Enhancing the Role of Multi-agent Systems in the Development of Intelligent Environments

Davide Carneiro[1], Paulo Novais[1], Ricardo Costa[2], José Neves[1]

[1] CCTC, Departamento de Informática, Universidade do Minho, Braga, Portugal
[2] CIICESI, College of Management and Technology of Polytechnic of Porto, Felgueiras, Portugal
[1] {dcarneiro, pjon, jneves}@di.uminho.pt, [2]rcosta@estgf.ipp.pt,

Abstract. The development of Intelligent Environments is a complex challenge. This complexity arises, in part, from the amount of different devices that need to be seamlessly integrated in a common and homogeneous environment, despite the fact of each device having its own characteristics. This heterogeneity of devices is particularly risky when one passes from the specification to the implementation phase, where all unexpected things start to happen. Multi-agent systems are the paradigm par excellence for implementing Intelligent Environments. However, traditionally, agents are only used at the implementation phase. In this paper we propose a new 3 step approach in which agents are used during all the development process, playing undoubtedly a much more preponderant role and making the path from the specification to the implementation a much easier and controllable one, always having in mind the challenges of the development of Intelligent Environments.

Keywords: Multi-agent systems, Distributed Computing, Ambient Intelligence.

1 Introduction

Ambient Intelligence (AmI) [7] has been revealed in the last years as one of the most promising and exiting research fields. In fact, the possibilities that arise from this merging of Ubiquitous Communication, Ubiquitous Computing and Intelligent User Interfaces are what a few years ago only science fiction visionaries could dream on. Ambient Intelligence may even be considered to be a revolution as it completely changes the way we see and we interact with computers, definitively ending with the long-established desktop paradigm.

In AmI the most important change is that the user is placed at the center of the system. We have no more to move close to a computer and use it according to its ways. Now, computers come to surround us, disguised behind our common appliances, and constantly interact with us, sometimes without us noticing it, and constantly providing services and taking actions that empower our experience.

However, as in every new technology, there are challenges that need to be addressed [2001]. Ambient Intelligence is not an exception. Most of the challenges arise from the fact that it is built out of many different components. In fact, it is common to find objects as different as sensors, personal devices, regular computers or actuators. Other than physical components, we must also consider

the software ones: databases, decision support systems, inference engines or remote services, only to name a few. We cannot fail to remember that each of these components has its own characteristics: a personal device has limited power capability and uses a wireless communication protocol, a sensor uses a dedicated cable and its manufacturer's communication protocol and an actuator may use either the power line or a wireless protocol to communicate.

It is a fact that Multi-agent systems can and have been used to overcome these difficulties. The classical approach is to design a specification of the problem, and then either simulate it or advance right to the implementation phase. It is commonly accepted that simulation is a great tool for the development of such complex systems: it allows for the test and assessment of the system before its actual implementation. Advancing to the implementation phase without at least simulate the desired system may therefore be irresponsible and carry a larger and more costly implementation phase. However, simulation by itself, is not a guarantee of much better results. In fact, when one looks at some current AmI simulation projects [8, 11], it is easy to conclude that the simulation architecture as well as the technologies used, are sometimes so different from the technologies used for the implementation that the moving on to that phase may be almost as costly as doing it without having used simulation.

What we propose is a 3 step process that fully exploits the possibilities of agents during all its phases: specification, simulation and implementation. This will allow, as many other approaches, to develop agent-based Intelligent Environments. However, having in mind the final agent-based architecture since the first stage of development, will allow the development of a better suited specification as well a simulation platform that truly resembles the implemented system, definitively making the implementation a smoother and more controlled process.

2 The Classical Approach

The use of agents for the development of Intelligent Environments constitutes no novelty. It is a paradigm so suited to this task that it is easy to find several projects that implement the services of AmI on top of agents. See for example [9] and [3]. Classically, what is done is to specify a set of agents, each one with well defined roles (e.g. temperature control, presence control, agenda management, power management). The agent paradigm is in fact very suited to do this task.

Agents have the autonomy to take their own decisions towards the achievement of their goals, can communicate with other agents and can even negotiate in order to have some influence on the beliefs of others [10]. Negotiation [5] is in these environments very important as it is vital to solve the common cases of contradictory objectives (e.g. the comfort agent wants to raise the temperature in the room but the energy control agent wants to save energy).

More than that, agents can hide the singularities of the devices they represent. A temperature agent, for example, is the specialist in reading the temperature from a given sensor. No other agent needs to have knowledge about how to read the temperature, all they have to do is to request the temperature value to the temperature agent, over a standard communication protocol.

