A 3D Simulation Framework for Safe Ambient-Assisted Home Care

Carlos Velasquez, Christophe Soares
INESC Porto and UFP
University Fernando Pessoa
Porto, Portugal
{*carlosv,csoares*}*@ufp.edu.pt*

Ricardo Morla
INESC Porto, FEUP
University of Porto
Porto, Portugal
rmorla@inescporto.pt

Rui S. Moreira, José M. Torres, Pedro Sobral
ISUS Unit @ FCT
University Fernando Pessoa
Porto, Portugal
{*rmoreira, jtorres, pmsobral*}*@ufp.edu.pt*

***Abstract*—The Safe Home Care project focuses on assembling safe home assisted-living environments built on autonomous Off-The-Shelf systems. We argue that these smart spaces will contribute to relieve the pressure on health systems by providing the means for ambulatory and daily life assistance. However, the integration of disparate sovereign systems will not be easy to accomplish since the number of interaction scenarios will be impossible to predict and evaluate a priori. Therefore, we propose the SHC reflective middleware framework, conceived with two goals in mind: i) manage the safe integration of off-the-shelf systems (cf. interference-free) by exploiting reflection and 3D virtual world simulation; ii) provide non-intrusive pervasive interface mechanisms for home assisted-living actors. In this paper, we focus specifically on the first goal by providing the means for generating 3D simulations; the states generated during simulation are then analyzed by a graph-pruning algorithm to perceive feature interactions in pre-deployment phases. We evaluate our approach on specific home care use cases.**

***Keywords*-3D simulation, safe home care, interference-free, reflective middleware, graph-based interference pruning.**

I. Introduction

The world population is getting older. Home care plays an important role in elderly care as it can be integrated in daily routines without much disruption and target preventive services to those with higher risk factors, potentially decreasing health care costs [1]. Numerous technologies exist and have been proposed for home care [2], often acquirable as off-the-shelf (OTS) components that are independent of other components in the home. Functional interactions between these components are difficult to predict a priori and may result in beneficial or detrimental behavior of the home as a whole. We are particularly interested in capturing functional interferences between OTS systems (e.g., two or more systems requiring simultaneous user attention, temperature being adjusted differently by two systems, etc.) but also interferences that may be due to physical features (e.g., location, sound and electromagnetic interferences). The SHC system aims to address such interferences between components in the home (cf. feature interactions), i.e., represent and detect interacting features and handle consequent malfunctions or miss behaviors.

The general approach of the Safe Home Care (SHC) project [3] is to capture the behavior of the home and its components and match it against the normal or expected behavior by pruning states in an observed sequence of states. The observed sequence of states of the home and its components can be captured directly from the home by introspection through sensors and management interfaces, or generated by simulation. The simulation-based approach has the potential for exploring more scenarios, which in turn may trigger the detection of additional interactions. Additionally, simulation can be used to anticipate potential interactions before deploying the new components in a home. In this paper we present a framework for simulation and detection of interactions between components in a home care scenario. This simulation framework is integrated in our SHC system providing introspection and interference detection, which we present in Section 2. We present the requirements, architecture, and working modes of the simulation framework in Section 3, and evaluate it with a specific home care scenario in Section 4. We conclude with a review of the related work in Section 5 and with final remarks in Section 6.

II. The SHC System

A. Goals

The next generation of home smart spaces will be crammed with diverse OTS system that may be independently assembled. Therefore, it is imperative to be able to assess their compatibility and detect possible interferences, before deployment and to dynamically monitor and manage interactions between these systems, after deployment. The SHC system is being built with these goals in mind, i.e., i) provide a simulation environment to exhaustively explore different interaction scenarios between the OTS systems and ii) reify real-time state information that can be used to monitor and safely adapt home environments. In this paper we focus on the simulation aspects that may able us to recreate critical or unpredictable OTS systems interactions and, thus, evaluate if these systems can coexist or if additional management components should be incorporated to resolve identified interferences.

