Intelligent Agents and Services for Smart Environments

AISB '08

AISB 2008 Proceedings Volume 8

GLoRiClass

AISB ABERDEEN - 2008

UNIVERSITY OF ABERDEEN

AISB 2008 Convention
Communication, Interaction and Social Intelligence

1st-4th April 2008

University of Aberdeen

Volume 8:
Proceedings of the AISB 2008 Symposium on Intelligent Agents and Services for Smart Environments

Published by
The Society for the Study of Artificial Intelligence and Simulation of Behaviour

http://www.aisb.org.uk/convention/aisb08/

ISBN 1 902956 67 2

Contents

The AISB'08 Convention ii
Frank Guerin & Wamberto Vasconcelos

Symposium Preface iii
Flávio Soares Corrêa da Silva & Stefania Bandini

Institutional Social Networks for Ambient Intelligence 1
Guido Boella, Leendert van der Torre & Serena Villata

Smarts Agents and Smarts Environments: a Predictive Approach to Replanning 7
Alfredo Garro, Sergio Greco & Fabio Palopoli

Semantic Web Services for Intelligent Responsive Environments 13
Crhistian Alberto Noriega Guerra & Flávio Soares Corrêa da Silva

A Middleware for Smart Environments 22
Crhistian Alberto Noriega Guerra & Flávio Soares Corrêa da Silva

The AISB'08 Convention: Communication, Interaction and Social Intelligence

As the field of Artificial Intelligence matures, AI systems begin to take their place in human society as our helpers. Thus it becomes essential for AI systems to have sophisticated social abilities, to communicate and interact. Some systems support us in our activities, while others take on tasks on our behalf. For those systems directly supporting human activities, advances in human-computer interaction become crucial. The bottleneck in such systems is often not the ability to find and process information; the bottleneck is often the inability to have natural (human) communication between computer and user. Clearly such AI research can benefit greatly from interaction with other disciplines such as linguistics and psychology. For those systems to which we delegate tasks: they become our electronic counterparts, or agents, and they need to communicate with the delegates of other humans (or organisations) to complete their tasks. Thus research on the social abilities of agents becomes central, and to this end multi-agent systems have had to borrow concepts from human societies. This interdisciplinary work borrows results from areas such as sociology and legal systems. An exciting recent development is the use of AI techniques to support and shed new light on interactions in human social networks, thus supporting effective collaboration in human societies. The research then has come full circle: techniques which were inspired by human abilities, with the original aim of enhancing AI, are now being applied to enhance those human abilities themselves. All of this underscores the importance of communication, interaction and social intelligence in current Artificial Intelligence and Cognitive Science research.

In addition to providing a home for state-of-the-art research in specialist areas, the convention also aimed to provide a fertile ground for new collaborations to be forged between complementary areas. Furthermore the 2008 Convention encouraged contributions that were not directly related to the theme, notable examples being the symposia on "Swarm Intelligence" and "Computing and Philosophy".

The invited speakers were chosen to fit with the major themes being represented in the symposia, and also to give a cross-disciplinary flavour to the event; thus speakers with Cognitive Science interests were chosen, rather than those with purely Computer Science interests. Prof. Jon Oberlander represented the themes of affective language, and multimodal communication; Prof. Rosaria Conte represented the themes of social interaction in agent systems, including behaviour regulation and emergence; Prof. Justine Cassell represented the themes of multimodal communication and embodied agents; Prof. Luciano Floridi represented the philosophical themes, in particular the impact of society. In addition there were many renowned international speakers invited to the individual symposia and workshops. Finally the public lecture was chosen to fit the broad theme of the convention – addressing the challenges of developing AI systems that could take their place in human society (Prof. Aaron Sloman) and the possible implications for humanity (Prof. Luciano Floridi).

The organisers would like to thank the University of Aberdeen for supporting the event. Special thanks are also due to the volunteers from Aberdeen University who did substantial additional local organising: Graeme Ritchie, Judith Masthoff, Joey Lam, and the student volunteers. Our sincerest thanks also go out to the symposium chairs and committees, without whose hard work and careful cooperation there could have been no Convention. Finally, and by no means least, we would like to thank the authors of the contributed papers – we sincerely hope they get value from the event.

Frank Guerin & Wamberto Vasconcelos

The AISB'08 Symposium on Intelligent Agents and Services for Smart Environments

A smart environment is endowed with autonomous behaviour. It can sense what occurs within itself and its surroundings, and adapts its actions accordingly.

Intelligent multiagent systems and Semantic Web Services constitute relevant technologies to design and implement smart environments. They provide for decentralised and robust platforms upon which autonomous software modules can interact with themselves and with sensors and actuators.

Among the challenges related to the design and implementation of smart environments, we highlight:

- Technical requirements related to real-time response using heterogeneous devices with possibly limited storage and processing capabilities.
- Socio-technical requirements due to smart environments being inhabited by humans, who typically produce partially reliable information.
- Ethical requirements related to smart environments interfering directly with social activities. Data collected by smart environments can raise privacy and ownership issues.

Applications of smart environments are many and diverse: they can be employed to enhance the human experience in interacting with systems for entertainment, arts and culture, public services and education.

This workshop is devoted to the exploration of multiagent systems and Semantic Web Services to support the design and implementation of smart environments. We are interested in theoretical and foundational issues, as well as project reports of practical applications.

Flávio Soares Corrêa da Silva & Stefania Bandini

Institutional Social Networks for Ambient Intelligence

Guido Boella[1] and **Leendert van der Torre**[2] and **Serena Villata**[3]

Abstract. Ambient Intelligence creates scenarios in which the reality is modified by devices that augment the possibilities of social interaction. In this paper we propose an approach based on institutions to model the virtual reality created by an application of Ambient Intelligence. Starting with our previous results that give a model of institution in terms of the relations of power and dependence, defined by means of a description of goals and skills of a single agent, our aim is to show with this framework, thanks to the help of an example, how to model the social structures developed in a system of Ambient Intelligence thanks to the notion of institution.

1 INTRODUCTION

The social network theory is emerged as a key issue in modern sociology, communication and information science. This theory has been used to connote complex sets of relationships between members of social systems, from an interpersonal point of view to an international one.

Ambient Intelligence [14] is a vision that enables intelligent environments by means of pervasive technology. This vision puts human users at the center of the discussion, but technological devices and humans are seen as equal inside the environment, collaborating, to improve both human being and machine performance. Ambient Intelligence offers many benefits. For example, by instrumenting public and private spaces to understand their users activities and requirements, embedded intelligent systems can react by guiding an elderly person, helping students to improve their learning and others relevant contributions. Artificial intelligence is the key technology for enabling and catalyzing this vision. In particular, AI theories, like the Multiagent one [17], will make it possible to model complex realities that represent new levels of interaction among groups of humans that are created by the use of technological devices.

A social network is a social structure composed by nodes and arcs where nodes usually represent individuals or organizations while arcs represent dependencies among nodes. Arcs can represent various types of dependencies like financial exchange, conflict, trust or friendship. Our framework, using the methodology of dependence networks, presents a kind of social networks called institutional social networks with the aim to describe an environment of Ambient Intelligence and its inner features with the application of the concept of institution. An institutional social network is a social network that represents set of individuals regulated by norms and in which it is present the application of social roles to each individual involved. We model Ambient Intelligence applications as institutional social networks using the multiagent model, because it seems to be a descriptive framework able to represent all facets that are present in an environment of Ambient Intelligence. We use our previous results of [7] as framework and we apply it to an environment of Ambient Intelligence with the aim to give a compact and understandable model. This breaks down in the following questions:

- How to define an approach based on the multiagent model and using social networks to describe an environment of Ambient Intelligence?
- How and why to model the use of a technological device in Ambient Intelligence as a new level of reality?

A Multiagent system can be viewed as an environment populated by agents. These agents interact with each others creating a complex net of dynamics inside the system. The study of these dynamics and, as a consequence, of the social structures [2] (such as groups and collectives) is an important aim in the field of Multiagent Systems. In a single agent framework, to achieve a given goal an agent has to be able to do it. On the contrary in Multiagent frameworks, especially those in which agents are heterogeneous and have different abilities, it is possible that, when an agent is not self-sufficient with respect to some goal, he can resort to some other agent, given that the latter cannot be self-sufficient itself in every respect. Hence, agents benefit from the interaction with the other agents and cooperate with them to achieve the goals of the other agents of the system. This makes clear the existence of relations as power and dependence that are the base of the social and organizational structure of a system.

In a scenario of Ambient Intelligence, we can see humans as the agents of a Multiagent system or associated with agents assisting them that have to interact with each other. The reality that describes the internal state of the agent, in terms of its goals and beliefs can be defined the private realty of the agent. This kind of reality can be viewed not only as private beliefs and goals of the agent but, from a multiagent point of view, also as the relationships among agents of the system that describe the power to achieve goals and can be visualized thanks to dependence networks. The use of a technological device, such as a pocket pc, establishes a new level of reality in the relationships between involved people. This reality is created thanks to only the use of the technological device and can be defined as the public or institutional reality. This new kind of reality can be considered as the public version of the private one because it contains public beliefs and goals of the agent. Moreover, the relationships among agents change thanks to the addition of the institutional reality and institutional social networks have to represent also the institutional powers to achieve goals.

For example, if students use a pocket pc during a lesson at the same time the teacher can limit the number of messages that they can send to their school friends not nearby. In this case, we have on one hand the private reality of the relationships among people (a student can

[1] Department of Computer Science, University of Turin, Italy, email: guido@di.unito.it
[2] Computer Science and Communications, University of Luxemburg, Luxemburg, email: leon.vandertorre@uni.lu
[3] Department of Computer Science, University of Turin, Italy, email: villata@di.unito.it

speak with his school friends, also if the teacher won't) and, on the other hand, the institutional reality of the relationships developed thanks to the device that improves a new type of communication (the teacher can block the possibility to send messages among students). The example shows the two types of reality that have been created by the use of the technological device. It is important to note that also the roles of the people involved in the system are relevant to establish the relation among people in both realities. From a methodological point of view, inspired by Sichman and Conte [15], we use the notion of agent dependence to create dependence graphs extended in [3], in order to highlight the topology and the symmetries of dependencies. Giving the agents the ability to reason about their social relations it enables us to model the institutional reality, moreover it makes it possible to proceed from a hierarchical view of institutional design to a more dynamic approach, where the agents are able to define their own powers, obligations and permissions on the actions performed by the other agents. We use a scenario for Ambient Intelligence to illustrate the different facets and cases in which the two levels of reality are more evident.

In this paper we don't treat the management of the system of pocket pc from a point of view of the implementation of software agents and of the architecture of the system [12] and the development of systems to support human organizations [10].

The layout of this paper is as follows. In Section 2 we introduce our running example of Ambient Intelligence for teaching a lesson with the supply of pocket pc for the students and the teacher. In Section 3 we formalize the concepts of power and social dependence networks. In Section 4 we apply the dependence framework to role-based institutions to model the scenario for Ambient Intelligence. Related work and conclusions end the paper.

2 THE ENHANCED CLASSROOM SCENARIO

The scenario we will describe is based on the use of technological devices during a lesson in a classroom. Our aim is to create a situation in which the participants (in this case, students) augments their possibilities of interaction thanks to the presence of the technological interface. Each student and the teacher have been provided with a pocket pc that augment the possibilities of communication among the students and also between the students and the teacher. We will follow the development of a lesson to underline the discordance from a common lesson. First, the teacher arrives in the classroom. At this point it is important to note that the privileges associated to the two types of pocket pc are different. In fact, the pocket pc of the teacher is provided by a software that lets him to see the flow of messages that the students send to each other (also with a graphic visualization) and he has also the possibility to stop selectively this type of messages if the students don't pay attention to the lesson. The reasons of this attention to the messages lies in the great importance that the possibility to communicate has for new generations. This is a fundamental component of their lives and the technological devices add new possibilities. In that way, the teacher can detect groups that take a form inside the class. From the point of view of emotional intelligence this is a relevant information. The number of groups can be changeable compared with the total number of students of a class. Moreover, these groups have not to be disjoint each other and some students can belong to more than one group. From the point of view of the social relationships, these groups represent ("happy islands" where these relationships are very intense because the members are strictly related to each others (thing that is documented by the flow of messages) by connections that can represent requests, acceptance of favors and meetings. In this context, we can also put in evidence the negative connotation of groups because they seem to be also close entities where it is difficult to enter for, for example, a new student arrived from another school. In this way the teacher can remedy if he notes that someone remains secluded from the other students trying more to involve him in the activity of the class.

Another point of view that has to be considered is the one of the presence of groups created virtually by the teacher with the aim to put together students with the same level of preparation in the subject without moving physically. First, this subdivision has not to be seen as a sort of discrimination because the reasons underlying a worse performance can be multifaceted, like the origin from another school, a different mother language and many others. The subdivision has the aim to help the teacher to do in parallel different kinds of lesson. For example, if the teacher is explaining a new topic, like the post-impressionist movement, he can send a preliminary material on the artists of this movement like Gauguin and Seurat, to the students that are on a base level and another material, consisting in quotations from the critic Rewald about this movement, to the students of high level. In this way, explaining the same topic, the teacher has given to the different students the kind of material more appropriate to them, allowing so a better preparation to all. The same point of view can be applied to the questions that the teacher poses to the students; as a matter of fact, he can pose questions with different level to the different groups to help the learning of everyone. The teacher can also individuate a representative for each group and he can send the material and the questions only to him. The representative is the only member of the group that has to send the answer of the group to the teacher and eventually the questions of the group to the teacher on the lesson. The students with a low level in the subject feel themselves in an ambient that, staring from their level, encourage themselves to do better, without the bad situation in which they don't understand anything because the lesson is too hard and advanced for them. The students of high level have so the possibility to increase their knowledge without hearing a lesson of a lower level that becomes boring. The analysis of the flow of messages is possible evidently only with the adoption of these devices and in a common lesson it is not possible to do that. Moreover, the subdivision of the material or, in particular, of the questions is more difficult, in particular the point of the questions. Another thing that has to be considered is that the teacher doesn't make public the groups that are created, so the students cannot be mocked by the others if they are in the group of lowest level. In a common lesson it is not possible or, better, do this distinction, taking care also of these problems is very difficult. For example, The messages can underlines two spontaneous groups of students. The first one composed by students S4, S5, S7 and S10 and the second one is composed by students S1, S2, S3, S4, S6, S8, S9. As can be seen, student S4 can belong to both the groups and so the groups are overlapping. Moreover, there can be other two different groups that represent two groups created by the teacher to divide the students of different level of preparation on the matter. The low level group is composed by students S2, S6, S7, S10 while the high level group is composed by students S1, S3, S4, S5, S8, S9. The teacher so poses two different questions to the two groups. An advantage of this type of learning is the possibility, connecting the pocket pc, to take part to the lesson as well if the student is ill or not in the classroom. This can help those students that have to stay at home (or also at the hospital) for long periods, but maintaining the physical possibility to follow a lesson. In this way, the student doesn't feel himself out of the class and gets behind also respect with the program of the subjects. This type of integration increases the sensation to be in

a ambient that helps them moreover all these things are not possible without the technological devices. Going on with the lesson, the teacher can start a new topic. The previous topic was the impressionist movement. Before starting with the new matter, the teacher would like to understand if the previous one is clear so he puts some questions to the students. The questions are posed using the pocket pc and they appear on all the pocket pc of the students. For example, the teacher can ask "What is the painting that represents the beginning of the impressionist movement?". The students who knows (or thinks to know) the answer can write it and than send it to the teacher (the answers can be send also to the class). The teacher reads the answers that appear on his monitor and then he can give the correct answer to the whole class, underlining or not the students that have supplied it. If we think to the same situations but without the use of any type of device, the students who want to answer the question have to do it in front of the whole class, risking to be mocked if it is wrong and to give a bad impression to the teacher. Because of one of these two reasons, a student can choose not to do answer to the question. These motivations can be solved also using anonymity in the messages. In practice, it is possible to use of pocket pc with a software that associates an alias or avatar to every student like "Student 1", "Student 2", ..., etc. In this perspective, another interesting reason to use the anonymity is the problem of prejudices of the teacher. If a student is considered to not have a gift for a particular subject, for example art, it is possible that the teacher has a different behavior with him during the lessons and during the answering of the questions. If the answering is done with the anonymity such behaviors disappeared automatically.

