# Realistic Human Behaviour Simulation for Quantitative Ambient Intelligence Studies

Fabio Veronese[1], Andrea Masciadri[1], Anna A. Trofimova[1], Matteo Matteucci[1], and Fabio Salice[1]

[1]*Department of Electronics, Information and Bioengineering* - Politecnico di Milano - Polo Regionale di Como, via Anzani 42, 22100, Como, ITALY

December 6, 2016

**Abstract** Smart Homes technologies development is oriented toward intelligent services for the dweller. Designing the Artificial Intelligence which plays behind the scenes in a Smart Home requires large datasets for several reasons: training machine learning algorithms, tuning parameters, system testing and validation. Usually such tasks are carried-out on real-world data, requiring long time and additional costs to be collected, checked and labeled. Accelerating the development and limiting costs, a behaviour simulator can digitally reproduce environments and behaviours of the dwellers, in controlled conditions and in short time. This work presents a simulator capable of generating or reproducing the routine of a person in terms of Activities of Daily Living (ADLs). Moreover, the activity scheduling can be used to generate synthetic data from sensors deployed in a virtual environment. For the ADL schedule generation, an innovative model based on the person status (represented by needs) and habits is used, while two alternatives are proposed to generate home automation data: an agent-based model (with deterministic behavioural pattern descriptions) and a stochastic one (modeling the ambient response based on sample data activations distributions). The whole simulation/emulation chain is evaluated comparing the characteristics of the obtained data with a real world dataset. This comparison proves that synthetic data respect the distributions of the corresponding real world dataset ADLs and sensors activations.

## 1 Introduction

Currently, many research projects are addressed to the development of Artificial Intelligence (AI) systems. They concern the process leading from Home Automation (HA) [1], where appliances, lights, windows, shutters, heating system, etc. are automatic; to Smart Homes [1], where thanks to sensors an intelligent software (i.e. Ambient Intelligence - AmI) drives the environment automation; to Active and Assisted Living (formerly Ambient Assisted Living – AAL) [2], where AmI is directed to the assistance of the dweller. Usually, these projects need large amounts of data to train their AI, either to build their knowledge or to obtain parameters and configurations. These requests are usually addressed by building real world installations, test fields, pilot environments, or exploiting third parties datasets in which one or more persons live in a sensorized (home) environment. These solutions are viable, at the cost of volunteers recruiting campaigns, money investments, long acquisition times or, in case they are recorded by third parties, one has to find the right recording settings, meeting the needs of the intended AI design.

An alternative approach relies on the design of a simulator to generate synthetic data. Advantages regard the reduction of times and costs, and the possibility to define in detail the behaviour of the observed persons as well as the response of the environment. Generating synthetic data means to take into account the complexity of the emulated elements; in particular, it requires replicating the logic of spontaneous choice to perform home ADLs, as well as their natural scheduling over the day.

In this work the simulation of HA data for Ambient Intelligence studies is approached by splitting the task in two levels: the ADLs scheduling generation and the environmental sensor data simulation. The rest of this work is organized as follows: presentation of the state of the art for HA data collection and simulation, introduction of simulation models for the ADL scheduling and two for HA data generation, quantitative experimental results from data generation and simulation, conclusions and future works.

## 2 Previous Work

In recent years, the increase of computational power has enabled the development of a great quantity of software able to simulate real-life aspects by modeling the behaviour of the ecosystem actors and the environment response. In this section, the state of art of digital simulators concerning the behaviour of a person in an indoor (domestic) environment is discussed; in particular, the attention is devoted to software for domestic environments emulation and agent-based simulators. The distinction between Agent Based Modeling and Simulation (ABMS) and Smart Environment Simulators, even if not particularly sharp, helps to identify the main approaches that can be retrieved in previous works. The former tries to model the elements of the HA system and the users, as independent agents communicating and interacting with each other based on their relation. The latter, instead, focus on the detailed simulation of the ambient response rather than modeling in details the human behaviour, letting the user decide what the main agent should do.

## 2.1 Agent Based modeling and simulation

Agent-Based Modeling and Simulation (ABMS) is a modeling approach that has gained attention over the last 15 years. This trend is evidenced, as mentioned in Macal et al. [3], by the increasing number of publications and Agent-based software. ABMS is becoming widespread due to the following factors:

- *Modularity:* Defining the behaviour of each agent makes it possible to simulate, as a whole, more complex systems than with monolithic approaches. By splitting the complexity over more agents, it is possible to produce more detailed and realistic systems;
- *Interdependency:* ABMS makes it possible to model parts of the system as independent agents. In this way it is easier to manage interdependencies within the whole model as it happens with objects in object-oriented programming;
- *Computational Power:* Nowadays computational power is advancing rapidly permitting the computation of large-scale microsimulation models which were not plausible just a few years ago.

Unfortunately there is no universal agreement on the definition of agent in the context of ABMS research. According to Bonabeau et al. [4], an agent is an independent component within a simulation or a model. A stricter definition, given by Casti [5], states that the behaviour of an element must be both independent and adaptive to be considered an agent. In other words, an agent is an element that can learn from the environment and dynamically change its behaviour with respect to the context. According to Jennings et al. [6], the most important characteristic for an agent is autonomy in taking decisions. Macal et al. [3], specify other characteristics an agent is required, such as the interaction capability with both the environment and other agents; as a result, behaviour must depend both on the state in which the agent is and on the context in which it is collocated. Moreover, an agent bases its decisions on a feedback system, performing a continuous comparison between the expected results of its actions and the environment status. To let it decide what choice to take also a goal is needed.

In the literature, according to Macal et al. [7], it is possible to divide ABMS platforms into two different families: *General Tools* and *Specific Tools*. A General Tool is an instrument able to simulate agents behaviour, but it can also be used to code environmental aspects. This broad group includes, to some extent, also the high level programming languages such as Java, C++ or MATLAB. Indeed, by implementing rules it is possible to realize both simple and complex systems. According to Macal et al. [7], General Tools also include spreadsheets with macro programming, such as MS Excel, probably the simplest approach to modeling. As a case study, it is possible to cite the work made by Bower and Bunn [8]; in their work, Excel has been used as an ABMS tool to model and simulate bilateral market mechanisms for electricity trading. On the other hand, the use of MS Excel (as well as other General Tools) generally produces models with limited agent diversity, restricted agent behaviours, and poor scalability compared to other, specific, approaches. Specific Tools instead are purposefully programmed software applications, devoted to the simulation of Smart Homes. In the following we will report the state of the art situation about those software.

**NetLogo [9], [10]** is a free and open-source software under GPL license available on Github which provides a programmable modeling environment for the simulation of natural and social phenomena. It is particularly well suited for modeling complex systems evolving over time. It is written mostly in Scala, a scalable Object-Oriented language [11], while some parts are coded in Java. A classical example developed in NetLogo is the Wolf-Sheep Predation model [12] [13]; in this model, two families of agents, i.e., wolves and sheep, move randomly around the environment trying to survive, sheep are eating grass and avoiding wolves who are hunting them. Even if NetLogo is a powerful software able to scale with large amounts of agents, it reveals its limitations when trying to implement complex models or to configure fine details in the simulation.