B. Overview

The system architecture is organized in two major levels, Base and Meta-Level, connected through a reflective Mid-

 ISBN: 978-1-61208-171-7

dleware layer. The Base-level comprehends all the software and hardware necessary to interact with the physical environment. The Meta-level is composed essentially by a 3D Virtual Environment and a relational database. The 3D virtual environment, implemented in OpenSim, is the primary interface with the SHC system; the database, implemented in MySQL, is the repository of all the information reified from our System. The Reflective Middleware layer provides the connection glue between both levels and offers: i) a management interface (developed in PHP); ii) an interference engine (developed in Java), and iii) a simulation framework, which incorporates a C# .NET component to control/interact with the Avatar, a series of PHP scripts for scheduling the simulation and several LSL scripts (Linden Script Language) for programming the behavior of the different simulation elements.

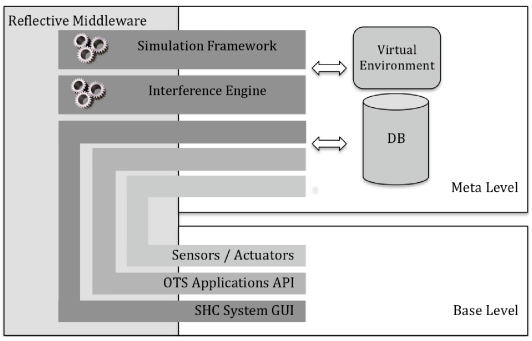

Figure 1. System Architecture

C. Simulation

The simulation framework makes use of OpenSim, a free version of the Second Life Simulator. This is a general-purpose programmable 3D platform, which offers intrinsic 3D modeling characteristics (e.g., notion of space, volume, time, behavior, etc.). For example, the sound propagation may be attenuated or blocked by a wall; thus communication between objects may be affected by their coordinates/location. Hence, the recreation of the home environment becomes easier and physically analogous to reality. OpenSim offers also a scripting language (cf. LSL), which permits to associate behavior to simulation objects (cf. Prims or Primitives). These building blocks can be re-shaped, re-colored and programmed to respond to events (e.g., touch, listen, etc.). Finally, by providing an open source framework unbolts and stimulates future contributions from the research community. Another advantage offered by the proposed simulation framework is the possibility to fully simulate a Human being, either by using an autonomous-deterministic agent (upcoming, a more proactive-intelligent agent) or a user-controlled agent. The former follows a specific scheduled scenario, testing as many predicted interactions as possible; the later, permits us to exploit unpredictable reactions of human controlled avatars (cf. virtual human representations) thus introducing variability in the simulation scenario. Finally, the proposed simulation framework integrates with the the SHC interference engine that analyses the observed states of OTS systems to detect unpredictable behavior. These states may be introspected both from the base-level components or generated by the simulation framework.

D. Interference

Our approach for interference detection relies on a graph representation of system state sequences derived from the interaction between residents and OTS systems. Directed graphs were used to represent: i) the expected behavior of isolated systems (Graph of Expected States - GoES), i.e., all state sequences resulting from the regular operation of each system; ii) the observed behavior of combined systems (Graph of Observed States - GoOS), i.e., the actual state history of all elements built via runtime introspection.

The GoES represents how applications should behave (i.e., without interference) while the GoOS represents what is the current/observed systems behavior. An algorithm is used to extract the expected behavior from the observed behavior (cf. State Pruning Algorithm – SPA [3]). If the SPA ends up without being able to prune every state sequence in GoOS then it assumes one of two things: i) there are interferences or feature interactions that should be handled; ii) there are state sequences or malfunctions not captured in the existing GoES that should be considered.

This paper proposes a simulation platform that collects state changes from different kinds of systems (e.g., entertainment, communication, health related devices). Using the SPA on this large state database it is possible to expose unforeseen interactions between applications that otherwise are hard to notice.

