Novais P., Costa R., Carneiro D., Neves J., Inter-Organization Cooperation for Ambient Assisted Living, in Journal of Ambient Intelligence and Smart Environments, IOS Press, ISSN 1876-1364, Vol. 2, No. 2, pp.179–195, 2010.

Inter-Organization Cooperation for Ambient Assisted Living

Paulo Novais[a,*], Ricardo Costa[b], Davide Carneiro[a] and José Neves[a]
[a]*Departamento de Informática-CCTC, Universidade do Minho, Campus de Gualtar, Braga, Portugal*
[b]*College of Management and Technology - Polytechnic of Porto, Felgueiras, Portugal*

Abstract. In the last years we have witnessed to a substantial increase on the number of people in need of care services, especially among the elderly, a phenomenon related to population ageing. However, this is becoming not exclusive of the elderly, as diseases like obesity, diabetes, and blood pressure have been increasing amongst young adults. This is a new reality which needs to be dealt by the healthcare sector, specifically the public one. Given these new scenarios, the importance of finding new and cost-effective ways for health care delivery are of particular relevance, especially when it is believed that these new patients should not be removed from their natural, day-to-day life, environment. The evolution of the, so called, new technologies may pay here a very important role as they may become part of the solution for this new problematic. Actually, they are already been used as, in recent years, several projects have raised in this relatively new area of work. These projects, although legitimate ones, were essential for delineating a path to pursue for others to come, as they were in some case, very simple ones (e.g. panic buttons), and, especially, reactive ones. In this paper, we are going to present how we are trying to evolve these projects a step further, through the introduction of proactiveness as a key factor, taking advantage of "new", as in applied to this areas, techniques of decision making, idea generation, argumentation and data quality, applied, not only to the in transit information, but also to the one provided by the several intervenient as well as themselves. In order to be able to pursue this delineated path, a new approach for knowledge representation, reasoning, and even for problem solving is proposed. To achieve these goals, the VirtualECare environment is presented, together with its sustaining infrastructure and architecture. Particular attention will be paidto how it may be used to simulate a virtual Assisted Living Environment in order to, later, better monitor real ones, attending to its customers' needs.

Keywords: Inter-Organization Cooperation, e-Health, Ambient Intelligence, Simulation, Monitoring

[*]Corresponding authors. E-mail: pjon@di.uminho.pt.

Novais P., Costa R., Carneiro D., Neves J., Inter-Organization Cooperation for Ambient Assisted Living, in Journal of Ambient Intelligence and Smart Environments, IOS Press, ISSN 1876-1364, Vol. 2, No. 2, pp.179–195, 2010.

1. Introduction

Although, in the last years, health care costs have been rapidly rising, the quality of the provided service is not following the same trend and, in non rare situations, it is not only poor but simply inexistent [17]. Given this negative scenario, changes have to be made in the care provision area and new approaches to this problem need to be found and followed. One of these approaches may well be the use of the emerging technologies, as the exploitation of collaborative networks, especially in the health care sector, which have not been, yet, studied and applied in depth in this field. Undeniably, patients in a general way and the ones with chronic diseases in particular, have to be empowered through the use of IT-enabled disease management processes, in order to allow them, in not rare cases, to take control of the situation and, under an economical perspective, lower treatment costs. Under this new scenario, they will stay at their own premises, being object of real time monitoring, leading to a continuous evaluation of critical data, allowing the achievement of the desired, and already mentioned, proactiveness. It is also believed that family and relatives should also be empowered, as they should have real time access to all the results of the collected information.

Following similar lines of thought, several IT health care projects have been analyzed [2, 7, 10, 14, 18, 24, 32], some being simple "panic buttons", others domotic technologies with the intention of using intelligent sensors, and the most advanced ones focusing on the development of context-aware interfaces for elderly or the development of "smart-homes" for aging-in-place. There have also been found generic frameworks with the intention to cater for the integration of these same systems like, for instance, the Telecare one [6].

The main objective of future projects should be to extend Telecare-like systems a step forward, giving birth to VirtualECare alike [26]. In order to be able to accomplish this auspicious objective, we have looked to the roles that collaborative networks and learning systems may be able to play within this innovative process engaged in a smart home for care of patients, especially elderly, and have suggested a framework which will allow any organization to strategically model a conducive to innovative collaborative environment. Such a framework will also be able to identify key areas of Inter-Virtual Organization Cooperation for Care of the Elderly, which should be addressed in line with the collaborative requirements of the several intervenient care providers. In order for such a result to be achieved new paths must be pursued, relying on a mix of different contributions from the Artificial Intelligence field such as Collaborative Networks, Ambient Intelligence and Knowledge Representation tools, coupled whit different computational paradigms and methodologies for problem solving, such as agent-based systems and (GDSS) Group Decision Support Systems [12]. For that purpose, spaces (e.g. houses, buildings, critical areas in hospitals) must be enriched with smart artifacts and then, through the use of automated or semi-automated Group Decision Support Systems, health related problems and others may be detected and, automatically, solutions may be presented.

2. Ambient Intelligence

Ambient Intelligence (AmI), a relatively new paradigm within the Information Technology world, allows people to be empowered through a digital environment that is aware of their presence and context, being sensitive, adaptive, and responsive to their actual needs, habits, attitudes and emotions [30, 19, 20]. AmI can be, in a very simplistically attempt, defined as a merger of ubiquitous computing with social user interfaces. It is built on top of advanced networking technologies, defined by a broad range of mobile devices and other objects and, adding adaptive user-system interaction methods based on new insights in the way people interact with computing devices, presents, as a result, digital environments created to improve quality-of-living for their users [3]. These context aware systems are achieved combining ubiquitous computing, communication, and entertainment, with human behaviour (e.g. natural interaction and intelligence) mixed with different perceptions of the world that come, among others, from the disciplines of Artificial Intelligence, Psychology or Mathematics, coupled whit different computational paradigms and methodologies for problem solving [16, 29].

2.1. Inter-Organization Cooperation

In Inter-Organization Cooperation there are factors that tend to be surrounded in the local milieu, which according to Dosi [13], can be seen as the social embedded processes that allow organizations to obtain outside complementary knowledge and be innovative in the course of interaction among different actors, i.e.

the local or regional milieu needs to include not only the substances related to the service structure or economics terms, but also social, cultural and institutional ones [22]. Thus, in the interaction of the different actors, the cooperation elements can be found in a kind of common language, social relationship, norms, values and institutions, which, in our work, will be set in terms of an extension to the logic programming language, being their knowledge bases built as logical theories that found their foundations on this extension [25].

These cooperating elements, together with the supporting platform being developed, will configure a Multi-agent System (MAS) [33] that will enrich its building architecture with features such as expansibility, scalability, autonomy or proactiveness.

Conclusions are supported by deductive proofs, or by arguments that include conjectures and motivate new topics of inquiry, i.e. if deduction is fruitless the agent inference engine resorts to abduction, filling in missing pieces of logical arguments with plausible conjectures to obtain answers that are only partly supported by the material available to the inference engine.

2.2. Smart Homes

Some good examples of AmI applications are the, nowadays emerging, smart homes [4, 31]. Smart homes are normal day-to-day environments enriched with sensors and actuators, in order to be able to acquire information about its users. This grouped information, also called context, helps to describe the status and surroundings of each user, thus it is important to have as much information about each one as possible, so that the, autonomous, decisions taken by the system can, gradually, become optimal. This information may comprehend, for instance, the user location and/or vital signs, other user's presences, house climacteric conditions, or even past experiences and interactions [5].

With all the above presented information and a set of interaction rules, these (smart) homes are able to decide, at each moment, how to control the embedded appliances in order to better manage the inside environment. These appliances are perfectly normal ones, but empowered with computational and communicational capabilities. Grouped they compose a network of devices, including not only sensors, like smoke detectors, switches, pressure sensors and RFID, among others, as well as electrical devices as ovens, air conditioning systems or microwave controllers, with the particularity of all been able to work together. There are even some devices than can be worn or carried by any user which, recurring to Personal Area Network (PAN) protocols and technologies, are able to maintain a close interaction between them and the system. One of the main applications in this field is to, in real time, read the vital signs of any user, thus allowing him to be under constant surveillance and to have an increased dose of mobility.