**SWARM [14]** is a software engine and a library for the multi-agent simulation of complex systems. It was developed in 1994, at the Santa Fe Institute with a focus on artificial life applications and studies about complexity; it is the first ABMS software with hierarchical organization. A key aspect of this software is the conceptual division between the model coding part and the testing part; in this way, conducting experiments on the model does not need programming skills. SWARM includes also an important concept of hierarchy in which each context, called swarm, can contain lower levels of swarms which are integrated to the higher level. For all these reasons, SWARM is considered the most powerful and flexible simulation platform. However, it has had a long life: it was designed before Java had become a mature language. SWARM was coded in Objective-C, and nowadays it shows some issues related to the lack of strong typing and problems in data protection. Indeed, a malevolent user can monitor and control any simulation object, no matter how protected it is, also directly from the graphical interface.

**The Recursive Porous Agent Simulation Toolkit (Repast Symphony or Repast S for short) [15]** is a free, cross-platform and open-source agent-based modeling toolkit. It was developed at the University of Chicago's Social Science Research Computing Lab with a focus on agent based simulations in social sciences. It is very similar to Swarm, both in philosophy and appearance, providing some libraries for creating, running, displaying and collecting data from simulations. It was born as a Java re-coding of Swarm, however, with respect to Swarm, Repast has a moderate learning curve permitting even inexperienced users to build complex models. According to Grids ABMS comparison [16], Repast has the greatest number of functionality among ABMS packages, making it used and very popular.

**MASON [17]:** (Multi-Agent Simulator of Neighborhoods) is a free and open-source fast discrete-event multiagent library coded in Java, able to handle both heavy custom-purpose simulations and lightweight simulation tasks. MASON contains also a model library and visualization tools in 2D and 3D. It was designed to serve as the basis for a wide range of multi-agent simulation tasks, ranging from swarm robotics to machine learning and social complexity environments. It was born as a branch of the Repast project, but its popularity and documentation is not well recognized.

**Siafu [18]:** is a large-scale ABMS, written in Java and provided with a pleasant GUI. It makes it possible to reproduce worlds and scenarios, designing a modular world composed by three elements: agents, places and the context therein. It is worth noting that agents are represented as state machines in which status changes are triggered by context switches or by random factors. In particular, in complex contexts, simple random factors obstruct the realization of complex human behaviours due to limitations imposed by state machines. Moreover, Siafu makes it possible to gather context data for each agent, simplifying statistical analysis. Some implemented examples are the living simulation of a few inhabitants in a city scenario or in a smaller context such as the working life in an office.

All the reported works have a common ground, reproducing a designed action/activity in a virtual environment, permitting tests and evaluation of the Smart House response in those situations. However in most of these software environments the physiological variations in the every day routine quantities (e.g. the amount of time we spend to consume meals, how many times we go to the toilet, how much time we spend in bed, etc.) are not simulated, while in few they are simply randomly reproduced. The approach we propose differs from the state of the art introducing an innovative scheduling algorithm based on a model acting to balance needs and routine of the person.

## 2.2 Smart Environments Simulators

The other main approach to HA data simulation consists in designing the virtual representation of a building, a dwelling or even a single room. These can be called Smart Environment Simulators and their main aim is to reproduce the response of the virtual AmI installed in the emulated environment. As already introduced, the main drawback of this approach is the need for a user that triggers the virtual sensors activation or guides an avatar or an agent in the sensorized space, producing the desired sensor activity. Differently from ABMS, this category of simulators is more focused on the environment than on the behaviour of the dweller living in it.

**eHome [19]:** is a project developed in Aachen, Germany. The research team has implemented, via hardware, a set of advanced domotic services, such as “music-follow-person” and “comfort-wake-up”. The first one permits to the dweller, who decides to listen to music or podcasts inside an environment, to move freely among rooms without turning the speakers on and off. The system monitors the person position adjusting the volume and activating the speakers according to the room in which the user is located. While the “comfort-wake-up” service computes the optimal wake-up time, considering various static and dynamic factors, such as the first appointment of the day, usual breakfast time, scheduled showers and traffic jams. As a result, eHome tunes home appliances, such as the heating system, the morning alarm or the coffee machine for breakfast to be ready at the right time. The installation of advanced domotic services inside real homes requires an effort; therefore, the research team decided to test the system with a software simulator, substantially reducing test-time and costs. eHome is a point-and-click software for 2D indoor simulation: once started, the user can select a person inside the environment and walk her/him around rooms interacting with objects such as doors, chairs or appliances.

**ISS (Interactive Smart Home Simulator):** is a simulator developed at the Chonman National University [20], in Korea. It is able to reproduce a set of behaviours defined through rules on devices installed inside a digital home. Also in this case, the simulator has been designed to reduce the costs of a hardware implementation of the system. The main goal of the project is user tasks automation. The system behaviour is based on a set of if-then-else rules by which every object of the house can change its status. The user can not directly interact with the system, a function called "Generate Event" randomly extracts an action, such as "go to sleep", for the inhabitant person, updating the status of the involved devices.

**TATUS:** is an environmental 3D simulator developed at Trinity College, in Dublin. This simulator is able to reproduce smart technological features, such as the identification of people on room entrance and exit, or the recognition of behavioural patterns and intentions of those living and working in the space, providing services to authorized persons only. Generally speaking, it reproduces via software the context-awareness capability of smart devices within rooms. Context awareness is the property of pervasive computing systems to detect changes in the environment, and modify their response accordingly. Also the realization of TATUS comes from the recurring problems involving costs and logistics when implementing suitable test environments.

These simulators, on one hand are developed focusing on the virtual environment characteristics rather than the agent's, on the other, they provide a fixed behaviour of the environment, designable by the user, but not trainable on other real world data nor with probabilistic characterization. In this work we address also these two points, proposing an environment model trainable and with probabilistic connotation.

## 3 Real World Datasets

The need for a simulator comes from the need to reduce costs related to real-life data collection. Nevertheless, many research groups have collected, analyzed and published some sets of HA data for different purposes.

As discussed by Alemdar et al. [21] two Smart Home dataset categories can be identified. The first group is composed by systems with a high impact and influence on dwellers life. They require the installation of more than one hundred sensors within a smart environment, and, generally, these projects work on the human interaction with Smart environment and objects but they do not provide any healthcare focus. **Gator Tech Smart** house [22], **Georgia Tech Aware** home [23] and **PlaceLab** [24] are some examples of this category.

