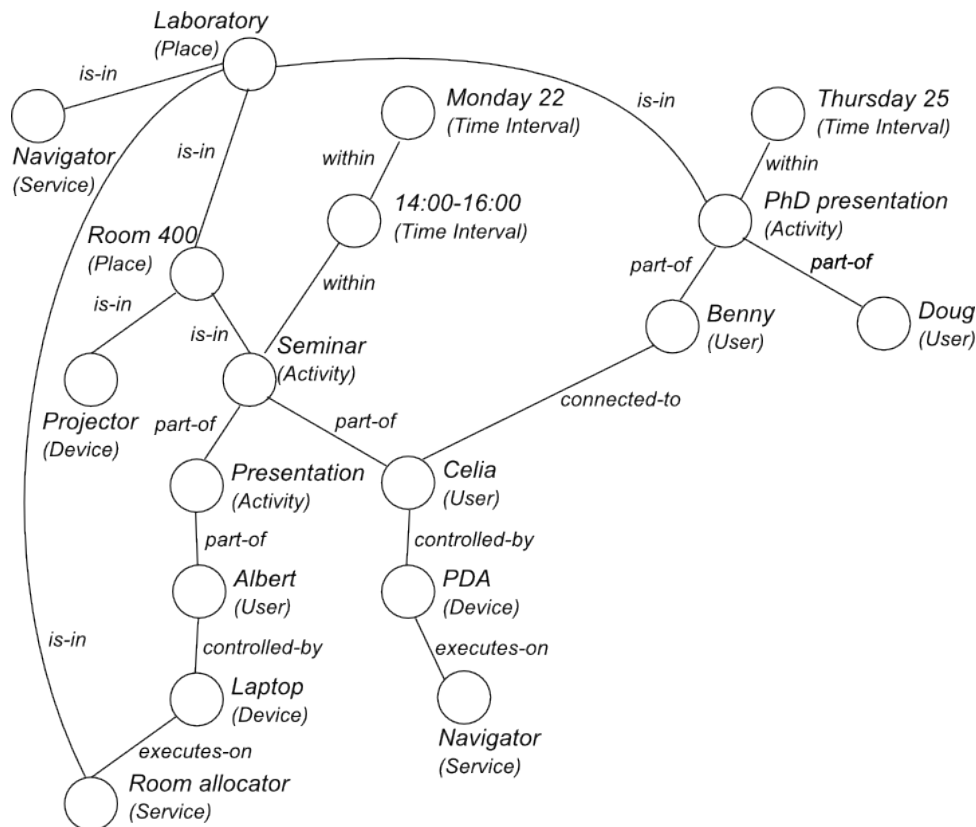


Fig. 6. Structure of agents for the scenario presented in Section 6. The figure displays the names of the agents, the types of the agents, and the relations between the agents.

If the information itself relates to a certain context, the agent needs to send it only to the hierarchy that is related to that context.

## 6. AN EXAMPLE

Take the following scenario: on Monday 22nd, there is a Seminar in Room 400 of the Laboratory. The presenter is Albert. One of the participants is Celia, who is not working within the Lab, so the AmI services that exist there must assist her to find her way to the room. At some point, the projector that is in the room brakes down, because of a malfunction of its cooling. At this point, Albert needs to find another room for the presentation. Also, other people that are giving presentations in the same room in the following days should be announced, so they would first check if the projector is working.

The structure of the multi-agent system is presented in Figure 6. It uses the types of agents and the relations from figure 5: The Seminar is an *Activity* that takes place in Room 400 (which is part of the Laboratory), *within* the interval 14:00-16:00 of Monday 22. The Projector is a *Device* that *is in* the Room. *Part of* the Seminar there is a Presentation, that is held by Albert, which is a *User*. The *User* Celia also takes part in the Seminar. Since she is not a member of the Laboratory, the agent that handles the services offered by the Laboratory produced an instance of the Navigator agent (a context-aware *Service*) that was sent to Celia's PDA, in order to help her reach Room 400.