Another positive aspect of this new kind of environment is the user interaction. In fact, the interaction with this, apparently regular, group of devices is so simple and natural that the singularities and, in some cases, complexity behind them is unnoticed. These homes tend to be so easy and intuitive to use that anyone, without any special knowledge and/or skills about computational systems, is able to interact with them. This, by itself, combined with the fact that these new systems follow the user-centred paradigm, constitutes a major shift in traditional human-computer interaction. Until now, to be able to interact with a computational system, it was necessary to approach it, and to do it according to its means or interfaces, which not always are very intuitive. In the actual trends, computational systems are merged in the existing environments, perfectly disguised in our common day-to-day devices, without a perfectly identifiable central convergence point. We, in the other hand, are under constant interaction with them, although most of the time without noticing it, and are becoming the long deserved central element of the interaction model. We may say that smart homes are also based on this approach.

As it can be easily seen, this is a promising new field of research, and numerous research projects can already be found: Domus [28], Aware Home [1], MavHome [9] and Gator Tech Smart Home [15]. All of these projects pursue the same objective, which is also said to be the objective of AmI: to place the person in the centre of the system, providing relevant services so that his life is improved. Being these advantages very significant to, any person, it is not hard to imagine how important they can become for impaired or persons with chronicle diseases or even for elderly ones living alone and/or isolated, bringing safety, comfort, well being and constant surveillance.

2.3. Simulation

Simulation consists of creating an alternative virtual reality to represent a real object. When making a simulation, generally one expects to predict how a

given object or system will behave when introduced in the real world. Nowadays, it is possible to draw conclusions about the behaviour of, almost, any system being studied and about its feasibility without having to actually physically build them. It is obvious that one has to select the most relevant features or behaviours of the system under study, so that the results are as accurate as possible.

Some of the common uses of simulation are the modelling of natural systems (e.g. weather forecasts, storm evolution, and earthquake damage), testing and optimizing new technologies and the construction of new or special buildings (e.g. the new skyscrapers, dams), just to name a few.

In AmI, simulation is yet a relatively unused tool, but is introducing several benefits, especially for projects that are developing new architectures or systems that are either too complex, expensive or risky for an implementation without prototyping.

2.4. OSGi

The OSGi (Open Services Gateway Initiative) Alliance [27] is a non-profit corporation created in March 1999 and is now formed by companies such as Mitsubishi Electric Corporation, Siemens AG, Oracle Corporation, Nokia Corporation or Sun Microsystems. Its aim is to promote interoperability of applications and services delivered and managed via networks. The OSGi Service Platform is a Java-based application server for networked devices, ranging from computers or mainframes to mobile phones or other handheld devices, which means it can be deployed to any platform running the Java Virtual Machine.

OSGi intends to establish standards in Java programming, highly specific, catering for the sharing of Java classes that may be achieved in terms of a services platform paradigm. The use of this technology will let developers build Java applications on a modular basis. The resulting modules are called bundles, which are not only competent to provide services, but also to use services provided by other bundles. In OSGi, a bundle can be installed, started, stopped or un-installed at run-time and without any kind of system reboot, which makes OSGi-based technologies very modular and dynamic.

R-OSGi is an extension to OSGi which allows the access to services provided in remote OSGi implementations, in a completely transparent way, much like if they were local services.

The core component of the OSGi specification is the OSGi Framework. This framework provides a standardized environment for running computing programs, being divided into four layers: the execution environment, the modules layer, the life cycle layer and the service registry.

This technology suits, very well, AmI since one is able to complement the other. AmI can very simplistically be seen, as a way of providing services to its users, thus a way of providing them is essential. This is where OSGi comes up as a natural way of implementing AmI services, whether they are just java classes or something else able to run hidden behind java classes.

The interest in such a technology is obvious since in AmI there are multiple entities communicating with each other, providing and requesting services, and there is generally also the need for controlling or monitoring the system remotely. All the advantages associated to Java (e.g. portability and modularity) are fully exploited by the platform, granting it can be successfully deployed to devices with low resources and easily changed after implementation and during run-time.

3. The VirtualECare Project

The VirtualECare project [8] main objective is to present an intelligent service-based architecture able to monitor, interact and provide its customers with healthcare services of the utmost quality. This system will be interconnected not only with other healthcare institutions, but also with leisure centres, training facilities, shops and patient relatives, just to name a few.

The VirtualECare architecture is a distributed one with their different modules interconnected through a network, each one with a different role (Figure 1). A top-level description of the architecture machinery is given below:

- SupportedUser – elderly people with special health care needs, whose clinical data is sent to the CallCareCenter and redirected to the Group Decision Support System. This user should be constantly monitored, inside and outside its environment so the data must be provided in real time to the interested parts. It is the central component of the architecture and all the other components must work together to ensure its safety and well being;

- Environment – the elderly natural environment, provided with sensors, in which the clinical data is sent to the Group Decision Support System through the CallCareCenter, and the remaining one redirected to the CallServiceCenter. The data provided by this module must also be constantly available and analyzed so a reliable network connection is mandatory. The environment can be the user home, a hospital room or a day centre, just to name a few. The main actions of the other components towards the environment are to maintain the comfort and security parameters;

- Group Decision – This module is responsible for the long term planning regarding the health care of the patients. It should be composed of specialized staff like nurses and doctors as well as Recommendation Systems and tools for time and space distant meetings. In the overall this module should be able of planning all the issues related to visits to doctors, tests, automatically scheduling all this according to the user agenda;

- CallServiceCenter – Entity with all the necessary computational and qualified personal resources, capable of receiving and analyze the diverse data and take the necessary actions according to it;

- CallCareCenter – Entity in charge of computational and qualified personal resources (i.e. healthcare professionals and auxiliary), capable of receiving and analyze the clinical data, and take the necessary actions according to it. The user may make voice calls to this service and request assistance or advises and the service should respond and, if necessary, contact other modules like the Group Decision;

- Relatives – individuals that may have an active role in the supervising task of their love ones, being able to give precious complementary information about them and being able to intervene, in a complementary way, in specific crises (e.g., loneliness). Being an important part of the equation, the relatives should also have access to the health status of the patient so that they are constantly aware of its situation.

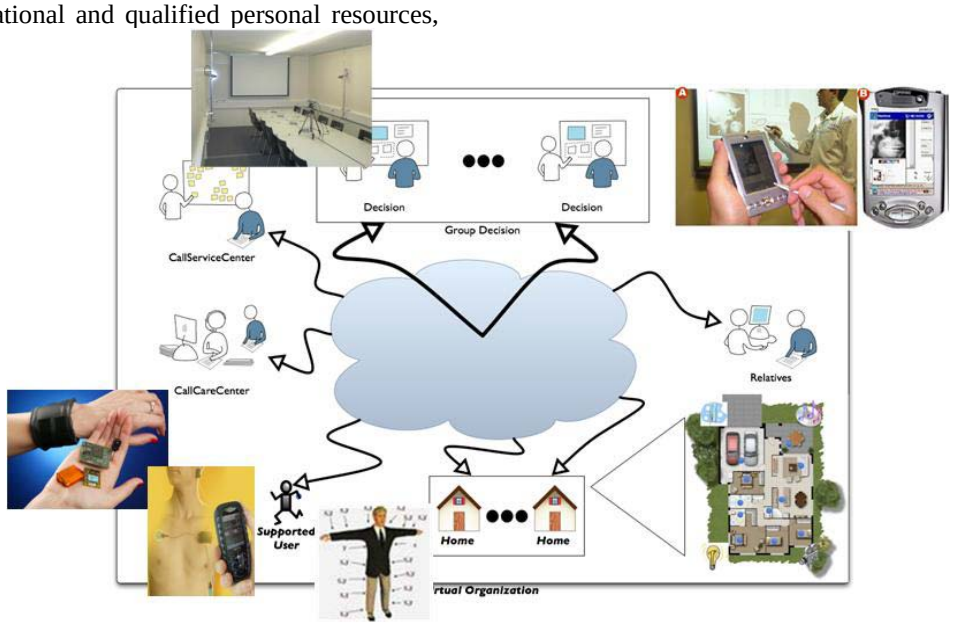

Figure 1: The VirtualECare Architecture

In order for the Group Decision Support System to make its work, it has to collect the opinion of specialized staff (e.g. nurses, paediatrics, cardiologists). There is also the need to have a digital profile of the SupportedUser, allowing a better understanding of his/her special needs. In this profile we can have several different kinds of relevant information, ranging

from the patient Electronic Clinic Process (ECP) to their own personal preferences (e.g. musical, gastronomic) passing by their own personal experiences, which can be used to better understand and satisfy their needs and expectations.