Another point can be that the teacher can choose every week a different student to substitute him during the lesson to answer to the questions of the other students. During the week all the questions done to the teacher are re-addressed to the pocket pc of the chosen student, that has to answer to them. This can be seen as a sort of training for the student or it can be used as method to do the exam of the matter. As said, the students can send messages to each other, apart from the possibility to send messages to the teacher. Such type of messages can contain different kinds of communications, from communications inherent to the lesson to communications inherent to a date for the next afternoon. First, it seems necessary that the teacher has the possibility to stop messages among the students if these are the reasons for a loss or a decrease of the attention of the class. The teacher is provided with a software that allows him to see the number of messages that the students send each other in real time. So, if this number overcomes a given threshold he can decide to stop the messages.

During the explanation of a new topic the teacher would like to know if the matter and the method used to treat it are considered interesting by the class. This is an important type of feedback that allows the teacher to know the degree of interest of the students and eventually to do some changes to make the lesson more interesting. In fact, the students can have a previous knowledge of a particular aspect of the topic, for example because they have treat it during another lesson of another subject. This type of feedback can be considered more realistic thanks to the presence of the anonymity that helps the students to be sincere. From the point of view of the students, this feedback return them a sensation of interest for their thoughts. A problem that can come out is that to the teacher can arrive too many questions at the same time and he is not able to answer. Moreover, the risk is that the teacher answers always to the questions of the same students and never or very rarely to the ones of the others (there is always anonymity). To manage this problem it is necessary to set a protocol that allows to every student to communicate to the teacher. This type of protocol can be used also to manage the answers that arrive from the students or from the representatives, if present. So it is possible to establish a protocol like the Delphi method that is used in the field of business to obtain answers to a problem from a panel of independent experts through a number of rounds. Since the physical space helps social interaction, the lesson has to be supported by a virtual ambient into the pocket pc with the features of visibility, awareness and accountability (a so called translucent system) as seen in Erickson [11]. The above scenario underlines the big difference that the use of the pocket pc brings to a traditional situation as a lesson. The institution adds new powers that are not possible in a situation without technological devices such as the possibility to communicate not only with the student seats nearby but also with the students on the other side of the classroom, the possibility to communicate with the class when we are ill at home or the possibility to stop the flow of messages among students or the creation of virtual groups inside the class. There is also the possibility to change the dependencies among agents, so if the teacher selects a student to answer to the questions send by the other students instead of the teacher, these students depend now from the selected student to obtain answers. We can argue that the net of dependencies among the participants of the lesson can be represented as an institutional social network where nodes represent people and arcs the dependencies created by the powers to achieve one own goals.

3 POWER AND SOCIAL DEPENDENCE NETWORKS

A simple representation of an agent is characterized by a set of features like the set of goals that he wants reach, the set of his beliefs and the set of skills that represent his capabilities. When an agent is put in a system that involves also other agents, he can be supported by the others to achieve his goals if he is not able to do them alone, thanks to the concept of power. In a Multiagent system, the concept of power, taken from the basic notions of Castelfranchi's social model [9], represents the capability of a group of agents (possibly composed only by one agent) to achieve some goals (theirs or of other agents) performing some actions without the possibility to be obstructed. The power of a group of agents can by defined as follows:

Definition 1 (Agents' power) $\langle A, G, pow : 2^A \rightarrow 2^{2^G} \rangle$ *where* A *is a set of agents,* G *is a set of goals. The function* pow *relates with each set* $S \subseteq A$ *of agents the sets of goals* $G_S^1, \ldots, G_S^m$ *they can achieve.*

Example 1 shows a set of agents and a set of goals taken from the scenario and what are the goals that each agent can achieve even if these aren't their own goals.

Example 1

- *Agents* $A = \{E, S, M, P, K\}$ *where agent P represents the teacher and the other agents represent the students. The teacher has created two groups for the base level of preparation on the subject Art and the high one. The two groups are* $\{S, M\}$ *and* $\{E, K\}$ *where the chosen representatives are agents E and S.*
- *Goals* $G = \{g_1, g_2, g_3, g_4\}$ *where* g_1*: to obtain the material sent by the teacher for its group from his pocket pc,* g_2*: to communicate with other students using the pocket pc,* g_3*: to obtain some feedbacks via pocket pc on the topic of Post-impressionism and* g_4*: to obtain an answer to the question "Why artists as Van Gogh or Gauguin are often considered as impressionists?".*

- $pow((S,E),(g_1))$,$pow((K,S,M,E),(g_3))$,$pow((P),(g_2,g_4))$

3.1 Social dependence networks

In order to define the relations that exist between the agents of the system in terms of goals and powers to achieve these goals, we adopt the methodology of social dependence networks as developed by Conte and Sichman [15]. In these models, an agent is described by a set of prioritized goals, and there is a global dependence relation that explicates how an agent depends on other agents for fulfilling its goals. For example, *dep*$(\{a,b\},\{c,d\}) = \{\{g_1,g_2\},\{g_3\}\}$ expresses that the set of agents $\{a,b\}$ depends on the set of agents $\{c,d\}$ to see to their goals $\{g_1,g_2\}$ or $\{g_3\}$. A social dependence network can be defined as follows:

Definition 2 (Social dependence networks) *A social dependence network is a tuple* $\langle A, G, dep, \geq\rangle$ *where:*

- A *is a set of agents and* G *is a set of goals.*
- *dep* $: 2^A \times 2^A \rightarrow 2^{2^G}$ *is a function that relates with each pair of sets of agents all the sets of goals on which the first depends on the second.*
- $\geq$: $A \rightarrow 2^G \times 2^G$ *is for each agent a total pre-order on goals which occur in his dependencies:* $G_1 \geq (a)\ G_2$ *implies that* $\exists B, C \subseteq A$ *such that* $a \in B$ *and* $G_1, G_2 \in depend(B,C)$.

We show how to model example 1 as a social dependence network where agents are related with each others by a set of dependencies created by power to achieve goals.

Example 2 *Consider the following social dependence network* $DP = \langle A, G, dep, \geq\rangle$*:*

1. *Agents* $A = \{E,S,M,P,K\}$ *and Goals* $G = \{g_1,g_2,g_3,g_4\}$
2. *dep*$(\{M\},\{S\}) = \{\{g_1\}\}$*: agent* M *depends on agent* S *to achieve the goal* g_1.
   dep$(\{K\},\{E\}) = \{\{g_1\}\}$*: agent* K *depends on agent* E *to achieve the goal* g_1.
   dep$(\{K,S,M,E\},\{P\}) = \{\{g_2\}\}$*: agents* $\{K,S,M,E\}$ *depend on agent* P *to achieve the goal* g_2.
   dep$(\{K,S\},\{P\}) = \{\{g_4\}\}$*: agents* $\{K,S\}$ *depend on agent* P *to achieve the goal* g_4.
   dep$(\{P\},\{M,K,S,E\}) = \{\{g_3\}\}$*: agent* P *depends on agents* $\{M,K,S,E\}$ *to achieve the goal* g_3.
3. *Agents K, M and S have the following pre-order on goals:* $\{g_1\} > (E)\ \{g_2\} \succ (P)\ \{g_4\}$ *and* $\{g_1\} \succ (S)\ \{g_2\}$ *and* $\{g_1\} \succ (P)\ \{g_4\}$.

3.2 Agent view

The passage from the concept of power to the social dependence networks can be explained using the concept of α-ability, as said by [3]. In fact, the definition of a social dependence network based on the abilities of agents and goals can be done using the notion of power as the so called α-ability, that is the capability of a group of agents to assure a state of affairs, independently from what the other agents do. This concept, coming from the classical game theory [13], does not consider the presence of useless agents in the system, so it is necessary to require that all the agents of the system play a profitable role in the achievement of the set of goals. In general, a dependence concerns the possibility of a group of agents to satisfy goals of agents, with the condition that in the group all members should be useful

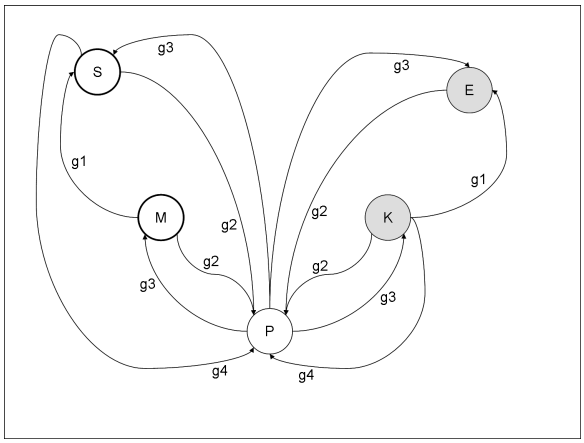

Figure 1. Social Dependence Network of Example 2

to the fulfillment of goals. After the definition of the relationships among the agents of the Multiagent system, the next step to perform is the modeling of the two levels of reality that emerge in a system of Ambient Intelligence such as by our scenario.
The first level of reality is the one that describes beliefs, goals and skills of the agent, the real ones. Skills or abilities of a group of agents (or of a single agent) play a relevant role as regards the power to achieve goals but power does not consist only of the group's abilities to achieve affects. There should also be at least one agent that desires those effects. In our model, skills can be represented as beliefs common to every agent, about the environment. Another component that has to be mentioned are the rules, also called effect rules in [3]. The environment of a multiagent system can be described by a pair of states, the initial one and the next one. A rule is conditional both on the initial state and on the actions performed by the agents. The link between rules and power is given by the concept of goal that contains rules and is contained in the definition of power. For example, in our scenario student S1 can have the goal to communicate with student S3 and student S6 can have the belief that the questions done by the other students to the teacher are useless. This level of reality can be called Agent view and can be defined as follows:

Definition 3 (Agent view) $\langle A, F, B, G, X,$ *beliefs*$: A \rightarrow 2^B$ *goals*$: A \rightarrow 2^G$, *skills*$: A \rightarrow 2^X, R : 2^X \rightarrow 2^G\rangle$ *consists of a set of agents* A*, a set of facts* F*, a set of beliefs* $B \subset F$*, a set of goals* $G \subset F$*, a set of actions* X*, a function beliefs that relates with each agent the set of its beliefs, a function goals that relates with each agent the set of goals it is interested in, a function skills that describes the actions each agent can perform, and a set of rules* R *that relate sets of actions with the sets of goals they see to.*

Example 3 shows, continuing by previous examples, how is constituted the agent view both from the point of view of sets and from the point of view of functions.

Example 3

- *Agents* $A = \{E,S,M,P,K\}$ *and Goals* $G = \{g_1,g_2,g_3,g_4\}$.
- $B = \{b_1,b_2,b_3\}$ *where* b_1*: answer the questions is useless,* b_2*: feedback is useful,* b_3*: communicate with schoolfriend is funny.*
- $X = \{x_1,x_2,x_3\}$ *where* x_1*: communicate with school friends,* x_2*: answer to the questions,* x_3*: require feedback and* x_4*: distribute material*

- $goals(M) = \{g_1, g_2\}$, $goals(K) = \{g_1, g_2, g_4\}$, $goals(P) = \{g_3\}$, $goals(S) = \{g_1, g_2, g_4\}$, $goals(E) = \{g_1, g_2\}$
- $beliefs(M) = \{b_3\}$, $beliefs(K) = \{b_3\}$, $beliefs(P) = \{b_2\}$, $beliefs(S) = \{b_1, b_3\}$, $beliefs(E) = \{b_1, b_3\}$
- $skills(M) = \{x_1, x_3\}$, $skills(K) = \{x_1, x_2\}$, $skills(P) = \{x_2, x_3, x_4\}$, $skills(S) = \{x_1, x_2, x_4\}$, $skills(E) = \{x_1, x_2, x_4\}$
- $rules(\{x_1\}) = \{g_2\}$, $rules(\{x_2\}) = \{g_4\}$, $rules(\{x_3\}) = \{g_3\}$, $rules(\{x_4\}) = \{g_1\}$

4 INSTITUTIONAL MECHANISM

The addition of a technological device changes in a relevant way the relationships among agents, giving a different aspect also to different roles composing an institution. For example, in our scenario we can recognize three different roles: the role of the teacher that has more power as regards the other roles, the one of the representative of a group that is not a common student because he has the power to communicate to the teacher and, finally, the role of the common student. In a multiagent perspective, roles are instances to be adjoined to the agents which play the role and they can be called also social roles. Obligations and permissions are a fundamental feature of normative positions of roles but, in general, we need also powers to specify normative or institutional positions. For more details, see [4].
The second level of reality is the one that describes public beliefs and goals of the role played by an agent and represents the institutional level. For example, taking again our scenario, student S3 can have the public goal to answer to the questions of the students instead of the teacher, so in spite of his private beliefs, he has in his public ones the utility of the answering to these questions. All the other students expect that Student S3 will conform to his role otherwise he will be sanctioned or even enforced. At this level becomes important, as previously said, the role of the agents because to some roles are associated more powers than to other ones. The role of the teacher, for example, has the power to change beliefs and goals of other roles, changing the institutional reality. Social institutions are entities which exist thanks to the collective acceptance of the public believes and goals and the rules regulating them. A role can not do any institutional action without the consent of the social entity (the system in which agents are). The reason is that social entities are not material ones and depend just on the collective acceptance. This level, called Institutional view, can be defined as follows:

Definition 4 (Institutional view)
$IV = \langle RL, IF, RB, RG, IX, beliefs\colon RL \rightarrow 2^{RB}, goals\colon RL \rightarrow 2^{RG}, skills\colon RL \rightarrow 2^{X \cup IX}, IR : 2^{X \cup IX} \times 2^{RB} \rightarrow 2^{IF}, roles : RL \rightarrow A\rangle$ *consists of a set of role instances RL, a set of institutional facts IF, a set of public beliefs attributed to roles $RB \subset F \cup IF$, a set of public goals attributed to roles $RG \subset F \cup IF$, a set of institutional actions IX, a function $ibeliefs$ that relates with each role the set of its public beliefs, a function $igoals$ that relates with each role to the set of public goals it is committed to, a function $iskills$ that describes the institutional actions each role can perform, and a set of institutional rules IR that relates sets of institutional actions, sets of facts and institutional facts with the sets of institutional facts they see to. A function $roles$ assigns a role to its player in A.*

Example 4

- *Agents* $A = \{E, S, M, P, K\}$
- $RL = \{Te, Re, St\}$ *where role Te is the role associated to the teacher, the role Re is the one associated to the representative of a group of students and the role St is the one associated to a common student.*
- $RB = \{rb_1, rb_2\}$ *where rb_1: possibility to send messages to the other school friends, rb_2: answering to the questions put to the teacher is useful.*
- $RG = \{rg_1, rg_2, rg_3, rg_4, rg_5, rg_6\}$ *where rg_1: answer to the questions sent to the teacher, rg_2: give next turn for asking a question, rg_3: sent the questions to the representative, rg_4: give feedbacks to the teacher, rg_5: give authorizations to download the material sent by the teacher, rg_6: send messages to other school friends.*
- $IX = \{ix_a, ix_b, ix_c, ix_d, ix_e, ix_f, ix_g\}$ *where ix_a: authorize to download the material, ix_b: stop the flow of messages among students, ix_c: put tasks in public goals and common points in public beliefs of every student, ix_d: give a absence note to the teacher, ix_e: give the turn to the next group asking questions according to the order of reservations, ix_f: delete public beliefs incompatible with public goals, ix_g: set the substitute of the teacher.*
- $IF = \{if_a, if_b, if_c, if_d\}$ *where if_a: turn as substitute of the teacher agent E, if_b: permission to download the material, if_c: possibility exchange messages with shoolfriends, if_d: turn of group with as representative agent S.*
- IR: *irules* $(\{ix_a\}) = \{if_b\}$, *irules* $(\{ix_b\}) = not\{if_c\}$, *irules* $(\{ix_g\}) = \{if_a\}$, *irules* $(\{ix_e\}) = \{if_d\}$.
- *IV:*
  - $igoals(Te) = \{rg_2\}$, $igoals(Re) = \{rg1, rg_4, rg_5, rg_6\}$, $igoals(St) = \{rg_3, rg_4, rg_6\}$.
  - $ibeliefs(Te) = \{rb_2\}$, $ibeliefs(Re) = \{rb_1, rb_2\}$, $ibeliefs(St) = \{rb_1\}$.
  - $iskills(Te) = \{ix_b, ix_c, ix_e, ix_f, ix_g\}$, $iskills(Re) = \{ix_a\}$, $iskills(St) = \{ix_d\}$.
  - $roles(Te) = \{P\}$, $roles(Re) = \{S, E\}$, $roles(St) = \{K, M\}$.

