Human behaviours simulation in ubiquitous computing environments

Teresa Garcia-Valverde, Francisco Campuzano, Emilio Serrano and Juan A. Botia
University of Murcia, Spain. E-mails: {mtgarcia,fjcampuzano,emilioserra,juanbot}@um.es

***Abstract*—Ambient Assisted Living (AAL) systems' main goal is to augment live quality of elderly people, by using ICT based systems. In this paper, we are concerned with the artificial reproduction of a physical environment (i.e. a house) and an elder (i.e. the attended) living in such environment. An agent based social simulation system is used for such purpose. Such simulator will allow the integration of ubiquitous computing appliances, services and applications in such environment. A realistic reproduction of human behaviour in the simulator helps, in this context, in the validation of silent monitorisation, diagnosis and action based applications. Proofs are given in the paper which demonstrate the level of reality reached by comparing the artificial behaviour with real ones.**

***Index Terms*—Ubiquitous computing, Ambient Assisted Living, behaviour simulation, user modelling.**

I. Introduction

The main thesis presented in this paper is the following: agent based social simulation (ABSS) [1] may help in the engineering of Ambient Intelligence systems. ABSS is a simulation paradigm in which the focus is put on the definition of the separate components of the simulation in an isolated manner. In such simulation runs, the emergence of behaviours is the main subject under study. And the metaphor of agent is used for specification of single components and interactions among them. An ambient intelligence [2] system is a set of appliances, services and applications which silently surrounds and interact with the user in an intelligent manner. In such kind of systems, the user is the central entity of the model. Starting from the user, services and applications are built.

A main difficulty one may find in the development process of an AmI system is that of testing and validation. Testing is the process of executing a program with the intent of finding errors in the code [3]. Such errors must be debugged [3]. Some of the errors may be found by using a Unit approach with the system under test (SUT). Common errors which are found in this stage are related to common programming mistakes (e.g. values of variables out of range, shoddy checking of return values from methods and so on). Thus, robustness of the program is a must here. But a more elaborated test set may be defined in order to assess the functionality of the system (i.e. it behaves as expected). But, if the main issue in AmI systems is a smooth interaction with the user, an Unit based approach is no more valid here. It is clear that the user, or at least a model of the user, should be incorporated in the development process in order to measure to what extent, the SUT is behaving as expected when interacting with him. In this paper, an approach to test and validate AmI systems in a stage prior to deployment is presented. The main idea behind this is that the user is modelled with a computational model and integrated into an ABSS model which incorporates as a simulated artifact, the environment, the hardware (i.e. mainly sensors and interfaces with the user) and integrates the real software (i.e. services and applications). The real software is precisely the SUT here.

The proposal is articulated by means of a methodological work. Such a methodology is a set of procedures which guides the developer in the definition, creation, testing and validation of the AmI system. It is based on a methodology previously described by Gilbert et al. [1]. It comprises the creation of the necessary ABSS models and how they should be employed to find errors in AmI services. The application of the methodology is exemplified in a real domain. The application domain is AAL (Ambient Assisted Living). An AAL system is an ICT based solution which is devoted to augment quality of live for elderly people. In this case, the interest is focused on a system called Necesity [4]. It is based on a sensor network deployed through the house and a central processing unit. Sensors include presence, pressure and open-door. The system is designed to work on single person environments (e.g. an elderly who lives alone and independently in his house). It is in charge of monitoring activity regime of the elderly 24x7 in a manner that when his activity pattern is anomalous, an emergency response is started. The rest of the paper will demonstrate that an artificial reproduction of a house, sensors and the attended, together integrated with the real software, helps in the fine tuning of the activity pattern management software.

The rest of the paper is structured as follows. Section II introduces the methodology used for the engineering of AmI services. Section III introduces the computational models employed to artificially reproduce the behaviour of the attended. Such behaviour is based on a probabilistic and hierarchical automata which governs the activity and location of the modeled elderly in each instant of time. In section IV, the validation approach is presented. It is based on the statistical contrast of artificial data traces obtained by simulating the automata just mentioned with similar traces coming from real users in similar context.

II. AVA, an Agent based methodology for the Validation of AmI systems

This section explains an agent based methodology for the validation of AmI systems called AVA. This methodology