All these processes are taken care by the Group Decision Support System, thus removing the complexity from the user point of interaction with the system. The way these decisions are achieved is out of the scope of this paper and will therefore not be addressed here, however all the strategic decisions and more complex planning and reasoning tasks are taken care by the GDSS.

This solution will help healthcare providers to integrate, analyze, and manage complex and disparate clinical, research and administrative knowledge. It will provide tools and methodologies for creating an information-on-demand environment that can improve the quality-of-living, safety, and quality-of-patient care.

3.1. The VirtualECare Infrastructure

We have designed a first proposal for a generic, configurable, flexible and scalable infrastructure as presented in Figure 2. It is expectable that on top of it an extensive number of services will progressively arise. These services must, and will be, developed according to open standards, thus allowing the coexistence of several, different software systems interacting with each other through the use of common messages under standard protocols.

The fundamental components of the proposed infrastructure are depicted below [8]:

- Secure Communications – in order to have all the components to interact, a secure communication infrastructure is mandatory;
- Management – it is in charge of the configuration and monitoring of the processes (or parts) involved;
- Resources – it responds for every component registration and the managing of the resources catalogue;
- Authentication – every component must authenticate by itself in order to be able to interact with the others;
- Recommendation – it provides problem solving recommendations;
- Monitoring – it is concerned with all the sensors, reporting its status and data collected to the GDSS.

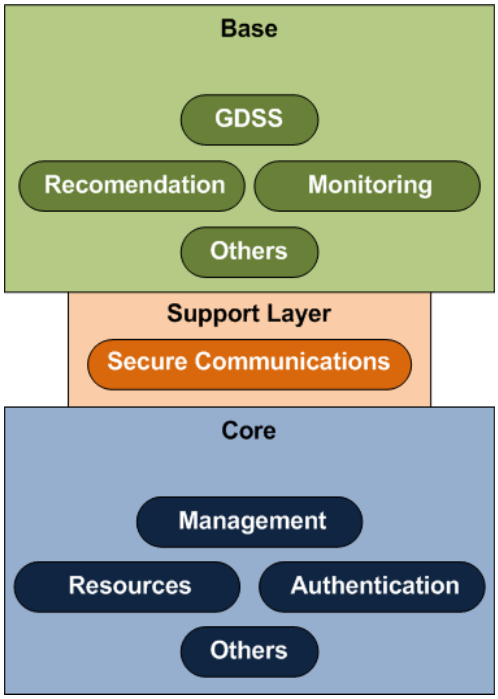

Figure 2: VirtualECare Infrastructure

3.2. The VirtualECare Architecture

The VirtualECare architecture is a dynamic and distributed one, built on a series of eventually geographically distant elements, which may, not only, "enter and/or leave" it at any time, but also be modified, as the services they provide (Figure 1). The architecture main components are: the *SupportedUser* and his/her Premise (SUP), the Monitoring System (MS), the Recommendation System (RS), the Group Decision Support System [8], the Database (DB) and the HL7 Translator module [16], among others, which leads us to an ubiquitous computational environment.

In this section, we are going to address some of our main issues, namely: how we have made the architecture a distributed one, and at the same time, modular, dynamic, extensible, flexible, scalable, heterogeneous and compatible. In order to be able to achieve these characteristics, we decided to adopt open and widely used technologies, such as OSGi, R-OSGi and Web-Services (WS) [23] and open stan-

dards and specifications such XML or the ones defined by the Foundation for Intelligent Physical Agents (FIPA). This approach became essential to ensure the communication and proper compatibility between the different above presented components, as they are a platform independent way of sharing information over a network. It also allows us to implement the architecture based on the client-server paradigm. As an example, the RS uses the WS provided by both the SUP and the DB, also providing in the form of a WS, the generated recommendation to the GDSS.

The information shared through the several WSs is defined in XML format, in terms of FIPA-ACL messages. This FIPA standard allows a description of the main content of the message without having to rely on ontology's, languages or speech-acts, making possible to messages to be forwarded and sent to the receivers without the need of content check.

However, it was also considered a way of structuring the actual content of any message in XML format, like the room temperature, a view of the Electronic Health Record (EHR), the whole EHR or a recommendation from the RS.

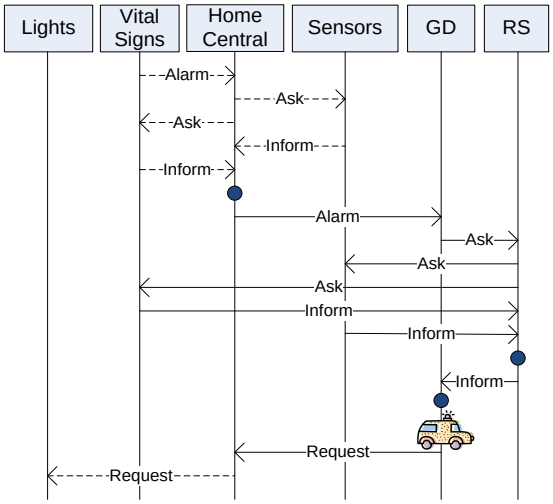

Figure 3: A Sample of a Communication Sequence Diagram

On Figure 3, we present a simplified example of a communication sequence where dashed arrows represent R-OSGi services being invoked, regular arrows stand for FIPA ACL messages being exchanged through WS and circles denote some major processing or communication with local bundles (using OSGi). All the processes are triggered by the bundle responsible for monitoring the vital signs of the Supported User. This bundle detects an irregular heart beat and warns the Premise OSGi, where the Multi-agent System is running using R-OSGi. The MAS requests information from the advanced sensors in another OSGi and asks again about the cardiac rhythm to the bundle that started the process to ensure that there was no reading error. Having gathered the information, the MAS decides that it cannot do anything to correct the situation and informs the GDSS, sending the key values. In turn, the GDSS contacts the Recommendation System which reads all the values of the sensors of the Premise and generates a recommendation which is then sent back to the GDSS. After communicating with some more members of the architecture (like specialized doctors) and having into consideration the answer from the Recommendation System, two actions are triggered: an ambulance is sent to the Premise and lights in the room of the user are turned on.

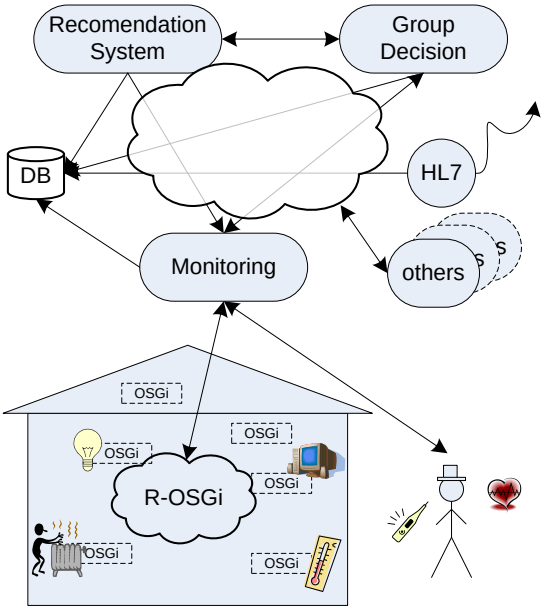

Figure 4: VirtualECare Architecture - Services Point of View

In Figure 4 we present a simplified view of our architecture. The arrows represent WS which permit the several components to exchange information. The arrows can be seen as "it uses service from" pointing from the client to the server. The Premise is a little more detailed, showing OSGi and R-OSGi subcomponents, which are responsible for interconnecting different kinds of elements.

Let us now detail the technologies used, by moving to a more close view of the architecture. At this level, two well known technologies where consid-

ered: OSGi and R-OSGi. These technologies are used in our architecture to achieve two main objectives at the component level: grant the compatibility and the communication between the members of the architecture and establish a hierarchical logical organization .