Example 4 shows the institutional view applied to the scenario. In this framework each participant is assigned with a set of public beliefs and goals, describing what he can do (e.g., authorize to download material) and should do (e.g., give feedback to the teacher). The agent that represents the teacher has the function of facilitator and so he has the aim to give the turn to the next student that desires to put a question and eventually to give the task to answer to these questions to a student. Agents with the role of representative can have the institutional goal to manage the questions that the members of its group want to send to the teacher while agents with the role of common students can perform the institutional action that send a message to a school friend on the other side of the classroom. The two levels, the public level and the private one have to be related together. To pass from the Agent view to the Institutional one we need a function that takes the private beliefs and goals of the agent and returns the public ones. The difference between the two sets of beliefs is not trivial, because there can be beliefs that remain from the passage from the private set to the public one, beliefs that disappear from the private set to the public one and, finally, beliefs that are present only in the public set and not in the private one. The same considerations can be done for goals. The difference between the private level and the public one is the existence of power. An agent can have the power to delete or add new goals and beliefs in the public sets of another agent such as the case in which the teacher stops the flow of messages and this action in our model is represented by a deletion of goals (the goal to send a message to other students) from the public set of goals

of students. The separation of the sets of public goals and beliefs has the aim to avoid contradictions between what the agent believes and what it has to perform (its goals) [1]. Our scenario allows to enforce the behavior of the agents in the institution, for example, by blocking them from making statements which contradict common beliefs, or by performing (virtual) actions which are not allowed (e.g., taking a turn in the wrong situation). We have to argue that these examples illustrate that our two level model is compact and understandable and succeeds in modeling all the facets that rise from an environment of Ambient Intelligence.

5 RELATED WORK

The formal model can be extended with the obligations, as done by Boella and van der Torre [5]. In this work, to model obligations they introduce a set of norms, associated with each norm the set of agents that has to fulfill it and what happens when it is not fulfilled. In particular, they relate norms to goals in the following two ways. First, each norm is associated to a set of goals. Achieving these normative goals means that the norm has been fulfilled; not achieving these goals means that the norm is violated. They assume that every normative goal can be achieved by the group, that means that the group has the power to achieve it. The second point is that each norm is associated to another set of goals which will not be achieved if the norm is violated, this is the sanction associated to the norm. We assume that the group of agents does not have the power to achieve these goals, otherwise they would avoid the sanction. An interesting approach to the application of the notion of institution to multiagent systems is defined in [16]. Electronic Institutions (EIs) provide the virtual analogue of human organizations in which agents, playing different organizational roles, interact to accomplish individual and organizational goals. As in human societies, it seems necessary the need of regulatory structures establishing what agents are permitted and forbidden to do. EIs introduce sets of artificial constraints that articulate and coordinate interactions among agents. In this approach, roles are defined as patterns of behavior and are divided into institutional roles (those enacted to achieve and guarantee institutional rules) and non-institutional roles (those requested to conform to institutional rules). Another approach to EIs is given by [6]. In this approach they propose the use of 3D Virtual Worlds to include humans into software systems with a normative regulation of interactions. Their methodology has two independent phases: the specification of the institutional rules and the design of the 3D Interaction environment. The normative part can be seen as defining which actions require an institutional verification assuming that any other action is allowed. Inside the 3D Interaction Space, an institution is represented as a building where the participants are represented as avatars. Norms determine the consequences of user actions that are modeled as commitments and these commitments may restrict future activities of the users. In the last two works, unlike us, the methodology is applied to an practical approach without a formal definition of the concept of institution and a description of its dynamics while they are similar to our one in the establishment of a different level of reality related to the institution.

6 CONCLUSIONS AND FUTURE WORK

We have observed as the possibility of interaction increases in a scenario of Ambient Intelligence thanks to the technological devices. We have presented the problem using as basis an example of scenario from Ambient Intelligence that describes a lesson done with the help of the device of a pocket pc.

We have defined an approach based on the Multiagent model, using social dependence networks, with the aim to describe the reality of an environment of Ambient Intelligence. The concept of power is used to model the reality of our scenario and the dependencies that the power sets in the system are represented as social networks, using the methodology of dependence networks.

To model the use of a technological device in Ambient Intelligence as a new level of reality, we have based our framework on the concept of institution with the aim to give a compact and realistic model of the reality. We have formalized the concept of institution, relating it with the one of power and we have applied this result to social dependence networks, obtaining institutional social networks.

Presently we are working on the definition of a dynamic model of the institutional view to represent the changes in the dependencies with the application of institutional actions. A first step in this direction seems to be in the dynamic social networks [8] that aim to bring together traditional social network theory and multiagent systems and contain multiple types of nodes and multiple types of connections among them with the feature to be larger dynamic. We are also defining measures on social dependence networks and their variations with the previously cited dynamics.

REFERENCES

[1] G. Boella, R. Damiano, J. Hulstijn, and L. van der Torre, 'A common ontology of agent communication languages', *Applied Ontology*, **2**, (2007).

[2] G. Boella, L. Sauro, and L. van der Torre, 'Social viewpoints on multiagent systems', *Proceedings of AAMAS'04*, (2004).

[3] G. Boella, L. Sauro, and L. van der Torre, 'From social power to social importance', *Web Intelligence and Agent Systems*, (2007).

[4] G. Boella and L. van der Torre, 'The ontological properties of social roles in multi-agent systems: Definitional dependence, powers and roles playing roles', *Artificial Intelligence and Law Journal*, (2007).

[5] G. Boella and L. van der Torre, 'Power in norm negotiation', *Proceedings of KES-AMSTA'07*, (2007).

[6] A. Bogdanovych, M. Esteva, S. Simoff, C. Sierra, and H. Berger, 'A methodology for developing multiagent systems as 3d electronic institutions', *Proceedings of AOSE@AAMAS'07*, (2007).

[7] P. Caire, S. Villata, L. van der Torre, and G. Boella, 'Conviviality masks in role-based institutions multi-agent teleconferencing in virtual worlds', *Proceedings of AAMAS'08*, (2008).

[8] K. M. Carley, 'Dynamic network analysis', *Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers*, 133–145, (2003).

[9] C. Castelfranchi, 'The micro-macro constitution of power', *Protosociology*, **18**, 208–269, (2003).

[10] H. Chalupsky, Y. Gil, C. K. Knoblock, K. Lerman, D. Oh, D. V. Pynadth, T. A. Russ, and M. Tambe, 'Electric elves: Applying agent technology to support human organizations', *Proceedigs of AAAI*, (2001).

[11] T. Erickson and W. A. Kellogg, 'Social translucence: an approach to designing systems that support social processes.', *ACM Trans. Comput.-Hum. Interact.*, **7**(1), 59–83, (2000).

[12] J. Masthoff, W. W. Vasconcelos, C. Aitken, and F. S. Correa da Silva, 'Agent-based group modelling for ambient intelligence', *Proceedings of AISB'07*, (2007).

[13] B. Peleg, 'Effectivity functions, game forms, games, and rights', *Social choice theory*, **15**, 67–80, (1998).

[14] P. Remagnino and G. L. Foresti, 'Ambient intelligence: A new multidisciplinary paradigm', *Systems, Man and Cybernetics*, **35**, 1–6, (2005).

[15] J. S. Sichman and R. Conte, 'Multi-agent dependence by dependence graphs', in *AAMAS'02*, pp. 483–490, (2002).

[16] C. Sierra, J. A. Rodriguez-Aguilar, P. Noriega, J. L. Arcos, and M. Esteva, 'Engineering multi-agent systems as electronic institutions', *European Journal for the Informatics Professional*, (2004).

[17] M. Wooldridge, 'An introduction to multi-agent systems', *The journal of policy, regulation and strategy for telecommunications*, **7**, 33–51, (2005).

Smarts Agents and Smarts Environments: a Predictive Approach to Replanning

Alfredo Garro[1] and **Sergio Greco**[2] and **Fabio Palopoli**[3]

Abstract. There is an increasing interest in the design and development of smart environments as they can be used to implement valuable applications aiming at improving the quality of life. However, another important issue, which has not been yet fully investigated, concerns the design and development of agents able to effectively and efficiently act in the new and emerging smart environments characterized by an increasing complexity and openness. This paper proposes a novel approach for dynamic (re)planning for agents which come into and would act in such kind of smart environments. The flexibility of the approach makes it exploitable also in the design of the replanning capabilities of smart environments both conceived as single rational entities that as a multi agent systems.

1 INTRODUCTION

A Smart Environment can be defined as "a region of the real world that is extensively equipped with sensors, actuators and computing components" [23]; it can sense what occurs within itself and its surroundings and adapts its actions accordingly to achieve its specific goals (e.g. to make inhabitants' lives more comfortable [4]). There is an increasing interest in the development of smart environments as they can be used to implement valuable applications aiming at improving the quality of life [5]. However, suitable paradigms and technologies are required for tackling the development of smart environments due to their ambitious goals [4]. Moreover, an important feature which makes more difficult their development is their *openness*: the entities which *inhabit* the environment are heterogenous, not known *a priori* and may change across time.

Several research efforts deal with the design and development of smart environments [1, 7, 22]. However, another important issue, that has not been yet fully investigated, concerns with the design and development of autonomous (artificial) entities (agents) able to effectively and efficiently act in such kind of open environments [2]. In fact, as a smart environment is endowed with goal-directed and autonomous behaviors [5], autonomous entities, which come into and would act in such an environment have to be able to properly interact with the environment and to constantly adapt their behavior to the behavior of the environment itself. Even if a smart environment provides several advanced services which can be used by these entities for achieving their goals in a more effective manner, the goal-directed and autonomous behavior of the environment makes it a complex and dynamic environment in which to act so requiring to these entities more *smart* capabilities. In particular, a key capability is planning. To support this statement, it can be useful to envisage a scenario concerning an autonomous entity (agent) which come into an open smart environment and is engaged in executing a plan for reaching a goal. Due to the complex and dynamic nature of the environment, during the plan execution it may occur an event which does not let the agent to continue the execution of the plan so it becomes necessary to build a new plan for keeping the goal reachable. Typically, during the replanning phase the agent reacts to the occurred event by applying reactive rules. After a certain amount of time the new plan is ready for the execution so the agent can switch from the reactive behavior to the execution of the new plan. However, mainly due to the complexity and the autonomous behavior of the environment, the state of the environment at the switching time can be very different from the state in which the event has occurred. If this possible evolution of the environment is not accurately considered in building the new plan, the computed plan could not be executable.

This paper proposes a novel approach for dynamic (re)planning which is based on a hybrid agent architecture and is specifically conceived for agents acting in smart environments. In particular, a prediction module is used to predict the environment state in which the agent will switch from the reactive behavior to the execution of the new plan. In addition, as at the switching time the conditions in which the agent should start the execution of the new plan could not be exactly the predicted one, it is computed and executed a "pre-plan" able to bring the agent in the conditions in which it is possible to start the execution of the new plan.

Although there are several approaches for dynamic (re)planning [12, 14, 24, 26, 28], the proposed predictive approach differs from the others in the way that the deliberation is evaluated with respect to the future state in which the deliberation output (i.e. the plan) will be executed. As far as the authors are concerned, there are few similar approaches reported in the literature [3, 11].

Moreover, although there are several studies and proposals which aim at providing smart environments with planning capabilities [8, 15, 16, 17, 18, 20, 21, 25], the problem concerning the planning capabilities of (artificial) autonomous entities which come into and would act in smart environments has been not explicitly and deeply investigated. Furthermore, it is worth noting that the flexibility of the proposed approach make it exploitable also in the design of the (re)planning capabilities of a smart environment both conceived as a single rational entity [5] that as a multi agent system [22]. In the former case the smart environment generates and executes plans to reach its own goals. Again, due to the possible events which could occur during the execution of these plans, it

[1] Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, Italy, email: garro@unical.it
[2] Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, Italy, email: greco@unical.it
[3] Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, Italy and Exeura S.r.L., email: f.palopoli@exeura.com

can be necessary to replan to keep the goals reachable, so dealing with the problems highlighted in the above described scenario. In the second case, the autonomous agents, which compose the smart environment and cooperate for achieving the environment goals, should be provided with suitable planning and replanning capabilities as they have to deal with several potentially unexpected events which must be properly managed.
This paper is organized as follows: in Section 2 the proposed replanning approach is presented and contextualized for smart agents which act in smart environments; Section 3 reports an example application concerning an Agent which acts in a (smart) office; eventually, in Section 4 conclusions are drawn and future works delineated.

2 A PREDICTIVE APPROACH TO REPLANNING

In this section, a new approach for dynamic replanning based on a novel agent architecture ($\mathcal{HPA}$ - Hypersphere and Prediction based Architecture) is presented. In particular, this approach has been conceived for dealing with the scenario discussed in Section 1 concerning an autonomous entity (hereafter $\mathcal{HPA}$ Agent) which come into an open smart environment and is engaged in executing a plan for reaching a goal.
The $\mathcal{HPA}$ Agent is supposed to be benevolent [10] and to act not in contrast with the smart environment rules and goals. The smart environment allows the $\mathcal{HPA}$ Agent to gather knowledge about its goals and behaviors in order to facilitate both actual and future interactions between the Agent and itself.

2.1 Basic Definitions

The considered scenario is as follows: the goal of an $\mathcal{HPA}$ Agent is to transform a state $s_0 \in S_0$ (the set of the Agent's initial states), into a state $s_g \in S_g$ (the set of the Agent's goal or final states). In order to get the state s_g, starting from the state s_0, it executes a plan p. During the plan execution the Agent's state changes due: (i) to the effects of the executed actions which strongly depends on the environment in which they are performed; (ii) to the events which concurrently happen in the environment in which the Agent is acting.
Given a set of possible states $S = \{s_1, ..., s_n\}$ and a set of actions $A = \{a_1, ..., a_m\}$, a plan $p =< a_i, ..., a_j >$ is a sequence of actions belonging to A. The set of actions define a transition function and the application of an action a_i to a state s_j, denoted by $a_i(s_j)$ gives a new state s_k. The notation $a_1.a_2(s_j)$ will be used in substitution of $a_2(a_1(s_j))$.

Definition 1 *An $\mathcal{HPA}$ Agent is a tuple $\mathcal{AG} = (S_0, S_g, S, P, R)$ where S denotes the set of its states, $S_0 \subseteq S$ denotes the set of its initial states, $S_g \subseteq S$ the set of its final states, P the set of plans, where for each $p \in P$ there is a $s_0 \in S_0$ such that $p(s_0) \in S_g$, and R denotes the set of reactive rules, where each $r \in R$ is a sequence of actions such that there is a $s \in (S - S_0)$ and $r(s) \in S$.*

An $\mathcal{HPA}$ Active Agent $\mathcal{AAG} = (s_0, s_g, s, p, q, R)$, derived from an $\mathcal{HPA}$ Agent $\mathcal{AG} = (S_0, S_g, S, P, R)$, is characterized by an initial state $s_0 \in S_0$, a final states $s_g \in S_g$, the current (or observed) state s, the already executed plan p and the plan to be executed q. □

An $\mathcal{HPA}$ Active Agent $\mathcal{AAG} = (s_0, s_g, s, p, q, R)$ is in a correct state if $q(s) = s_g$; otherwise, it is in a wrong state if $q(s) \neq s_g$.

Definition 2 *An influential unexpected event e is an action, not necessarily belonging to A, which is executed before q such that $e.q(s) \neq s_g$.* □

Thus, when an Active Agent $\mathcal{AAG} = (s_0, s_g, s, p, q, R)$, evolves into an Agent $\mathcal{AAG} = (s_0, s_g, s', p, q, R)$, where s' is a wrong state, it is supposed that an influential unexpected event e, modifying the correct state s of the Agent into the wrong state s' (i.e. $e(s) = s'$), has occurred. Moreover, if after the execution of an action a, belonging to A, the Agent evolves in a wrong state $s' \neq a(s)$, it is assumed that there has been an influential unexpected event e such that $a.e(s) = s'$.

Definition 3 *Given an $\mathcal{HPA}$ Active Agent $\mathcal{AAG} = (s_0, s_g, s, p, q, R)$, an uninfluential unexpected event e is an action, not necessarily belonging to A, which is executed before q such that $e.q(s) = s_g$.* □

Thus, when an Active Agent $\mathcal{AAG} = (s_0, s_g, s, p, q, R)$, evolves into an agent $\mathcal{AAG} = (s_0, s_g, s', p, q, R)$, such that $q(s') = s_g$, it is supposed that an uninfluential unexpected event e, modifying the state s of the agent into another correct state s' (i.e. $e(s) = s'$), has occurred. Moreover, if after the execution of an action a, belonging to A, the Agent evolves in a correct state $s' \neq a(s)$, it is assumed that there has been an uninfluential unexpected event e such that $a.e(s) = s'$.