III. Simulation Framework

A. Requirements

The simulation framework has four major requirements. First, like other general purpose frameworks, it should be possible to integrate real world elements to generate actions/events and stimulate virtual world representatives; such causal connection will enable us to test the integration of new OTS systems alongside with existing/deployed systems; this connection though cannot be reflected back on the real world since the new simulated prims do not have their counterpart real objects (e.g., a simulated air conditioning prim cannot change the temperature in the real environment). Second, the framework should provide the means to facilitate the creation of simulations; this is achieved by the use of pre-programmed prim inventories; such prims coupled with a set of base LSL scripts may reduce the time to shape the appearance and behavior of simulation scenarios. Third, the framework should provide the means to select/use different types of simulation, i.e., deterministic, intelligent

 ISBN: 978-1-61208-171-7

and user-driven simulation; the deterministic simulation follows/triggers a scheduled set of events for a given scenario and permits to analyze the pre-programmed reactions of surrounding prims; the user-driven simulation permits to introduce some variability in the simulation environment through the use of an Avatar directly controlled by a human; the intelligent simulation also follows a given schedule but using prims programmed with probabilistic action/reaction models. Fourth, the simulation framework should provide several levels of introspection, i.e., enable to select/unselect the type of information that may be collected from base-level (e.g., select which sensor type systems or APIs we may use); this will influence the amount of data generated during simulation, and will allow us to adapt the simulation to the level of introspection facilities that we may have of the environment. The simulation framework is fully integrated in the SHC Reflective Middleware. This allows its use for detecting interferences between deployed and new OTS systems, thus proving a powerful tool for the integration of ubiquitous systems.

B. Architecture

The SHC Middleware uses a deterministic manager component for driving simulations based on pre-scheduled events and pre-programmed prim's behavior. This manager uses two different components: the scheduler, which is a PHP/MySQL component that triggers events on prims (e.g., VoIP call, Drug Dispenser alarm, etc.); the client agent, a C#/ OpenMetaverse component, that permits to control the Avatar via scheduled events, just like a prim, replicating programmed user activity (cf. Full Simulation mode) [4] It is also possible, for a human, to directly control an Avatar via any Second Life Viewer (cf. Semi Simulation mode). A 3D scenario uses several prims, one for each OTS system. Prims have their own scripts, which allow to individually program and customize different actions and reactions. We use a Master Control prim that serves as a communication proxy between the base-level elements and their counterpart 3D meta-level representations. This unique prim permits to use only one HTTP connection between the base and meta-levels, thus simplifying the scripts on the other prims.

All prims listen for commands sent by the Master Control prim and act according to their programmed behavior. For example, if the Master Control sends the message "Phone Ring" then the phone prim reacts by changing its state to "Ringing". Every state change is registered in the SHC database. The Master Control may also be used by the SHC middleware to reflect prim state changes back into their base-level originals. The simulation may be driven by pre-scheduled events that trigger state changes on specific prims. These events will propagate state changes throughout other prims that will react according to pre-programmed scripts (cf. deterministic control). To increase variability the simulation framework offers also the possibility for a human controlled prim to drive the simulation, i.e., to trigger event changes through direct ad hoc interactions with prims. In addition, base-level elements may also trigger state changes in their reified prims, increasing again simulation variability. Next sub-sections will detail these two types of simulation supported by SHC.

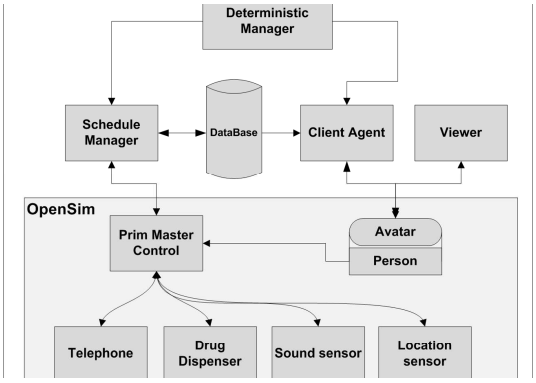

Figure 2. Simulation Architecture

C. Full Simulation

All prims have scripts, which represent pre-programmed deterministic reactions to certain events. The Schedule Manager uses the Master Control prim for conveying events to these prims. Each prim listens for given channels, thus facilitating different types of communications (e.g., sound, network, etc.). The Client Agent uses a login to control an Avatar (cf. moveto(), touch(), etc) and, through it, interact with the environment or other prims. For instance, if the telephone starts ringing and the Avatar hears it then, it will move toward the phone prim for answering the call (touching it).