Let us look, as an example, to the Premise components. The Premise is made of physical parts (e.g. 1-Wire sensors and X10 actuators) and logical parts (e.g. software agents). 1-Wire sensors are used for measuring environmental values and X10 actuators to control appliances and equipments. Software agents are responsible for taking basic reactive actions like control the temperature or call for help in case of need. Moreover, the house may have a big number of rooms and floors which, like the rest of the components, can vary along the time. Therefore, and first of all, one needs to logically organize all these components. In order to achieve this purpose, we create several OSGi implementations. For each group of similar sensors, in each room, a bundle is created. This means that, for each room, there will be a bundle reading values from the temperature sensors, another one reading the values from the luminosity sensors, and so on. These bundles provide, as a service, the mean value of the last values obtained from the respective sensors. As for the actuators, there is one bundle controlling each equipment or appliance, which is able of sending X10 commands to the equipment it controls. The services these bundle provide are the X10 commands that may be issued to each equipment.

The bundles of the same type, in each floor, run in the same OSGi implementation, i.e. in each floor there is an OSGi implementation for the temperature bundles, another one for the luminosity bundles, and so on. Likewise, there is an OSGi implementation for the appliances of each type on each floor (e.g. an OSGi for lights, another for air conditioning). In addition, we may have also OSGi implementations inside the Premise, like a MAS to handle un-schedule situations. Each of these OSGi implementations have at least one additional bundle: a R-OSGi bundle which provides remote access to the implementation. This bundle acts as the bridge between the exterior and the sensors or actuators controlled by the OSGi implementation. In the case of the sensors, this bundle is remotely requested to provide the values of sensors and, in the case of the actuators, is through this bundle that the X10 commands arrive to the correct appliance. In this case, the bundle also guarantees that the command is valid and that the consequences of it being executed do not go against pre-established security policies (e.g. establishing the temperature of the air conditioning to undesirable values). R-OSGi is therefore the way of integrating each piece inside an OSGi-based component of the architecture, granting the communication among OSGi implementations.

Finally, let us describe how a MAS is merged inside this system. The MAS is responsible for regularly checking the values of the sensors, acting on the actuators accordingly (e.g. if the temperature suddenly drops, turn the heat on) and calling for help in case of need, as well as registering all the events into the Database. The aim is to make accessible and visible the functionalities of an agent (e.g., its methods) as services to other bundles. It would not be advisable to convert each agent into an OSGi bundle, since it would increase the development time and throw away the advantages of MAS based methodologies for problem solving. Therefore, the decision was to create an OSGI bundle that could make the bridge between regular bundles and Jade: the MAS bundle. This bundle can deal with an Agent Container (AC) and implement the methods declared on the interface of the agents in that AC as its own services. Moreover, this bundle must be able to start and stop agents, which in practice, corresponds to the start and stop of the services provided by them. The bundle, upon the reception of an invocation for an offered service from any other bundle, sends the invocation to the correspondent agent and delivers the respective result to the calling bundle. It must be noted that an agent, when trying to satisfy an invocation, may require the services provided by the other bundles currently available. This is possible through the MAS bundle.

As for the interface between the MAS bundle and the Jade system, a JadeGateway agent (JGa) is being used. The task of this agent is to act as a bridge between Jade and non-Jade code. This agent is created when the MAS bundle is started, along with the other agents. The JGa has access to which services are provided by each running agent; therefore, whenever a request from a service arrives to the MAS bundle, it knows to which agent the request should be forwarded. Likewise, if an agent needs to use a service from another bundle, it contacts the MAS bundle, which is responsible for contacting the correct bundle, invoking the service and forwarding the result back to the agent. This way, we created a bundle which allows for Jade instances, the agent platform that we adopted, to run behind OSGi implementations in a completely transparent way.

We have hereby detailed our architecture. At a high level, it is composed of components which share information based on the concept of services. Each

one of these components can then be detailed and looked closer in terms of its inner structure (e.g. sensors, actuators, MAS). The communication inside the components is based on OSGi and R-OSGi open standards, granting extensibility, modularity, dynamics and a hierarchical logical organization of the pieces that make part of each component.

4. The Simulation Environment

Before implementing the above described architecture it is advisable to create a simulation environment that will allow for the system to be tested and its performance assessed. In our case, we clearly need to study the behaviour of the system when specific cases occur, ranging from the reactive cases (e.g. react to an abnormal temperature change) to the more complex ones (e.g. there is no movement in the room where the person is for a long period of time) [8]. This is in fact one of the main advantages of simulation: it enables us to study specific scenarios that can hardly occur, but are still possible, without having to face the consequences or challenges of creating them.

We are therefore using simulation for studying the behaviour of the future environment and improve the architecture, before all the equipment is acquired. Our simulation consists of a premise, the environment around the premise and the user itself. When implementing this, one has to select which parameters will be object of analysis. As in each simulation, choosing which parameters to simulate is a major phase of the development. In this work we decided to simulate all the main parameters that influence the well being of the person inside its environment.

Therefore we identified the key components of our simulation. We will simulate first of all the physical characteristics of the environment, i.e. the rooms and the way they are organized and connected in the house. As for the rooms themselves we had to identify the main characteristics that influence the environment inside walls and therefore the well being of the person. We have chosen as characteristics of the rooms the insulation, the amount of glass, the existence of window blinds and the capacity of the devices in that room to interfere with the environmental parameters (e.g. capacity of air conditioning, capacity of dehumidifier). Other elements such as house orientation or sun exposure could be considered but that would bring too much complexity to the configuration of the simulation.

Following the same line of thought, the external weather conditions were also considered since, in most of the houses, they are the main influence in the inside environmental parameters. To do this we are simulating an outside weather station that generates the values of the weather upon its configuration, according to probability functions that shape the natural evolution of the weather.

Another major player is the user and the actions performed by he/she inside the house. When we are in our homes, doing our regular activities, we are constantly influencing the environmental parameters: if we turn on the oven the temperature in the kitchen rises. We are therefore simulating the user, its movement across the house and the activities being performed at each moment so that the influences f those activities can be noted on the environment.

Talking about the user, another important parameter to simulate are the vital signs. A constant monitoring of the vital signs is essential in an intelligent environment. When simulating them, we can feed the data to the VirtualECare platform and test how it reacts to some specific scenarios. The simulation of the vital signs also takes into account the action being performed as our actions influence our vitals. Having this knowledge about how a given action interferes with the vital signs, we believe that in a later stage of development, we will be able of advising the user not to do some activity due to its influences on his vital signs (e.g. stop exercising if the heartbeat is too high).

At last one more important component of our homes that influences our well being: the actuators. Actuators here can be seen as any device that performs an action on the environment being therefore able of changing it. They are also being simulated and comprehend devices like air conditionings, dehumidifiers, lights, window blinds, among others.

In the whole we will be looking at the temperature, luminosity, movement and humidity and the emergence of fire, flood and gas alarms. In terms of the user, we will be looking at the heart beat rate, the body temperature, the blood pressure or the respiratory rate. All these values will be simulated having in mind the user activities and its context, trying to be as realistic as possible.

The importance of such simulation tools goes beyond the mere test and assessment of the VirtualECare platform. If these simulations reach a full development, they can be used in real case scenarios, to predict what could happen given the actual state of the house or of the user. It will be possible to determine the best and worst that can occur given that state as well as the probability of each one happening,

allowing for real life systems to take better and wiser decisions.

To develop our simulation tool we will be using OSGi and R-OSGi technology. This means that the current architecture and logic organization of the components will shape the real one, fastening and improving the implementation phase.

4.1. The Simulation Architecture

As it was said before, this tool was developed having in mind the final architecture of VirtualECare so it is based on OSGi and R-OSGi standards. It is organized as follows. Each simulated parameter is generated in one OSGi bundle, i.e. the temperature sensors are simulated in one bundle, the luminosity sensors are simulated in another bundle, and so on. The same happens for the house appliances, existing one bundle for each type of appliance.

There is a bundle for the interfaces that allow for an intuitive configuration of the simulation through a Guided User Interface (GUI) or to show its state as it runs. The control bundle is the bundle that starts the simulation. It is responsible for controlling it, requesting values from the sensor bundles and passing them to the interface bundle, updating it. There are, yet, the log bundle, which logs all the events into the database, and a basic alarm bundle which constantly checks the sensors and fires alarms. Using R-OSGi and WS, the simulation platform is connected to the rest of the VirtualECare architecture, then making possible, not only to check its state at any time, but also to make the simulation results available to the GDSS service, as well as to any other service, whenever needed.