The sequence of actions which compose a reactive rule is different from a plan (i.e. $P \cap R = \oslash$) as it is not conceived for reaching a goal state starting from an initial state but for reacting to an influential unexpected event during the replanning process as described in Section 2.2.

2.2 The Replanning Process

The main activities which characterize the $\mathcal{HPA}$ Agent behavior are showed in the Figure 1. Each activity is executed by a dedicated module as described in details in Section 2.3.

The $\mathcal{HPA}$ Agent behavior is as follows. At time t_0, in the state s_0, the $\mathcal{HPA}$ Agent starts to interact with the smart environment by executing a plan p_0, such that $p_0(s_0) = s_g$, in order to get the goal state s_g. At time t_j an influential unexpected event happens (see Definition 2). Thus, the Agent evolves in a state s' from which, by keeping executing p_0, it is not possible to get the goal state s_g.
The $\mathcal{HPA}$ Agent percepts the event, analyzes it and recognizes that an influential unexpected event happened, so it stops executing the current plan (which is became ineffective) and, concurrently, it both adopts a reactive behavior and starts replanning.
The *reactive behavior* consists in the selection and execution of reactive rules from the set R of the Agent (see Definition 1). More in detail R can be conceived as a table in which reactive rules are selected by using the current state as an index. In other words, from this time, until no switch to any new plan is performed, from an external observers point of view, the $\mathcal{HPA}$ Agent is characterized by a purely reactive behavior.
Concurrently to the execution of the reactive behavior, the $\mathcal{HPA}$ Agent starts replanning by performing the following activities:

1. *Time bounds definition*. The $\mathcal{HPA}$ Agent defines the computational and execution time bounds for the other activities which compose the $\mathcal{HPA}$ replanning process (see Section 2.2.1 for a discussion).

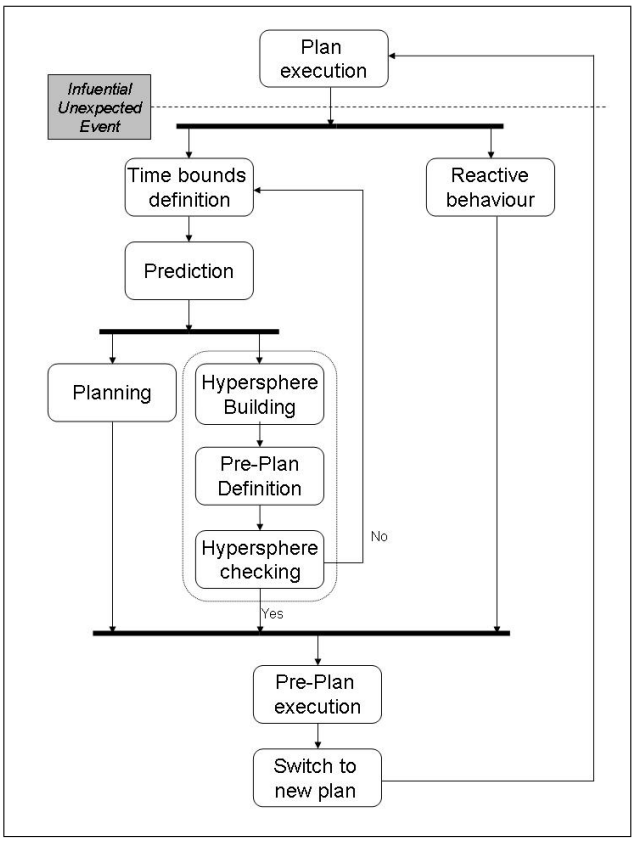

Figure 1. The $\mathcal{HPA}$ approach to replanning

2. *Prediction*. The $\mathcal{HPA}$ Agent computes the future state s_e in which it should be at the time t_e at which the new plan execution should start. The prediction must be computed in ΔT_{prd} seconds (t_e and ΔT_{prd} have been set during step 1).
3. *Planning*. The $\mathcal{HPA}$ Agent computes the new plan which has as initial state s_e and final state s_g. The new plan must be computed in ΔT_{pld} as set during step 1.
4. *Pre-planning, Hypersphere Definition & Checking*. This activity, which is executed in parallel with *Planning*, is required because, due to the dynamics and complexity of the environment, there is no guarantee that the Agent state will be exactly equivalent to s_e at time t_e. The idea is to compute a set of states in the *neighborhood* of s_e from which it is possible to reach s_e, at latest at time t_e, by executing a set of actions (pre-plan). Specifically, the *Pre-planning, Hypersphere Definition & Checking* activity consists of the following consecutive steps:

   (a) *Hypersphere Building*. The $\mathcal{HPA}$ Agent computes the set of states (*Approximation Hypersphere*) from which it is possible to reach the state s_e in a certain amount of time (ΔT_{ppe}) by executing a pre-plan. The Hypersphere must be computed in ΔT_{hsd} defined during step 1.

   (b) *Pre-plan Definition*. The $\mathcal{HPA}$ Agent defines the set of actions to be performed before executing the plan in order to reach the state s_e from a state belonging to the *Approximation Hypersphere*. The pre-plan must be computed in the time interval ΔT_{ppd} defined during step 1.

   (c) *Hypersphere Checking*. The $\mathcal{HPA}$ Agent tests at time $t_e - \Delta T_{ppe}$ if it is in a state which belongs to the *Approximation Hypersphere*.

The subsequent behavior of the $\mathcal{HPA}$ Agent depends on the outcome of the *Hypersphere Checking* step as follows:

- *case 1:* at time $t_e - \Delta T_{ppe}$ the $\mathcal{HPA}$ Agent has reached a state which belongs to the *Approximation Hypersphere*. In this case it stops executing the reactive behavior and starts the execution of the computed pre-plan. After the execution of the pre-plan, the $\mathcal{HPA}$ Agent, which is now in s_e (if no other influential unexpected event has occurred in the meantime), can start executing the new plan.
- *case 2:* at time $t_e - \Delta T_{ppe}$ the $\mathcal{HPA}$ Agent has reached a state which does not belong to the *Approximation Hypersphere*. In this case it does not stop executing the reactive behavior and restarts the execution of the replanning process from the *Time bounds definition* activity which consequently remodulate the previously computed time bounds.

It is worth noting that if during any phase of the described replanning process another influential unexpected event occurs it causes the immediate ending of the process and the starting of a new one for managing the occurred event (the $\mathcal{HPA}$ Agent starts reacting to the new event and replans again). However, this case is risk free, also if a large number of influential unexpected events occur, since the Agent simply resets the replanning process while continuing the execution of the reactive behavior. In the following subsections further details on some relevant aspects of the proposed approach to replanning are discussed.

2.2.1 Time bounds definition

The *time bounds definition* activity defines the time bounds for the execution of the above mentioned activities. In particular, it allows us to define the following time bounds:

- ΔT_{prd}: maximal time to compute the prediction output;
- ΔT_{hsd}: maximal time to define the Approximation Hypersphere;
- ΔT_{ppd}: maximal time for defining the pre-plan;
- ΔT_{pld}: maximal time for defining the new plan;
- ΔT_{ppe}: maximal time for executing the pre-plan.

It is worth recalling that the temporal execution sequence is as follows: first, the *Prediction* activity is executed; next, the Agent executes in parallel the *Pre-planning, Hypersphere Definition & Checking* activity and the *Planning* activity. The *Hypersphere Checking* execution time is negligible. From these considerations it is possible to compute the execution time t_e, the time at which the new plan can be executed:

$$t_e \geq t_{bde} + \Delta T_{ppe} + max(\Delta T_{hsd} + \Delta T_{ppd}, \Delta T_{pld})$$

where t_{bde} is the time at which the execution of the *time bounds definition* activity ends.
The *time bounds* are properly tuned on the basis of the results of the previous replanning processes in which the Agent has been engaged. Two typical situations which cause the adjustment of the *time bounds* are the following:

- at $t_e - \Delta T_{ppe}$ the Agent is in a state which does not belong to the computed *Approximation Hypersphere*, and thus it is necessary to

restart replanning. Probably, in this case, the dynamic of the environment are increasing so requiring a more fast replanning ($t_e - t_j$ decreases and thus the *time bounds* need to be consequently remodulated);
- a large number of influential unexpected events occur while the agent is replanning, so causing frequent resets of the replanning process and forcing the agent executing a reactive behavior. In this case can be necessary to remodulate the *time bounds* to reduce the duration of the replanning process so trying to execute a new plan before the next influential unexpected event occurs.

These tuning capabilities make the $\mathcal{HPA}$ approach suitable for highly dynamic environment and able to scale in situations characterized by a large number of influential unexpected events, possibly due to a large number of agents acting in the same environment.

2.2.2 *Prediction*

After the definition of the time bounds, the prediction module starts; its output is a prediction of the state s_e at the time t_e at which the agent should start to execute the new plan.

The inputs of the prediction procedure are:

- a description of the influential unexpected event;
- a description of the state at time t_j - the time at which the agent has observed the influential unexpected event;
- the set R of reactive rules of the Agent;
- information about the (future) behavior of the smart environment (both in function and not in function of the occurrence of the influential unexpected event).

On the basis of these inputs, the prediction procedure calculates s_e. It is worth noting that the smartness of the environment makes it possible to gather useful information about its future behavior which can increase the accuracy of the prediction respect to what happen in *classic* environments where the Agent can only make assumptions on the environment evolution.

2.2.3 *Hypersphere Building*

Pre-plan and Hypersphere Definition are introduced in the $\mathcal{HPA}$ approach because there is no guarantee that the agent state will be exactly equivalent to s_e at time t_e. In fact, the output of the prediction procedure might be imprecise, for several reasons:

- the computational time available to the prediction activity is limited, and this fact influences the prediction degree of accuracy;
- an *influential* unexpected events might have occurred after t_j - it is worth recalling that:
  - the prediction is based on the state at time t_j;
  - the occurrence of an *influential* unexpected event causes the restart of the replanning process;
- the reaction time may vary, and the response of the environment to a reaction could be slightly different from the prediction module's output;
- in the same way, even if the agent has acquired information on the (future) behavior of smart environment, it is very difficult to predict the exact effects of the actions that the smart environment will perform after t_j.

Thus, it is very difficult to predict exactly the agent state at time t_e. However, it is easier to predict that at time $t_e - \Delta T_{ppe}$ the Agent could be in the *neighborhood* of the state s_e. If it is the case, the agent can perform a sequence of corrective actions (a pre-plan) in order to get the state s_e. The distance of the neighbor states of s_e must be properly limited so that it is possible to perform correcting actions (pre-plan) for getting state s_e no later than time t_e. More formally:

Definition 4 *An Approximation Hypersphere of a state s_e and time bound ΔT_{ppe} is a set of states S such that for each state $s_i \in S$ there is a pre-plan which is able to transform the state s_i in a state s_e in a time less than or equal to ΔT_{ppe}.* □

It is worth noting that in the $\mathcal{HPA}$ approach each state s_e has associated a unique pre-plan (defined on the basis of the properties of s_e) and that each neighbor state s_i is such that the application of the pre-plan brings the Agent to state s_e before time t_e. Observe that it is possible to generalize by considering more alternative neighbors and pre-plans which could be defined in parallel, but this is outwith the aims of this paper.

2.3 The Hypersphere and Prediction based Architecture

The replanning process described in Section 2.2 is supported by the Hypersphere and Prediction based Architecture ($\mathcal{HPA}$) for replanning shown in figure 2.

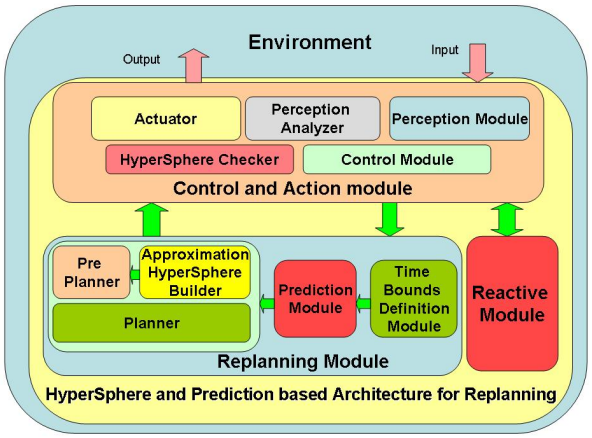

Figure 2. HPA Architecture for Replanning.

The $\mathcal{HPA}$ architecture consists of three macro modules: a *Control and Action Module*, a *Reactive Module* and a *Replanning Module*.

The *Control and Action Module* consists of the following submodules:

- *Perception Module*, which percepts the environment through a set of sensors;
- *Perception Analyzer*, which processes the perceptions received by the *Perception Module*;
- *Actuator*, which executes actions received both from the *Reactive* and the *Replanning Module*;

- *Control Module*, which controls the other submodules and determines the behavior of the agent by coordinating the *Reactive Module* and the *Replanning Module*. In particular, if the *Input Analyzer* identifies an influential unexpected events, the *Control Module*:
  - asks to the smart environment information about the future behavior of the environment (which can also depend on the occurred event);
  - stops the execution of the current plan;
  - activates both the *Reactive* and the *Replanning Module*.

  During the reaction of the Agent (see Section 2.2) the *Control Module*:
  - requires the *Actuator* the execution of reactive actions received from the *Reactive Module*;
  - stops the current replanning process and restarts the *Reactive* and *Replanning Module* in case of a new influential unexpected event;
  - at time $t_e - \Delta T_{ppe}$ requires the *Hypersphere Checker* to check if the Agent state belongs to the *Approximation Hypersphere*, and on the basis of the result requires the *Actuator* to execute the computed pre-plan or it restarts the *Replanning Module* and requires the *Actuator* to continue executing the reactive actions (see Section 2.2);

The *Replanning Module* is responsible of the replanning process described in Section 2.2. It consists of the following submodules:

- *Time Bounds Definition Module*, which defines time bounds for the replanning process as described in Section 2.2.1;
- *Prediction Module*, which computes the future state in which the Agent should be when the new plan execution should start (see Section 2.2.2);
- *Planner*, which computes the (new) plan for reaching the goal state starting from the future state predicted by the *Prediction Module*. It is possible to adopt several algorithms for the planning task [9, 13];
- *Approximation Hypersphere Builder*, computes the set of states in the neighborhood of the state predicted by the *Prediction Module* from which it is possible to reach the predicted state by executing a pre-plan in a given amount of time (see Section 2.2.3);
- *Pre-Planner*, which computes the pre-plan for reaching the state predicted by the *Prediction Module* starting from a state which belongs to the *Approximation Hypersphere*. Again, it is possible to adopt several algorithms for the pre-planning task [9, 13];

The *Reactive Module* is responsible of the reaction to an expected influential event; in particular, it selects a sequence of actions and sends them to the *Actuator* for their execution.

3 AN APPLICATION EXAMPLE

The application example envisages a scenario similar to that presented in [19, 27]. It consists of a mobile robot (Worker) which acts in a warehouse to properly store goods on the basis of their typologies. The warehouse is a smart environment. In particular, the Worker's goal is to move a certain quantity of goods from a point A to a point B. The map of the environment, its expected behavior, and a plan p_0 to move from A to B are available. The internal architecture of the Worker is the $\mathcal{HPA}$ architecture presented in Section 2.3. At some time t_j, the Worker's *Perception Analyzer* detects that an unexpected obstacle lies on its way to B (e.g. a shelf is fallen down). Assume that it categorizes the obstacle as an influential unexpected event. At this point, the Worker's *Control Module* starts two concurrent processes:

- it stops the execution of the plan p_0 and starts the execution of the reactive behavior;
- it starts replanning.