Given the event described at the beginning of this section, two things happen. First, Albert issues a search for a room

with a projector. He does this with the help of the Room Allocator, which is a context-aware *Service* offered by the Lab. This search will be sent from agent to agent, starting from the agent Albert. The search will only be sent to agents to which the search is relevant: the search relates to a Room (which is a *Place*) and to a Projector (which is a *Device*) so the search will follow only relations of the type *is-in* between *Places* or between a *Place* and a *Device*. Also, agents will first search in their sub-trees before searching in their parent. When a room will be found, the participants will move and the agents that are concerned by the movement (Seminar, Celia, Albert, etc.) will become part of the new room's sub-tree of agents.

A second piece of information that will be sent through the system will be that the Projector in Room 400 is not working. This will be shared by Albert's agent with agents to which the information is considered relevant. The information will reach the Room 400 agent, the Laboratory agent and finally the PhD Presentation – an *Activity* agent – and its organizer – Doug. Although the connection to Doug can also be done by means of users Celia and Benny, the information will not take this path, because it is not relevant to them – there is no need for those agents to have it. It may also be possible that Benny's agent receives the message. But as Benny is not the presenter, his agent (being aware, according to his internal knowledge, of Benny's role in the activity) will reason that this information is not relevant to Benny and will discard it.

## 7. CONCLUSIONS

In the implementation of AmIciTy:Mi a simple, local behavior for agents was created, that allows agents, with only very little knowledge, to share information so that, at the global level, information reaches the agents that are potentially interested by that piece of information.

The realization of the Ao Dai project showed that agents can be placed in a hierarchy that will reflect the structure of the user's context, making context-aware communication and action natural and easy to implement.

Using elements of these previous approaches, this paper presents an architecture for a multi-agent system to serve as a layer of an AmI system. This architecture is based on a topology of connections between agents that reflect the relation between different types of context. An example was presented, showing how this architecture would work, on a simple scenario.

As future work, formal semantics for the agent topology, as well as for the operations with it, should be developed, as in the CLAIM language. This way it would be easy to prove that the system cannot reach any inconsistent state – i.e. a structure inconsistent with the presented rules – and that information will follow the intended paths.

Implementation on a large scale system is mandatory for validating of the architecture. Since CLAIM is based on Java, and the Sympa platform manages the creation and mobility of agents, as well as all their interactions, it will be reasonably easy to deploy the system in a larger environment. Validation will also need a more complex scenario than the one presented in the paper.

## ACKNOWLEDGEMENTS

The authors would like to thank Cristian Gratie, who participated in the implementation of AmIciTy:Mi, Nguyen Thi Thuy Nga and Diego Salomone, who implemented the Ao Dai project in CLAIM, and prof. Amal El Fallah Seghrouchni, who supervised the realization of the Ao Dai project.

## REFERENCES

- Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4), 263–277.
- Cabri, G., Ferrari, L., Leonardi, L., and Zambonelli, F. (2005). The LAICA project: Supporting ambient intelligence via agents and ad-hoc middleware. *Proceedings of WETICE 2005, 14th IEEE International Workshops on Enabling Technologies, 13-15 June 2005, Linköping, Sweden*, 39–46.
- Cardelli, L. and Gordon, A.D. (2000). Mobile ambients. *Theor. Comput. Sci.*, 240(1), 177–213.
- Chen, G. and Kotz, D. (2000). A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College.
- Chen, H., Finin, T.W., Joshi, A., Kagal, L., Perich, F., and Chakraborty, D. (2004). Intelligent agents meet the semantic web in smart spaces. *IEEE Internet Computing*, 8(6), 69–79.