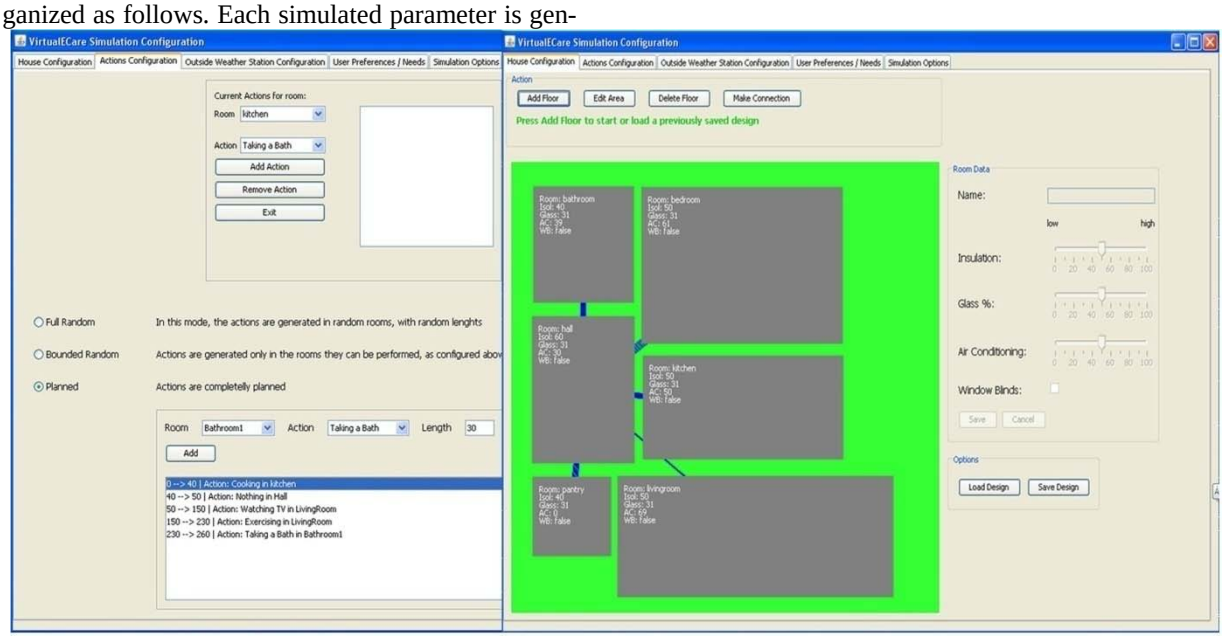

Figure 5: Some Screens of the simulation platform: actions configuration on the left and house drawing on the right.

Therefore, it is possible to have an outline of the premise under observation, as well as a picture of the weather conditions, the vital signs or even the actions to be performed by the human being under observation. On the other hand, one may leave these parameters in random mode and let the simulation flow. All this is detailed in the next sections.

The selection of OSGi and R-OSGi technologies has not been a random act. In fact, the advantages provided can be fully exploited not only by our architecture but by any service based architecture. OSGi allows for any component to be hidden behind a service, thus allowing for a group of modules to intercommunicate and share information. We have successfully built experimental modules on which we connect 1-Wire sensors, a database and a Jade based Multi-agent System, proving that this idea is feasible. This means that when we evolve to the next stage of development, we will not need to do major changes to the architecture, we will only need to change a bundle that randomly generates a value by a bundle that generates that value based on a sensor. The re-

maining bundles that use that value do not need to know where it came from, only need to use it.

These technologies also allow to reduce development time and costs as after having developed a bundle the services it provides can be used by any other, without the need to know how that bundle works or what is behind it, significantly increasing service compositionality.

Having said all this, another major advantage arises from the use of these technologies. Given the modular and distributed nature of OSGi, it allows the bundles to be used by any other OSGi platform at any time. This means that this simulation platform can be used by other applications. The use of the GUI bundles is not mandatory for the simulation to run so other developers could develop their own configuration or running GUIs that use all or just some of the information generated by the simulation bundles.

Let us now detail the specification of the architecture in terms of the OSGi bundles. The simulation platform is made of 15 bundles with the following characteristics:

- Simulator – This bundle is responsible for controlling the simulation and its execution. It updates all the values on the interface bundle and therefore uses the services that provide the values to update the interface (e.g. getTemperature, getHumidity).
- Simulation Interface – The *Simulation Interface* bundle does not use any service from other bundles, it only provides the services needed for the values displayed to be updated as well as for drawing the house and the devices and respective states in a canvas.
- Temperature – The *Temperature* bundle generates the temperature values of the simulated sensors. For doing so it uses services from the *house*, *weather station*, *actions* and *air conditioning* bundles, the ones that influence the generation of the simulated temperature values. It provides as a service the temperature of each of the rooms.
- Dehumidifier – This bundle simulates a dehumidifier and therefore it provides services for turning dehumidifiers on or off as well as for adding dehumidifiers to the house during the simulation configuration process.
- Actions – the *Actions* bundle has as main tasks to generate the actions of the user as configured in the first step of the simulation process. It provides therefore services for setting the settings of the actions simulation. Moreover it provides as services the location of the person, the current activity as well as the characteristics of that same activity (how it influences the vital signs and the environment of the house).
- StartGUI - The *startGUI* bundle does not provide any service. It is however responsible for starting and configuring all the remaining bundles as the user initializes the simulation process. This means that when the user configures the simulation and hits the *start* button, this bundle passes the each value of the simulation to the correct bundles and starts the simulation process.
- Weather – This bundle simulates a 1-Wire outside weather station. It does not use any service from the other bundles however it is used by many of them for generating their own values. It offers as services values of external weather parameters like temperature, wind speed or humidity.
- Vitals – The *Vitals* bundle simulates the vital signs of the monitored user. For doing this it uses services from the *actions* bundle (since actions influence the vital signs of the person). As services, it provides the five main vital signs: respiratory rate, heartbeat rate, systolic pressure, diastolic pressure and body temperature.
- House – This bundle simulates the house. It provides services for adding rooms with specific configurations to the house layout and the connections between them. It also provides as services the characteristics of each room so that other bundles can use them, such as the interface bundle that draws the house.
- Luminosity – The *luminosity* bundle is responsible for simulating the values of the luminosity inside each room, values that are provided as a service. For generating these values the luminosity bundles uses services from the *weather station*, *window blinds* and *lights* bundles.
- Lights – The *lights* bundle simulates the lights of each room. It does not use services from any other bundle but provides as services the functionalities associated to lights control: lightsOn, lightsOff, lightsUp and lightsDown.
- Humidity – This bundle simulates the values of the humidity inside each room of the simulation. For doing this it uses values from the *weather station*, *actions* and *house* bundles.

- AC – This bundle simulates the air conditioning devices in the house. It therefore provides as services the state of each device and the possibility of changing that state.
- Window Blinds – The *Window Blinds* bundle simulates the window blinds distributed across the house. They are added on the first stage of the simulation and provide as services the state of the blinds as well as the functionalities of moving the blinds up or down.
- Reactive – This bundle uses services from most of the other bundles so that it may gather as much information as possible. It then reactively takes some basic actions on the house actuators so that the house environment may meet the preferences or needs of the user.

4.2. Initializing the Simulation Process

The first move is to shape the premise, looking at each room, the filling, the air conditioning, the windows, among others attributes. On the other hand, one may put aside or load a previously saved design, instead of always draw on a new one (Figure 5).

The weather conditions are also object of attention. Under this tab, we may select one of the two running procedures: Random and Planned. Under the Random mode, one selects for each parameter its mean value and the variation that may occur. The values are then randomly generated inside these limits, following a Gaussian distribution. Under the Planned mode, one may set the exact outside environmental parameters at all times.

It is now possible to set user-related parameters. As the final system aims to monitor the user vital signs these should also be simulated to test the inference mechanisms that try to evaluate the health status of the user. By simulating the vital signs, one can cause specific cases to occur and see how and how fast the system reacts to certain vital signs configurations and this way improving these inference mechanisms. When configuring the User Vital Signs two modes are possible: the Random mode and the Planned mode (figure 6).

