Smart Home for Elderly: Modeling and Simulation

Yacine Kissoum
Faculty of Sciences, Computer Science Department
Université 20 Août 1955, Skikda. Algeria
kissoumyacine@yahoo.fr

Sara Kerraoui
Faculty of Sciences, Computer Science Department
Université 20 Août 1955, Skikda. Algeria
kerraouisara@hotmail.fr

Mohamed Lamine Boughaouas
Faculty of Sciences, Computer Science Department
Université 20 Août 1955, Skikda. Algeria
moha.boughouas@gmail.com

Abstract – The development trend of computing is that the intelligence will become ambient. It is a one-user multi-device paradigm. These devices are transparent to the user and they can react in a proactive and non-intrusive manner. Moreover, among the general population those most likely to benefit from the development of this technology are the elderly and dependent people. They often wish to continue living independently in their home as opposed to being forced to live in a hospital. For that, homes should be smart in order to provide safety, security to reduce falls, disability, stress, fear or social isolation. Nevertheless, much of processes used for smart home design are ad-hoc. To cope with this drawback, while taking into account both dimensions of time and space, it is required to use a suitable formal model that is able to handle smart home domain specific nature. This paper proposes a technique that uses reference nets to model and simulate a smart home which is sensitive, adaptive and responsive to elderly's health, needs and preferences.

Keywords – Ambient intelligence, Multi agent systems, Nets within nets, Elderly, Timed nets, Smart home.

1. INTRODUCTION

In recent years, there has been an important growth in the field of Ambient Intelligence (AmI) [5, 6], involving major changes in the daily lives of people. The vision of Ambient Intelligence implies a seamless environment of computing and advanced networking technology that is aware of human presence, personalities, needs and which is capable of responding intelligently to spoken or gestured indications of desire, and even in engaging in intelligent dialogue. However, the development of systems that clearly fulfill the needs of AmI is difficult and not always satisfactory. It requires a joint development of methods, techniques and technologies based on services. An AmI based system consists on a set of human actors and adaptive mechanisms which work together in a distributed way. Those mechanisms provide on demand personalized services and stimulate users through their environment according specific situation characteristics [1].

Moreover, the percentage of elderly in today's societies keeps on growing. With the current trends in population demographics, it is becoming increasingly difficult for governments worldwide to fully support the health and social care systems [11]. The use of smart technologies, including smart homes could arguably relieve the pressure on aged care health and social support services [10]. The challenge with smart home technologies for elderly is to create a home environment that is safe and secure to reduce falls, disability, stress, fear or social isolation [4].

Like any other system, smart home development starts with a high level model and proceeds through a process of refinement, simulation, verification, implementation and test. Much of processes used for smart home design are ad-hoc. By time, all the gritty details of implementation are taken care with the original system description has pretty much been lost, causing a lack of design oversight and a surplus of one-time-only design artifacts [2]. Besides, very little can be done within a smart home system without an explicit or implicit reference to where and when the meaningful events occurred. For a system to make sensible decisions it has to be aware of where the inhabitants are and have been during some period of time. These insights, together with other information, will provide important clues on the type of activities the user is engaged in and the most adequate response.

So, to cope with the ad-hoc nature of smart home systems design process and to take into account both dimensions, space and time, to understand key elements of a situation under development, it's required to use a suitable formal model that is able to handle smart home domain specific nature. Thanks to Köhler et al [6] who proposed a model that deals with complex systems in an elegant and intuitive manner without losing formal accuracy. A model built upon a formalism that has a formal semantics to support verification, timed simulation and execution. The proposed approach for modeling smart home proposed so far is based on such a model called Reference net [10]. Reference nets are based on the nets within nets paradigm that generalizes token to data types and even nets.

The paper is structured as follows: section 2 surveys related work. In section 3 we give a short introduction on the paradigm of reference net and show how this paradigm can be used to model the mobility concept in multi-agent special case. Section 4 describes the case study which will be modeled in section 5; Sections 6 discusses the result of simulation. Finally the section 7 concludes this work.