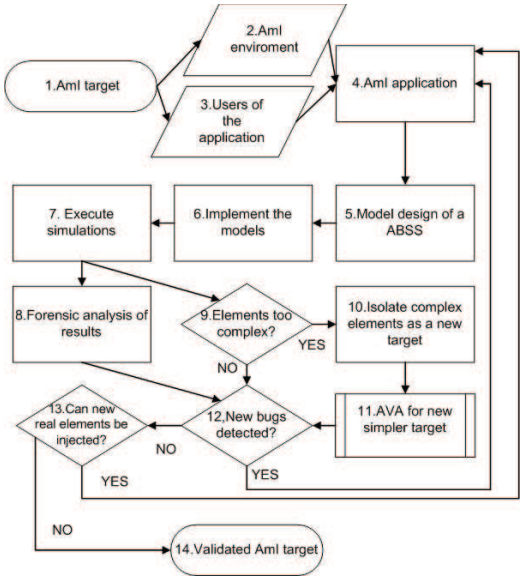


Fig. 1. AVA, An Agent based methodology for the Validation of AmI systems expressed as a flowchart

proposes the development of ABSS in order to validate AmI applications. AVA is an extension of the methodology promulgated by Gilbert et al. [1] for the development and use of general ABSS. The two main innovations with regarding the classical methodology by Giltber et al. are: (1) the existence of a step to generate simpler simulations and (2) the consideration of including real elements in the simulation to get more realistic results. This section will show the advantages of these innovations for the specific purpose of validating AmI applications. The AVA methodology is expressed as a flowchart in figure 1[1].

Gilbert et al. [1] defines the target of a social simulation as some "real world" phenomenon which the researcher is interested in. The AVA methodology is proposed to validate an AmI application including their interactions with the environment and users. Thus, AVA starts considering an AmI target (step 1) which includes an environment, users and an AmI application which may be finished or at an advanced stage of development. Typically, the use of ABSS to generate knowledge involves a necessary familiarization with the domain in the first step. This is required to generate models of the target in the following steps of the methodology [1]. The main elements to be studied in an AmI system are: the environment (step 2), users (step 3) and the AmI application (step 4). Note that while the environment and the user are inputs, the application system is a process. In principle, the environment and users are external and do not support changes. On the other hand, the AmI application can be modified. This application will be refined along the iterations of AVA to get a realistic validation. This paper discusses the performance of these steps for an AAL system for elderly people

The development of an AmI system model is performed in step 5. The model design associates the real system with a representation of this system (the model) [6]. Here, the AmI application must be modelled but also the users and the environment. These models are necessary to validate the AmI application because they interact with it. Moreover, a realistic validation of the application needs realistic models for users and environment. Therefore, models must describe reality before being as simple as possible [7]. Section III explains the construction of a user model for an specific AmI application.

Step 6 deals with the implementation of the AmI system in a simulation language. The implementation from the concepts of the model is not a trivial task [6]. A general programming language or a specific one of the available frameworks for the development of ABSS can be used for the construction of the simulation. The second option is much more convenient because several of the typical tasks in the construction of ABSS have been included in this kind of software packages [1]. Examples of these tasks are scheduling agents' actions or building basic environments. The web of the Open Agent Based Modeling Consortium[2] nowadays lists 22 of these frameworks. Section III shows the use of a specific software package for the implementation of a realistic environment model: 3D Sweethome.

After building the simulation, this must be executed (step 7). Quick, cheap and numerous experiments can be performed thanks to the ABSS. These executions produce large amounts of data regarding the behaviour of users, environment and application models. Forensic analysis, step 8, is an offline analysis to be conducted on the data stored from the previous step. The analysis should consider whether the AmI application functionality is correct. Furthermore, this step must validate that the behaviour of users and environment models is consistent with the observed reality (steps 2 and 3). Without this validation, the theories generated from simulations have no relation to reality (as they are based on non-descriptive models). Therefore, the functionality of the AmI application model is linked to the users and environment models. Section IV deals with the validation of the users model for an AAL system

One of the innovative points in the AVA methodology is the use of simple simulations as a means to validate descriptive simulations. Step 9 checks if any elements of the simulation are too complex. In this case, complexity means that it is difficult to assess whether the behaviour of an element is the expected one. For example, some behaviours of users can be so complex that they need to be evaluated in isolation. An example of this type of behaviour would be the resolution of collisions on the motion of a large number of agents. This behaviour is a problem to be studied itself and its validation would be much more complicated with additional elements in

[1]The flowchart uses standard elements of classic flowcharts [5] as flow of control (represented as arrows), processes (represented as rectangles), decisions (rhombus), input/output (parallelograms), start and end symbols (ovals) and predefined processes (rectangles with vertical lines at the sides).

[2]OpenABM Consortium website: http://www.openabm.org/

the simulation (more users' behaviours, a realistic environment model, a realistic model of an AmI application, etc). In these cases, the methodology proposes to consider these complex elements as an object of study itself (step 10) and repeat the AVA methodology for them (step 11). The reuse of models and code in this new iteration will be direct because a more descriptive simulation is available as result of the previous steps. Once the complex element is validated in a simpler simulation, which is the final result of the methodology, the next step for the overall simulation (step 12) can be performed.

Step 12 checks if the developer has found errors in functionality. If that is the case, the AmI application of step 4 must be modified in order to correct these errors and the process repeated. Besides the primary objective (validating the AmI application), is typical to find bugs of previous steps at this point in the form of implementation failures or unrealistic models.