D. Semi Simulation

A pre-ordered set of events may be too limited for generating enough realistic prim interactions and thus influence our ability to detect all possible interferences. To increase more randomness in the simulation, a human may control the Avatar to generate ad hoc interactions. Reifying real-time state changes, on OTS systems, into their prim representatives will have the same effect on generating unpredictable interactions and state changes.

The Avatar may be controlled through a Second Life viewer or the Client Agent (e.g., move, touch, etc.). The Client Agent uses a prim attached to the Avatar to be enable to indirectly control it. This prim is also used to interact with the environment, simulating the Persona features (e.g., hearing certain events).

IV. Application Scenario and Simulation

A. Scenario

The particular scenario to which we applied our simulation represents an excerpt of the daily routine of Maria,

 ISBN: 978-1-61208-171-7

our elderly persona. Like every morning, Maria watches her favorite TV show and, at 10:30 AM, the Drug Dispenser (DD) triggers the alarm light and buzzer reminding her it is time to take medication. Unlike every morning, however, this time the phone rings just after the dispenser alarm is triggered. She answers the phone and spends more time talking to her friend than the DD alarm timeout, which disengages the light and buzzer. According to the DD behavior, it will then send a notification to the server, indicating that a pill was not taken.

Figure 3. 3D home scenario with Maria, DD and VoIP interactions.

The depicted scenario includes 2 different OTS systems (cf. VoIP Phone and DD) and a Persona. These systems were inserted into a virtual environment in order to simulate their interactions.

B. Simulation

This simulation model has two OTS systems plus the User Avatar. It is the correlation between the states of these three elements that our simulation will explore. As the scenario says, at 10:30 AM the DD alarm should sound, this involves a couple of steps. The DD API, simulated as a prim inside the OpenSim, checks every 30 minutes for the need of a pill intake, this is achieved through the following steps: first, the API sends a message to the Master Control Prim, using the virtual channels for communication, requesting a database check at the requested time (10:30 AM); second, the Master Control Prim receives the message and, through an HTTP request method, contacts a PHP server, using the information that the message brought, more accurately the day and time needed. So far all is done in the virtual environment. Third, the PHP server connects with the MySQL database using a PHP method for the query. Fourth, once the query returns the result, the server uses the connector of the Master Control Prim and reconnects with the virtual environment returning the result of the query. The connector is actually a URL that can be used as an HTTP Socket but with the difference that it connects directly with a specific prim inside the virtual environment. Fifth, the Master Control Prim receives the HTTP request and replies to the DD API the result; in this case the result is DDRING to start ringing. This way the DD API sends the result message to the DD Prim, that starts Ringing, which means sending the message “DD Ringing” in the channel chosen as environment sound, and changes its color and brightness to simulate the blinking. All these steps are performed in milliseconds, which means that so far the usage of the Master Control Prim is not a bottleneck. The reason to use the DD API inside the OpenSim was the clock. This way, if there is the need of speeding up the clock for decreasing the total simulation time, the only clock needing tampering will be the simulator clock.