The second group is composed of low impact systems. In these contexts few sensors are used and they focus on human activity detection and health status monitoring. In Table 1 some data collection projects are summarized together with their collection modalities. Some of the projects reported in Table 1 consider more than one dweller, indeed the persons' recognition is a very complex topic if the system uses only common sensors. Experiments lasted from few hours to more than 12 months. This broad range of time durations is due to many aspects; first of all budget limitation, indeed involving people in these projects has high costs. The same holds true when considering the sensors

**Table 1:** *Smart Home projects comparison*

| Project | Multi Residents | Duration | Sensors | Activities | Occurrences |
|---|---|---|---|---|---|
| ARAS | Yes | 2 months | 20 | 27 | 1023-2177 |
| CASAS | Yes | 12+ months | 20-86 | 11-16 | 37-1513 |
| Van Kastereen | No | 28 days | 14 | 7 | 245 |
| UvA | No | 2 months | 14-21 | 10-16 | 200-344 |
| Domus | No | 11,5 hours | 78 | 0 (user feelings) | NA |
| Mit | No | 2 weeks | 77-84 | 13 | 176-278 |

number; even if systems have a low impact on dwellers life, they can be composed of a high number of sensors. Generally speaking, their number depends on the size of the smart environment and, if applicable, on the number of signals they want to recognize.

Simple sensors data need to be structured in complex clusters called activities. Even if all these projects study human life, they consider different activities performed by dwellers; a number between 7 and 27. **Domus** needs a special mention since the volunteers were asked to autonomously define their activities, to analyze and categorize them offline. For what concerns the number of occurrences of each activity, these are not proportional to the number of sensors readings but to their annotations. **ARAS** (Activity Recognition with Ambient Sensing) is a project for the automatic human ADLs recognition. The experiments consisted in two houses instrumented with 20 sensors able to capture dweller's activities and movements. These have provided a ground truth annotation for 27 different activities. In each house, ARAS staff has recorded a full month of information containing sensor data and activity labels, resulting in a total of two months data. Datasets are freely available on the project's website [25].

The **CASAS** Smart Home is a project developed at the Washington State University to provide an interdisciplinary research platform for intelligent environments. It is possible to identify two main goals towards the development of this project. The first one is the maximization of the user's comfort recognizing, discovering and tracking the user's activities. The second one is to minimize costs, such as those related to maintenance and energy saving. Datasets are freely available on the project's website [26].

T. **Van Kastereen** [27], at the University of Amsterdam, has installed, inside the real house of a 26 years old volunteer living alone, a set of 20 sensors. The aim of this project was to annotate person ADLs starting from a collection of binary data. The experiment has left 28 days, providing more than 2000 sensor events; obtained data have been annotated using two different approaches: Conditional Random Fields and Hidden Markov Model. Also in this case datasets are freely available on the project's website [28].

Gathering real world data has been proven by the previous experiences to be a challenge. Bureaucratic procedures, high hardware costs, very long data gathering time, high costs of installation, system upgrade and removal, or possible faults are issues that should not be underestimated. Moreover, the collected datasets present several drawbacks, and not always they fit research purposes.

## 4 Human Motivation Modeling

Even if employing simulators can represent an answer to the lack of data, a common weak point of the state of the art is that most of the available software rely on the user to decide the activity performed by the agent in the virtual environment. The simulator hereby presented, on the contrary, only leaves the user to configure few aggregate parameters describing the agent attitude, while taking care of generating a realistic daily schedule of human activities, based on such a defined profile.

The approach we propose is more suited for the description of human behaviour especially for long periods thanks to its characteristics. Indeed it avoids the design of a fixed ADL scheduling for the generation of all days, preferring to model a human-like choice of each ADL along the day, and including also a random variability to represent free will and modeling residuals. This makes the model more complete and mimicking the human decision making process.

As introduced, SHARON (Simulator of Human Activities, ROutines and Needs) is designed to reproduce the human behaviour in terms of ADL scheduling, by extracting, from an input set of activities, the most suited, based on a score function. This function is based on a two step modeling of the human motivation: the needs, making people perform ADLs in order to maintain wellness, and the habits, making more probable for someone to carry on certain ADLs at some specific times in the day, week, etc. Each ADL is further composed of sub-tasks, that are executed by the simulated user who interacts with the virtual environment sensors.

The set of needs $\mathcal{N}$ and activities configured in the present work are:

- **Hunger**, as the need for feeding;
- **Tiredness**, as the necessity to sleep;
- **Stress**, as the necessity to relax;
- **Boredom**, as the desire to have leisure time;
- **Loneliness**, as the need for social activities;
- **Uncleanness**, as the need to take care of the personal hygiene;
- **Excretion**, as the necessity to fulfill biological needs;

Moreover, it was necessary to consider also house related factors such as **Low Stock** and **Untidiness**.

Firstly, let $a$ be a generic activity in the set of ADLs, and $\mathbf{a}$ the one going on the current moment $t$. Each need $n \in \mathcal{N}$ is associated with a status $\sigma_n \in [0, 1]$, that is assumed to change linearly as the simulation evolves. This relationship expresses the dependency from a spontaneous growth rate $\gamma_n$ of such need and the effect $\zeta(\mathbf{a})$ of the ongoing activities on it as follows:

$$\sigma_n(t) = \sigma_n(t-1) + \gamma_n + \zeta(\mathbf{a}) \ . \tag{1}$$

The result is saturated, so to be bound inside the range $[0, 1]$. A need status affects the ADL scoring through a linear function $N_a(\cdot)$, representing the urge to perform a certain activity $a$ given the related person's needs $\sigma_n(t)$. This is implemented using a set of weights $w_{n,a} \in [0, 1]$ as follows:

$$N_a(\sigma_n(t)) \triangleq \sum_{n \in \mathcal{N}} w_{n,a} \cdot \sigma_n(t) \ . \tag{2}$$

However, this does not fully represent human activities motivation. Many activities are performed in a habitual or conventional time of the day. Examples can be tea time, main meals, or night time sleep. Such time dependency is modeled in this work by a function $\theta_a(t)$ which assumes a value in $[0, 1]$ for each simulation step in the day: the higher the value, the more likely the activity in that time quantum. To avoid deterministic series of events, the final likelihood $\Theta_a(t)$ is obtained through applying a probabilistic heuristic to $\theta_a(t)$. For each simulation step a random variable $u \approx \mathcal{U}\{0, 1\}$ is extracted:

$$\Theta_a(t) = \begin{cases} \theta_a(t) & u > \theta_a(t) \\ 1 & u \leq \theta_a(t) \end{cases} \tag{3}$$

The ADL most likely to be performed is the highest in a ranking identified by a metric driven by the time and needs factors hereby described. Considering the activity $a \in$ ADL at time $t$, its score function $S_a(t)$ is defined as:

$$S_a(t) = \alpha(\mathbf{a}) \cdot N_a(\sigma_n(t)) \cdot \Theta_a(t) \ ; \tag{4}$$

where $\alpha(\mathbf{a})$ represents a factor discouraging activity change (i.e., 1 for the ongoing activity $\mathbf{a}$ and 0.8 for all the others). The overall method for behaviour simulation can be formalized as in Algorithm 1.