The final decision, step 13, checks if actual elements of the AmI system can be connected to the simulator. The AVA methodology proposes to inject or connect real elements in the simulation progressively in order to make more realistic validations[3]. This process is called *"reality injection"* and the basic idea is that real elements can coexist with simulated elements. After connecting real elements, the methodology must be repeated from step 4 to improve the application and the models. The result is an exhaustive validation which is as realistic as possible. The obvious question is why models and simulations are necessary if real elements (as real users) can be injected. The answer is that a model, by definition, is somewhat easier to study than the modelled reality. The purpose of including real elements in simulations is to improve the realism of the models. Then, in subsequent iterations, the real elements will not be included because the pure simulations allow faster and cheaper tests.

Finally, if models are descriptive enough, the bugs found in the functionality of the AmI application model will correspond to failures of functionality in the real AmI application. These failures should have been corrected in each iteration of the AVA methodology. Therefore, the result of the methodology (step 14) is that the AmI target is exhaustively validated.

III. Realistic Behaviour Modelling

In this section, the particular models used, in the application of the AVA methodology in the AAL domain, are introduced. In section III.A, it is presented how the physical environment (i.e. the house and furniture) and sensors were defined. Section III.B refers to the production of realistic computational models for elders living in such environment and making sensors to react on their presence. Having such models (i.e. the house, sensors and persons) within a simulation, and its integration with the ubiquitous computing software, such software can be tested.

[3]Notice that this not involves necessarily a Participatory Multiagent Simulation. The real elements do not have to be humans playing the role of simulation components. These elements can be software applications, hardware or even parts of the environment.

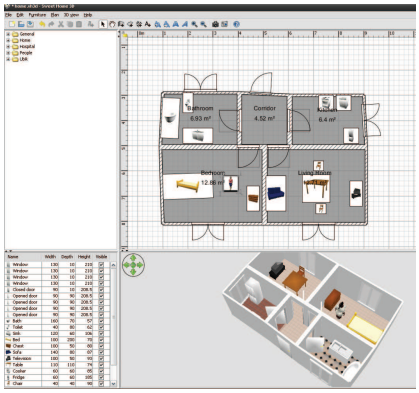


Fig. 2. A plane model in Ubik's editor and its 3D representation

A. Environment Modelling

For Multi-Agent Based Simulation (MABS), it is available Ubiksim[4], a simulator developed by University of Murcia that works over MASON[5]. It has integrated an environment modeling tool based on SweetHome3D[6], an application conveniently adapted for modelling attended people and their environment. It is possible to create houses over a 2D plane, also it offers a 3D navigable view (figure 2). This view is used for simulation's visualization at real time.

In the physical environment generated by using the Ubik editor, a simple house (see figure 2) with a kitchen, a bathroom, a bedroom and a living room is modelled. Presence sensors are included in every room of the house. And a sensor for open door (it is necessary for knowing when the elder leaves the home) is also included in the outdoor. When a simulation is run, the person moves in the house and stimulates sensors when he is detected by them. Such sensors, through the ubiquitous computing software, generate events. And these events generate log entries. Such simulated log entries are used afterwards to check if the virtual elder behaves in a realistic manner (see section IV).

Notice that log entries (both in the simulator and the real setting) are generated by the same monitoring service which continuously checks if the elder may be suffering some problem, by using a pattern recognition approach (more details on this may be found in [4]) on the events coming from sensors. In the first case, the person is virtual, in the second it is real. But the monitoring service is the same.

B. Behaviour Modelling

The target of the modelling activity is a typical aged person, who lives independently and alone in his own house. As he lives alone, the following situation may occur: he may suffer some health problem and stay immobilised in the floor for too much time before anybody comes and notices

[4]UbikSim: http://ubiksim.sourceforge.net, last access: 20 May 2010

[5]MASON Toolkit: http://cs.gmu.edu/~eclab/projects/mason/, last access: 20 May 2010

[6]Sweet Home 3D: http://www.sweethome3d.eu/es/index.jsp, last access: 20 May 2010

that something is wrong. But it is possible to develop an ubiquitous computing system which detects it and generates some emergency response process [4]. By following the AVA methodology, we may use a simulated elder within a simulated environment to test such system before it is deployed in a real environment for pilot testing. Such simulated elder should be necessarily simulated along the 24 hours of the day, repeatedly for a determined number of weeks. For this, it is assumed that the day is divided into time slots (i.e. morning, noon, afternoon and night). In each time slot, it is also assumed that the simulated person behaves specifically for such slot.

The behaviour of simulated people are modelled probabilistically. In this approach, behaviours are defined as situations the agent should play in each moment. Transitions between behaviours are probabilistic. The underlying model is a hierarchical automaton (i.e. in a higher level there is a number of complex behaviours that the agent may play and once it is in a concrete state, within the state there is another automaton with more simple behaviours). So, the modelling of each behaviour is treated separately and the modeller is abstracted of unnecessary details. So, in the lowest level (basic actions), each state is atomic. An agent never conducts two behaviours of the same level simultaneously.