2. RELATED WORK

Smart home are often the ideal solution for individuals with different needs and abilities. This is why several projects were introduced in recent years. Pan et al. [5] for example presented a homecare service that tackles the problem of fall discovery using a tri-axial accelerometer and reporting such a discovery to an emergency center. The paper introduces a fall detector based on a neural network and a multi-agent architecture for requesting emergency services. The Aging in place project[1] from the University of Missouri gives seniors an independent space that allows guaranteeing adequate medical supervision with dedicated sensors. Soprano-UE project[2] (Services-Oriented programmable Smart environments for older Europeans), funded by the European commission, aims to help older people to have a more independent life. The main contribution of the project was the creation of middleware OpenAAL for software implementation of smart homes.This is an open software framework that takes as input the information from sensors installed in the house and that the active services to perform actions on the environment. The originality of this system is to provide two layers for semantic representations of the environment: the first is a description of sensors and their states, and the second describes the situation of the inhabitants, activities, location, and emergency. Another project is the DAT project [3] that aims at building up a smart home environment where people with disabilities can improve their abilities to cope with daily life activities by means of technologically advanced home automation solutions. The project has a threefold purpose. The smart home will be used as a physical setting, where clients with disabilities can follow individual programs aimed at improving their independence in the home environment.

3. TIMED NETS AND MOBILITY

Reference nets [12] are a graphical notation that are especially well suited for the description and execution of complex, concurrent processes. As for other net formalisms, there exist tools for the simulation of reference nets called Renew (for Reference Net workshop). Reference nets extend black and colored Petri nets by means of net instances, nets as token objects, communication via synchronous channels, and different arc types. Definitions of these extensions are given in [8, 9]. While pure Petri nets capture the causality and conflict situations of a system nicely, there are reasons to add a notion of time to the formalism in order to model additional dependencies. This is especially true in the case of simulations of real-world systems.

In timed nets, a time stamp is attached to each token. It denotes the time when the token becomes available. Delays may be used with arcs in order to control the time stamps of token and the firing times of transitions. A delay is added to an arc by adding to the arc inscription the symbol @ and an expression that evaluates to the number of time units. For

[1] http : //aginginplace.missouri.edu/

[2] http://www.soprano-ip.org/

example, x@t indicates that the token value x has to be move after t time units.

Look to Figure 1. This is an example model of a simple traffic light where yellow light is omitted. On the left hand side net, the token in the place green has to move to the yellow place after 30 units of time. This will synchronize the transition inscribed with by the channel *this:go1()* which make the token on the place red (on the right hand side net) moving to the green place. Subsequently after 30 units of time, the transition inscribed by the channel *this:go2()* on the right hand side net become enabled and the traffic light will switch to red. Note that cars, modeled as mobile agents, are created every 10 units of time on the lower nets (*a:new cars_1()* and *b:new car_2()*), each being numbered accordingly. Cars on either side cannot move only if the transition *isL1Green* (respectively *isL2Green*) is enabled. Such transitions are modeled using test arcs. Test arcs, which have no arrowheads, allow a single token to be accessed (tested) by several test arcs at once.

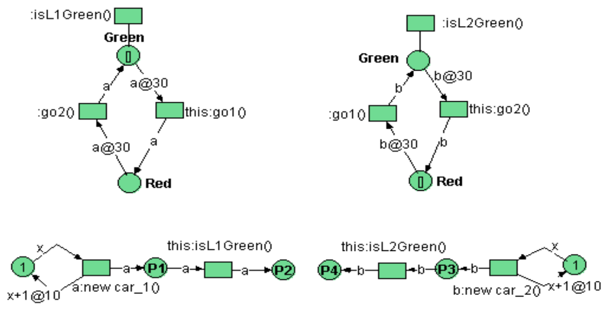


Figure 1: The traffic light example.

This example gives an idea how the interplay between object net (token net) and system net can be used to model mobile entities moving through a system net. Such a net offers or denies possibilities to move around, while the mobile object net moves at the right time by activating respectively, the transition that is inscribed with the precondition of the channel, and the move transition in the system net. Without the viewpoint of nets as tokens, the modeler would have to encode the mobile agent as a data structure for example. As a consequence, the inner actions of the mobile entity cannot be modeled directly, so they have to be lifted up to the system net, which seems quite unnatural. By using nets within nets we can investigate the concurrency of the system and the mobile agent in one model without losing the needed abstraction [10].

To investigate this modeling method, below we introduce the smart home for elderly case study. Then we show how we model this example by means of reference net paradigm.

4. SMART HOME FOR ELDERLY CASE STUDY

Let us imagine a house with several rooms: living room, kitchen, bed room, next room, bath room and the front yard. To be smart, such a house should contain a large set of services that cooperate to simplify the life of the owner, to make energy saving, and to provide comfort and security solutions. Figure 2 show an example of a smart home where various home devices are scattered over rooms to constitute a cooperating environment that provide smart services to the owner.