For simplicity, the only trivial reactive rule for obstacle avoidance could be as follows:

RULE1:
if (there is an obstacle on your trajectory) **then**
    send an alert message to the environment;
    turn 90 degrees to your left;
    repeat
        move ahead at a speed of 2 Km/h;
    until (no obstacle is detected)
end if

The input to the replanning process is composed by relevant data about: *(i)* the environment and the Worker state at time t_j; *(ii)* the expected behavior of the environment (e.g. the environment may unlook some doors along some alternative paths to keep point B reachable). Those data also includes a description of the obstacle and its location, the location of the Worker, its speed, and so on.
After the *Time bounds definition*, the next main step of replanning is the *Prediction*. The *Prediction Module* of the Worker, on the basis of the above described information, predicts the (future) state at time t_e (the time at which the new plan's execution must start). For instance, if the parameter $t_e - t_j = 10$ seconds the execution of the new plan (that does not exist yet) must start 10 seconds after the time t_j in which the influential unexpected event occurred. The *Prediction Module* accesses the warehouse map, the set R of Worker's reactive rules, and the information regarding the expected behavior of the environment. Firstly, it matches its knowledge of the influential unexpected event - *there is an obstacle on the Worker's way* - against the set of reactive rules, deducing that the Worker should choose RULE1. Then it predicts, also taking into account the fact that the environment may unlook some doors along some alternative paths to keep point B reachable, that the Worker at time t_e should be in a point of the warehouse of coordinates (x_e, y_e). At this point, if the environment was static, nothing more is necessary: the *Planning Module* knows that at time t_e the Worker will be in a well determined point of the warehouse so it is sufficient to return a new plan (the initial state and the goal state (unchanged) are known). But the environment is complex and dynamic, and the amount of time to predict the Worker's future location is bounded, thus the prediction has a fixed level of accuracy. There is no guarantee that the Worker really will be at (x_e, y_e) at time t_e. However, in the $\mathcal{HPA}$ approach, it is sufficient that the state reached by the Worker is in the neighborhood of the predicted state at time $t_e - \Delta T_{ppe}$; more specifically, it is sufficient that the Worker's location is inside a "circle" whose centre is (x_e, y_e): the circle is the *Approximation Hypersphere*. According to the *Approximation Hypersphere* definition, the circle's size has to be large enough to allow the robot to move from every point inside the circle to the point (x_e, y_e) in an amount of time equal (or less than) ΔT_{ppe}.
If, at time $t_e - \Delta T_{ppe}$, the current state belongs to the *Approximation Hypersphere* (the Worker is in the circle), the Worker stops the execution of the sequence of reactive actions and initiates the pre-plan's execution. If the Worker reaches in useful time the point (x_e, y_e), it starts the execution of the new plan.

If anything goes wrong, the Worker would continue working according to the reactive behavior and it starts to replan.

4 CONCLUSIONS AND FUTURE WORK

In recent years enormous interest has grown around the design and development of smart environments. This paper has presented a novel approach to (re)planning that provides effective planning capabilities to agents that come into and act in smart environments. The main novelty is that, while standard approaches compute the plans with respect to the current state, in the $\mathcal{HPA}$ approach the computation of the plan is based on the (future) state in which the plan execution will start. In order to effectively replan, the agent exploits information received by the smart environment about its current and future behavior. Due to its flexibility, the $\mathcal{HPA}$ approach can be easily adapted in order to provide smart environments themselves with effective (re)planning capabilities in the case in which they are modelled and implemented as (multi)agent systems.
Efforts are currently underway to: *(i)* further investigate the interrelationships among the different *time bounds* which drive the replanning process in order to define specific strategies for their (optimal) setting; *(ii)* extensively experiment the $\mathcal{HPA}$ approach in real testbeds and more complex scenarios.

REFERENCES

[1] J. C. Augusto and C D. Nugent (Editors). *Designing Smart Homes: The Role of Artificial Intelligence*. Lecture Notes in Artificial Intelligence, vol. 4008. Springer, Berlin, 2006.

[2] M. Broxvall, M. Gritti, A. Saffiotti, B. S. Seo, and Y. J. Cho, "PEIS Ecology: Integrating Robots into Smart Environments", *In Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, Florida, May 2006.

[3] A. Coddington and M. Luck, "A Motivation-based Planning and Execution Framework", *In International Journal on Artificial Intelligence Tools*, vol. 13(1), pp. 5-25, 2004.

[4] D.J. Cook and S. Das, "Smart Environments: Technology, Protocols and Applications", Wiley-Interscience, 2004.

[5] D. J. Cook, M. Youngblood, E. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, "MavHome: An Agent-Based Smart Home", *In proceedings of the Conference on Pervasive Computing*, 2003.

[6] D. J. Cook, M. Youngblood, and S. K. Das, "A Multi-agent Approach to Controlling a Smart Environment". *Designing Smart Homes* 165-182, 2006.

[7] S. K. Das, and D. J. Cook, "Designing Smart Environments: A Paradigm Based on Learning and Prediction". *Mobile, Wireless, and Sensor Networks*. John Wiley & Sons, Inc., 2006.

[8] S. K. Das, D. J. Cook, A. Battacharya, E. O. Heierman, and L. Tze-Yun, "The role of prediction algorithms in the MavHome smart home architecture", *IEEE Wireless Communications*, vol. 9, pp. 77-84, 2002.

[9] M. Ghallab, D. Nau, and P. Traverso, "Automated Planning: Theory & Practice", Morgan Kaufmann, 2004.

[10] N. Jennings and M. Wooldridge, "Agent Theories, Architectures, and Languages: A Survey", *Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag, Berlin, 1995.

[11] M. Hadad, S. Kraus, Y. Gal, and R. Lin, "Temporal reasoning for a collaborative planning agent in a dynamic environment", *Annals of Mathematics and Artificial Intelligence*, vol. 37(4), pp. 331-379, 2003.

[12] X. H. Hu and J. W. Dang, "Hybrid Agent Model to Design Real-time Distributed Supervising and Control Systems", *In Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision*, pp. 1-5, Singapore, 2006.

[13] M. LaValle, "Planning Algorithms", Cambridge University Press, 2006.

[14] K. H. Low, W. K. Leow, and M. H. Ang, "Hybrid Mobile Robot Architecture with Integrated Planning and Control", *In Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, 2002.

[15] R. Levinson, "The planning and execution assistant and trainer (PEAT)", *Journal of Head Trauma Rehabilitation*, vol. 12, pp. 85-91, 1997.

[16] A. Mihailidis, G. Fernie, and J. C. Barbenel, "The use of artificial intelligence in the design of an intelligent cognitive orthotic for people with dementia", *Assistive Technology*, vol. 13, pp. 23-39, 2001.

[17] A. Mihailidis, B. Carmichael, and J. Boger, "The use of computer vision in an intelligent environment to support aging-in-place, safety, and independence in the home", *IEEE Transactions on Information Technology in Biomedicine*, vol. 8, pp. 238-247, 2004.

[18] A. Mihailidis, B. Carmichael, J. Boger, and G. Fernie, "An intelligent environment to support aging-in-place, safety, and independence of older adults with dementia", *In Proceedings of the 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications*, Seattle, Washington, 2003.

[19] F. Mizoguchi, H. Hiraishi, and H. Nishiyama, "Human-robot collaboration in the smart office environment," *In Proceedings of the 1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Atlanta, USA, 1999.

[20] M. C. Mozer, "Lessons from an adaptive house," *in Smart environments: Technologies, protocols, and applications*, D. Cook and R. Das, Eds. Hoboken, NJ: Wiley & Sons, pp. 273-294, 2005.

[21] M. C. Mozer, "An intelligent environment must be adaptive", *IEEE Intelligent Systems and Their Applications*, vol. 14, pp. 11-13, 1999.

[22] A. Omicini, A. Ricci, and G. Vizzari, "Building Smart Environments as Agent Workspaces", *In Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, Paris, France. June 18-20, 2007.

[23] P. Nixon, S. Dobson, and G. Lacey, "Managing Smart Environments", *Workshop on Software Engineering for Wearable and Pervasive Computing*, Limerick, Ireland, 2000.

[24] T. Patkos, A. Bikakis, G. Antoniou, M. Papadopouli, and D. Plexousakis, "Distributed AI for Ambient Intelligence: Issues and Approaches", *Ambient Intelligence*, Lecture Notes in Computer Science, Springer Berlin, vol. 4794, pp. 159-176, 2007.

[25] M. E. Pollack, "Intelligent technology for an aging population", *AI Magazine*, vol. 26, pp. 9-24, 2005.

[26] H. Sarmadi, "Fuzzy Hybrid DeliberativelReactive Paradigm", *Formal Approaches to Agent-Based Systems*, Lecture Notes in Computer Science, vol. 3228, pp. 281-286, 2004.

[27] D. Weyns, K. Schelfthout, T. Holvoet, and T. Lefever, "Decentralized control of E'GV transportation systems", *In Proceedings of AAMAS Industrial Applications*, pp. 67-74, 2005.

[28] D. E. Wilkins, K. L. Myers, J. D. Lowrance, and L. P. Wesley, "Planning and Reacting in Uncertain and Dynamic Environments", *Journal of Experimental and Theoretical AI*, vol. 7, no. 1, pp. 197-227, 1995.

Semantic Web Services for Intelligent Responsive Environments

Crhistian Alberto Noriega Guerra and **Flavio Soares Correa da Silva** [1]

Abstract. In this work we propose a model for intelligent responsive environments based on semantic web services, in which services are located in distributed devices and the interaction between devices and the environment takes into account the dynamics of contextual information and the availability of services. In the proposed model, all functionalities in the environment are provided as structures of services, which are automatically discovered and executed to support the users in specific tasks. We illustrate our ideas through a prototype implementation in our university library, in which location aware information is provided to users to help them locate items of their interest.

1 Introduction

Intelligent responsive environments are physical environments which can sense the necessity of certain services and provide them to users at the appropriate time and location. For example, a student in a library could check from his PDA what books are available, based on the courses in which he is enrolled in that academic term. Once a book is selected, the environment could inform the student, through his PDA, about related books that can also be relevant for his studies. The environment could also determine the location of the PDA - and hence of the student - and provide directions for the student to find the desired books, as a sort of "electronic compass". Finally, the environment could search for other users who could be close to the student and share similar interests with him, and inform the student about these users.

In order to provide users with automated services that enable these functionalities, an environment must identify the required services based on the goals and the context of the users. The required infrastructure for an intelligent responsive environment includes ubiquitous computing, ubiquitous communication, adaptive user interfaces, and the capability to collect and interpret contextual information [1, 14, 32].

In the present work we propose a model for intelligent responsive environments based on semantic web services [4]. We employ the *Web Services Modeling Ontology - WSMO*, which belongs to the *Web Services Management Framework - WSMF* [13] for the semantic specification of services in the intelligent responsive environment. The WSMO is based on four elements: (1) goals, (2) services, (3) mediators and (4) ontologies. These elements relate to each other through the WSMF, which is based on a particular flavor of description logics called *F-Logic* [21].

[1] University of Sao Paulo, Brazil, email: {cnoriega,fcs}@ime.usp.br. This work has been partially supported by CNPq and Microsoft Research. The authors wish to thank the anonymous reviewers for their suggestions and corrections to the paper.

Our model is based on the notion of *task driven computing* [39], in which the users of an information system must focus on their tasks of interest, instead of the means to accomplish these tasks.

We introduce the utilization of an *Assumption based Truth Maintenance System - ATMS* [9] to manage the provision of services, based on the relations between preconditions and effects for each service. If the context of a user changes, the updated available services provide new alternatives to reach the user goals. The ATMS identifies minimal sets of preconditions that suffice to obtain a given effect, based on the descriptions of the available services.

Any functionality in an environment based on our model is provided through web services running on mobile or fixed devices. Our prototypical implementation employs two existing middleware systems for ubiquitous computing: (1) The WSAmI middleware [18] provides a protocol for the discovery and location of web services; and (2) The MoCA middleware [30] provides the means to collect contextual information about mobile devices in the environment.

In section 2 we discuss the importance of context awareness for intelligent responsive environments, and how this concept can be implemented using *task-driven computing*. In section 3 we briefly review the notion of assumption-based truth maintenance systems, and how they can be employed to help manage the services in intelligent responsive environments. In section 4 we discuss semantic web services and their relevance to intelligent responsive environments. In section 5 we introduce our model for intelligent responsive environments, based on semantic web services. In section 6 we present our prototypical implementation at a university library. Finally, in section 7 we present some conclusions and proposed future work.

2 Context-awareness and Task-driven computing

An intelligent responsive environment must be sensitive to the context of its users. In the present work we adopt as a definition of *context* what is found in [10]: context is any information that can be employed to characterize the situation of an entity. An *entity* can be a person, a location or any physical object that is relevant in the interactions between the environment and its users. In our work, we consider that people can be identified by digital devices they carry with them, such as intelligent responsive phones and PDAs.

A *context sensitive* system uses contextual information to provide relevant information and services to users [8, 39]. Once the information about the status of a device has been captured, the environment must process it and match it with a contextual model to infer contextual information. Many existing systems for intelligent responsive environments adopt a layered approach, in which sensed information is used to feed a contextual model, and services are provisioned to users based on this model [34]. The overall quality of services in these systems is bounded by the quality of the contextual model.

We have implemented context awareness in our system through *task-driven computing*. When a user moves to a different location in the environment, services and available information must be updated and reconfigured accordingly. *Task-driven computing* [39] is a technique to design systems in such way that users interact with high-level descriptions of tasks they wish to see accomplished, instead of dealing with lower-level internal services provided by the system. Using task-driven computing, we have *tasks* as an intermediate level of abstraction between the goals and intentions of users and the available resources in the environment that change depending on context.

Tasks and sessions are connected through four architectural concepts: (1) Task management - management of relations between tasks and contexts; (2) Encapsulation of services - appropriate abstraction of services to be offered to the accomplishment of tasks; (3) Management of sessions and services - configuration, discovery and coordination of services; and (4) Environment management - discovery and management of information related to devices and services related to each context.

Several lower level infrastructures provide the means to implement these concepts, e.g. Jini, Universal Plug and Play - UPnP and Service Location Protocol - SLP. These infrastructures, however, address mainly the management of individual services. Task-driven computing is implemented on top of these infrastructures, to allow the management of composed services.

Other systems provide the means to implement service composition, e.g. Enterprise Java Beans, CORBA and Web Services. These systems, however, do not address *what* services should be combined. This is indeed a complex problem, presently studied by many research groups [39, 34, 28, 35].

3 Assumption-based Truth Maintenance Systems

The goal of an assumption-based truth maintenance system (ATMS) is to identify minimal sets of assumptions that can justify a conclusion. An ATMS can work in conjunction with a logical inference system, to help search a knowledge base and find useful connections between logical assertions. The details including implementation issues - about ATMS can be found in [9].

Essentially, an ATMS manages the following: (1) Premises - a premise is a justification - i.e. a logical assertion - that does not depend on other conditions to be true. Roughly speaking, it corresponds to a fact in a PROLOG program; (2) Assumptions - an assumption is a logical assertion whose own set of assumptions is empty; (3) Justifications - a justification is a Horn clause that implements a logical assertion; (4) Nogoods - a nogood is the negation of a premise; (5) Environments - an environment in an ATMS has a meaning different from the one we have used in this work. Essentially, an ATMS environment is a set of assumptions that justifies a conclusion; and (6) Labels - a label is a set of ATMS environments.

Given an arbitrary logical assertion, the ATMS makes use of a specific algorithm to search its set of justifications and find the label that corresponds to it. The label that is built by the ATMS algorithm is such that the environments are minimal, i.e. they correspond to minimal sets of assumptions that can be used as alternatives to provide logical support to the selected assertion.

We have borrowed these concepts to support the management of tasks in intelligent responsive environments. Essentially, we have built an implementation of the original ATMS, in which assumptions are directly connected to available services in the environment, and logical assertions are connected to tasks. When a task is presented to the environment, it uses the ATMS and the currently (i.e. at the present context) valid justifications to find the services it needs to accomplish that task. This is a necessary step for service composition, which must be complemented by the appropriate resources to orchestrate the identified services.

4 Semantic Web Services

A web service is the abstraction of a functionality that should be provided by a concrete agent [38]. A web service must be well defined and self contained. Operationally, a web service is specified as a computational resource which employs SOAP (*Simple Object Access Protocol*) packages over HTTP (*Hypertext Transfer Protocol*). The description of web services is made using WSDL (*Web Service Description Language*). It comprises the public methods and parameters available for remotely calling a service.

Web services have been extended with semantic descriptions. An extension of the OWL language [4] has been proposed specifically to encode ontologies for semantic web services. This extension has been called OWL-S (*Ontology Web Language for Services*) [26].

We have adopted the *Web Services Modeling Ontology* - WSMO [29] for the semantic description of the services available in the environment. This ontology has been built for the specification of semantic web services, to allow the discovery, execution, monitoring and composition of web services. We have used this ontology to encode the services provided by the environment and how they relate to each other. The WSMO is part of a larger framework, the WSMF (*Web Service Management Framework*) [13].