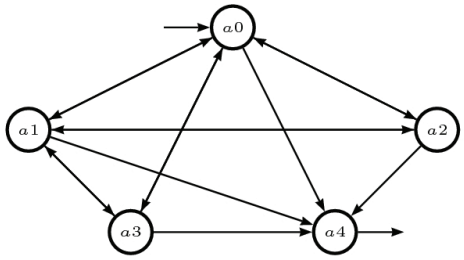
The behaviours used for modelling elders are of three types:

- Monotonous behaviours: the kind of behaviour the elder manifest always approximately in the same time slot, and on a daily basis (e.g. sleeping, having meal, medication and so on).
- Non monotonous behaviours: the kind of behaviour the elder usually manifest, not bounded to a concrete time slot, and repeated within a non constant period (e.g. going to the toilet, having a shower, cleaning the house and so on).
- Any time behaviours: such behaviours will sometimes interrupt others the elder is already doing, and will be generated regardless they were already generated in a temporal proximity (e.g. in his spare time).

A probabilistic automaton [8] is defined as the quintuple $(Q, V, P(0), F, M)$ where Q is a finite set of states, V is a finite set of input symbols, $P(0)$ is an initial state vector, F is a set of final states and M is a matrix that represents probabilities of transition for every state. In this definition, transition's probabilities depend on time. According to different daily time slots, the agent's behaviour acts on a different pattern. There are time slots for eating, sleeping and taking medication (Monotonous behaviours).

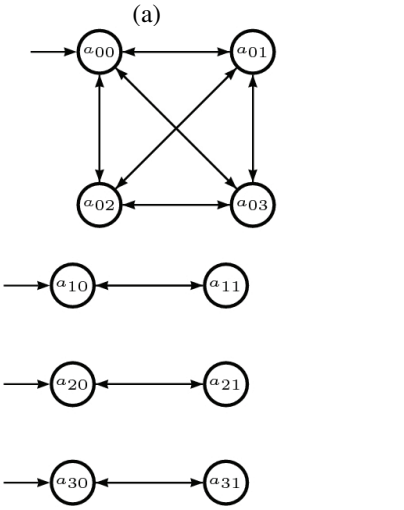
Notice that, when the elder is at any state, the necessity of changing to another state may arise. But this is not done immediately. Moreover, a number of different changes (i.e. transitions) may be pending simultaneously. Thus, a list of pending tasks (i.e. or events) is maintained. Such tasks are ordered by a static priority (e.g. going to the toilet goes before cleaning).

For generating transitions in real time, probability distribution functions are used according to the type of the behaviour and its features. These distributions are member of the exponential family [9], [10]. Section IV defines all distribution functions used.



$Q = \{a_0 = NormalTime, a_1 = MedicationTime, a_2 = MealTime, a_3 = SleepTime, a_4 = Anomalous\}$

(a)



$Q = \{a_{00} = SpareTime, a_{10} = MedicationTime, a_{20} = MealTime, a_{30} = SleepTime, a_{x1} = ToiletTime, a_{02} = ShowerTime, a_{03} = CleanTime\}$

(b)

$Q = \{a_{200} = GoingToFridge, a_{201} = GoingToCooker, a_{202} = Cooking, a_{203} = GoingToTable, a_{204} = Eating\}$

(c)

Fig. 3. (a) Level 0 automaton, (b) Level 1 automata, (c) Level 2 automaton for state a_{20}

Notice that monotonous behaviours must be activated at specific time hours or within specific time intervals. For example, in the case of *MedicationTime*, a new necessity of changing to such state will be generated exactly at the time to take medicines. In the case of MealTime and SleepTime, the necessity is generated within a time interval. The distribution that models these transitions is bounded in this time slot. It must be assured that agent eats and sleeps every day, because of that, if not transition is generated into the time slot, a transition is generated at the end time.

To provide more realism, an automata hierarchy is introduced representing each state by lower level automata which define more specialized behaviours.

As shown in figure 3(a), in level 0 the initial state

is a_0 =*NormalTime*. For every one of the other states $\{a_1$ =*MedicationTime*,a_2 =*MealTime*,a_3 =*SleepTime*$\}$ exist a list of times generated by a probability function. When a time counter arrives to one of these times, a transition to state owner of the list is added to pending tasks list. When the action is finished, the automaton returns to initial state if there is not another pending task. The final state is a_4 =*Anomalous*, if it is reached, the execution will be stopped.

In level 1 (figure 3(b)) non monotonous behaviours are represented. There is an initial state in every refinement where the person does the main task of the upper level (eating, sleeping or taking medication). The state a_{x1} =*ToiletTime* is considered in every refinement of level 0. However, the states a_{02} =*ShowerTime* and a_{03} =*CleanTime* only may be activated in normal state of level 0.

In the lowest level (level 2), the new automata define some specialized actions refining every state of level 1. These new states do not give relevant information, but the agent gains more realistic behaviours.