So far the DD is ringing, but a minute later the phone starts Ringing as well. To test a different, more direct way of communication, instead of a scheduled call, it was created a phone like page to simulate a call directly from a web page. With the scheduled DD API our approach functions without the need for a user, but for the Phone a user as to simulate the call, so at 10:31 AM the user inserts in the web page a request for a call. This request is a simple button that uses the Master Control connector just as before, and sends the message “PhoneRing” to the virtual environment. Inside the virtual environment the Master Control Prim sends the received message, using the communication channels, to the Phone API Prim, named Asterisk. This API, in a way similar to the DD API, communicates with the Phone telling it to Ring. The ringing event consists in sending a message “PhoneRing” to the environment sound channel and changing the phone color simulating the light in the visor. As of now, both OTS systems are Ringing. According to the planned scenario the Persona will answer the Phone before the DD. But for this the Avatar needs to approach the phone prim and touch it, simulating the answering state. At 11 AM the DD gives a timeout and stops ringing. As mentioned, there are two alternative avatar simulations, Full and Semi Simulation. In Full Simulation, our approach uses the Openmetaverse Library (LIBOMV) in order to create a client agent, login the Avatar in the virtual environment and use a schedule table to perform human like actions. In our framework, after the login process has been achieved, the avatar will be waiting for orders. In reality the code serves to create a BOT (cf. Robot), normally used in video games to create Artificial Intelligence characters; ours react to commands like moveto and touch. For example, the command moveto(x,y,z) is used to send the avatar to the coordinates of the phone and the command touch(prim id) is used to answer the phone (both the coordinates and id are stored in the MySQL database). The connection to database is made through the MySQL Library for .NET. In Semi Simulation, the User will login in the VR, through a viewer, and may control the avatar. The interaction with other prims will be direct through “touch” or indirectly by walking in a zone with sensors. To answer the phone, the user must move to the Phone Prim and then touch it. The Phone Prim, when touched, uses the same communication process to send a message to the Phone API saying it was

 ISBN: 978-1-61208-171-7

answered. The Phone API sends a message saying the same to the Master Control Prim that will use the same process as before, communicate with the server indicating the new phone state. The server will record this state in the database. In both types of simulation, the Avatar has an attached Prim that will send a message directly to the Master Control Prim indicating the states of the Avatar (something like a human API). In our example, the Avatar answers the Phone and the system waits for the next interaction that will be the timeout triggered by the DD API. This time the API needs not to check the schedule, and sends two messages: first to the DD prim, saying StopRinging that will make it change to its original color and send a message to the environment saying "low" (meaning low noise); second to the Master Control with the timeout notification. The Master Control will send the message to the server creating the new state.

Finally, the User will touch the Phone again, indicating that the call is over. Every time a state is changed in either of the prims the respective API (DD API, Phone API, or Avatar API) will inform the Master Control Prim, which will be responsible for sending the information to be recorded in the server database (cf. recorded states in Table II).

Table I
COMMUNICATIONS IN THE SIMULATION SYSTEM

Source	Destination	Message
Master Control	DD API	DD On
DD API	DD	DD On
DD	Environment	DD Ringing
Environment	Person	DD Ringing

Table II
SUBSET OF THE STATE TABLE ON SHC DATABASE

Element	Feature	Value	Timestamp	Source	Type
DD	Alarm	ON	10:30 AM	DD	Out
DD	Ringing	ON	10:30 AM	DD	In
Person	Needs Pill	ON	10:30 AM	Person	In
Phone	Call In	ON	10:31 AM	Phone	In
Phone	Ringing	ON	10:31 AM	Phone	In
Person	Receives Call	ON	10:31 AM	Person	In
Phone	Call	ON	10:32 AM	Phone	In
Person	Answers Call	ON	10:32 AM	Person	In
Phone	Ringing	OFF	10:32 AM	Phone	In
DD	Take Pill	OFF	11:00 AM	DD	In
DD	Notify	ON	11:00 AM	DD	Out
DD	Alarm	OFF	11:00 AM	DD	Out
DD	Ringing	OFF	11:00 AM	DD	In
Phone	Call	OFF	11:05 AM	Phone	In

C. Interference

The logical operation of the interference detection has three steps: i) create the GoOS – read state values from the SHC database and assemble a GoOS based on the known expected states; ii) prune GoOS – based on the GoES, remove correct state sequences from GoOS; and iii) interpret results – from the state sequences left in the graph try to identify the interference source. This last phase is currently under work; for now our focus is on interference detection without identifying the causality.