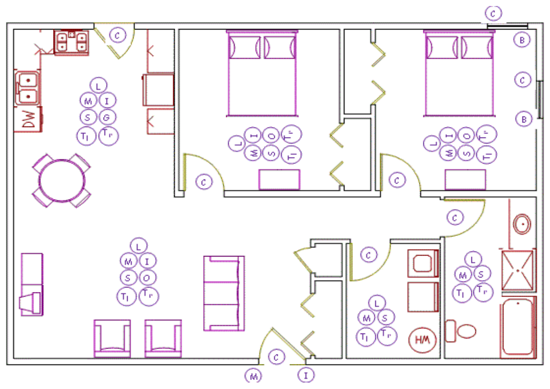


Figure 2: The smart home case study.

More precisely, to ensure security and safety of the owner the smart home uses information from temperature level sensors (T_l), temperature rate sensors (T_r) and smoke detectors (S) to detect possible fire and to take adequate actions. In the kitchen, where there is a possibility of gas leakage, a gas detector is installed (G). Gas leakage keeps an open eye to detect any gas leakage and take immediate action. There are image sensors (I) and motion detectors (M) in all places to identify any visitor and check whether he is authorized or no. Glass break sensors (B) and contact switch (C) are also installed on all windows/doors to monitor their safety. In addition, smart home provides energy saving solutions through installing occupancy sensors (O) in all rooms. Systems like lighting (L) will consider environmental conditions for their operation to help saving more energy.

The overall system will be designed according to MULAN [7] architecture. MULAN is implemented in RENEW[3] and has the general structure as depicted in Figure 3 Each box describes one level of abstraction in terms of a system net.

[3] www.renew.de

The net in the upper left side of Figure 3 describes an agent system, which places contain agent platforms as tokens. The transitions describe communication or mobility channels, which build up the infrastructure. By zooming into the platform token on place p3, the structure of a platform becomes visible. The central place agents host all agents, which are currently on this platform. Each platform offers services to the agents. Agents can be created (transition *new*) or destroyed (transition *destroy*). Agents can communicate by message exchange. Two agents of the same platform can communicate by the transition *internal communication*. *External communication* only binds one agent, since the other agent is bound on a second platform somewhere else in the agent system. Also mobility facilities are provided on a platform: agents can leave the platform via the transition *send agent* or enter the platform via the transition *receive agent*.

Agents are also modeled in terms of nets. They are encapsulated, since the only way of interaction is by message passing. Agents can be intelligent, since they have access to a knowledge base. The behavior of the agent is described in terms of protocols, which are again nets. Protocols are located as templates on the place protocols. Protocol templates can be instantiated, which happens for example if a message arrives. An instantiated protocol is part of a conversation and lies in the place conversations.

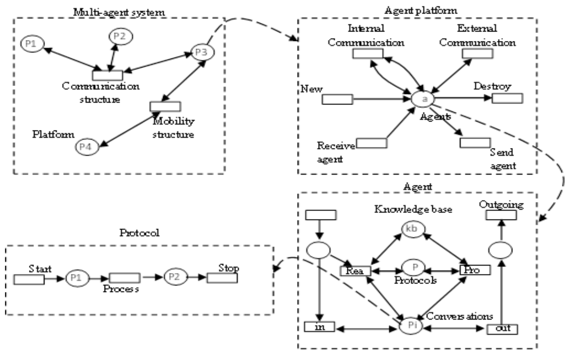


Figure 3: Agent systems as nets within nets (adapted from [9])

5. MODELING SMART HOME COMPONENTS

Using nets within nets as a modeling paradigm allows for the direct use of system models at execution time. This can be exploited as follows: The overall system will be designed as a system net with places defining locations (rooms) in or in front of the house: Transitions model possible movements between these rooms (green transitions), Figure 4. Due to the paper size limitation, only a subset of services will be presented here to highlight the modeling concept. The idea can be further extended to any smart home room and to any additional services.