With the sequence of actions in figure 3(c) the person cooks before eating. It refines state a_{20} of level 1, the agent is not going to be static in the kitchen, because it must be going to different places and spends some time in every state.

In figure 4 a base implementation for agent's behaviour is presented. First of all, all possible automata's states are iterated for initializing lists which contain time instants (lines 13-24). These time instants are generated by a probability distribution function, and they represent when a transition to its associated state is going to be launched. After of that, the automaton begins to run. At every time instant, if actual time is equal than first item of a time list, a transition is added to pending tasks list (lines 57-60). When there is one state with higher priority than actual state in pending tasks list, a transition is also generated. Then, if actual state is unfinished, it is stored as a pending task and a new state is reached (lines 46-50). When this new state is finished, leaved state may be resumed.

The configuration parameters for the whole simulation involve:

- Temporal limit in every room before entering in anomalous state (t_{max})
- The probability distribution parameters according to the kind of behaviour
- Time slots of routinary temporal behaviours, like eating or sleeping $(ini_s, end_s, s \in MealTime, SleepTime)$

IV. Validation of the approach

From section II, it is clear that user modelling is a means for testing AmI services and applications without a real environment. This task is performed in the third step of the AVA methodology. Regarding to the validation, other important steps within AVA are the steps from 5 to 8. Model validation is needed in order to show that models are able to describe the users' behaviours. The rest of the section shows how the model validation has been approached. But basically, activity data from real users is compared (in statistical form) with the same type of data produced by the artificial models.

```
1 Let pt be a list of pending tasks ordered by priority
2 Let states be a list with all posible automata's states
3 Let times be a list of time instants when transitions
4 are going to be generated
5 Distribution Functions to model the ocurrence
6 of monotonous behaviours:
7 Let mb be the function to model monotonous behaviours
8 Let nb be the function to model non monotonous behaviours
9 Let ab be the function to model anytime behaviours
10
11 //Instants of time initialization
12
13 for all s in states do
14   if isAnytime(s) then
15       times(s)<-ab()
16   else if isMonotonous(s) then
17     //Initial and final instants of bounded time slot
18     i <- ini(s)
19     e <- end(s)
20     times(s)<-mb(i, e)
21   else if isNonMonotonous(s) then
22     times(s)<-nb()
23   endif
24 endfor
25
26 actualTime <- 0
27 //actual state is defined by 3 numbers, one per level
28 level0 <- 0
29 level1 <- 0
30 level2 <- 0
31 actualState<-newState(level0,level1,level2)
32
33 //Once initialized the times, the automata begin to move
34
35 //If an anomalous state is reached, the execution stops
36 while(level0(actualState)<>4)
37   //If actual task ends, initial state is activated
38   if timeLeft(actualState)=0 then
39     actualState <- newState(0,0,0)
40   endif
41   //If next state has higher priority than actual state,
42   //actual state becomes a pending task and it is stored
43   //in pt
44   if (size(pt))>0
45     nextState <- first(pt)
46     if priority(nextState)>priority(actualState) then
47       add(pt,actualState)
48       actualState<-nextState
49       remove(pt,nextState)
50     endif
51   endif
52   for all s in states do
53     time <- first(times(s))
54     //If actual time matches first time instant in
55     //times, the task associated with this time instant
56     //is added to pending tasks
57     if actualTime=time then
58       remove(times(s),time)
59       add(pt,s,priority(s))
60     endif
61   endfor
62   actualTime <- actualTime + 1
63   //Decrement remaining time for finishing actual task
64   time <- timeLeft(ActualState) - 1
65   setTimeLeft(actualState, time)
66 endwhile
```

Fig. 4. Realistic behaviour implementation

A. Data Preprocessing

Activity data from real users were obtained within a pilot project devoted to the validation of the Necesity system. Around 25 users, all elderly people living independently, were used in such Pilot project. In this paper, data coming from three users, under monitorisation during two months, were used. Data is in the form of a Necesity log. The logs offer the

possibility to represent where the user is, at any moment, at the house (including also if he leaves the house or he is seated or sleeping). Thus, three different data sets, with log entries corresponding to sensor events are available. These data sets need further processing.

Validating the artificial models is assuring that the right probability distribution function is used to reproduce the transition between the different states (i.e. behaviours). Thus, data series are obtained from log data as they were a random number series generated by the corresponding probability distribution. Such series are used afterwards in a goodness of fit test. For example, in case of the non monotonous behaviours and anytime behaviours, preprocessing the log data involves the extraction of the time series of the moments in which each event is produced. These time series only include values within the typical awake period of the corresponding user. Obviously, while the user are sleeping, his daytime routines change.