## 4.1 Parameters and Training

The proposed method for behaviour simulation is based on a set of parameters, defined by the formerly explained equations.

ADL: the set of the performed/available activities;

$\mathcal{N}$: the set of needs, related to the defined set of ADLs;

**Algorithm 1** Overall ADL scheduling algorithm, extracting the activity $a$ to be performed

```
loop
  Update time instant $t$
  for each Need $n$ in $\mathcal{N}$ do
    $\sigma_n(t) = \sigma_n(t-1) + \gamma_n + \zeta(\mathbf{a})$
  end for
  for each ADL $a$ do
    if $a$ is active then
      $\alpha(a) = 1.0$
    else
      $\alpha(a) = 0.8$
    end if
    for each Need $n$ in $\mathcal{N}$ do
      $N_a(\sigma_n(t))$ += $w_{(n,a)} \cdot \sigma_n(t)$;
    end for
    Var rand = genRandom $\in$ [0, 1]
    if rand $< \theta_a(t)$ then
      $\Theta_a(t) = 1.0$
    else
      $\Theta_a(t) = \theta_a(t)$
    end if
    $S_a(t) = \alpha(a) \cdot \sum_{n \in \mathcal{N}} N_a(\sigma_n(t)) \cdot \Theta_a(t)$
  end for
  $\mathbf{a} = \underset{a}{\operatorname{argmax}}(S_a(t))$
end loop
```

$w_{n,a}$: representing the relations linking needs and activities;

$\gamma_n$: the spontaneous grow rate for each need;

$\zeta(\mathbf{a})$: the effect of each activity on specific needs;

$\theta_a(t)$: likelihood the user performs an ADL in a specific instant of the day.

Obtaining proper values for these parameters can be a challenging task. To perform such task two methods requiring a very limited effort can be proposed:

**Interview analysis:** thanks to a brief interview with the subject it is possible to gather a rough estimate of the usually performed ADLs, their time profile and the needs they satisfy. The collected information has a limited validity, since it is biased by the point of view of the interviewed person.

**Real data set / diary analysis:** having the possibility to collect information over a longer period of time it is possible to get more reliable estimates. For instance we could record data from a Smart Home installation (if available), or require the dweller to precisely fill in a diary. Quantitative values for the simulation can be extracted by analyzing the ADLs patterns in such data.

## 5 Modeling Home Automation Environment Response

One of the crucial aspects to make the simulator a powerful tool able to generate synthetic data is the translation of the virtual inhabitant's ADL into HA sensors activations. To obtain this, we need to model the environment, the agent, and their interaction, so that the generated data are a possible representation of the real-world data. In the following, the authors describe the core component of the SHARON simulator, which transforms the synthetic ADL scheduling into simulated HA data.

Two approaches have been adopted to describe the data generation: the first considers as a central aspect the inhabitant behaviour, while the second is based on the stochastic modeling of the data generation process. This double choice enables either the description of a completely artificial environment and agent, or the extension and modification of already existing datasets. In the following, the two are detailed and the resulting data analyzed.

### 5.1 Deterministic Agent-based Activity Pattern (DAAP)

The most intuitive model able to represent the activation mechanism of HA sensors along the execution of an ADL is based on the description of the inhabitant behaviour according to the specific ADL. An ADL, indeed, can be decomposed in a sequence of actions or tasks, each to be performed in a specific place of the house. Moreover, every sensor is supposed to activate, with a certain probability, whenever the agent is inside a specific area of the house. Thus, to represent an ADL, a pattern can be used which is made of different positions the agent visits for a specific amount of time, performing a given action.

### 5.1.1 Agent Modeling

The inhabitant is modeled as an agent, able to move around the house. No specific interaction with objects or sensors is defined, the presence of the agent is the only factor triggering changes in the environment. In particular the agent is characterized by a set of coordinates $\mathbf{x}_a = \{x_1, x_2\}$ that represent its position in the house, a walking speed $s_a$, quantifying the maximum distance between two subsequent positions. The path between places in the house is designed knowing the location of walls, doors, and furniture on a discrete map composed by tiles: the sequence of tiles connecting starting point to the target point with the minimum overall distance becomes the agent trajectory.

### 5.1.2 Environment Modeling

The environment of the virtual house is described by two main characteristics. A generic sensor $s$ is described by the tuple $\mathcal{D}_s : \{\mathbf{x}_s, \mathcal{A}_s, P_s\}$ representing respectively its position, its triggering area, and its status distribution probability. Since most of the sensors have binary status, in the following, they will be considered as *active* or *inactive*; the activation of the sensor is triggered when the agent is in its activation area ($\mathbf{x}_a \in \mathcal{A}_s$) according to the outcome of a Bernoulli trial $\mathcal{B}(1, p_s) = P_s$ outcome.

### 5.1.3 Pattern Modeling

Given that the human behaviour has a certain variability and that each high level ADL might correspond to a different set of actions or tasks (e.g., cooking two different recipes, or relaxing on the sofa or on the bed), several patterns of actions are associated to a single activity. To model such variability each pattern is associated to a probability of being chosen. As formerly mentioned the pattern $\mathcal{P}$ itself is a sequence of target positions $\mathbf{x}_{tk}$ for the agent and time durations $\theta_k$, thus formally $\mathcal{P} : \{p_c; < \mathbf{x}_{t1}, \theta_1 >, \ldots, < \mathbf{x}_t n, \theta_n >\}$. The actual pattern, as executed by the agent, implies that the agent first moves toward the target position, then it remains in such location for the necessary time, moving to the following target when the expected duration is passed. This enables not only the activation of the ADL-specific sensors, but also the transitory triggering caused by movements (e.g., motion sensors, doors opening, lights switching, etc.). Time lapses can have both a precise duration or a percentage of the overall ADL time: thus, due to the variability of the ADL duration, this requires an action to last at least a certain amount of time to be completed. The overall data generation algorithm is formalized in Algorithm 2.