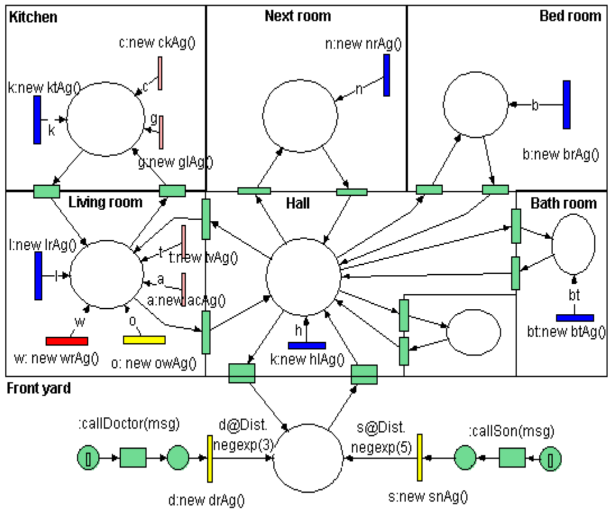


Figure 4: The smart home system net.

To be smart, rooms should be sensible to the human presence. Being sensible demands recognizing the user, learning or knowing her/his preferences, and the capability to exhibit empathy with or react to the user's mood and the prevailing situation. This is why we have attached to each room a cognitive stationary agent (blue transitions). These agents are: the hall agent, the kitchen agent, the living room agent, the bed room agents, the next room agent and the bath room agent. Also, from all the electronic devices scattered over the house we have selected those laying in the living room and in the kitchen, namely: the TV agent, the AC agent, the cooker agent and gas leakage detector which are modeled again as agents (brown transitions). There is also one particular agent (red transition) responsible of checking the blood pressure and/or glucose rate of the elderly at random times of a day. It may take the form of a bracelet hanging on the wrist of the elderly. Finally, this model of house is filled with life by implementing human house inhabitants (yellow transitions). These ones are the owner (the elderly); his doctor and let us say his son (the creation and arc inscriptions of these two agents will be detailed so far). All these agents share the same structure depicted in lower right side of Figure 3 (of course their knowledge bases and protocols are different).

The behaviors of these agents are described in terms of protocols. The protocol selection can basically be performed pro-actively or reactively. This distinction corresponds to the bilateral access to the place holding the protocols: *protocols*. The only difference in enabling and occurrence of the transitions *rea* and *pro* is the arc from the place *incoming* messages to the transition *rea* (see agent box of Figure 3). So it may only be enabled by incoming messages. Both the reaction to arriving messages and the commencement of a new conversation is influenced by the *knowledge base* place.

Human preferences and moods are part of these agent's knowledge bases. In simple cases the knowledge base place can be implemented for example as subnets. This is true in the case of sensors (modeled as reactive agents). Unfortunately, using such implementation for cognitive agents leads to a closed model that is often needed to rethink completely before introducing any modification. For this kind of agent advanced implementations of the knowledge place as the connections to an inference engine are also possible. In this way, we will make use of Java Expert System Shell[4] (JESS). As for other expert systems, JESS is composed from a rule base which corresponds to the knowledge base and a working memory which corresponds to the fact base. Its inference engine is composed from the pattern matcher which decides what rules to fire and when and the agenda which schedules the order in which activated rules will fire. The execution engine is responsible for firing rules and executing other code.

At home, the behavior of the owner is highly dependent on the timeframe in which he is located. Intuitively these slots are morning, noon, afternoon and evening. Thus, different scenarios related to such parts of the day can be imagined. The *''wakeup mode''*, the *''leave home mode''*, the *''return home mode''*, the *''evening mode''* and the *''go to sleep mode''* are examples of these scenarios.

Before discussing one of these scenarios, we start by defining the working memory and the rule bases of each our agents. A particular attention will be given to the living room and the kitchen agents. Indeed, such agents must keep an open eye on the elderly living in this house. For this, it is necessary that they store information about this person, its environment as well as other agents. Also, they must exchange their respective information to ensure the welfare of the elderly. For that, each JESS rule engine holds a collection of knowledge nuggets called facts. Every fact has a template. The template has a name and a set of slots, and each fact gets these things from its template. We start by defining the ACL message template as following:

```
(deftemplate ACLMessage (slot
communicative-act) (slot sender) (slot
receiver) (slot conversation-id) (slot
protocol) (slot language) (slot
ontology) (slot content))
```