The data preprocessing for the monotonous behaviours is slightly different. This kind of behaviour is usually produced in a bounded time slots. For example, having dinner, having lunch or sleeping are behaviour which occur during specific time periods of the day. So, the preprocessing involves the extraction of the behaviours events inside the time slots. The time slots can be slightly different for each person, but it is possible to define an approximation of them which will be valid for all (see in section III the configuration parameters). Finally, in each slot time, the time intervals between each ocurred event are measured. The extracted time series are composed of these time intervals.

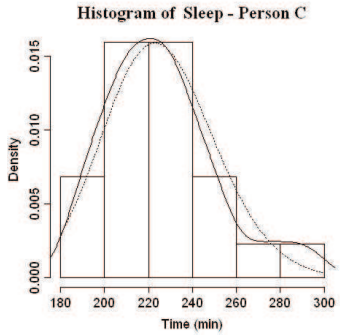
B. Model Diagnosis

Validation is one of the most important issues in a simulation system. Validation consists in the determination that the simulated model is an acceptable representation of the real system, for the particular objectives of the model [11]. There are many techniques for validating simulations [11], [12], and specially, for validating agents based on simulated models [13].

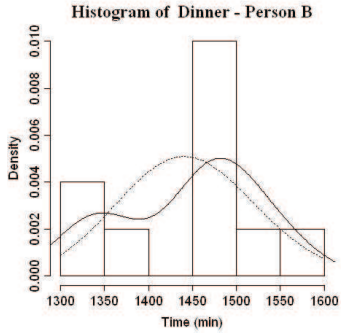
The models which describe social processes, as the model proposed here, are generally hard to validate. In this approach, the behaviour is probabilistically modelled. However, some statistic tests should be done to assume that a probabilistic model is reasonable to explain the data. This process is called model diagnosis. And this section is devoted to make the diagnosis of the models presented above. The most serious problem that one usually faces in this kind of validation is the lack of real data [14]. However, in this work the data from the Necesity project is available and can be used.

From these preprocessed data, some histograms for different behaviours and people are shown in Fig. 5. The sample density is shown with a black line. The dashed line shows the probability density function of the theoretical distribution that models that behaviour.

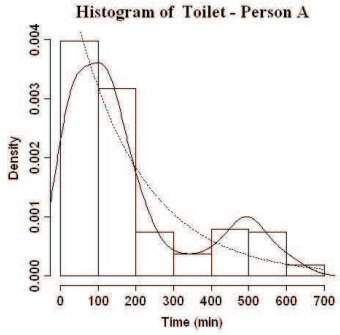
Graphs 5 (a) and (b) show two monotonous behaviours, sleeping and having dinner (having lunch is similar). In these kinds of behaviours the event that raises the behaviour occurs



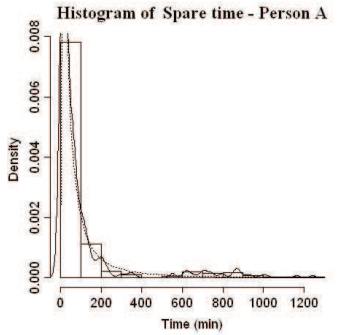
(a)



(b)



(c)



(d)

Fig. 5. (a) Minutes between 21:00 hours and the instant the attended C goes to bed, (b) Time between dinners for attended B, (c) Time between uses of the toilet for attended A, (d) Time between spare time for attended A

during a time interval. The interval is usually the same for each attended person, i.e., a person usually goes to sleep or to have dinner at the same hours. Because of this, the curve of the behaviour can be fitted to a gamma distribution. The gamma distribution is usually employed as a probability model for waiting times, in this case, the waiting time until the next event of the behaviour. So, a gamma distribution is suitable for modelling monotonous behaviours.

A kind of non monotonous behaviour, i.e. going to the toilet, is shown in Fig. 5 (c). In this case, the behaviour is fitted with an exponential distribution. Non monotonous behaviours are also waiting time models, but they are defined in wake periods. This is a special case of the gamma distribution which can be modelled with an exponential distribution.

Finally, the group of anytime behaviours are behaviours which will be often interrupted for the other behaviours. Because of this, size of intervals in an anytime behaviour are small due the interruptions. This causes the characteristic curved heavy-tailed distribution, specifically, the Pareto II distribution, also known as Lomax distribution.

Gamma, exponential and Lomax distributions are members of the exponential family of probability distributions. Actually, they are special cases of the beta prime distribution (also known as beta distribution of the second kind, Beta II). The Beta II distribution nests many important distributions as the gamma, the exponential or the Lomax distribution. The gamma and the Lomax distributions are special cases of Beta II. On the other hand, the exponential distribution is a special case of the gamma distribution $\Gamma(\alpha, \beta)$ when $\alpha = 1$, and a special case of the Lomax distribution with some restrictions [15], [16]. So, the Beta II distribution can be used here as a generalized distribution for all group of behaviours: monotonous, non monotonous and anytime behaviours. Then, each behaviour can be specified according to its features in order to obtain a better fitting.