### 5.1.4 Parameters Configuration

The agent based approach used to model the ADL execution allows the configuration of the agent simulation parameters in an easy and intuitive manner in case the goal is the generation of fully synthetic data. Indeed the characterization of the model using high level concepts enables the design of high level parameters, whose tuning can be easily done without referring to a precise set of data. As an example, to model the pattern of

**Algorithm 2** Data generation using the Agent-based modeling

```
Initialize variables
loop
    Update simulation time $t$
    if $t$ after the ending time of $a$ then
        Update $a$ from scheduling
        Get random pattern $\mathcal{P}_a$ based on $p_c$
    end if
    if $t$ after the current task ending $\tau_k$ then
        Get new task $k$ from $\mathcal{P}_a$
        Get new task location $\mathbf{x}_k$ from $\mathcal{P}_a$
        $\tau_k = t + \theta_k$ from $\mathcal{P}_a$
    end if
    if $\mathbf{x}_a \neq \mathbf{x}_k$ then
        Update $\mathbf{x}_a$ on the path toward $\mathbf{x}_k$ given $s_a$
    end if
    for each sensor $s \in \mathcal{S}$ do
        if $\mathbf{x}_a \in \mathcal{A}_s$ AND $P_s$ then
            Sensor $s$ is *active*
        else
            Sensor $s$ is *inactive*
        end if
    end for
end loop
```

the ADL *"Watching TV"* the first target location is the TV, which has to be switched on, taking a very short and fixed amount of time, then the rest of the time, the agent will be on the sofa. This description can be immediately translated in an execution pattern, moreover, considering that in the 75% of the cases the agent sits on the sofa and in the 25% it sits on the armchair, simply results in two similar patterns with different probabilities $p_c$.

## 5.2 Mixture of Markov Renewal Processes (MMRP)

The sensors behaviour response related to an activity execution can be also reproduced in a stochastic manner, with a model trained on a real dataset. Through such a model, it is possible to simulate the activations of binary sensors as the result of a set of stochastic decisions based only on probability distributions computed on data.

### 5.2.1 Home Automation Data Representation

The HA data collected in a smart home at a given time instant can be defined with a *sensorset* (SS): a vector of $N$ variables representing the status of all the sensors installed in the house. Formally, taking into account only binary sensors, a SS for every time instant $t$ can be denoted as:

$$ss(t) = \{s_1(t), \ \dots \ , s_N(t)\} \, , \quad s(t) \in \{0, 1\} \ ; \tag{5}$$

where 1 and 0 stand for active and inactive status respectively.

Analyzing a real-world dataset, the number of different SSs that appear is smaller than all the possible sensors' status combinations ($2^N$). This is due to the limited happening probability of unusual situations, for instance when the sensors are all active at the same time, as result of sensor correlation in activities. Anyway, considering an analysis of real-world data used for modeling with a sufficient extent, it can be assumed that all the relevant SSs have been collected.

### 5.2.2 Activity Templates Modeling

The human behaviour cannot be completely described at the sensors activations level only defining the ongoing ADL. Indeed, an activity can be performed in several ways, which differ in terms of duration and order of the performed sub-tasks. For example, the activity "Lunch" can be performed from the same actor as "Family lunch" which requires hours of work and several tools or "Fast lunch" just opening the fridge. To comply with this, it is advisable to introduce the concept of *Activity Template.* An Activity Template $\tau_{(k,a)}$ represents a specific way of performing an ADL $a$:

$$\tau_{(k,a)} \triangleq \left\{ p(\tau_{(k,a)}|a), \bar{p}_{\text{init}}(ss), \overline{\overline{SS}}, \lambda \right\} \ ; \tag{6}$$

it is characterized by the probability of being executed given the ongoing ADL (a priori probability) $p(\tau_{(k,a)}|a)$, the probabilities of the sensorsets to be the initial state $\bar{p}(ss_{\text{init}})$, the SSs transition probability matrix $\overline{\overline{SS}}$, and the SSs change rate $\lambda$.

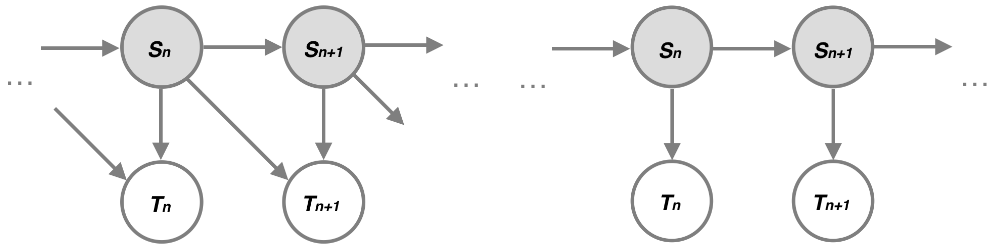


**Figure 1:** *Semi-Markov Chain representing HA sensorsets of a given Activity Template. Considering that the future SS duration does not depend in any way on the current SS, the standard Semi-Markov Chain (a) can be simplified as (b).*

In detail, the SS changes can be modeled by a stochastic random system with finite state space $E = \{ss_1, \dots, ss_R\}$, where states represent sensorsets and can be described by a Semi-Markov Model (SMM). Formally, if $S = (S_n)_{n\in\mathbb{N}}$ is a chain which records system states and $T = (T_n)_{n\in\mathbb{N}}$ is the sequence of discrete time intervals that the system spends in the states, the resulting discrete-time semi-Markov kernel [29] can be denoted by:

$$q_{ij}(k) \triangleq p(T_n = k, S_n = ss_j | S_{n-1} = ss_i) \ . \tag{7}$$

By taking into account that the time spent in a state does not depend on the previous states, the conditional independence assumption can be applied to modify the kernel as follows (Fig.1):

$$q_{ij}(k) \triangleq p(T_n = k, S_n = ss_j) p(S_n = ss_j | S_{n-1} = ss_i) \ ; \tag{8}$$

where $p(S_n = ss_j | S_{n-1} = ss_i)$ is a probability to be in state $ss_j$ if the previous state is $ss_i$ and $p(T_n = k, S_n = ss_j)$ is the probability that the system spends $k$ time units in state $ss_j$.

### 5.2.3 Sensorset Duration Modeling

The process of sensorsets change can be described with a sequence $J$ of time points in which the changes happen, such a sequence can be modeled as a Renewal Process $N(k)$[29]:

$$N(k) = \sum_{n=1}^{\infty} 1(J_n \leq k) \to N \sim \text{Pois}(\lambda) \ ; \tag{9}$$

i.e., $N$ follows a Poisson distribution with rate $\lambda$. Furthermore, according to the relation between $J$ (instants in which a SS change happens) and $T$ (waiting times) the distribution of the latter is defined as exponential [30]:

$$p(T_n = k, S_n = ss_j) = 1 - e^{-\lambda k} \ . \tag{10}$$

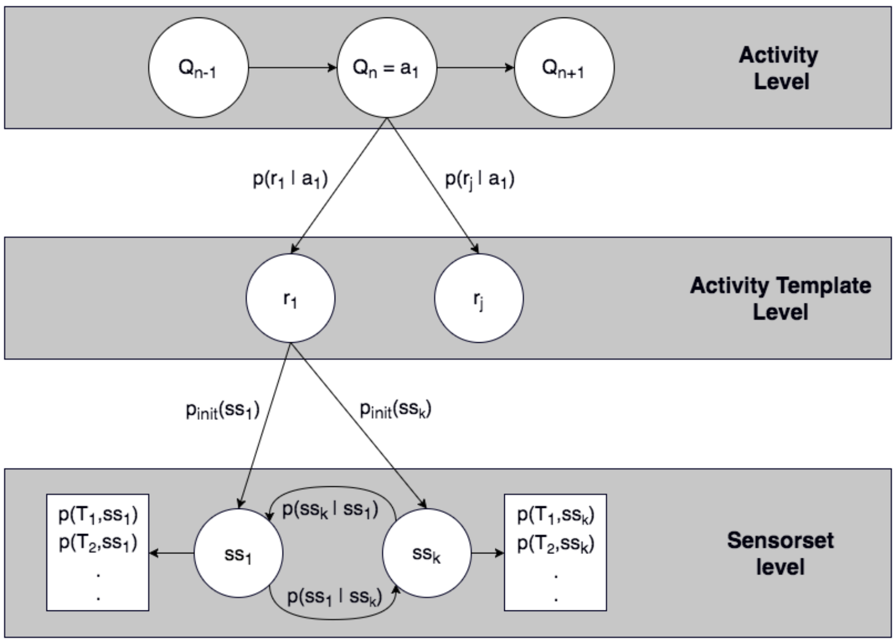


**Figure 2:** *Graphical representation of the HA simulator model in three layers: the activity template to be performed is chosen according to the current activity.*

In Figure 2 it is shown the graphical representation of the proposed model of the HA simulator divided in three layers, the first of which refers to the temporal sequence of activities. Given that the current activity $Q_n$ is the activity $a_1$, the activity template $r_i$ to be performed is given by the a priori probability $p(r_i|a_1)$, while the initial sensorset $ss_i$ is given by the initial probability $p_{init}(ss_i)$. Finally, the transitions at the sensorset level are given by the transition matrix $\overline{\overline{SS}}$ and the probability to have a waiting time $T$.

### 5.2.4 Parameters Configuration

Given that the proposed stochastic model needs the estimation of activity templates a priori probability $p(r_i)$, initial states probabilities $p_{init}(ss_i)$, SSs transition probability matrix $\overline{\overline{SS}}$, and SSs change rate $\lambda$, it is necessary to extract them from a set of real-world data. This procedure requires a HA dataset annotated with the ground-truth related to the performed activities. Each contiguous portion of the dataset, characterized by the same label, is called *activity instance.*

For every activity in the dataset it is possible, using the Jarvis-Patrick clustering method, to group the related activity instances in homogeneous groups $c_{(k,a)}$, which will be considered as activity templates. Jarvis-Patrick is a clustering method based on similarity between neighborhood data units [31]. It can identify arbitrary shaped clusters, that in case of no prior knowledge on shape, like in our case, becomes very import advantage. Moreover, it does not require explicitly specify a desired number of clusters and does not allow the absorption of small clusters by bigger ones. For such a method, we used a similarity metric based on SSs transitions. Thus, let $M$ be the transition matrix, where $M(i,j)$ is the number of transitions from $ss_i$ to $ss_j$. The distance measure between two activity instances $I_m$ and $I_n$ can be defined as follows:

$$d(I_m, I_n) = \sum_{i,j} (M_m(i,j) - M_n(i,j))^2. \tag{11}$$

For each cluster the first parameter to be computed is the a priori probability of the corresponding template. This can be derived by considering the number of instances in such a cluster, over the total number of instances of such activity. Formally for a generic activity instance $I_m$:

$$p(\tau_{(k,a_i)}) = \frac{\#(I_m \in c_{(k,a_i)})}{\sum_{\alpha=a_i} \#(I_m \in c_{(j,\alpha)})}. \tag{12}$$

Given that $\bar{p}_{\text{init}}(ss) = \{p_{\text{init}}(ss_1), \ldots, p_{\text{init}}(ss_R)\}$, the initial probability of a sensorset $p_{\text{init}}(ss_r)$, considering every instance $I \in c_{(k,a_i)}$ related to a template $\tau_{(k,a_i)}$, can be computed as:

$$p_{\text{init}}(ss_r) = \frac{\#(t : ss(t) = ss_r \wedge ss(t) \in I)}{\#(t : ss(t) \in I)}, \tag{13}$$

Defining $\overline{\overline{SS}}_{i,j}$ as the transition probability of $ss_i$ with given $ss_j$ within every instance $I \in c_{k,a_i}$ can be computed as follows:

$$\overline{\overline{SS}}_{i,j} = \frac{\#(t : ss(t-1) = ss_i \wedge ss(t) = ss_j \wedge ss(t) \in I)}{\sum_q^R \#(t : ss(t-1) = ss_i \wedge ss(t) = ss_q \wedge ss(t) \in I)}. \tag{14}$$

Finally, the rate $\lambda$ needed to calculate the waiting times distribution for the Poisson process $N(t)$ [32] (Eq.9) is given by:

$$\lim_{t \to \infty} \frac{\mathbb{E}[N(t)]}{t} = \lambda, \tag{15}$$

where $\mathbb{E}[\cdot]$ is the expected value.

## 5.3 Some notes on the proposed models

Some considerations can be done analyzing the characteristics of the two models even before producing any datum. The main difference between the proposed models is that DAAP is meant to be designed by hand while MMRP requires a dataset to be trained on.

Indeed to find a proper DAAP configuration it is necessary to represent the semantic of an ADL execution inside an activity pattern. This implies that the pattern can be tuned, also with fine adjustments, directly by the designer and modified directly. Moreover, the model needs also the design of the environment characteristics and the sensors characterization: even if it represents a potential source of unrealistic information, this still permits the creation of synthetic data without any real installation. Given that the process is based on a semantic level, it should be difficult to reproduce real data coming from a dataset. In such a case the data should be annotated with a great amount of information, concerning the environment, the sensors, the implemented activity characterization, etc.

Conversely, the MMRP model requires only the dataset with the annotation concerning the ADL performed. This enables to extend the dataset with homogeneous (but never identical) synthetic data, while neglecting information about the environment, the sensors, and the semantic of Activity Templates. Clearly this forbids tuning the parameters by hand, to modify them and in general to intervene on the configuration extracted from a single homogeneous dataset. As a mild drawback, MMRP will consider also the noisy portions of the dataset, potentially resulting in noisy data: even if realistic, this might severely affect the quality of the synthetic data. Thus, whereas needed, it is advisable to filter the data, inhibiting artifacts.

# 6 ADL Scheduling: Experimental Results

Before verifying the end-to-end results of the overall simulator, it is worthy to evaluate the ADL scheduling generation alone. The further section will be instead devoted to HA data simulation results and models comparison. To validate the effectiveness of the simulator in reproducing humans' behaviour, the model has been validated on the data set provided by ARAS [21]. This dataset consists of a detailed description of the activities performed by some dwellers during 30 days and on the data collected in their house. Such data have been divided into two sets: training set (23 days) and validation set (7 days). The training set was used to tune the system parameters and generate 300 simulated days. To reduce the complexity, ADLs were grouped semantically and reduced from 27 to 17.