[4] http://herzberg.ca.sandia.gov

It is important to note that elderly preferences described in each agent's working memory, depend on the room in which he is. It will be favorite program TV or multimedia playlist for the living room agent, water temperature for the bath room agent, light dimming and air conditioner levels for the bed room agent. The following templates define the owner of the house (the elderly) and people close to him.

```
(deftemplate elderly (slot name) (slot
sex) (slot age) (multislot disease)
(multislot TVprefer) (multislot
Musicprefer))
(deftemplate related (slot name) (slot
sex) (slot age) (slot type) (slot
telephone))
```

Other facts necessary to our room's agents are given in the following. They let them, among others, to create the desired ambiance once the owner enters the living room or the kitchen.

```
(deftemplate enter (slot name) (slot
room) (slot time))
(deftemplate television (slot state)
(slot channel))
(deftemplate airCond (slot state) (slot
temperature))
(deftemplate coffeeMaker (slot drink)
(slot sugar))
```

Also, and because this house is owned by an elderly suffering from one or more diseases (usually chronic), agents should be able to analyze the significant factors related to these diseases. Therefore, our agents will use the following templates:

```
(deftemplate bloodpressure (slot
maxVal) (slot minVal))
(deftemplate glucose (slot rate) (slot
state))
```

Finally, we must note that in a smart home, devices are often used to build an environment in which many features of the home are automated. But in some situations the user decides whether to perform tasks alone (manual mode) or allow devices throughout the home to realize its tasks (automatic mode).

Assigning values to these slots allows room's agents to identify persons having access to the house. An example of authorized persons is as follows:

```
(assert (elderly (name Ben) (sex male)
(age 69) (disease Diabetes
Hypertension) (TVprefer News Docs
Entertainment) (foodprefer tea fish
salad))
(assert (related (name Tim) (sex male)
(age 45) (type doctor) (telephone
077777777))
```

```
(assert (related (name Bob) (sex male)
(age    35)    (type    son)    (telephone
06666666))
```

Now we can define some JESS rules. Every JESS rule has two parts, separated by the "=>" symbol. The first part consists of the LHS pattern. The second part consists of the RHS action. The LHS of a rule consists of patterns which are used to match facts in the working memory, while the RHS contains function calls. Examples of such rules are the following. In the first one, as soon as the owner enters the living room, the agent will check the current time (morning in this case) then turn on the television and air conditioner with the owner's desired channel and temperature. In the second, the kitchen agent will ask the coffee machine to prepare a tea without sugar.

```
1 :(defrule rule1
(enter {name == Ben && room == living
room && time >= 8 && time <= 10})
(elderly {Tvpref == news})
    =>
(assert (television (state ON) (channel
NSN)))
(assert     (airCond     (state     ON)
(temperature 25)))
2 :(defrule rule2
(enter {name == Ben && room == kitchen
&& time > 10 && time <= 12})
(elderly {foodpref == Tea})
    =>
(assert (coffeeMaker (drink Tea) (sugar
No)))
```

In addition, the following rules allow the bracelet agent to decide on the health of the elderly based on his blood glucose.

```
3 :(defrule rule3
(glucose {rate < 0.60 })
    =>
(assert  (ACLMessage  (communicative-act
INFORM)  (sender  wrAg)  (receiver  lrAg)
(content         "hypoglycemia         ")
(conversation-id "call son"))
4 :(defrule rule4
(glucose {rate > 0.70 && rate <1.10 &&
state == fasting })
    =>
(assert  (ACLMessage  (communicative-act
INFORM)  (sender  wrAg)  (receiver  kiAg)
(content " normoglycemia "))
5 :(defrule rule6
(glucose {rate > 1.10 })
    =>
(assert  (ACLMessage  (communicative-act
INFORM)  (sender  wrAg)  (receiver  kiAg)
(content     "     hyperglycemia      ")
(conversation-id "call doctor"))
```

Let us focus on the elderly agent. Suppose that we are in the "*wakeup mode*", the execution of the of the JESS program laying in the elderly agent knowledge place (modeled in our case as mobile agent) concludes with a valid fact "*take breakfast*". Note that both JESS and Renew are written in Java. More precisely, references net are themselves Java objects. Making calls from JESS code to nets is just as easy as to make calls from nets to JESS code. So, Driven by this fact, the elderly agent selects a plan to execute: "*enter kitchen*". Actually the plan execution corresponds to the selection and the instantiation of the exact protocol and commencement of the conversation. Figure 5 shows such a protocol.

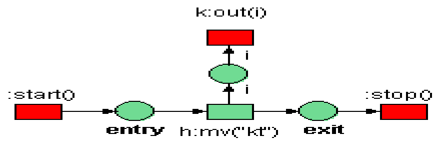


Figure 5: The enter kitchen protocol.