In the first mode, the user vital signs can be configured to run randomly, inside pre-determined values. When using this mode, one selects the mean value of each vital sign and the variance it may have. The values are then generated using a Gaussian distribution that uses these values as its mean and standard deviation. The decision on using this probability function was due to its performance when used to simulate natural phenomena. As an example, if we configure the heart beat mean to 80 bpm and its variance to 20 bpm, then we would have a simulated heart beat between 60 and 100 bpm, being however a value around 80 much more probable to happen than one around 90.

In the Planned Mode, the vital signs are completely planed and it is possible to determine its exact values at each moment. With this mode we can for example simulate a heart attack for a given time and see how the system reacts to it.

The final values of the vital signs are however not the ones that are generated by the Gaussian functions. The simulation takes these values and modifies them according to the action the user is performing at that time. This means that if we configure the simulation of the heartbeat to a mean value of 80 bpm and the user is exercising, the heartbeat observed in the simulation will be considerably higher during the time the user is exercising. In another hand, if the user is sleeping, it will be slightly lower.

It is important that the system knows which actions influence which vital signs. By knowing that a certain activity influences some vital sign by a certain amount, we can advise the user to stop doing it or even to take another action that leads to a better state of the user. As an example, in the simulation, the action *exercise* increases the heartbeat, the body temperature, the respiratory rate and the room temperature and humidity. Therefore, if the heartbeat is too high and the user is exercising, the system can advise it to stop or even go rest so that its heartbeat can lower. Similar information is associated to each action that is simulated.

Additionally, some more information about the user can be provided. There is the user activity level, already mentioned before, that determines the rate at which new actions are generated when no action is being performed. The user level of richness is used by the system as a factor on the decision making process. When deciding which action to take when the temperature rises in the exterior, the system can decide to lower the window blind or turn on the air conditioning. The most effective action would be to turn on the air conditioning. However, if the user level of richness is low, the system could opt for acting on the blinds. It is at this level that this parameter is used. As for the user needs and preferences, the use is much the same. The user might like a colder environment but the doctor also has a word to say and it might be better for him to be in a hotter environment. When the system acts on the environment, all these factors are weighted to try to find the optim-

al solution that provides both the comfort for the user and the adequate environment. The third factor is the economical one that is, as said before, contained in the user level of richness parameter.

Under the last tab, we may select some control-related parameters such as the simulation speed, the simulation length or log information and save path. The simulation speed can also be controlled when the simulation is running. It is also possible to pause and play the simulation as well as advance it tick by tick.

4.3. Simulating User Actions

In every environment, there must be users for all this to make sense. More than that, the users interact with the system and are, probably, the most unpredictable part of it. A user can change the environmental parameters of the house he is in by using the actuators or through the common action he performs on the house. As an example, if the user decides to take a bath he is increasing the temperature and humidity on the bathroom. The simple fact of interacting with certain devices interferes with the environmental parameters: if the user turns on the oven to cook a meal, the temperature in the kitchen will rise.

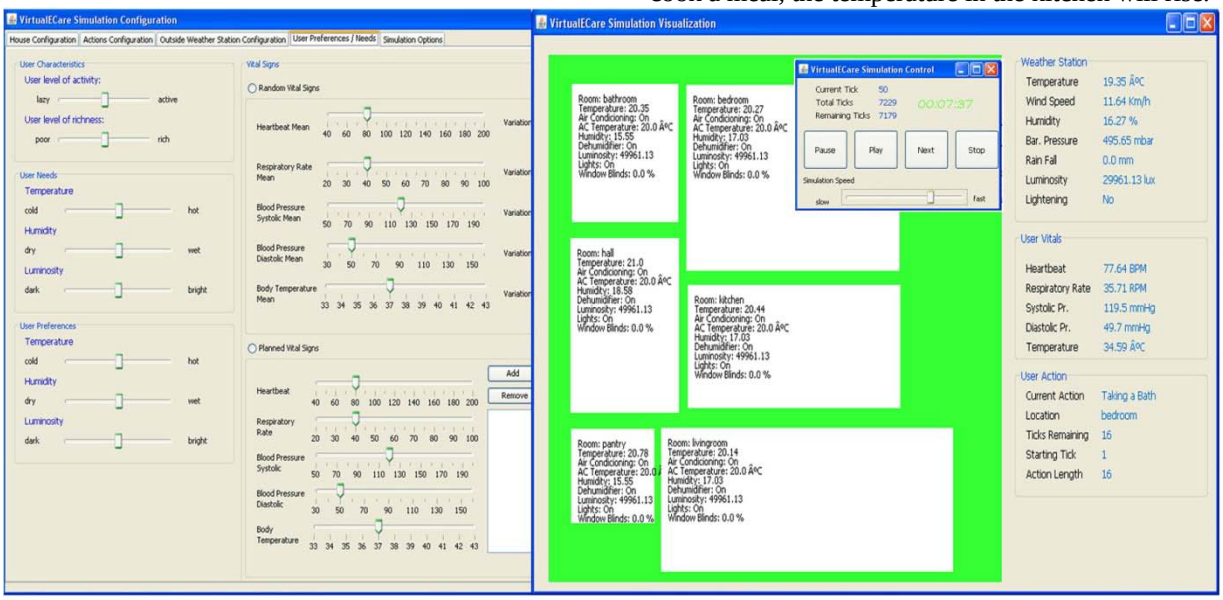

Figure 6: Some screens of the simulation process: the vitals and preferences/needs configuration and the simulation visualization tool with the control menu.

This justifies the importance of simulating the user and its actions inside the house.

In the simulation tool that was created, it is possible to simulate a wide range of common actions that one generally performs inside a house (Figure 5). There are three running modes for the user actions simulation: Full Random, Bounded Random and Planned.

In the Full Random mode, one has no control over the user actions being performed inside the house. The only restriction is the user level of activity which dictates the rate at which new actions are generated when no action is being performed and the maximum length of the action which is 20 ticks. The resulting behavior is actions being performed in random rooms, with random lengths and starting at random tick. In this mode, the oddest things can occur, like the user having a bath in the hall.

In the Bounded Random mode, the actions are generated randomly, with random lengths, just like in the Full Random mode. However, in this mode, it is possible to configure which actions can be performed in which room. This mode enables a more realistic simulation without however having to worry about completely specifying what is going to happen in what tick in which room. When configuring the simulation, we choose this mode and say that in the bathroom the user can only take a bath and make

exercise, then only these actions will be generated for this room, having however random lengths.

The last mode is the one where we can completely specify what we want to happen in the simulation, concerning user actions. In the Planned mode, one can select which action will be performed in which room at each time with what length. This way, it is possible to completely specify the actions of the user during the simulation. As an example we can look at figure 5, where Planned Mode is selected. When the simulation starts the user will be cooking in the kitchen for 40 ticks. After finishing, the user will be in the hall doing nothing during 10 ticks. He will then be watching TV in the living room during 100 ticks and exercising in the same room for 80 ticks. At last, the user will be taking a bath during 30 ticks in the bathroom.

4.4. Simulating Sensors

For generating the data concerning the environmental parameters, there is a group of different sensors that are being simulated. There is an outside virtual weather station that will be later replaced by a real 1-Wire pre-assembled one. This weather station (as the real one) can provide information about the wind speed and direction, temperature, humidity, barometric pressure, rain fall, sunlight intensity and lightening. With this we have access to the knowledge regarding the environment around the house. In the real case, this is useful not only for informing the user but also to be accessed by Group Decision Support Systems, when deciding some recommendation for the client. In fact, all the simulated data can be remotely accessed by external entities, like the VirtualECare GroupDecision module. In the simulation, this weather station has yet another very important role: it is the base of all the simulation. This means that the simulated parameters inside the house depend, in first hand, on the values of the same parameters outside, much like what happens in the real life. The exact way that the external environment affects the inside environment depends then on factors like insulation, house exposure to sunlight, among others.

The temperature and luminosity sensors are distributed along the house to monitor these parameters. They will be replaced by the DS18B20 and D2Photo 1-Wire sensors. The humidity sensors are used mainly in the bathroom, to detect that the user is having a shower and in the rest of the house, to monitor the air humidity. They will be replaced by the TAI8540A 1-Wire Humidity Module. The fire, flood and gas sensors as well as the movement sensor, in other hand, are X10 based. The fire, flood and gas sensors are used to detect dangerous situations that threat the life of the user and the real ones will, respectively be, the PR8307, PR8306 and PR8808. The movement sensor is used to determine in which room the user is in and the real one will be the PR8070.