- Costantini, S., Mostarda, L., Tocchio, A., and Tsintza, P. (2008). DALICA: Agent-based ambient intelligence for cultural-heritage scenarios. *IEEE Intelligent Systems*, 23(2), 34–41.
- Dey, A., Abowd, G., and Salber, D. (1999). A context-based infrastructure for smart environments. *Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE'99)*, 114–128.
- Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., and Burgelman, J. (2001). Scenarios for ambient intelligence in 2010. Technical report, Office for Official Publications of the European Communities.
- El Fallah Seghrouchni, A. (2008). Intelligence ambiante, les défis scientifiques. presentation, Colloque Intelligence Ambiante, Forum Atena.
- El Fallah Seghrouchni, A., Breitman, K., Sabouret, N., Endler, M., Charif, Y., and Briot, J. (2008). Ambient intelligence applications: Introducing the campus framework. *13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'2008)*, 165–174.
- El Fallah Seghrouchni, A., Florea, A.M., and Olaru, A. (2010a). Multi-agent systems: a paradigm to design ambient intelligent applications. In M. Essaaidi, M. Malgeri, and C. Badica (eds.), *Proceedings of IDC'2010, the 4th International Symposium on Intelligent Distributed Computing*, volume 315 of *Studies in Computational Intelligence*, 3–9. Springer. Invited paper.
- El Fallah Seghrouchni, A., Olaru, A., Nguyen, T.T.N., and Salomone, D. (2010b). Ao Dai: Agent oriented design for ambient intelligence. In *Proceedings of PRIMA 2010, the 13th International Conference on Principles and Practice of Multi-Agent Systems*.
- Feng, L., Apers, P.M.G., and Jonker, W. (2004). Towards context-aware data management for ambient intelligence. In F. Galindo, M. Takizawa, and R. Traunmüller (eds.), *Proceedings of DEXA 2004, 15th International Conference on Database and Expert Systems Applications, Zaragoza, Spain, August 30 - September 3*, volume 3180 of *Lecture Notes in Computer Science*, 422–431. Springer.
- Hagras, H., Callaghan, V., Colley, M., Clarke, G., Pounds-Cornish, A., and Duman, H. (2004). Creating an ambient-intelligence environment using embedded agents. *IEEE Intelligent Systems*, 12–20.
- Harter, A., Hopper, A., Steggle, P., Ward, A., and Webster, P. (2002). The anatomy of a context-aware application. *Wireless Networks*, 8(2), 187–197.
- Hellenschmidt, M. (2005). Distributed implementation of a self-organizing appliance middleware. In N. Davies, T. Kirste, and H. Schumann (eds.), *Mobile Computing and Ambient Intelligence*, volume 05181 of *Dagstuhl Seminar Proceedings*, 201–206. ACM, IBFI, Schloss Dagstuhl, Germany.
- Henricksen, K. and Indulska, J. (2006). Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing*, 2(1), 37–64.
- Hong, J. and Landay, J. (2001). An infrastructure approach to context-aware computing. *Human-Computer Interaction*, 16(2), 287–303.

- Lech, T.C. and Wienhofen, L.W.M. (2005). AmbieAgents: a scalable infrastructure for mobile and context-aware information services. *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands*, 625–631.
- Olaru, A. and Gratie, C. (2010). Agent-based information sharing for ambient intelligence. In M. Essaïdi, M. Malgeri, and C. Badica (eds.), *Proceedings of IDC'2010, the 4th International Symposium on Intelligent Distributed Computing, MASTS 2010, the The 2nd International Workshop on Multi-Agent Systems Technology and Semantics*, volume 315 of *Studies in Computational Intelligence*, 285–294. Springer.
- Olaru, A., Gratie, C., and Florea, A.M. (2010). Context-aware emergent behaviour in a MAS for information exchange. *Scalable Computing: Practice and Experience - Scientific International Journal for Parallel and Distributed Computing*, 11(1), 33–42. ISSN 1895-1767.
- Ramos, C., Augusto, J.C., and Shapiro, D. (2008). Ambient intelligence - the next step for artificial intelligence. *IEEE Intelligent Systems*, 23(2), 15–18.
- Sadeh, N.M., Gandon, F.L., and Kwon, O.B. (2005). Ambient intelligence: The MyCampus experience. Technical Report CMU-ISRI-05-123, School of Computer Science, Carnegie Mellon University.
- Satoh, I. (2004). Mobile agents for ambient intelligence. In *Proceedings of Massively Multi-Agent Systems I, First International Workshop, MMAS 2004, Kyoto, Japan, December 10-11, 2004, Revised Selected and Invited Papers*, volume 3446 of *Lecture Notes in Computer Science*, 187–201. Springer.
- Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *IEEE Personal communications*, 8(4), 10–17.
- Spanoudakis, N. and Moraitis, P. (2006). Agent based architecture in an ambient intelligence context. *Proceedings of the 4th European Workshop on Multi-Agent Systems (EUMAS'06), Lisbon, Portugal*, 1–12.
- Suna, A. and El Fallah Seghrouchni, A. (2004). Programming mobile intelligent agents: An operational semantics. *Web Intelligence and Agent Systems*, 5(1), 47–67.
- Weiser, M. (1995). The computer for the 21st century. *Scientific American*, 272(3), 78–89.