This protocol is selected pro-actively. After its instantiation, the transition *mv("kt")* produces a performative *i* that define the entering agent's identity, and which is directed to kitchen agent main interface over the *k:out(i)* channel; subsequently the agent terminates (by enabling the *:stop()* transition). The *k:out(i)* is met by synchronizing with the same transition in the kitchen agent net (*:in(i)*) which induce the enabling of transition *rea* on agent main net. Influenced by the identity of the person entering the kitchen and the inference results provided by the JESS program laying in knowledge base of the kitchen agent, this one will instantiate, first, the *"welcome"* protocol. So, knowing the entering person's identity and preferences, this protocol will welcomes the entering person, then verify whether the manual or the automatic mode is selected (transition *h:isManual()*), in the first case, the kitchen agent will do nothing but keep a close eye on the elderly. In the second, it produces a performative *d* defining the kind of the drink to prepare and over the channel *c:out(d)*, it will send it the *''request drink''* protocol (Figure 6). With the same manner this protocol terminates by enabling the transition *:stop()*.

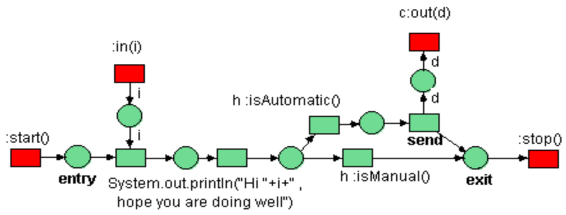


Figure 6: The welcome protocol.

After that this agent will instantiate the *"request drink"* protocol (the automatic mode). So receiving the kind of required drink, the transition *compose* produces a performative containing a string that is directed over the channel *c:out(s)* to the coffee maker agent; subsequently the protocol is blocked waiting for an answer message. An arriving answer enables the transition *answer*. After occurrence of such a transition the protocol is not blocked any further, and the receiver will process the answer. Two situations are possible: if the received answer is positive, then the agent will update his beliefs; otherwise the kitchen agent will undertake adequate actions. In either case the protocol instance is deleted (by enabling the *stop* transition), Figure 7.

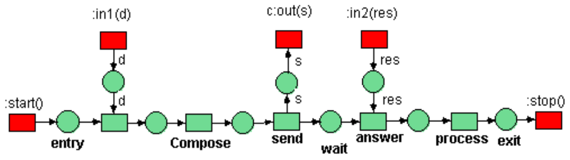


Figure 7: The request drink protocol.

Figure 8 shows the prepare drink protocol. The transition *:in(dr)* is responsible of passing the kind of drink required (tea, coffee, or milk). Then the protocol starts by verifying randomly if the required quantity for the drink preparation is less to the available one. If yes, the *prepare* transition is enabled (note that drink preparation may take five minutes on average). Otherwise, the transition *supply* is executed and the transition *:out(res)* will inform the sender (the kitchen agent) that his request is fulfilled or not.

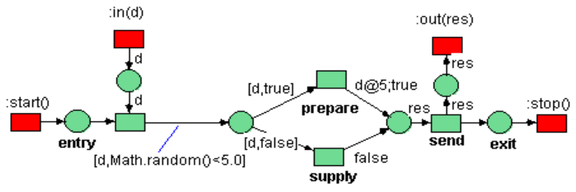


Figure 8: The prepare drink protocol

The above protocols describe how agents collaborate to assist elderly in the *wake up* scenario. Let us see how these agents take care of elderly health. Remember the bracelet agent that checks the glucose rate of the elderly at random time of a day. It is modeled as a mobile agent (having the same structure as other agents) that is transported by the elderly agent. Now, suppose that the manual mode is activated, the cooker state is on and the bracelet agent has detected a hyperglycemia. Once received, the room agent in where the elderly is, instantiates the protocol depicted in Figure 9. This one starts by executing the transition inscribed by the channels *callSon* and *callDoctor*. It is modeled to meet by synchronization the same transitions in the house net (see Figure 4). The doctor, for example, takes three hours on average with a negative-exponential distribution to reach the front yard place. After that, the agent will check whether the manual mode is activated or not. If so, the agent will switch to automatic mode so it can broadcast safety messages to other agents by enabling the transition *this:Safety()* (turn off the cooker, the TV, the alarm, etc.).