## 6.1 Validation Metrics

As discussed in previous sections, human habits are described as a distribution profile in which, for each minute, a likelihood is provided for the system to perform a determined ADL. The purpose of the validation is to rate similarities between the distributions obtained during the simulations and the one extracted from the ARAS dataset. In the literature it is possible to identify numerous distances between statistical distributions. Many of them are not true metrics because they violate one or more of the three requirements of a metric, that of non-negativity, symmetry and triangular inequality. To the scope of this work, three different metrics have been employed to evaluate the results of simulations. As it will be detailed in the following paragraphs, each of them describes differently the distance between the statistical distributions of the activities: applying all of them makes it possible to better characterize such a distance.

### 6.1.1 Bhattacharyya distance

In statistics, the Bhattacharyya distance ([33], [34]) is a measure of similarity between two discrete probability distributions. It is related to the Bhattacharyya coefficient, which is a statistical measure of the overlap of two sets of samples. This metric is widely used in many fields, such as the extraction and the selection of features, image processing, phone clustering and speech recognition. The Bhattacharyya distance (BD) is defined as:

$$\mathrm{BD} = \left\{ 1 - \frac{\sum_i \sqrt{\{P_i \cdot Q_i\}}}{\sqrt{\sum_j P_j \cdot \sum_k Q_k}} \right\}^{\frac{1}{2}},$$

where $P_i$ and $Q_i$ are the two normalized histograms expressed as unidimensional arrays. BD varies between 0 (perfect match) and 1 (total mismatch); however tends to produce high distances even when distributions are still quite similar.

### 6.1.2 Earth Mover Distance

The Earth Mover's Distance (EMD) is a method to evaluate dissimilarity between two distributions. Intuitively, given two distributions, one can be seen as a mass of earth properly spread in space, the other as a set of holes in that same space. Then, the EMD measures the least amount of work required to fill the holes with earth. Within this reasoning, a unit of work corresponds to carrying a ground unit by a unit of distance from the ground.

Assuming a discrete domain, EMD can be computed by solving an instance of the transportation problem, applying the so-called Hungarian algorithm [35]. In particular, representing distributions as one-dimensional array of discrete areas (called "bins") the EMD is computed by keeping track of how much "earth" needs to be transported between adjacent bins. An interesting characteristic of EMD is that its cross-bin distance is not affected by single bin differences.

EMD is often used in the computer vision field for pattern recognition and to compare generic summaries or surrogates of data records called signatures. EMD, in its multidimensional computation formulation, has not found widespread application outside the computer vision community due to its prohibitive computational costs. However, applying in the mono-dimensional formulation, its complexity is affordable and it can be formalized as the following:

$$\mathrm{EMD}_0 = 0 \tag{16}$$

$$\mathrm{EMD}_{i+1} = (Ai + \mathrm{EMD}_i) - B_i \tag{17}$$

$$\mathrm{TotalDistance} = \sum_{i=0}^{I} |\mathrm{EMD}_i| \tag{18}$$

Where $A$ and $B$ represent the two discrete distributions, $\mathrm{EMD}_i$ is the partial EMD and "TotalDistance" is the final distance. EMD value is proportional to the distance between profiles. The smaller its value, the better are the obtained results. Its range has a lower bound 0 obtained when the two distribution are identical while it has no upper bound. Indeed, it depends both by the cardinality and by the values of the two distribution.

Being not bound to an interval, it better represent the distance when distributions are very dissimilar.

### 6.1.3 Kullback-Leibler divergence

Kullback-Liebler divergence is a special case of a broader class of divergences called f-divergences. This metric is a non-symmetric measure of the difference between two probability distributions P and Q. P represents the original distribution of data considered as reference, while Q is an approximation of P. In detail, the Kullback-Leibler divergence of P from Q, denoted as $\mathrm{KLD}(Q||P)$ measures the information loss obtained when Q approximates P. To interpret results it is possible to consider produced values as the entropy, the higher it is the broader are the differences. For discrete probability distributions it is defined as:

$$\mathrm{KLD}(Q||P) = \sum_{i=0}^{I} Q_i \cdot \ln \frac{Q_i}{P_i} \ . \tag{19}$$

## 6.2 Results

The distributions of the activities along the days (computed through histograms with 1440 bins, one per minute), were extracted from the generated data as well as from the training and validation datasets (Fig. 4,5). Then, Bhattacharyya Distance [36], Earth Mover's Distance [37] and Kullback-Leibler Divergence [38] between the simulation results and the validation set were computed. As visible in Table 2 the performances of the SHARON ADL simulation on the habitual actions are good, considering all the three used metrics. However, some ADLs are showing very high results. This can be explained computing the metrics also between the training and the validation sets: in such a way it emerges that in those cases the validation can only be a limited representation of less frequent ADLs since it does not comprise enough days. Indeed, it is possible to notice that the simulator behaves as expected, since the metrics of the simulated data are very close to the validation ones (Fig. 3).

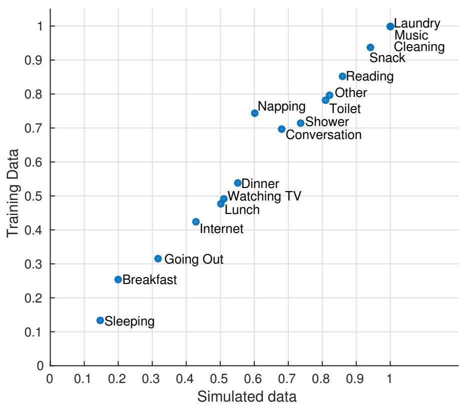
Moreover, when comparing the profile of distribution histograms (along the day) of an activity extracted from the simulated and ARAS real data, it comes clear that the SHARON simulator is reproducing the routines with a good approximation. In the reported histograms (Figures 4,5) the three sets of distributions are represented as follows: the orange line represents the simulated profile, the blue one the training set, and the gray one represents the test set.

## 6.3 Discussion

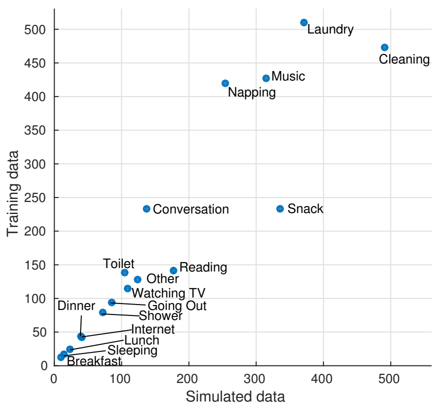
Wheres the state of the art simulators focus on the agent executing a set of codified and designed actions in a virtual space without a particular interest in the human motivation behind them, the methodology hereby proposed makes the motivation modeling the central point. Indeed building the daily actions balancing between the satisfaction of needs and the accordance to the personal habits, is a more realistic model compared