The WSMO defines *service capabilities* as 4-tuples of the form assumptions, pre-conditions, effects and post-conditions (*APEP*). Based on web service APEPs, a client can find the adequate service for its goal. Assumptions and effects work as constraints on the specification of the context of the environment. Pre-conditions and post-conditions correspond to inputs and outputs of the execution of a service. As an example, the assumption

```
?someStudent[hasDepartment hasValue "IME"]
          memberOf fex#StudenFenix
```

establishes that a service is provided only to students of the *IME* department, and the pre-condition

```
?someBook memberOf lib#Book
```

establishes that the service requires a `Book` as input.

The WSMO defines two visions for *service behavior*: (1) choreography - informs the clients how the web service works and (2) orchestration informs other web services about the behavior of the service.

The behavior of the service is characterized as a *finite state machine*, in which *transition rules* determine the consequences of specified pre-conditions. If the pre-conditions of a rule are fulfilled, then the consequences are achieved. The client stores the state of the execution of services. When a rule is used, its consequences are inserted into the state. A transition rule transition has the form:

```
if (?someStudent[hasLocation hasValue
       ?someLocation] memberOf fex#StudentFenix)
then
    add(?relatedbooks memberOf lib#RelatedBooks)
```

This rule is interpreted as: if the client state contains the student location then a list of relevant books can be found.

The WSMF defines an execution environment for semantic web services, which however until the date of the preparation of this paper had not been implemented.

Each transition rule relates to a concrete method of the WSDL interface. We have defined execution properties for each transition rule (method) of the web service. These properties are related to: (1) types of the results of the method, which can be *persistent* or *transient*; and (2) frequency of execution the method, which can be *once* or *many* times.

For example, location dependent services may need to be updated frequently and relate to the user instead of the device. Such services may be classified as *trasient-many* methods.

5 Proposed Model

Our proposed model for intelligent responsive environments is based on task-driven computing. It employs an ATMS to identify, given a task, the services whose composition can accomplish it. The actual communication and exchange of services between devices is based on semantic web services.

In this section we provide a brief overview of the model. A detailed account of our model, as well as of how to access the corresponding code of its implementation, can be found in [15].

5.1 Architecture

Our proposed architecture is based on three components: (1) Environment server - this component manages the relation between components and devices, as well as the discovery and execution of services. It is located on a fixed node in a computer network; (2) Service providers - these components are the components that, when executed, offer some service to the environment. Service providers can be fixed or mobile; and (3) Environment clients - these are executed in mobile devices.

These components are depicted in Figure 1. The client sends to the environment server updated contextual information, task definitions and knowledge base facts. The environment server discovers and executes appropriate services and updates the user knowledge base with contextual information of the device and results of service executions. Figure 2 shows the components of the environment server.

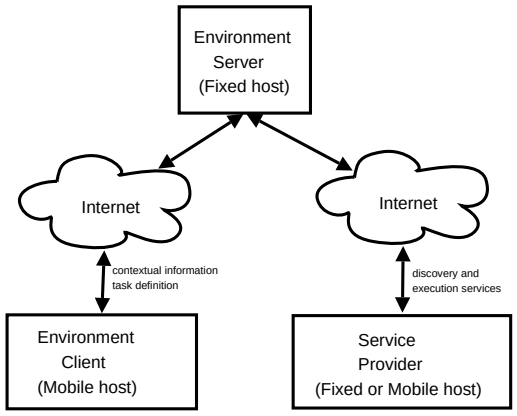

Figure 1. Basic components of the proposed architecture.

For each device in the environment are created an associated ATMS and knowledge base.

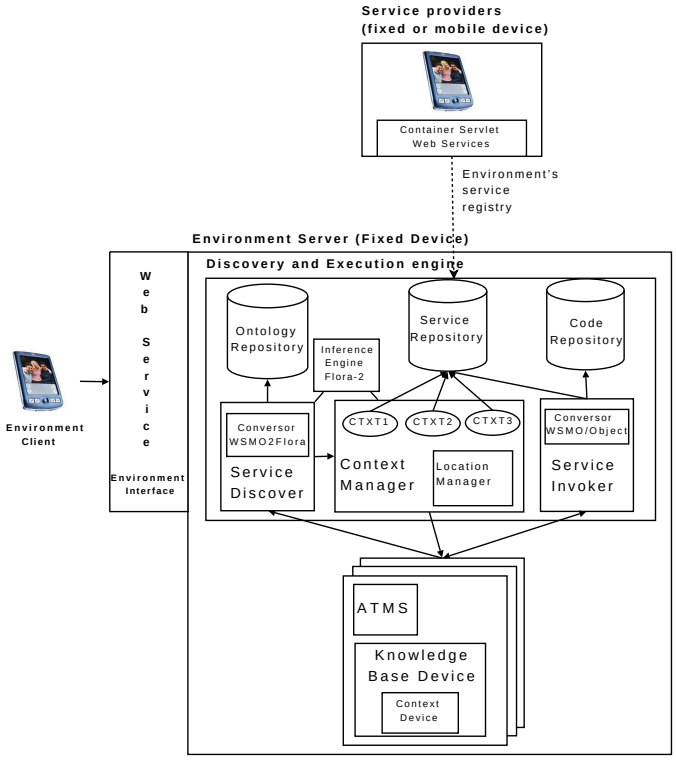

Figure 2. Components of the environment server.

The proposed architecture defines three repositories: (1) Services repository - stores the semantic specifications in WSMO of the services that are available in the environment. We have implemented access to this repository using resources available from WSAmI. The services discovery mechanism searches this repository to discover the available services; (2) Ontologies repository - stores all ontologies employed in descriptions of services in the environment. The required ontologies are retrieved from this repository to process inferences accordingly in order to build service compositions; (3) Code repository - stores Java classes dynamically generated from services, which are loaded in real time for the automatic execution of services.

A *context manager* performs the efficient management of services in the environment, based on declared connections between pairs context / service. Services that are not connected to any specific environment are generic services, considered to be available in all contexts. The composition of services is built based on inferences performed using F-logic, in order to discover implicitly available services. The context manager also monitors the context of a device and updates its set of available information accordingly. This is implemented using MoCA [30] to match locations with contexts, thus providing the required relation between location and context to the system.

The available information and services identified by the context manager feeds the *services discovery mechanism*, which then triggers the ATMS to relate services and information with potential effects that can come out of them. These effects determine what tasks can be accomplished.

Once the required services to accomplish a task are determined, they are invoked using a *Dynamic Invocation Interface* - DII for *Jax-RPC*. DII requires the following concrete pieces of information to be triggered: the service execution end-point, the name of the spe-

cific method to be triggered, and the required objects that function as parameters of the method.

The same methods that implement the invocation of services capture the results of the corresponding services and update the information available to each device accordingly.

Each device has a set of pieces of information that characterize its context, which includes information about location. This is called the *knowledge base* of the device. The knowledge bases are constantly updated by the context manager.

The sequence of steps for discovery and execution of services is as follows:

Registry: A device is registered in the environment. The environment builds a knowledge base, an ATMS and a context for the device.

Specification of a task: A user sends to the environment the description of a task. This can be done explicitly and manually by the user, or automatically by a device carried by the user.

Identification of available services: The context manager retrieves the set of available services.

Matching of task and services: Using the ATMS, minimal sets of services are identified in order to accomplish the desired task.

Execution of services: The appropriate services are invoked, and the results of their execution are stored in the appropriate knowledge bases. If the task cannot be accomplished, pro-active help is provided to the user, suggesting nearby locations where the necessary services can be found.

The context manager updates the user's knowledge base with contextual information, including the user location. The context manager also stores service availability information in context as pairs such as:

```
< { ?someStudent[hasDepartment hasValue "IME"]
              memberOf fex#StudentFenix },
  {Library} >
```

This pair, for example, defines that the `Library` service is available for students of the `IME` department. This association allows the reduction of the search space and the management of service availability in the evironment.

The implementation of this module makes use of `CIS` and `LIS` services of the MoCA middleware [30] for contextual and location dependent information.

Facts in the knowledge base are stored as pairs of WSMO expressions and Java objects. The expressions describe the Java objects, and the objects are reified as WSMO expressions. Consider, for example, the pair:

```
< (?someStudent[hasLocation hasValue
       ?someLocation] memberOf fex#StudentFenix),
   {Student@1232fasf, Student@3g3122} >
```

In this example we find the definition of which objects have an attribute associated to the `hasLocation` ontology property. This association allows the retrieval of the appropriate objects in a WSMO expression for the execution of services.

The WSMO expression describes the attributes which have the Java objects. For example, in the previous case, the expression indicates which the `Student` objects have the attribute `location`, for the property `hasLocation`.

In this mapping we use the convention for ontologies, which uses `has` notation for named concept attributes; and the convention for classes, which uses `get-set` notation to access class properties.

5.2 Service discovery mechanism

The process for service discovery is divided in two phases: (1) logical service discovery, which uses service capabilities to match services with the user goals; and (2) ATMS based services composition, in which rule transitions are propagated and an ATMS is used to identify valid environments that characterize minimal sets of preconditions for services.

The ATMS manages the context of the environment and the knowledge base manages the context of the user and the device.

Services and ontologies are rewritten as F-Logic [21] assertions. A service must satisfy the following condition:

$$\exists Ser \bullet O, Ser_{pos}, KB_{disp} \models_{F-Logic} G \wedge Ser_{hip}$$

where Ser is the service, O are the ontologies, Ser_{pos} are the service post-conditions, KB_{disp} in the device knowledge base of the user, G is the user goal and Ser_{hip} are the assumptions. The services that satisfy the goal are selected for the next phase.

The transition rules of the services described in the previous discovery phase are rewritten as ATMS justifications. An ATMS justification has the form:

$$< effect_i, \{\{condition_1, condition_2, ...\}, \{\ldots\}, \ldots\} >$$

where $effect_i$ and $condition_i$ are the effect and conditions of a transition rule. For example, in the following transition rule:

```
if(?someStudent[hasLocation hasValue
       ?someLocation] memberOf fex#StudentFenix)
   then
add(?relatedbooks memberOf lib#RelatedBooks)
```

the ATMS justification is:

```
< ?relatedbooks memberOf lib#RelatedBooks,
{ {?someStudent[hasLocation hasValue
   ?someLocation] memberOf fex#StudentFenix}},
{ ?someStudent[hasLocation hasValue
  ?someLocation] memberOf fex#StudentFenix}
    => ?relatedbooks memberOf lib#RelatedBooks >
```

This justification is propagated in the ATMS. The conditions of the transition rule are added to the ATMS as assumptions. An ATMS assumption has the form:

$$< cond_i, \{\{condition_i\}\} >$$

Justifications are propagated in the ATMS to identify appropriate ATMS environments.

When there is a change in the availability of services, the ATMS must update its sets of assumptions. Each environment contains the minimal and consistent assumptions required to accomplish a task that is relevant for a user.

The ATMS of a device is updated with the services that are available in a context and useful for the user task. Pre-conditions of transition rules of services that are no longer available are withdrawn from the ATMS. A service is not available if we have (1) communication problems, e.g. the end-point of the web service is not accessible or (2) the service does not satisfy the constraints of the context environment, e.g. when the user moves to an area where the service cannot be executed.

The knowledge base manages the context of the device, and the ATMS manages the context of the environment and the availability of services required for the user task.

When adequate ATMS environments cannot be found, new services are searched to be placed as assumptions in ATMS environments. In other words, assumptions become sub-goals and discovery is restarted to look for further assumptions that support these sub-goals.

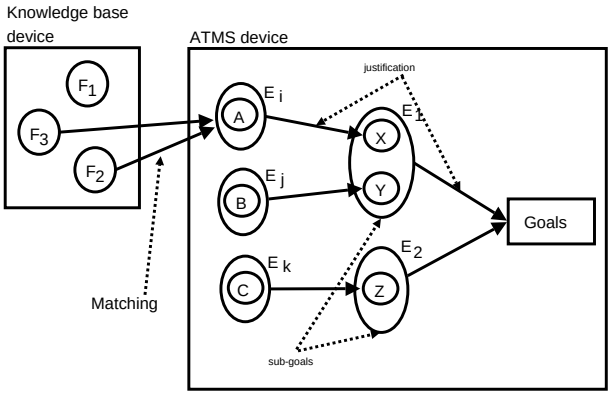

Figure 3. Matching ATMS and the knowledge base.

For example, in Figure 3, X and Y are the sub-goals and the environments E_i, E_j and E_k are the new environments found for this sub-goals. Thus, the previous ATMS state is:

```
< effect, {{X,Y},{Z}} >
```

Propagation of new justifications for the environments X, Y and Z produce the environments E_i, E_j and E_k. The new ATMS state is:

```
< effect, {{X,Y},{X,B},{Y,A},{A,B},{Z},{C}} >
```

Proactive support is provided to the user: when it is not possible to add new rule transitions to find an environment that can support the user task, recommendations are provided so that the user can move to a better location where perhaps the required conditions can be found. For example, if the environment has the assumption:

```
?somestudent[hasLocation hasValue "area1"]
     memberOf fex#StudentFenix
```

then, in order to satisfy this environment is recommended to the user to move to $area1$.

6 A Prototypical Implementation: The math library at the University of Sao Paulo (USP)

We have built an experiment to test our ideas at the Library in our Institute. In order to build this experiment, we have implemented an environment server on a high-end portable computer using J2EE and a generic client on a PDA using the Microsoft .NET platform.

The server has a fixed network node and an IP identifier. We have employed a JBoss applications server as a container of web services, and CORBA to manage the communication with the client.

In Figure 4 we show the main classes used in the implementation of the environment server:

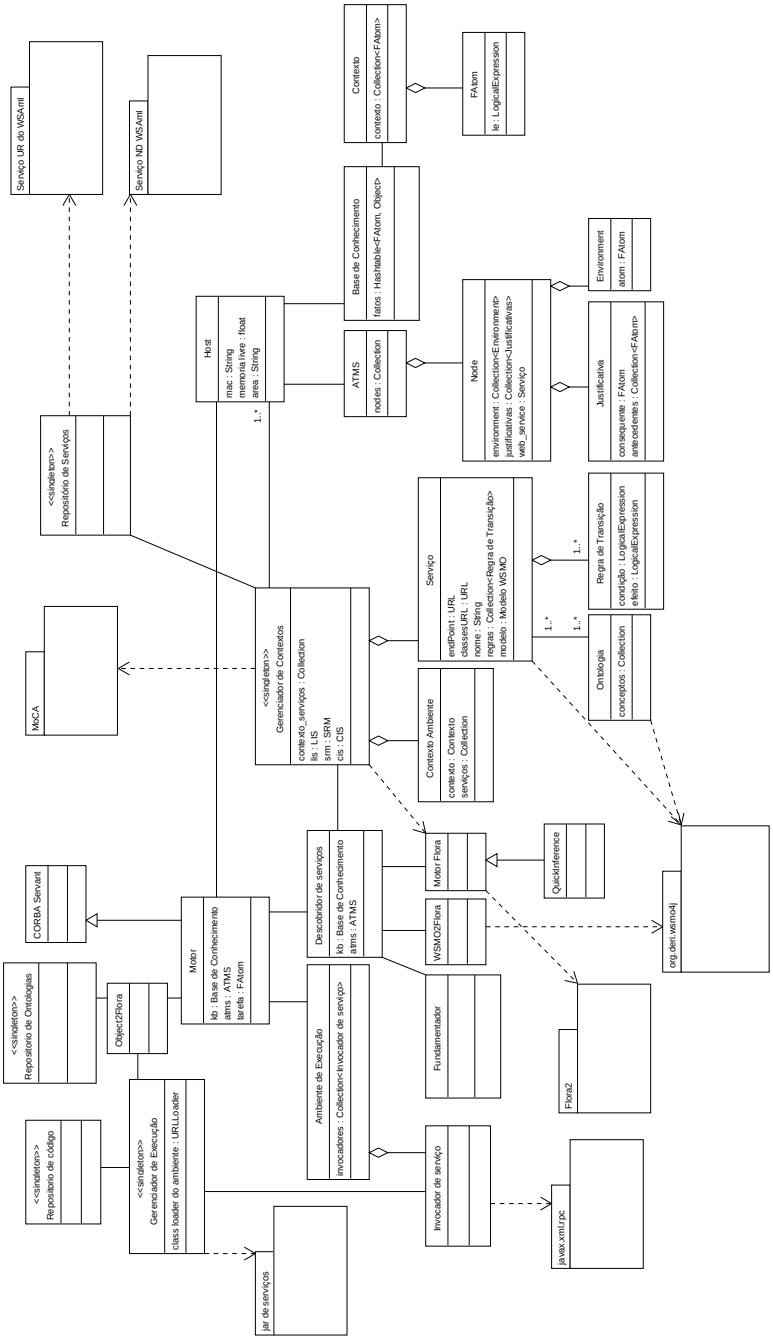

Figure 4. Main classes in the environment server.