As for the vital signs of the user, we intend to use the Vital Jacket developed in the University of Aveiro. By now, we are simulating the values of the heart beat, the body temperature, the blood pressure and the respiratory rate. This data is mainly to be used remotely by doctors or other services and to raise alarms in case of dangerous or abnormal readings.

So the question here is now how to generate the values. Should they be random? Should they follow a function? Should it be a mixture of both? We cannot forget that we also want to create specific scenarios. So, we need to find ways to both set the values manually and to let them evolve in a natural way.

As said before, the base of the simulated sensors is the outside weather station which means that the values inside depend on the values outside. The only configurable sensors on this simulation are, therefore, the weather station sensors. The way we choose to implement scenarios uses XML files. Each XML file starts with the address of the simulated sensor and is followed by a list of pairs of the type <tick, value>. Whenever a sensor bundle is started in the scenario mode, it searches for XML files that refer to any of the virtual sensors it controls. Let's say that we are talking about a temperature bundle and it finds a XML file that refers to temperature sensors it controls. The bundle starts counting the seconds since it started and whenever the value of the seconds passed is equal to tick, the value of the simulated sensor becomes value. As an example we can look at Program 1:

```
<?xml version="1.0"?>
<scenario>
     <address>
            1AB2CD44200A1
     </address>
     <events>
            <value tick = "500">
                  20
            </value>
            <value tick = "800">
                  22
            </value>
            <value tick = "2000">
                  24.5
            </value>
```

```
    </events>
</scenario>
```

Program 1: a scenario for a sensor.

This XML file would set the temperature of the sensor to 20 degrees 500 ticks after starting the bundle, to 22 degrees after 1000 and to 24.5 degrees 1500 ticks after starting the bundle, remaining like that until the bundle is stopped.

The same methodology is used for any of the other sensors being the only difference the values (that must be according to the type of sensor). Now the second case, when we don't want to set up static scenarios. Instead, we want to let the values flow and see what happens. In this mode, the values are generated using a Gaussian distribution. This distribution was selected due to its ability to shape natural phenomena. So, when configuring the simulation, in the "Outside Weather Station Configuration" tab, one selects for each simulated parameter the mean value and the variation that may occur and that will define the weather that will occur during the simulation. It is therefore possible to create a more stable weather or, in another hand, to create a weather configuration that can change rapidly and unexpectedly.

Let us see now how the values inside the house are generated. The first step is to identify the main factors that stand between the outside environment and the inside environment in our homes. These factors were already mentioned on the previous section and are the factors used when configuring each room (e.g. insulation, glass percentage, etc) so the values generated depend on these factors. There are two additional factors that were used when generating some of the parameters: the previous values and the number of adjacent rooms. We realized, when developing the simulation tool, that the temperature inside the house would change immediately after the state of the air conditioning changed, which does not corresponds to the reality. In our homes, when we change the state of the air conditioning, the environment takes a while to change and the factor time is used to add some delay to parameters like temperature or humidity. Luminosity, in another hand, changes instantly so the factor time is not necessary. As for the number of adjacent rooms, it is known that the environment in a room is affected by the environment in the rooms that share a common door and, in another hand, it also affects the environment in all those rooms so this is another factor to consider. As the formula that generates the values is too complex to be displayed here, the parameters it uses are depicted here:

- prevTemp – The previous value of the temperature in the room;
- airCondCapacity – The capacity of the air conditioning provided as a parameter on the environment configuration;
- insulation – The insulation of the room provided as a parameter on the environment configuration;
- airCondTemp – The temperature of the air conditioning on the room provided as a service by the actuators bundle.
- actionTemp – The temperature resulting from the action that the user is currently executing;
- consTemp – The contribution of the connecting rooms to the temperature of this room.

The simulation of the effects of the outside weather in different types of room is achieved by assigning different characteristics to the rooms.
As another example, we can look at the luminosity. In this simulation, the luminosity depends on three factors: the outside luminosity, the position of the window blinds and the fact of the lights being on or off. The values of the outside luminosity, in its normal mode, are also generated by a Gaussian function and are provided by the weather station bundle. The state of the lights and the window blinds is a service provided by the X10 bundle. So, if the lights are on, the luminosity level is always high. Otherwise, it depends on the existence of windows and, if they exist, the position of the window blinds. If they are down or no windows exist, the luminosity is low. The luminosity can be seen on Program 2.

```
if (hasWB)
{
 if (LightsService.isLightOn(room))
    lum =
(WBPos/100)*WeatherService.getLum(tick) +
lightsIntensity;
 else
    lum =
(WBPos/100)*WeatherService.getLum(tick);
}
else
{
 if (LightsService.isLightOn(room))
    lum = WeatherService.getLum(tick) +
lightsIntensity;
 else
    lum = WeatherService.getLum(tick);
}
```

Program 2: the function for the luminosity simulation.

where wbposition is the percentage of the position of the window blind (e.g. a window blind that is at the middle of the window corresponds to wbposition = 0.5). Similar processes occur for the rest of the sensors. While the weather station is ruled by Gaussian functions that can be configured when the simulation starts, the values of sensors inside the house depend on those external values and the other factors mentioned.

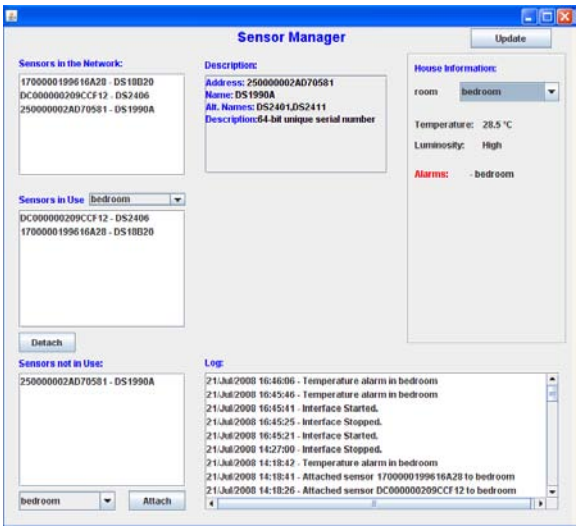

Figure 7: The Sensor Manager Interface

In order to help to understand its behaviour and how it interacts with the simulation platform, we have designed a few interfaces. In Figure 6 we can see the interface which shows the values of the sensors and alarms in the premise. It also shows the vital signs of the user and the values of the outside weather station.

In figure 7, we can see the interface of the Sensor Manager. It allows for sensors to be added or removed, to be paired with specific rooms and be moved between rooms. It also shows the information of each sensor (e.g. address, characteristics, location, values) as well as the environmental parameters of each room, the rooms which have sensors in alarming state and a log with all the historic information regarding sensors.

4.5. Simulating Actuators

The last important part of the simulation is the actuators. With what has been described until now, we have the ability to simulate the generation of values. With this last section, we gain the ability to change the way these values are generated, in a natural way. These values are generated, among other factors, according to the state of the virtual appliances. It is by changing the state of these appliances that we can interfere with the simulated environmental parameters.

The devices that are simulated include air conditioning, dehumidifier, window blinds, among others. Each one of the simulated appliances offers as services the ones that we expect to find in the real world appliance. Therefore, the virtual air conditioning module allows for its temperature to be set higher or lower as well as turning it on or off. The dehumidifier allows for it to be turned on or off and the window blinds allow to move up or down.

The effects of these actions are reflected in the environment and visible on the simulation, although the time to notice the variations may vary, depending on the parameter affected (e.g. a change in a window blind position would immediately affect the luminosity in the correspondent room while a change in the air conditioning temperature would take a while to show its full effect). The actions on the devices are visible not only on the effects on the environmental parameters as well as on the representation of the appliances on the simulation as the state of the appliances is depicted on the simulation interface (figure 6).

5. Results

The above presented simulation tool made possible the testing and assessment of the majority of the platform building components. This was, indeed, our initial main objective and has revealed itself a crucial step in our path to the implementation of the final model. Through simulation, we were able to study the behaviour of all the system modules before their deployment in a real and, consequently, risky environment thus increasing our confidence and allowed us to achieve the first validation of the presented system. It also highlighted the first flaws, making adjustments and testing a possibility.