Now, in the rest of the section, empirical evidences which support using gamma, exponential and Lomax for monotonous, non monotonous and anytime behaviours are given.

Notice that it is possible to estimate the distance between time series generated, as explained above, from real log data and time series generated by simulation of theoretical distributions. Such estimation is done by using some statistical test, like the Kolmogorov-Smirnov (K-S) test [17]. The K-S test is a nonparametric and distribution-free goodness-of-fit test. This means that they do not rely on parameter estimation or precise distributional assumptions [18]. The proposed model in this work does not assume any concrete distribution and does not require parameter estimation. This way, the K-S properties are suitable for the hypothesis test.

The K-S test and the chi-square test are the most commonly used and for large size sample both tests have the same power. However, the chi-square test requires a sufficient sample size in order to obtain a valid chi-square approximation [19], [20].

The K-S is a goodness-of-fit test to indicate whether it is reasonable or not to assume that a random sample comes from a specific distribution. It is a form of hypothesis testing where the null hypothesis says that sample data follows the stated distribution. The hypothesis regarding the distributional form is rejected if the test statistic, D_n, is greater than the critical value obtained from a table, or, which is the same, if the p-value is lower than the significance level. The significance level is fixed in this work at 0.05, which it is the value usually referred in statistical literature.

Table I shows the p-values obtained from the K-S test for each validated behaviour with the adequate distribution. The null hypothesis is that the behaviour sample data come from the stated distribution and it is rejected if p-value is lower than the significance level.

	Behaviour				
Person	Sleep	Dinner	Eat	Toilet	Spare time
A	0.404	0.311	0.361	0.111	0.086
B	0.488	0.467	0.542	0.079	0.108
C	0.337	0.489	0.575	0.137	0.103

TABLE I
P-VALUES

From these results, none of the stated null hypothesis can be rejected. Therefore behaviour of the attended people could be fitted by the specified distribution, as it is described above.

V. RELATED WORKS

Various approaches have been proposed to create autonomous characters. For example, in [21] every character is provided with a small KBS (Knowledge-Based System). Such method is very flexible, but defining the knowledge base is a complex and time-consuming task. A reasoning system is also used in [22].

The approach to behavioural autonomy presented in section III is based in [23], in that approach the idea is to develop agents that act and choose in the way actual humans do. The agents are represented using parametrized decision algorithms, and choose and calibrate these algorithms so that the agents' behaviour matches real human behaviour observed in the same decision context. For this purpose, they uses a parametrized learning automaton with a vector of actions associated that can be weighted to choose actions along the time the way humans would.

The decision of representing the automaton's transitions with probabilities instead of using a vector of strengths is based in [24], where the behaviour sequences are modelled through probabilistic automata (Probabilistic Finite-State Machine, PFSMs). Probabilistic personality influence implies that one cannot fully predict how a character will react to a stimulus.

In [25] and [26], the behavioural models described use a hierarchical structure of finite state automata similar as model described in section III. Each behaviour of a behaviour sequence is called a behaviour cell. At the top of the structure there is a behaviour entity with a finite state automaton composed of at least one behaviour cell. An elementary behaviour

is situated at the bottom of the hierarchical decomposition and encapsulates a specialized behaviour which directly controls one or more actions. The list of prioritized events is based in [27], where human agents have a pending task list. Priorities give more realism to human behaviours.

VI. Conclusion

In this work, a general behaviour model for different people is proposed. Adjusting the model to specific persons would imply using the suitable configuration parameters for the corresponding probability distributions governing transitions between states of the probabilistic automata. This process is part of a more general task which is testing AmI services and applications. For such purpose, the AVA methodology was presented. It has been applied for the validation of an AAL system called Necesity. More specifically, this paper is focused in producing models of humans (i.e. step 3 of AVA) and validation of the models (steps form 5 to 8).

Such an approach requires, as a means to validate the artificial behaviours, for all the artificial models, a method to check if gamma, exponential and Lomax distributions are well suited. Using the K-S test is possible to quantify the distance between the empirical distribution functions of two samples. Then the K-S test is used to validate the behaviours of the artificial models, by using real data of real elders. A high level of the p-value (higher that the significance level) means that the behaviours are drawn from the same distribution (the null hypothesis is not rejected). Notice that, in most cases, the obtained p-values are higher than the significance level and the null hypothesis is not rejected. But, it is also necessary to remark that a good fitting of the configuration parameters will always be required. And this is due to inevitable heterogeneity in behaviour of people (including elders). As a conclusion, it can be said that the proposed model is suitable for modeling probabilistically the behaviour of simulated people, as the work states.

Future works include a deep study of source data in order to generate a taxonomy of elders in terms of their behaviour within their houses, depending on mobility and habits. Such a taxonomy would be useful for an automatic parameter tuning of the models of elders. The user of the simulator, instead of configuring parameters by hand, would simply choose between a catalogue of elderly people patterns of behaviour.