Device engine: This class encapsulates the objects and methods required for the discovery and execution of services. Each instance of this class contains an ATMS and a knowledge base connected to a specific device. This class also implements CORBA objects to manage the connection between the server and the clients.

Host: Manages the information about a mobile device in the environment. The contextual information collected by MoCA is updated in this class.

Knowledge base: Manages the knowledge base of a device.

ATMS: Implements the ATMS corresponding to each device.

Service: Manages the information related to a service in the environment: execution end-point of the web service, an instance of the WSMO model that describes the service, the URL where the implementation of the service can be found, and a service unique identifier.

F-logic engine: Manages and performs inferences in F-Logic.

WSMO2F-logic: Translates WSMO service specifications into F-logic notation.

Services repository: This class is based on the services *UR* and *ND* found in WSAmI. It retrieves services registered in the WSAmI universal repository (*UR*) and searches for specific services that match with the service identifier *ND*, based on the semantic specifications of services declared in WSMO.

Context manager: Manages the relations between services and contexts.

Services discovery: Performs all required steps to identify sets of services given a task.

Service identifier: Connects each set of services given by the services discovery to the corresponding methods whose execution corresponds to those services.

Execution manager: Orchestrates the execution of the other classes.

Execution environment: Implements service invocation objects for each method to be executed.

Service invocation: Identifies the required parameters for the execution of methods that implement a service. Queries the knowledge base of a device to identify JAVA objects whose semantic description matches with the required parameters. When necessary, this class also updates the knowledge base.

Object2F-logic: Translates F-logic expressions into JAVA objects.

Access to the environment server is made through a web service which abstracts the relevant processes in the environment to the clients. The interaction between the clients and the environment server is implemented by the following functions:

`void register(HostData, ContextData)`**:** This function registers a device in the environment. It takes the MAC address of the device encapsulated in `HostData` and an initial context in `ContextData`. Contextual information is encoded as sentences such as `crhistian[hasNusp hasValue "5055668"] memberOf StudentFenix`.

`void setContextForHost(HostData, ContextData)`**:** This function updates the context of a device using the information contained in `ContextData`.

`boolean hasException(HostData, GoalData)`**:** This function starts up the process of discovery and execution of services. It takes the specification of a task in `GoalData` and returns `true` if the process is successful, or `false` if some exception is triggered, e.g. requiring pro-active behavior.

`WebServiceData[] getServiceForGoal(HostData)`**:** This function publishes the services and corresponding methods that were executed under normal conditions (i.e. when the previous function returns `true`).

`Exception getLastProActiven(HostData)`**:** This function publishes the services and methods that were executed under abnormal conditions (e.g. when pro-active behavior is required).

`Exception getLastMultipleObject(HostData)`**:** This function returns a list of potential objects related to a concept.

`PositionData getPosition(HostData)`**:** This function returns the location of the device.

`String[] getKB(HostData)`**:** This function returns the knowledge base of the device as a collection of WSMO expressions.

`void addFactToKB(HostData, String)`**:** This function adds a new entry to a knowledge base.

The client was implemented using C# and Microsoft Visual Studio .NET 2005. It runs on Windows CE 5.0 in an HP iPAQ Pocket PC. It is essentially a client for the web services provided by the environment server. Web services are the means to allow seamless interoperability between `.NET` and `J2EE`.

We have implemented a generic graphical user interface, so that users can interact with the environment using WSMO sentences. A user friendly interface should be built on top of this interface to make this system truly available to end users. In our specific prototype, the WSMO sentences represent books in the library, their bibliographical information and physical location.

We had to build a map of the library to configure the location service provided by MoCA. The library was organized in 104 areas.

We have built the following ontologies for this example:

`Library`**:** In this ontology we have built concepts related to books in a library, such as `Book`, `ListBook`, `BookTheme`, `BookPlace` and `RelatedBooks`.

`Fenix`**:** In this ontology we have built concepts related to academic records of university students[2]. It contains concepts such as `StudentFenix`, `TeacherFenix`, `CourseFenix` and `Deparment`.

`Ambient`**:** This ontology contains concepts related to the environment, such as `Device`, `User` and `Context`.

`Location`**:** This ontology contains concepts related to locations in the environment. The base concept is `Location`, with corresponding attributes *x, y* and *z*. It also contains the concept `Region` that refers to a composite area in the environment. For example, a book can be associated with a `Location` that belongs to a `Region`.

`People`**:** This ontology contains concepts that characterize the presence of people in the environment.

We have implemented the following services in our prototype:

`Fenix`**:** With this service the user can check the academic status of a student. The service provides the following information:

1. Relevant books according to the courses in which the student is enrolled.
2. List of courses in which the student is enrolled.
3. List of books that are available in this library and can be relevant to the student.

`Library`**:** This service encapsulates all functionalities provided by the library. It allows the following operations:

[2] The name *Fenix* relates to the information system used at the University of Sao Paulo for that purpose.

1. Enter a queue for the withdrawal of a book.
2. Find books that relate to a book of interest.
3. Find books that relate to a specific topic.
4. Find books that are physically close to the present location of the user.

`Location`**:** This service guides the user to find a book within the library. It takes a book ID and a student ID and location and returns directions to find the book.

All interactions between the user and the environment run through WSMO sentences expressing terms of the environment ontologies. A generic mobile client was implemented on a Pocket PC under Windows CE. This client implementation allows (1) to send goals definition to the environment server; (2) to add and remove fact of the knowledge base, including context information; and (3) to consult facts in the knowledge base and transient information.

Figure 5 shows the generic interface of the client application, and Figure 6 shows the interface for library environmet. It shows the directions presented to a user to find a selected book.

Figure 5. Generic client application running in environment device (Pocket PC).

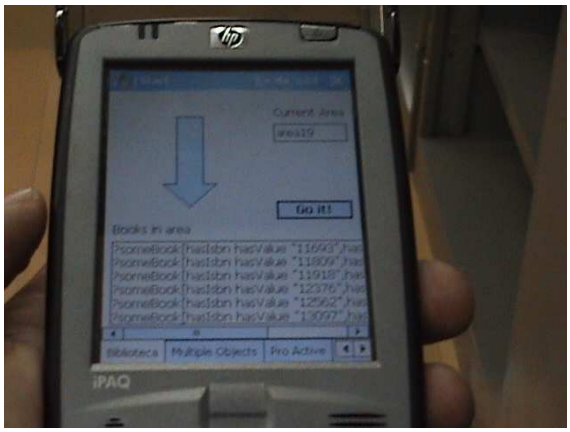

Figure 6. Library environment interface.

7 Conclusions and Further Work

In [12] some scenarios are presented in which intelligent responsive environments can be useful. These scenarios also pose challenges for research in ambient intelligence. We suggest three scenarios to show the potential of our proposed model:

1. *An environment that reacts to the presence of people:* In this scenario services are discovered according to the user location. For the user, it means that the environment reacts to her presence when she moves e.g. into the `"room1"` area.
   This scenario is based on a sensor-actuator model. Sensors are devices on the environment in charge of capturing contextual information and inform the actuators which handle physical changes in the environment. In our proposed model sensors and actuators are controlled through services.
   For example, consider a thermometer in area `room1` (sensor) and a heater in the same area (actuator). The thermometer can have the specification:

```
if(?user[hasLocation hasValue "room1"]
         memberOf User)
   then
add(?temperature memberOf Temperature)
```

   This transition rule defines that the sensor returns the temperature in area `room1`. The heater can have the specification:

```
if(?temperature[hasValueCelsius
                hasValue ?somevalue]
      and ?user memberOf User)
then
   add(?temperature
        memberOf EnvironmentTemperature)
```

   This transition rule defines that the heater returns the desired temperature for a given user. The implementation of the heater can contain further rules that specify what to do to move from `Temperature` to `EnvironmentTemperature`. The discovery mechanism allows sensors and actuators dynamically.
2. *A location aware recommendation system for a wide area:* In this scenario we consider a wireless network area such as a metropolitan area and a location system such as GPS or Placelab [23] for this type of environment.
   Services are associated to specified areas in the metropolitan area. When a user moves about, appropriate services are discovered and executed. For example, consider a recommendation system for consumer products. A user can add to his knowledge base the intention to buy some product, e.g. a refrigerator:

```
fridge[hasModel hasValue "LG"]
   memberOf Product.
preference[hasElements hasValue fridge]
   memberOf Preference.
```

   Environment services can be implemented for a shopping mall in the area under consideration, containing rules like e.g.:

```
if(?someprefer[hasElements hasValue
    ?someproduct] memberOf Preference
   and ?user[hasLocation hasValue ?somearea]
        memberOf User
)
   then
add(?someInfo memberOf ProductInformation)
```

   In this scenario, users can access their architectural components (knowledge base, ATMS, discovery and execution mechanisms) through the Internet without losing their states.

Additional service capabilities can be provided by adding semantic descriptions for those functionalities. For example, to buy selected products, service implements that functionality in a WSDL method associated to the next transition rule:

```
if(?someprefer[hasElements hasValue
      ?someproduct] memberOf Preference
   and
   ?creditcard[hasUser hasValue ?someuser,
               hasType hasValue ?someType]
            memberOf CreditCard
)
then
   add(?somestatus[hasStatus hasValue buyOK]
           memberOf StatusPayment)
```

3. *An environment capable of providing composite services:* In this scenario we can illustrate the implementation of architectural services with the purpose if accessing components (knowledge bases) of other devices in the environment, thus looking for alternative means to accomplish a task. Sophisticated planning and collaborative systems can be used to build compositions of services.

These three scenarios illustrate some possibilities in which intelligent responsive environments can be effectively employed to provide good services to users. Any implementation of an intelligent responsive environment based on our model requires the definition of: (1) a collection of ontologies, in which the semantic specification of services is built; (2) semantic definition of services in the environment and (3) service implementation with web services technologies.

In the case of a reactive environment, the main feature to be considered is the dynamic discovery and execution of services. In this scenario we can, for example, be interested in services that control the behavior of devices distributed across the environment, so that for example a printer, a digital display or an air cooling device is started when it senses the presence of a mobile device such as a PDA or a smartphone.

A location sensitive recommendation system for a wide area - e.g. a university campus or a whole urban area - requires that location information is provided by appropriate technologies - such as, for example, GPS. A user can inform the environment through his/her mobile device the interest in buying a specific product - say, a refrigerator. This information could be added to the device's knowledge base, which would then interact with different services and devices as the user moved about the environment. Interesting lower level implementation challenges become relevant in this scenario, such as the need to replicate the information and services provided by the environment server in geographically distributed workstations across the environment.

The composition of basic services can greatly enrich the capabilities of a intelligent responsive environment. In order to fully implement compositions, we would need to enrich our model with explicit composition capabilities, provided for example by automated planning systems. We envisage difficulties, however, to provide appropriate composition capabilities that perform well under the stringent real time requirements that characterize intelligent responsive environments and location aware systems in general.

All in all, we believe that our proposed model for intelligent responsive environments based on semantic web services and task-based computing has great potential in many application areas. The flexibility provided by web services and task-based computing is useful to support scalability and the seamless utilization of heterogeneous devices. The semantic specifications granted by semantic web services and semantic web technology is useful to design sophisticated interaction mechanisms. The utilization of ATMS to constrain the sets of relevant services to accomplish a given task has proven to be a useful tool to improve the computational efficiency of the model. Indeed, our experiments have indicated that this model can be implemented efficiently, to abide by the real time requirements that characterize intelligent responsive environments and location aware systems, at least for systems in which the composition of services does not require the utilization of highly sophisticated planning systems.

In the prototype system like the one desribed in this paper, it is very important to conduct user evaluation. This will reveal flaws in the system and should provide directions for future work. Currently, without any user evaluation data, it is difficult to assess how easy and useful the system is going to be in actual practice and what limitations it presents. We plan in the future to develop some user tests to assess and improve the system.

Further empirical analysis must also be done to evaluate how the system scales to large numbers of services and complex coordination tasks.

REFERENCES

[1] Alcaiz M. e Rey B., *New Technologies For Ambient Intelligence*, Ambient Intelligence, IOS Press 2005.

[2] Angele J. e Lausen G., *Ontologies in F-Logic*, Handbook on Ontologies in Information Systems. International Handbooks on Information Systems 29-50, Springer Verlag, 2004.

[3] Burbey I., *Ubiquitous Internet Computing*, WWW Beyond the Basics (`http://ei.cs.vt.edu/book/index.html`), Prentice Hall, 1998.

[4] Bussler C., Maedche A. e Fensel D., *A Conceptual Architecture for Semantic Web Enabled Web Services*, ACM Special Interest Group on Management of Data: Volume 31, Number 4, 2002.

[5] Bussler C., Maedche A. e Fensel D., *Web Services: Quo Vadis*, IEEE Intelligent Systems, 2003.

[6] Chen G. e Kotz D., *A Survey of Context-Aware Mobile Computing Research. Dartmouth Computer*, Science Technical Report TR2000-381. Department of Computer Science - Dartmouth College, 2000.

[7] Chen H., Finin T. e Joshi A., *An Intelligent Broker for Context-Aware Systems*, Adjunct Proceedings of Ubicomp 2003, USA, October, 2003.

[8] Cortese G., Lunghi M. e Davide F., *Context-Awareness for Physical Service Environments Ambient Intelligence*, IOS Press, 2005.

[9] de Kleer J., *An Assumption-Based TMS*, Artifificial Intelligence, Volume 28, Issue 2, 127-162, 1986.

[10] Dey A. K. e Abowd G. D., *Toward a better understanding of context and context-awareness*, GVU Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology, 1999.

[11] Davies J., Fensel D., e Van Harmelen F., *Towards the Semantic Web: Ontology-Driven Knowledge Management*, John Wiley Sons, 2003.

[12] Ducatel K., Bogdanowicz M., Scapolo F., Leijten J. e Burgelman J-C. *Scenarios for Ambient Intelligence in 2010*, *ISTAG*, Febrary 2001.

[13] Fensel D. e Bussler C., *The Web Service Modeling Framework WSMF*, Electronic Commerce: Research and Applications, 113-137, 2002.

[14] Gagglioli A., *Optimal Experience in Ambient Intelligence*, Ambient Intelligence, IOS Press 2005.

[15] Guerra N. A. Crhistian, *Um modelo para ambientes inteligentes baseado em servios web semnticos*, Tese de Mestrado IME-USP, Agosto 2007.

[16] Helal A., Mann W., Elzabadani H., King J., Kaddourah Y. e Jansen E., *Gator Tech Intelligent responsive House: A Programmable Pervasive Space*, IEEE Computer magazine, March 2005.

[17] Hansmann U., Merk L., Nicklous M. e Stober T., *Pervasing Computing*, Springer Second Edition, 2003.

[18] Issarny V., Sacchetti D., Tartanoglu F., Sailhan F., Chibout R., Levy N. e Talamona A. *Developing Ambient Intelligence Systems: A Solution based on Web Services*, In Journal of Automated Software Engineering. Vol 12. 2005.

[19] IST Advisory Group (ISTAG), *Ambient Intelligence: from Vision to Reality*, Ambient Intelligence, IOS Press 2005.

[20] Kifer M., Lara R., Polleres A. e Zhao C., *A Logical Framework for Web Services Discovery*, ICWS 2004.

[21] Kifer M. e Lausen G., *F-Logic: A Higher-Order Language for Reasoning about Objects, Inheritance, and Scheme*, International Conference on Management of Data archive ACM SIGMOD, 1989.

[22] Kleiner A., *Game AI: The Possible Bridge between Ambient and Artificial Intelligence*, *Ambient Intelligence*, IOS Press, 2005.

[23] LaMarca A., Chawathe Y., Consolvo S., Hightower J., Smith I., Scott J., Sohn T., Howard J., Hughes J., Potter F., Tabert J., Powledge P., Borriello G. e Schilit B., *Place Lab: Device Positioning Using Radio Beacons in the Wild*, In proceedings of Pervasive 2005, Munich, Germany.

[24] Masuoka R., Parsia B. e Labrou Y., *Task Computing - The Semantic Web meets Pervasive Computing*, `http://www.flacp.fujitsulabs.com`, Fujitsu Laboratories of America, Inc., 2004.