The simulation tool however, has been far more useful than we initially expected or intended it to be. More than a mere tool to test the communication mechanisms and the behaviour of our platform, this work revealed itself interesting for other applications. The generated time series configure a rich growing bank of values concerning the most varied data, ranging from house temperatures or luminosity to user

behaviour or vital data such as heartbeat rate or blood pressure. This data, associated to the scenario in which it was generated, configures a useful input to other projects in related fields such as domotics or health.

Due to the technologies used in the development of the architecture, we arrived to an easy and modular fashion of prototyping an intelligent environment. Therefore, it became easy to gradually replace simulated entities with real ones, starting from the point of a totally virtual environment, passing through a mixed one with some entities being simulated and others being real ones in the path to build the first real prototype. This methodology for developing intelligent environments has revealed to be a highly modular one, ideally suited for conducting tests that address each individual component, a group of components or the whole architecture.

More than that, the simulation tool revealed itself important for the project team to develop and test a wide range of Ambient Assisted Living oriented services before its actual implementation. One of the most important and complex services uses an inference engine in order to infer rules about how the user manages his environment. These rules are then used by a Case-based Reasoning model in order to manage the environment in a completely autonomous and proactive fashion [8, 11]. The services developed are now being implemented and tested in the real environment of a Portuguese Northern major hospital.

6. Conclusion

In order to provide an on time dignified and specialized healthcare to all the population in general, and to the elderly in particular, a new approach to the problem had to be devised in terms of the services to be provided, without delocalizing the incumbents or messing up with their routines, in a more effective and intelligent way.

The VirtualECare platform will keep growing with more improvements. A more evolved context aware module is being developed which uses an inference mechanism and logical predicates to try to obtain exact information regarding the context, using the information obtained from the sensors, and this way being able of inferring states. This information will then be used to proactively assist the user. In order to do that, the services description will have to be enriched with more semantic information so that they are selected in a more efficient way, according to factors like availability, location or costs versus utility and to the exact context of the user.

Therefore, our goal in this work leads us to idealize a platform and its supporting architecture, to hold on the VirtualECare project. More than that we created a simulation platform that can be used later when the project is fully developed, to predict the evolution of determinate cases and to determine the best and worst case scenarios, given the current state of the house or user. As this platform is based on open and highly modular technologies, it could easily be used by other similar health care or medical projects, which need to simulate the evolution of an assisted living environment.

References

[1] Abowd, G., Bobick, I., Essa, E. and Mynatt, W. “The aware home: Developing technologies for successful aging”. Proceedings of AAAI Workshop and Automation as a Care Giver. July 2002.

[2] Aguilar, J., Cantos, J., Expósito, G., and Gómez, P. “Tele-assistance Services to Improve the Quality of Life for Elderly Patients and their Relatives: The Tele-CARE Approach”. In The Journal on Information Technology in Healthcare 2, (2), pp. 109-117, 2004.

[3] Alamán, X., Ballesteros, F., Bravo, J., Fernández, D. Ambient Intelligence at Home: Facts and Future. In: Cepis Upgrade: Ambient Intelligence. Vol. VIII, issue 4. Novatica.

[4] Augusto, JC., Nugentm CD. 2006. Designing Smart Homes: the role of Artificial Intelligence. Springer, 2007.

[5] Augusto, C., and McCullagh, P. Ambient Intelligence: Concepts and Applications. Invited Paper by the International Journal on Computer Science and Information Systems, volume 4, Number 1, pp. 1-28, June 2007.

[6] Brown, S.J.: ‘Next generation telecare and its roles in primary and community care’, Health and social care in the community, 2003, 11, (6), pp. 459-462

[7] Camarinha-Matos, L.M., Castolo, O., and Rosas, J. “A multi-agent based platform for virtual communities in elderly care”. In A multi-agent based platform for virtual communities in elderly care, 2003.

[8] Carneiro, D., Costa, R., Novais, P., Neves, J., Machado, J., Neves, J., Simulating and Monitoring Ambient Assisted Living, in Proceedings of the ESM 2008 - The 22nd annual European Simulation and Modelling Conference, Le Havre, France, 2008.

[9] Cook, D. “Health monitoring and assistance to support aging in place”. JUCS, 12(1):15–29, 2006.

[10] Corchado, J. M. , Bajo, J. , Paz, Y., Tapia, D. “Intelligent Environment for Monitoring Alzheimer Patients”. Agent Technology for Health Care. Decision Support Systems, 44 (2) 382-396, 2008.

[11] Corchado, J.M., Bajo, J., Paz, Y. “A CBR System: The Core of an Ambient Intelligence Health Care Application”. Soft Computing Applications in Industry 311-330, 2008.

[12] Costa, R., Novais, P., Neves, J., Marreiros, G., Ramos, C., Neves, J. “VirtualECare: Group Decision Supported by Idea Generation and Argumentation.” in Pervasive Collaborative Networks, Luís Camarinha-Matos and Willy Picard (eds),

Springer-Verlag, Series: IFIP International Federation for Information Processing, ISBN 978-0-387-84836-5, 2008, pp 293-300.

[13] Dosi, G. "Sources, procedures and microeconomics effects of innovation". In Economic Literature 26, 1120-1171.

[14] Fraile, J.A., Bajo, J., Abraham, A., Corchado, J.M. "HoCaMA: Home Care Hybrid Multiagent Architecture," Pervasive Computing, 2009, pp. 259-285.

[15] Helal A., Mann W., Elzabadani H., King J., Kaddourah Y., and Jansen E. "Gator tech smart house: A programmable pervasive space". IEEE Computer magazine, pages 64–74, March 2005.

[16] Hinchley., A. "Understanding Version 3 - A Primer on the HL7 Version 3 Interoperability Standard". 4th edition, ISBN 3-933819-21-0, 2007.

[17] Holmlid, S. and Björklind, A. "Ambient Intelligence to Go". AmIGo White Paper on mobile intelligent ambience. 2003.

[18] IBM Global Business Services. "Healthcare 2015: Win-win or lose-lose?", 206.

[19] IST Advisory Group: Ambient Intelligence: from vision to reality, 2005.

[20] IST Advisory Group: Scenarios for Ambient Intelligence in 2010; Final Report, 2001.

[21] K. Chen and L. Gong. "Programming Open Service Gateways with Java Embedded Server(TM) Technology". Prentice Hall PTR, 2001.

[22] Malmberg, A. "Industrial Geography: agglomeration and local milieu". In Progress in Human Geography, vol. 20, 392-403, 1996.

[23] Monson-Haefel, R. "J2EE Web Services: XML SOAP WSDL UDDI WS-I JAX-RPC JAXR SAAJ JAXP". ISBN-10: 0321146182, 2003.

[24] Nehmer, J., Becker, M., Karshmer, A., and Lamm, R.: 'Living assistance systems: an ambient intelligence approach', in Editor (Ed.)^(Eds.) 2006. 'Book Living assistance systems: an ambient intelligence approach', pp. 43-50

[25] Neves, J. "A Logic Interpreter to Handle Time and Negation in Logic Data Bases". In ACM (ed.): The Fifth Generation Challenge 50-54, 1984.

[26] Novais, P., Costa, R., Carneiro, D., Machado, J., Lima, L., Neves, J., 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, ISBN 978-0-387-85690-2, 2008, pp 353-362.

[27] Osgi Alliance. "OSGi Service Platform: The OSGi Alliance", 2003.

[28] Pigot, H., Mayers, A., Giroux, S., Lefebvre, B. and Noury, V. Rialle."Smart house for frail and cognitive impaired elders". UbiCog '02: First International Workshop on Ubiquitous Computing for Cognitive Aids, Göteborg, Sweden, Sept. 2002.

[29] Ramos, C., Augusto, J.C., Shapiro, D. "Ambient Intelligence—the Next Step for Artificial Intelligence". IEEE Intelligent Systems 23(2): 15-18, 2008.

[30] Riva, G. "Ambient Intelligence in Health Care". In Cyberpsychology & Behavior, vol. 6, 295 – 301, 2003.

[31] Tang, P., Venables, T.: Smart homes and telecare for independent living Journal of Telemedicine and Telecare 2000 6: 8-14, 2000.

[32] Ohta, S., Nakamoto, H., Shinagawa, Y., Tanikawa, T. A health monitoring system for elderly people living alone, Journal of telemedicine and telecare, vol. 8, nº3, pp. 151-156, 2002.

[33] Wooldrige, M., An Introduction to Multiagent Systems, John Wiley & Sons, 2002.