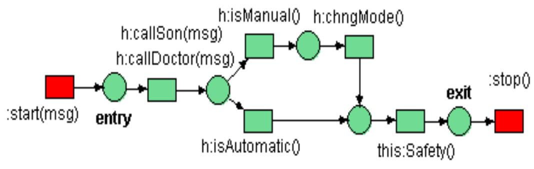


Figure 9: The emergency protocol.

6. THE SIMULATION

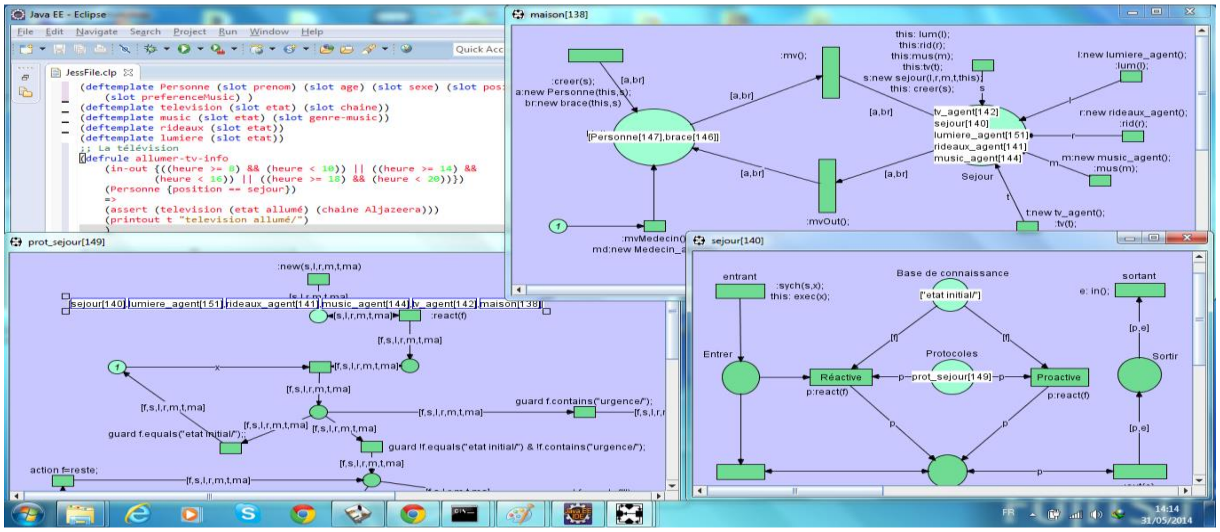
Having defined the features of the smart home components, the system is now ready for execution (simulation). Figure 10 shows an ordinary simulation state. Only the living room and the hall places are depicted. In addition the living room agent net and its protocol, the window in the background contains the JESS facts and rules representing the knowledge base of the living room agent.

It is now easy to see the state of the overall system by looking at the system net. In fact, it is possible to take control of the whole agent system implementing the living room agent scenario by just double clicking on one of the corresponding token. The tool RENEW will display the respective net allowing for a complete inspection of the running system without having to interrupt it. The advantage is twofold [10].

- Whereas a normal modeling process requires at least three stages to reach an executable program: (a) model the system, (b) implement the model and (c) write the visualization for the program, using the nets within nets paradigm, the modeling process concludes with a running system model.
- The visualization of the system model at execution time is indeed the implementation of the system. This eliminates several potential sources of errors shifting from model to implementation to visualization in an ordinary software design process.

7. CONCLUSION

In this paper we have presented an approach for modeling of cooperating ambient agent cohabiting an elderly's home. An important argument for using the reference nets paradigm is its strong expressiveness, openness and versatility without losing formal accuracy. Indeed, using such a paradigm, the modeling process concludes with a running system