3 The Three Step Approach

In section 2 we have seen the way that agents have been exploited until the present for the development of AmI. However, it is easy to conclude that agents play only a preponderant role in the implementation phase of the environment. Could they not be used since the beginning of the process? Wouldn't it be possible to develop a simulation platform whose architecture would match the final architecture, thus agent-based? It is our conviction that it is possible and that it significantly improves the process of the development of Intelligent Environments.

In fact, what we propose in this paper is a three step process in which the agent paradigm is present right from the beginning. In the first step the agents and their hierarchical structure are defined. In the second step a simulation platform is built that respects the architecture defined in the first step, with every agent simulating the services it will actually provide when implemented. We will call these agents Virtual Agents (VA). This naming convention is rather misleading as these agents are evidently real agents, living in a real platform, nevertheless acting in a virtual environment shaped after the image of the target environment. It will however be useful for distinguishing between these and the Real Agents (RA), the agents operating in the real environment. The path to the third step is taken when the VA are gradually replaced by the RA: similar agents that do not simulate their services but in fact provide them in the real environment. At some intermediary point there will be, in the same platform, Virtual and Real agents interacting. These two types of agents are actually the same and run in the same platform, thus allowing communication between them. The only difference is that RA work with real components (e.g. sensors, actuators, databases) while the VA works with virtual or simulated ones (e.g. virtual sensors, simulated user).

Step 1: Specification

The first step is to completely specify the problem. There are several main issues to have in mind when designing the specification of an intelligent environment. It is mandatory to know the type of environment that will be implemented (e.g. domestic, hospital, commercial, geriatric). Based on this, the main services to be provided will be depicted and therefore the devices needed to implement these services will also be known. The target type of users is as important as the type of environment, so that the services are personalized. In fact, most of the services need to be adapted according to the context of the user (e.g. role in the

environment, age, physical impairments, gender). Having defined these two key issues, work may start on the definition of the agents.

A methodology suited for building complex agent-based systems is called evolutionary development of systems. It consists in starting the development with a reduced number of agents, each one representing a group of services that can be grouped according to some criteria. Each of these agents is then sub-divided into more simple agents, and so on, resulting in a hierarchy that, as we go to lower levels, find more specific and simpler agents [4]. This has several advantages. In one hand, it is simpler to analyze the agents and detect eventual errors or unexpected behaviors as each agent has simple and well defined tasks. In the other hand, it makes it possible to reuse agents and services, resulting in more lightweight and scalable systems.This process of iterative subdivision is also known as compositionality. This concept denotes entities that can be described by the sub-entities that make part of it and the rules used to combine them. In the specific case of an agent-based platform, it can be described by the agents that make part of it and their relations (e.g. social, hierarchical), the sub-agents that make up each main agent and their relations, and so on.

When defining this hierarchical structure of agents, we will have in mind one central idea: the architecture that the hierarchy of the agents defines will be the architecture of the simulation platform and the architecture of the final environment. This means that the simulation platform and the real environment will be compatible and will even be possible for them to coexist.

Step 2: Simulation

The main objective of this second step is to look at the specification defined before and implement all the agents respecting their hierarchical structure. Moreover, each agent will simulate the services needed to implement the roles that were assigned to it. What we achieve is an architecture that matches the one of the final implementation of the system, with the same agents providing the same services, although in a simulated fashion. However, in order animate this simulation, two more agents are necessary. The first one, which we will call the Simulator agent, will be responsible for controlling the simulation-related parameters, namely the velocity of the simulation, its length or the time of the day. The second one is the Environment agent. This agent will simulate the environment in terms of its main characteristics, namely the environmental parameters, the physical properties, the users, the appliances, among others. This will make it possible for the agents of the platform to acquire the data from the environment that they need to take their decisions.

Once the simulation is running a wide range of tests and assessment can be performed on the architecture and their building blocks in a much more controlled environment. There are two main objectives here: to test the communication protocols defined previously and to test the services of the agents and their effects on the simulated environment.

Should any modification be necessary and an agent or the whole architecture can go back to step one, for a better specification, whether it is by better detailing

an agent, modifying its role on the architecture, rearranging the hierarchy, among others. We can now make the exercise of imagining doing these changes with the real implementation and the complexity of changing a system that has already been deployed. In fact, using this approach, one gets a simulation platform that is as close as possible to the architecture of the final implementation, which allows detecting situations that would, otherwise, only be visible after the implementation. Another advantage is that this is an iterative process that can be repeated until one reaches the desired architecture, with the desired behavior, and only then, with more confidence, advance to the implementation phase.