Table III
INTROSPECTED ENVIRONMENT INFORMATION

Case	Element	Feature	Value	Type	Kind
A	DD	Alarm	ON	OUT	APP
B	DD	Ringing	ON	IN	APP
C	DD	Take Pill	ON	IN	APP
D	DD	Take Pill	OFF	IN	API
E	DD	Alarm	OFF	OUT	APP
F	DD	Ringing	OFF	IN	APP
G	DD	Low Drug	ON	IN	API
H	DD	Notify	ON	OUT	API/APP
I	DD	Low Battery	ON	IN	API
J	DD	Upside Down	ON	IN	API
K	Phone	Call In	ON	OUT	API/APP
L	Phone	Ringing	ON	IN	APP
M	Phone	Call	ON	IN	API
N	Phone	Call	OFF	IN	API
O	Phone	Call In	OFF	OUT	API/APP
P	Phone	Ringing	OFF	IN	APP
Q	User	Needs Pill	ON	IN	APP
R	User	Receives Call	ON	IN	APP
S	User	Take Pill	ON	IN	APP
T	User	Take Call	ON	IN	APP

Table II represents a subset of the "state" table taken from 10:30 AM to 11:05 AM. This table results from the scenario presented in IV-A. In order to look for interference a GoOS is built from the database records (see Figure 4a). This is achieved through a matching table (Table III) where each case corresponds to an expected state. The state is characterized by the component of the system (Element), the activity (Feature, Value and Type) and the introspection source (APP, API or both). Next, the Pruning Algorithm uses the GoES (see Figure 4b) to remove correct sequence states from the GoOS. In this example, paths <A,B,D,H,E,F>, <R ,T> and <K,L,H,P,N> are removed. However, the SPA returns the <Q> state since state <S> was not observed, successfully identifying the interference, i.e., Maria does not take her medicine as described.

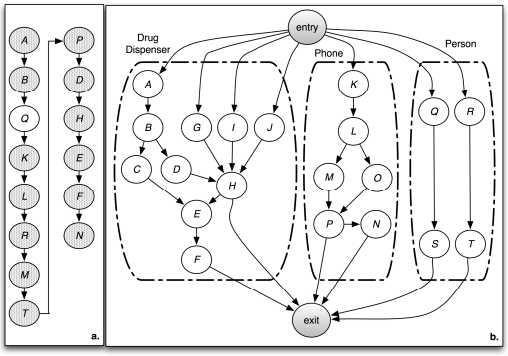

Figure 4. Graph of observed state - GoOS

V. Analysis of the Related Work

There are several simulation frameworks focusing specifically on 3D representations of ubiquitous computing environments. The UbiWise [5], from HP Labs, uses the Quake III Arena graphics engine and offers two clients: UbiSim (a 3D view of the virtual environment) and Wise (a close-up view of devices that users may manipulate). DiaSim [6] is a simulator for pervasive computing applications, coping with widely heterogeneous entities. UbiREAL [7] provides features to simplify the layout of 3D scenarios and simulate communications (from MAC to Application layers). The SHSim [8] is an OSGI-based Smart Home Simulator, which offers system configuration facilities and provides device transparent simulation. Only the last framework offers a real transparent connection between the real-world and the simulation environment, but both real and virtual devices must be OSGI compliant. A transverse difficulty in this area is the integration of OTS devices. Most of these systems offer a black box design (i.e., closing its internal architecture and behavior) and proprietary technology (e.g., communication protocols, programming languages, etc.), which poses difficulties to simulation. Another important limitation of theses frameworks is the lack of representation for human activities and interactions. Actually, in [9] a simulation model for self-adaptive applications, proposes two distinct representations: i) a high-level model describing the activities of inhabitants (e.g., take pill, answer phone); ii) a 2D model to map the activities of the former model, thus allowing to associate locations/objects to activities (e.g., "drug dispenser" to "take pill") and establishing usage profiles. Similarly, our approach offers also physical modeling capabilities but is not limited to location and may explore other 3D characteristics (cf. space, volume, etc.). This may explore, for example, the topology of the house and the location of an Avatar to understand if it is able to hear a system (e.g., Phone or DD) and move toward it (e.g., to answer the call or take the pill).