[25] OASIS - Organization for the Advancement of Structured Information Standards, *OASIS Reference Model for Service Oriented Architecture V1.0*, Official Committee Specification approved Aug 2, 2006.

[26] OWL Services Coalition, *OWL-S: Semantic Markup for Web Services*, `http://www.daml.org/services/owl-s/1.0/`.

[27] Paolicci M., Kawamura T., Payne T. R. e Sycara K., *Semantic Matching of Web Services Capabilities*, 2004.

[28] Project Oxygen, `http://www.oxygen.lcs.mit.edu/`, site acessado pela ltima vez 01/06/2007.

[29] Roman D., Lausen H. e Keller U., *Web Service Modeling Ontology (WSMO)*, `http://www.wsmo.org/TR/d2/v1.2/`, Working Draft D2v1.2, April 2005.

[30] Sacramento V., Endler M., Rubinsztejn H. K., Lima L.S., Gonalves K., do N ascimento F.N. e Bueno G., *MoCA: A Middleware for Develo-ping Collaborative Applications for Mobile Users*, ACM/IFIP/USENIX International Middleware Conference, Toronto, October, 2004.

[31] Schmidt A., *Interactive Context-Aware Systems Interacting with Ambient Intelligence*, Ambient Intelligence, IOS Press, 2005.

[32] Schilit N., *A System Architecture for Context-Aware Mobile Computing*, Phd Teses, Columbia University, 1995.

[33] Singh M. e Huhns M., *Service-Oriented Computing: Semantics, processes and agents*, John Wiley Sons, 2005.

[34] Sousa J. P. e Garlan D., *Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments*, Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture, August 25-31, 2002.

[35] Srivastava B. e Koehler J., *Web Service Composition: Current Solutions and Open Problems*, ICAPS 2003 Workshop on Planning for Web Services, 2003.

[36] Strang T. e Linnhoff-Popien C., *A Context Modeling Survey*, Workshop on Advanced Context Modeling, Reasoning and Management as part of UbiComp 2004.

[37] Xavier E. e Correa da Silva. F. S., *Expressing Systems Capabilities for Knowledge Coordination*, AAMAS'2002.

[38] W3C Working Group, *Web Services Architecture*, `http://www.w3.org/TR/ws-arch/`, W3C Working Group Note 11 February 2004.

[39] Wang Z. e Garlan D., *Task-Driven Computing*, Technical Report, School of Computer Science, Carnegie Mellon University, May 2000.

[40] WSMO Group, *The Web Service Modeling Language WSML*, `http://www.wsmo.org/TR/d16/d16.1/v0.2/`, WSML Final Draft 20 March 2005.

A Middleware for Smart Environments

Crhistian Alberto Noriega Guerra and **Flavio Soares Correa da Silva**[1]

Abstract. A *smart environment* is a context sensitive system based on ubiquitous computing, in which the environment interacts with its inhabitants through embedded dedicated devices. The design and construction of a smart environment requires the collaboration among several areas, such as (1) intelligent man-machine interfaces, (2) pervasive communications, (3) ambient intelligence, (4) scalable systems and (5) mobile computing. It can be very useful if a designer can abstract the required functionalities from these areas to design and build a smart environment. We propose a layer based middleware for mobile devices (mainly focusing on *smart phones*) for the intelligent interaction between devices. Each layer provides specific functionalities and serves as the ground upon which upper - i.e. more abstract - layers are built. We identify the following layers: (1) infrastructure and communications, (2) services and agents, (3) middleware services and (4) collective intelligence. We also propose a formalization of interactions for the specification of services in a smart environment, which is based on *ambient calculus* and also defines the relationships between layers.

1 Introduction

The interaction with computers in daily activities is useful only if computers can be helpful to the users, providing effective means to complete their tasks. These means can be presented as services, which must be consistent with the tasks at hand of users and also adequate considering the available devices for each user.

A smart environment provides to its inhabitants the computational services most adequate for each task at their hands. These services must be adapted according to each task, and can be provided cooperatively by a group of devices whose activities must then be coordinated.

The design of a smart environment can be based on a variety of approaches, as presented in [4], where the required functionalities to build a smart environment have been identified as (1) intelligent man-machine interfaces, (2) pervasive communications, (3) ambient intelligence, (4) scalable systems and (5) mobile computing.

Different approaches provide to the systems designed different middleware systems, frameworks and tools which have as one of their main goals the abstraction of the required functionalities to build a smart environment. For example, we have the Aura project in which task driven computing is the foundational model for interaction between the environment and its devices; the Oxygen project in which several tools have been defined to deal with ubiquitous computing; and projects like GAIA [19], GatorTech, EasyLiving and Smart Media Spaces, in which specific architectures have been proposed to support the required functionalities for smart environments.

[1] University of Sao Paulo, Brazil, email: {cnoriega,fcs}@ime.usp.br. This work has been partially supported by CNPq and Microsoft Research. The authors wish to thank the anonymous reviewers for their suggestions and corrections to the paper.

More recently, different approaches such as the Open Croquet project [10] have implemented a P2P middleware to build virtual worlds which, together with ubiquitous computing result on the union of real and virtual worlds [9].

Our proposed middleware is based on layers [17] to activate the interaction between devices. Each layer provides support to specific functionalities, as follows: (1) infrastructure and communications; (2) agents and services; (3) middleware services; and (4) collective intelligence.

Moreover, each layer abstracts the complexity of implementation of the corresponding concepts.

2 Goals and Motivation

Each individual nowadays carries with him/her a variety of mobile devices, such as smartphones, PDAs, notebooks, portable game consoles such as PlayStation Portable, etc. the mobility of these devices provides individuals with the capability to receive information anywhere and anytime. Following the tenets of ambient intelligence, the interactions with these devices and among devices must be intelligently managed so that it becomes transparent to individuals when performing their tasks.

Interactions must occur naturally and be unobtrusive to the activities of individuals. One way to attend this requirement is to specify interactions based on high-level and flexible descriptions of interactions and services, for example using ontologies to describe coordination protocols between autonomous agents [7].

In the following section we discuss the functionalities required to build a smart environment. In section 4 we describe the functionalities provided by each layer of our proposed middleware and provide examples of smart environments that could be founded on them. Finally, in section 5 we present some preliminary conclusions and proposed future work.

3 Required Functionalities for Smart Environments

Several projects have been introduced in the literature related to the design and construction of smart environments. In [4] we have studied some of these projects and detected the following required functionalities for a smart environment. The projects we have studied can be found in [22, 23, 24, 25, 26, 27, 28]:

1. A computational model to describe the interactions between the environment and its devices.
2. Architectures to support the proposed computational model.
3. A computational model to facilitate the representation of the physical environment as a computational environment.
4. Techniques and applications for ubiquitous computing and communications.

5. Adaptable environments and devices.
6. Intelligent interfaces between humans, devices and the environment.
7. Seamless interaction between any pair of devices [7, 8].

The middleware proposed in this work aims at supporting these requirements, whilst facilitating the design of sophisticated environments through the abstraction of specific functionalities and requirements in different layers of abstraction.

4 Architecture of the Proposed Middleware

In this section we describe the functionalities proposed for each layer in our middleware. We also suggest different projects that could be used to support the implementation of each layer in our middleware.

Each layer defines different entities as first class entities, as follows:

- The infrastructure layer transports messages. The messages include entities of the upper layers.
- The layer of agents and services conveys semantic descriptions of agents and services.
- The layer of middleware services provides pré-built services to convey specific functionalities to the environment.
- The layer of collective intelligence contains more ellaborate entities and artifacts in charge of the coordination of actions involving several devices in the environment.

The interactions between layers is defined using *ambient calculus* [15], so that each entity in a layer is considered na agent in the semantics of ambient calculus.

In figure 1 we have the layers of the middleware and corresponding projects related to each layer.

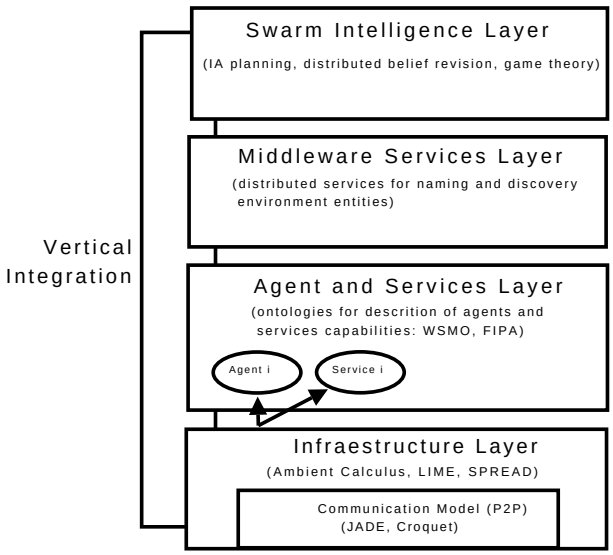

Figure 1. Middleware Layers and Corresponding Projects

4.1 Infrastructure

This layer provides the communications infrastructure for devices. It must support different communications technologies, such as Wi-Fi, Bluetooth and RF, and different architectures, such as P2P and client/server. It is based on the model presented in [20, 21], where messages are placed in a common space visible to the upper layers.

This layer must implement the message exchange infrastructure, which can be based on models such as JADE [11] - which is based on the FIPA model for message exchange in multiagent systems - and Croquet [10] to build collaborative multiuser applications.

4.2 Agents and Services

This layer must contain a repository of agents, services and semantic descriptions of their capabilities. It is responsible for the discovery of the required functionalities implemented in an agent or service that can be found in a device. In [4] we have introduced a mechanism for the discovery of services and agents based on ATMS (assumption-based truth maintenance systems), which matches functionalities with tasks [6]. Moreover, in [13, 14] we find proposals to standardize the descriptions of such functionalities so that other layers in the middleware can retrieve them.

4.3 Middleware Services

This layer contains services provided by the middleware itself and which are distributed across the devices in the environment. Examples of such services are yellow pages and discovery of entities from different layers, management of ontologies and semantic objects.

This layer provides to the other layers resources for:

- Management and maintenance of ontologies, implemented using mechanisms such as e.g. COBRA [3];
- Retrieval and processing of contextual information [2];
- Yellow pages and Discovery of entities based on mobility, as in [18].

4.4 Collective Intelligence

Contains more ellaborate entities (artifacts) to coordinate the entities of the lower layers [7, 8].

In [1, 12] several methods are presented to manage and coordinate services provided by heterogeneous agents. The difference in our case is that the coordination must manage entities of lower layers, i.e. representations of services and agents in different levels of abstraction.

The construction of these entities and artifacts must rely on concepts originating from:

- Game Theory;
- Distributed truth maintenance systems;
- Multiagent coordination;
- AI planning; etc.

5 Conclusions and Future Work

In [5] we find several scenarios of applications of ambient intelligence, with their corresponding challenges related to social and ethical issues, technological issues and industrial issues. As a result, the following topics are suggested as relevant for the construction of ambient intelligence:

1. Specialized hardware development;
2. Open platforms development;
3. Unobtrusive and invisible technologies;
4. Support to personal and social activities; and

5. Safety and reliability.

We intend to build proof-of-concept systems using our middleware, in which these topics are taken into account.

At present, we are working on the specification of the interactions between layers of abstraction, employing *ambient LCC* [15, 16] (Lightweight Coordination Calculus) - which is a variation of ambient calculus - for that. Our goal is to specify the interactions between layers in such way that different implementations of each layer can be built and used in our middleware, provided only that it is aligned with this specification.

REFERENCES

[1] Keith S. Decker and Victor R. Lesser, *Designing a Family of Coordination Algorithms*.

[2] Schmidt A., *Interactive Context-Aware Systems Interacting with Ambient Intelligence*, Ambient Intelligence, IOS Press, 2005.

[3] Chen H., Finin T. and Joshi A., *An Intelligent Broker for Context-Aware Systems*, Adjunct Proceedings of Ubicomp 2003, USA, October, 2003.

[4] Noriega G. Crhistian, *Um modelo para ambientes inteligentes baseado em servicos web semanticos*, Master dissertation, University of Sao Paulo - Brazil, Agosto 2007.

[5] Ducatel K., Bogdanowicz M., Scapolo F., Leijten J. e Burgelman J-C. *Scenarios for Ambient Intelligence in 2010*, *ISTAG*, Febrary 2001.

[6] Wang Z. e Garlan D., *Task-Driven Computing*, Technical Report, School of Computer Science, Carnegie Mellon University, May 2000.

[7] Ramparany, F., Boissier, O., Brouchoud, H, *Cooperating Autonomous Smart Devices*, In: sOc'2003, the Smart Objects Conference, pp. 182-185 (2003).

[8] Naohiko Kohtake, *Smart Device Collaboration for Ubiquitous Computing Environments*, Ubicomp2003.

[9] Jianhua Ma, Laurence T. Yang, Bernady O. Apduhan, Runhe Huang, Leonard Barolli and Mokoto Takizawa, *Towards a Smart World and Ubiquitous Intelligence: A Walkthrough from Smart Things to Smart Hyperspaces and UbicKids*, Pervasive Computing and Communication, IEEE, March 2005.

[10] David A. Smith, Alan Kay, Andreas Raab and David P. Reed, *Croquet - A Collaboration System Architecture*, First Conference on Creating, Connecting and Collaborating through Computing, 2003.

[11] Fabio Luigi Bellifemine, Giovanni Caire and Dominic Greenwood, *Developing Multi-Agent Systems with JADE*, Wiley, April, 2007.

[12] Omicini A., Zambonelli F., Klusch M. and Tolksdorf R., *Coordination of Internet Agents Models, Technologies, and Applications*, Springer, 2001.

[13] Xavier E. and Correa da Silva. F. S., *Expressing Systems Capabilities for Knowledge Coordination*, AAMAS'2002.

[14] Roman D., Lausen H. e Keller U., *Web Service Modeling Ontology (WSMO)*, `http://www.wsmo.org/TR/d2/v1.2/`, Working Draft D2v1.2, April 2005.

[15] L. Cardelli and A. D. Gordon, *Mobile Ambients*, Foundations of Software Science and Computational Structures, number 1378 in Lecture Notes in Computer Science, pages 140Ű155. Springer-Verlag, 1998.

[16] Sindhu J., Perreau de Pinninck Bas A., Robertson D., Sierra C. and Walton C., *Interaction Model Language Definition*, IJCAI Workshop AOMS Agent Organizations Models and Simulations, 2007.

[17] Sven Apel, Klemens Böhm, *Towards the Development of Ubiquitous Middleware Product Lines*, In ASE'04 SEM Workshop, volume 3437 of LNCS, 2005.

[18] Issarny V., Sacchetti D., Tartanoglu F., Sailhan F., Chibout R., Levy N. and Talamona A. *Developing Ambient Intelligence Systems: A Solution based on Web Services*, In Journal of Automated Software Engineering. Vol 12. 2005.

[19] Renato Cerqueira, Christopher K. Hess, Manuel Román and Roy H. Campbell, *Gaia: A Development Infrastructure for Active Spaces*, In Workshop on Application Models and Programming Tools for Ubiquitous Computing, September 2001, Atlanta, Geogia.

[20] Picco G., Murphy A. and Roman G., *Lime: Linda Meets Mobility*. In Proceedings of the 21stInternational Conference on Software Engineering (ICSE'99), Los Angeles, USA, 1999.

[21] Couderc P. and Banâatre M., *Ambient Computing Applications: an Experience with the SPREAD Approach*, Procs. of the 36th Hawaii Int'l Conf. on System Sciences (HICSS'03), IEEE Comp Soc, 2003.

[22] Steven Shafer, et al, *The New EasyLiving Project at Microsoft Research*, Proceedings of the 1998 DARPA/NIST Smart Spaces Workshop, July 1998.

[23] Project Oxygen, *http://www.oxygen.lcs.mit.edu/*.

[24] Project One.world, *http://www.cs.nyu.edu/rgrimm/one.world/*.

[25] Helal A., Mann W., Elzabadani H., King J., Kaddourah Y. e Jansen E., *Gator Tech Smart House: A Programmable Pervasive Space*, IEEE Computer magazine, March 2005.

[26] Sousa J. P. e Garlan D., *Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments*, Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture, August 25-31, 2002.

[27] HomeLab, *http://www.research.philips.com/technologies/misc/homelab/*, Philips Research.

[28] Kidd, Cory D., Robert J. Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth Mynatt, Thad E. Starner and Wendy Newstetter, *The Aware Home: A Living Laboratory for Ubiquitous Computing Research*, In the Proceedings of the Second International Workshop on Cooperative Buildings - CoBuild'99.