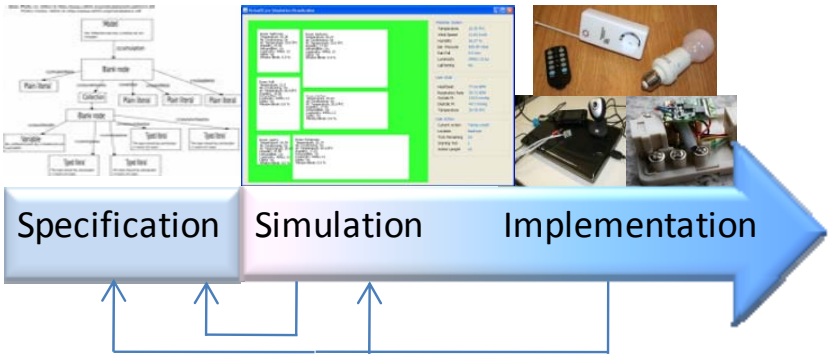

Fig. 1. The three steps of the development process.

Step 3: Implementation

The final step consists in implementing the architecture defined and simulated. Traditionally, one would have to implement all the agents and their services to, only then, start making the final tests to the architecture. However, using this approach, it is possible to gradually replace a VA for its equivalent RA, resulting in a much smoother and controlled transition from the simulation to the implementation phase. The only requirement here is that the RA has the same name and the same services signature of the VA it is replacing. This way, compatibility is kept with the remaining agents that eventually use its services. The main advantage here is the possibility of replacing one agent at a time, being possible to perform all the tests to each new RA while he is already integrated in a running platform, eventually a partially simulated one. If an error is found on a new RA, it can be removed from the architecture and again replaced by its equivalent VA, ensuring that the architecture will keep running so that other tests can be performed to other agents. This can be done individually to each new RA, maximizing the confidence on the final architecture and making it easier to isolate errors or wrong behaviors.

Hence, there is not an explicit point at which step two ends and step three starts. Instead, they overlap (Figure 1). Step three starts when the first VA is replaced for its equivalent RA and step three ends when the last VA is replaced. Evidently, during this process, if necessary, the process can go back to step one, for some

adjustments. This constitutes no major problem to a modular agent-based architecture.

4 Our Experience: The VirtualECare Project

The ideas depicted here are the result of the experience of this team in AmI projects. The most recent one, the VirtualECare project, is the soundest example in which this three step process has been applied with success. The VirtualECare project [6] is built on top of a Jade-based Multi-agent System in support of computational systems that seek to monitor and provide personalized health care services to its users or patients.

This way, our work in VirtualECare began by defining the main high level agents. The first configuration was composed of six agents: *Supported User*, *Environment*, *Group Decision*, *Call Service Center*, *Call Care Center* and *Relatives*. For the sake of space, we will not detail all the agents and their complete hierarchy here. We will, instead, focus on the Environment agent, the one that is undoubtedly more interesting given the ambit of this paper (Figure 2).

When we thought of this agent, we had in mind that we needed an agent that could provide all the important information about the environment, namely values from the sensors, the intrinsic characteristics of the environment or the appliances present. Evidently, this would result in a very complex agent. Therefore, it has been sub-divided into five agents. The *Physical Properties* agent deals with the physical properties of the environment such as the geographic distribution of the rooms, its insulation and its geometry, among others. The *Environmental Parameters* agent deals with all the sensorial information. The *Appliances* agent is responsible for communicating and issuing orders to every compatible appliance in the environment. The *External Environment* agent was made responsible of sensing the outside environment that surrounds the setting, namely in terms of the temperature, rainfall or luminosity. The *UIs* agent deals with all the interfaces with the user. The same process of sub-division was iteratively applied to the resulting agents as needed in order to minimize their complexity, as visible in Figure 2.

Having finished the Specification phase, the team advanced into the Simulation phase. In this phase a fully configurable simulation platform [2] has been developed and its architecture interactively tested and improved. This architecture, as stated before, matched the specification defined and the final architecture. Once we achieved a stable hierarchy of intercommunicating agents, we advanced into the implementation phase. This last phase consisted in developing the Real Agents. Taking as example the *Temperature* agent, we modified de code of the VA so that instead of simulating a temperature value, it actually read the temperature value of a given sensor. All the structure of the agent remained the same, including the signatures of the services in order to maintain the compatibility. The only thing left was to replace the RA for the VA.

This same process has been applied to other agents and has revealed to be a rather smooth one. At this moment, as we are not yet in possession of all the hardware required for implementing such architecture, we have a hybrid

architecture that mixes VA with RA. It is, however, a stable and polyvalent one that allows us to develop and test AmI services, which would otherwise be impossible.