References

[1] N. Gilbert and K. G. Troitzsch, *Simulation for the Social Scientist*. Open University Press, February 2005. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0335216013

[2] E. Aarts and J. L. Encarnação, "True visions: Tales on the realization of ambient intelligence," in *Into Ambient Intelligence, Chapter 1*. Springer Verlag, Berlin, Heidelberg, New York, 2005.

[3] G. J. Myers, C. Sandler, T. Badgett, and T. M. Thomas, *The Art of Software Testing, Second Edition*. Wiley, June 2004. [Online]. Available: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/0471469122

[4] J. A. Botía, A. Villa, J. T. Palma, D. Pérez, and E. Iborra., "Detecting domestic problems of elderly people: simple and unobstrusive sensors to generate the context of the attended." in *First Internationa Workshop on Ambient Assisted Living, IWAAL*, Salamanca, Spain, 2009.

[5] "Flowcharting techniques," *IBM GC20-8152-1 edition*, 1969.

[6] A. Drogoul, D. Vanbergue, and T. Meurisse, "Multi-agent based simulation: Where are the agents?" in *Proceedings of the Third International Workshop on Multi-Agent-Based Simulation MABS 2002, Bologna, Italy*, ser. LNAI 2581, J. S. Sichman, F. Bousquet, and P. Davidsson, Eds. Berlin Heidelberg: Springer Verlag, July 2002, pp. 1–15.

[7] B. Edmonds and S. Moss, "From kiss to kids - an 'anti-simplistic' modelling approach," in *MABS*, 2004, pp. 130–144.

[8] M. Rabin, "Probabilistic automata*," *Information and control*, vol. 6, no. 3, pp. 230–245, 1963.

[9] G. Darmois, "Sur les lois de probabilit a estimation exhaustive," *CR Acad. Sci. Paris*, vol. 260, pp. 1265–1266, 1935.

[10] B. Koopman, "On distributions admitting a sufficient statistic," *Transactions of the American Mathematical Society*, pp. 399–409, 1936.

[11] A. Law, W. Kelton, and W. Kelton, *Simulation modeling and analysis*. McGraw-Hill New York, 1991.

[12] K. Troitzsch, "Validating simulation models," *Networked Simulations and Simulated Networks*, pp. 265–270, 2004.

[13] M. Richiardi, R. Leombruni, N. Saam, and M. Sonnessa, "A common protocol for agent-based social simulation," *Journal of Artificial Societies and Social Simulation*, vol. 9, no. 1, p. 15, 2006.

[14] F. Klügl, "A validation methodology for agent-based simulations," in *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, 2008, pp. 39–43.

[15] J. B. McDonald, "Some generalized functions for the size distribution of income," *Econometrica*, vol. 52, no. 3, pp. 647–663, 1984.

[16] J. McDonald and Y. Xu, "A generalization of the beta distribution with applications," *Journal of Econometrics*, vol. 66, no. 1, pp. 133–152, 1995.

[17] H. Neave and P. Worthington, *Distribution-free tests*. Routledge London, 1989.

[18] D. Sheskin, *Handbook of parametric and nonparametric statistical procedures*. CRC Pr I Llc, 2004.

[19] F. Massey Jr, "The Kolmogorov-Smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[20] F. David and N. Johnson, "The probability integral transformation when parameters are estimated from the sample," *Biometrika*, vol. 35, no. 1-2, p. 182, 1948.

[21] G. Anastassakis, T. Panayiotopoulos, and T. Ritchings, "Virtual agent societies with the mVITAL intelligent agent system," in *Intelligent Virtual Agents*. Springer, 2001, pp. 112–125.

[22] H. Noser and D. Thalmann, "Towards autonomous synthetic actors," *Synthetic Worlds. TL Kunii and A. Luciani, John Wiley and Sons, Ltd*, 1995.

[23] W. Arthur, "On designing economic agents that behave like human agents," *Journal of Evolutionary Economics*, vol. 3, no. 1, pp. 1–22, 1993.

[24] L. Chittaro and M. Serra, "Behavioral programming of autonomous characters based on probabilistic automata and personality," *Computer Animation and Virtual Worlds*, vol. 15, no. 34, pp. 319–326, 2004.

[25] D. Thalmann, S. Musse, and M. Kallmann, "Virtual Humans' Behaviour: Individuals, Groups, and Crowds," *Proceedings of Digital Media Futures*, pp. 13–15, 1999.

[26] P. Bécheiraz and D. Thalmann, "A behavioral animation system for autonomous actors personified by emotions," in *Proceedings of the 1998 Workshop on Embodied Conversational Characters*. Citeseer, 1998.

[27] L. Temime, Y. Pannet, L. Kardas, L. Opatowski, D. Guillemot, and P. Bo "elle, "NOSOSIM: an agent-based model of pathogen circulation in a hospital ward," in *Proceedings of the 2009 Spring Simulation Multiconference*. Society for Computer Simulation International, 2009, pp. 1–8.