Java EE - Eclipse
File Edit Navigate Search Project Run Window Help
JessFile.clp
maison[138]
prot_sejour[149]
sejour[140]
:new(s,l,r,m,t,ma)
sejour[140],lumiere_agent[151],rideaux_agent[141],music_agent[144],tv_agent[142],maison[138]
:react(f)
guard f.contains("urgence");
guard f.equals("etat initial");
guard !f.equals("etat initial") & !f.contains("urgence");
action f=reste;
:creer(s);
a:new Personne(this,s);
br:new brace(this,s)
[Personne[147],brace[146]]
:mv();
:mvOut();
:mvMedecin();
md:new Medecin
this: lum(l); this:rid(r); this:mus(m); this:tv(t);
s:new sejour(l,r,m,t,this);
this: creer(s);
tv_agent[142] sejour[140] lumiere_agent[151] rideaux_agent[141] music_agent[144]
Sejour
l:new lumiere_agent(); :lum(l);
r:new rideaux_agent(); :rid(r);
m:new music_agent(); :mus(m);
t:new tv_agent(); :tv(t);
Base de connaissance
["etat initial"]
entrant
:sych(s,x); this: exec(x);
Entrer
Protocoles
Réactive
prot_sejour[149]
Proactive
p:react(f)
sortant
e: in();
Sortir
[p,e]

```
(deftemplate Personne (slot prenom) (slot age) (slot sexe) (slot pos
    (slot preferenceMusic) )
(deftemplate television (slot etat) (slot chaine))
(deftemplate music (slot etat) (slot genre-music))
(deftemplate rideaux (slot etat))
(deftemplate lumiere (slot etat))
;; La télévision
(defrule allumer-tv-info
    (in-out {((heure >= 8) && (heure < 10)) || ((heure >= 14) &&
            (heure < 16)) || ((heure >= 18) && (heure < 20))})
    (Personne {position == sejour})
    =>
    (assert (television (etat allumé) (chaine Aljazeera)))
    (printout t "television allumé/")
```

Figure 10: An example of simulation.

model. Also, this paradigm is selected not only for mobility purposes in multi-agent systems, but mainly because it is built upon a formalism that has a formal semantics to support verification, execution and timed simulation. We think that such robust and easy-to-use tool reduces considerably the large initial effort in term of man-hours required mainly in constructing and validating smart home models.
Additional services such as fire system, curtain control, climate control system, multimedia control system and so on, will be directly integrated in the design of the reference net based multi agent system architecture MULAN.

8. REFERENCES

[1] A. Ricci, Buda, C. and Zaghini, N. An agent-oriented programming model for SOA & web services. In 5th IEEE International Conference on industrial Informatics (INDIN'07),Vienna, Austria. pp. 1059-1064, 2007

[2] A. K. Nabih, M. M. Gomaa, H. S. Osman and G. M. Aly, Modeling, Simulation and control of smart home Using Petri Nets, International Journal of Smart home. Vol. 5 N° 3, July 2011

[3] A. Renzo, V. Gower, A. Caracciolo, G. Del Zanna and M. Di Rienzo, The DAT Project: A Smart Home Environment for People with Disabilities. LNCS 4061, pp.492-499, springer –Verlag Berlin Heidelberg 2006.

[4] J. Barlow, Venables T, Will technological innovation create the true lifetime home? Housing Studies 19: 795-810. 2004

[5] J.-I. Pan, Yung, C.-Y., Liang, C.-C., Lai, L.-F.: An intelligent homecare emergency service system for elder falling. In: World Congress on Medical Physics and Biomedical Engineering, pp. 424--428. Springer, 2006.

[6] M. Anastasopoulos, D., Bartelt, C., Koch, J. & Rausch, A. Towards a Reference Middleware Architecture for Ambient Intelligence Systems. ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications. 2005

[7] M. Duvigneau, D. Moldt and H. Rölke, Concurrent architecture for a multi-agent platform, in Proceedings of the Workshop on Agent Oriented Software Engineering (AOSE'02), LNCS 2585, Springer Verlag, 2003.

[8] M. Köhler, D. Moldt, and H. Rölke, Modeling the structure and behavior of petri net agents, J.M. ICATPN 2001, LNCS 2075, 2001, 224–241.

[9] M. Köhler, D. Moldt and H. Rölke, Modelling mobility and mobile agents using nets within nets, ICATPN 2003, LNCS 2679, 2003, 121–139.

[10] M. Morris, Ozanne E, Miller K, Santamaria N, Pearce A, Smart technologies for older people: A systematic literature review of smart technologies that promote health and wellbeing of older people living at home. IBES, The University of Melbourne, Australia. 2012.

[11] Agoulmine, Deen MJ, Jeong-Soo L, Meyyappan M, U-health smart home: Innovative solutions for the management of the elderly and chronic diseases. IEEE Nanotechnology Magazine 5: 6-11. 2011

[12] O. Kummer, Simulating synchronous channels and net instances. Workshop Algorithmen und Werkzeuge für Petrinetze 1998, 73–78.