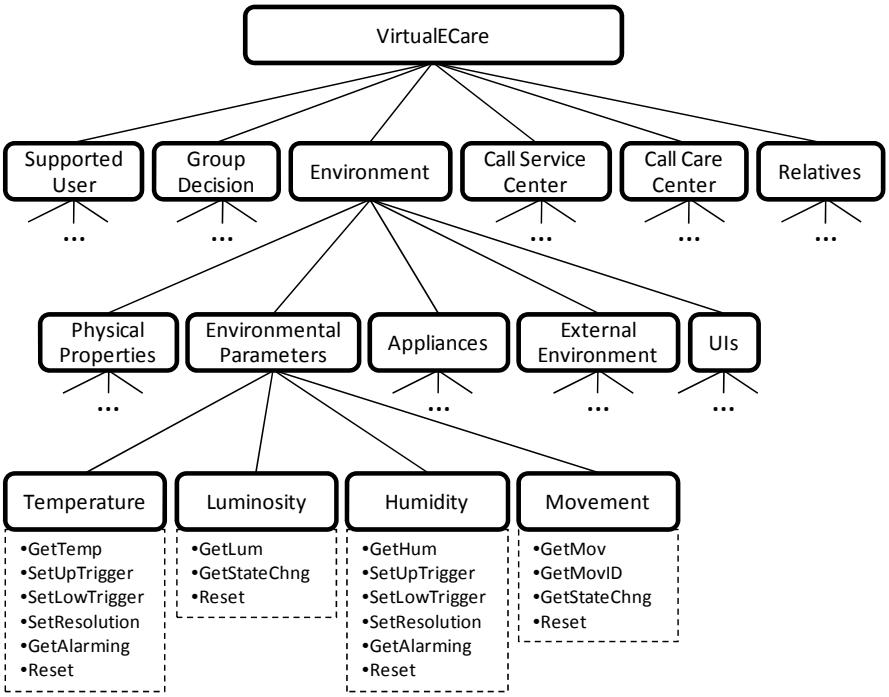

Fig. 2. A detail of the agent's hierarchy that make up VirtualECare environment. The leaf agents are the simple agents with their roles well defined in terms of the services they provide.

5 Conclusions

In this paper we have presented a new approach for the development of Intelligent Environments based on the agent paradigm. It is a three step process that has as main objective to fully exploit the capabilities of this paradigm from the very beginning of the development. It also constitutes a way of developing a simulation that actually matches the configuration of the final environment, which allows testing not only the agents but the architecture itself. This configures definitively a step ahead as the simulation platforms developed nowadays generally focus on the services themselves and completely forget the architecture, resulting in a simulation that is much different from the implementation. As we concluded during the development of the VirtualECare environment, this approach results in a development process that is faster, simpler, smoother and more homogeneous.

References

1. Aarts, E., and Roovers, R. 2003. IC design challenges for ambient intelligence. In Proceedings of the conference on Design, Automation and Test in Europe-Vol 1.
2. Carneiro, D., Costa, R., Novais, P., Neves, J., Machado, J. and Neves, J. 2008. Simulating and Monitoring Ambient Assisted Living, in Proceedings of the ESM 2008 - The 22nd annual European Simulation and Modelling Conference, Le Havre, France, October.
3. Fraile, J. A., Bajo, J., Pérez, B., and Sanz, E. 2008. HoCa Home Care Multi-Agent Architecture. In International Symposium on Distributed Computing and Artificial Intelligence.
4. Jennings, N., Faratin, P., Lomuscio, A., Parsons, S., Wooldridge, M., and Sierra, C. 2001. Automated Negotiation: Prospects, Methods and Challenges. Group Decision and Negotiation.
5. Kraus, S. 1997. Negotiation and cooperation in multi-agent environments. Artificial Intelligence, 94(1-2), 79–97.
6. Novais, P., Costa, R., Carneiro, D., Machado, J., Lima, L. and Neves, J. 2008. Group Support in Collaborative Networks Organizations for Ambient Assisted Living. In Towards Sustainable Society on Ubiquitous Networks, Makoto Oya, Ryuya Uda, Chizuko Yasunobu (eds), Springer-Verlag, Series: IFIP International Federation for Information Processing.
7. Riva, G., Vatalaro, F. and Davide, F. 2005. Ambient Intelligence. IOS Press.
8. Shirehjini, A. A. N. and Klar, F. 2005. 3DSim: rapid prototyping ambient intelligence. In Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies (pp. 303-307). Grenoble, France.
9. Tapia, D. I., Bajo, J., De Paz, F., and Corchado, J. M. 2006. Hybrid multiagent system for Alzheimer health care. Proceedings of HAIS.
10. Wooldridge, M. and Jennings, N.R. 1995. Intelligent Agents: theory and practice. In The Knowledge Engineering Review, p. 115-152.
11. Youngblood, M., Cook, D. J. and Holder, L. B. 2005. Seamlessly engineering a smart environment. In 2005 IEEE International Conference on Systems, Man and Cybernetics (Vol. 1).