VI. Conclusions and Future Work

Computer simulation of home care scenarios represents a powerful mechanism to gain a deeper insight about the unpredictable cascade of events, which can occur and their potential interference effects. In this work, we apply 3D simulation to model the behavior of coexisting OTS systems and understand their possible unplanned interactions, eventually, involving users. The presented simulation framework allows the incremental deployment of new OTS systems in the simulated scenario and, through this, detect in advance possible interference problems. Due to these facts, we argue that 3D simulation is a pivotal part of the proposed SHC reflective middleware framework.

In this paper, we have presented the simulation framework and have described an application scenario, which clearly illustrates the mechanics behind the simulation environment. Presently, we are focused in enriching the simulation model by creating alternative behavior models for the computer agent who acts on behalf of the human user being simulated. Another scheduled goal, to address in future work, will be the generation of datasets, as outcome of the simulation, that can be used to compare the performance of difference systems.

Acknowledgment

Christophe Soares thanks FCT, Portugal, for PhD Grant SFRH/BD/64210/2009.

Safe Home Care Project (Interference-free Home Health-Care Smart Spaces using Search Algorithms and Meta-Reality Reflection), sponsored by FCT Grant PTDC/EIA-EIA/108352/2008.

References

[1] F. U. B. of the Census., "65+ in the united states," U.S. Government Printing Office, Washington, DC, Tech. Rep. Current Population Reports, Special Studies, P23-190, 1996. [Online]. Available: http://www.census.gov/prod/1/pop/p23-190/p23-190.pdf

[2] P. Gonçalves, J. M. Torres, P. Sobral, and R. S. Moreira, "Remote Patient Monitoring in Home Environments," in *The First International Workshop on Mobilizing Health Information to Support Healthcare - MobiHealthInf 2009 (in conjunction with BIOSTEC 2009)*, Porto, Portugal, 2009. [Online]. Available: http://isus.ufp.pt/wp-content/uploads/2010/03/mobihealthinf2009.pdf

[3] R. M. R.S. Moreira, "Project safehomehealthcare: Interference-free home health-care smart spaces using search algorithms and meta-reality reflection; fct grant ptdc/eia-eia/108352/2008," 2010, http://isus.ufp.pt/2010/03/.

[4] "Openmetaverse library," http://www.openmetaverse.org/projects/libopenmetaverse, Last accessed March 2011.

[5] J. J. Barton and V. Vijayaraghavan, "Ubiwise, a ubiquitous wireless infrastructure simulation environment," *HP LABS*, 2002. [Online]. Available: http://www.hpl.hp.com/techreports/2002/HPL-2002-303.pdf

[6] W. Jouve, J. Bruneau, and C. Consel, "Diasim: A parameterized simulator for pervasive computing applications," *IEEE International Conference on Pervasive Computing and Communications*, vol. 0, pp. 1–3, 2009.

[7] H. Nishikawa, S. Yamamoto, M. Tamai, K. Nishigaki, T. Kitani, N. Shibata, K. Yasumoto, and M. Ito, "Ubireal: Realistic smartspace simulator for systematic testing," in *the 8th Int'l Conf. on Ubiquitous Computing (UbiComp2006*, 2006.

[8] Z. Lei, S. Yue, C. Yu, and S. Yuanchun, "Shsim: An osgi-based smart home simulator," in *Ubi-media Computing (U-Media), 2010 3rd IEEE International Conference on*, july 2010, pp. 87 –90.

[9] M. Huebscher and J. McCann, "Simulation model for self-adaptive applications in pervasive computing," in *Database and Expert Systems Applications, 2004. Proceedings. 15th International Workshop on*, aug.-3 sept. 2004, pp. 694 – 698.

 ISBN: 978-1-61208-171-7