**Table 2:** *BD, EMD, and KLD computed between SHARON Simulated data and Validation set (**S**), and between Training set and Validation set (**T**). The lower the distances, the better the dataset is represented; the smaller the difference between results **S** and **T** the closer the simulation is to real data.*

| **DATASET** | S | T | S | T | S | T |
|---|---|---|---|---|---|---|
| **METRIC** | **BD** | | **EMD** | | **KLD** | |
| **Breakfast** | 0.19 | 0.25 | 9.69 | 13.16 | 0.28 | 0.67 |
| **Lunch** | 0.50 | 0.47 | 22.54 | 24.59 | 4.67 | 4.87 |
| **Dinner** | 0.55 | 0.53 | 40.44 | 41.68 | 4.24 | 3.66 |
| **Snack** | 0.94 | 0.93 | 335.23 | 233.38 | 11.88 | 11.38 |
| **Sleeping** | 0.14 | 0.13 | 14.69 | 17.43 | 0.05 | 0.05 |
| **Watching TV** | 0.51 | 0.49 | 109.39 | 115.39 | 3.80 | 3.88 |
| **Shower** | 0.73 | 0.71 | 71.31 | 79.29 | 7.86 | 7.49 |
| **Toilet** | 0.80 | 0.78 | 104.63 | 138.55 | 8.91 | 9.14 |
| **Napping** | 0.60 | 0.74 | 254.12 | 419.23 | 7.21 | 8.70 |
| **Internet** | 0.42 | 0.42 | 38.68 | 43.67 | 2.56 | 2.60 |
| **Reading** | 0.85 | 0.85 | 177.24 | 140.83 | 10.07 | 9.95 |
| **Laundry** | 1.00 | 1.00 | 370.58 | 510.73 | 14.17 | 13.45 |
| **Phone** | 0.67 | 0.69 | 136.68 | 233.11 | 7.12 | 7.58 |
| **Music** | 1.00 | 1.00 | 314.33 | 427.16 | 13.57 | 13.63 |
| **Cleaning** | 1.00 | 1.00 | 490.8 | 473.72 | 14.62 | 13.99 |
| **Going Out** | 0.31 | 0.31 | 85.97 | 94.38 | 1.58 | 1.59 |
| **Other** | 0.82 | 0.79 | 124.38 | 127.76 | 9.34 | 9.13 |



**(a)**



**(b)**

**Figure 3:** *Graphical representation of Bhattacharyya (a) and Earth Mover Distance (b) for each ADL*

to the state of the art where usually the scheduling is deterministic with superimposed random variations.

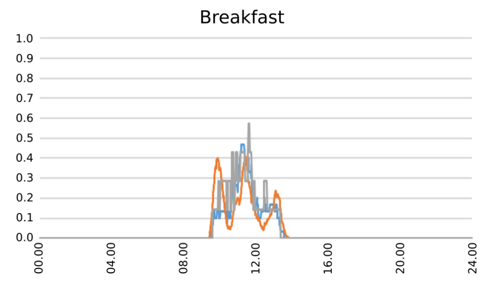
The model used in SHARON is moreover interesting since it adds a further level of semantics in the choice of ADL scheduling, replicating with simplicity complex phenomena (e.g., actions saturations, uniform distribution along a time window, etc.). Moreover, a key point with respect to the state of the art is the possibility to train the agent behaviour on real world data, obtaining an *extension* of the given dataset with any desired length.

Results on quantitative experiments prove that the behaviour replication is reliable. In particular it is clear that even designing only a restricted set of parameters it is possible to synthesize the actions distribution. Numeric evidences in Table 2 show that the activities daily distributions have been successfully replicated in most of the cases, with limitations where the actions instances in the dataset were only few.

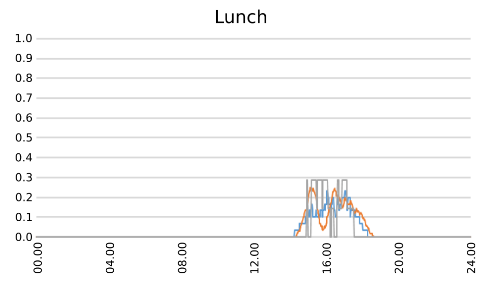
The application of the proposed method can contribute significantly to the simulation of realistic human domestic behaviour, bringing benefits to all the fields where it is interesting to replicate inhabitant's behaviour

## 7 HA Data Simulation: Experimental Evaluation

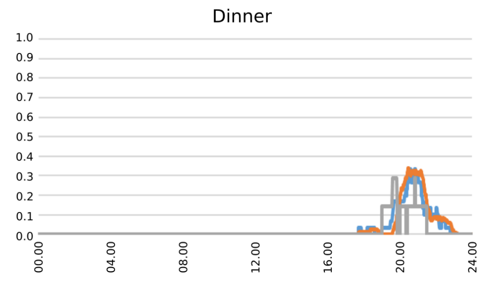
To complete the description of the simulator characteristics it is necessary to consider an experimental evaluation of the home automation simulated data. To such an aim the following section provides both qualitative and quantitative comparison of the two proposed models, i.e. the agent based and the stochastic ones, when trying to (re)produce the activity of home automation sensors generated by a specific ADL execution.



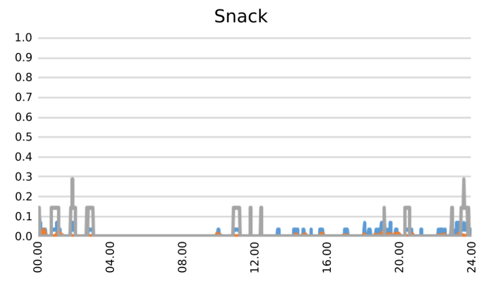
*Breakfast ADL is quite well defined in the later morning. The computed simulation covers well this trend.*



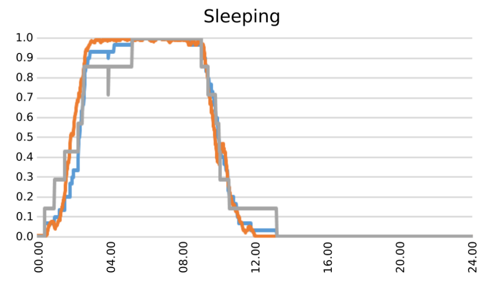
*Lunch ADL is defined by two nearby peaks during the first afternoon. The simulation is positioned within the same lapse of time.*



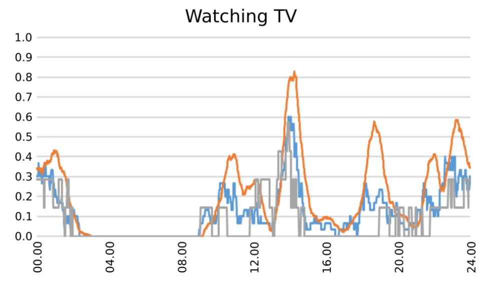
*Dinner ADL has a regular shape. Indeed it has only a peak centered in 1200 (8 P.M.).*



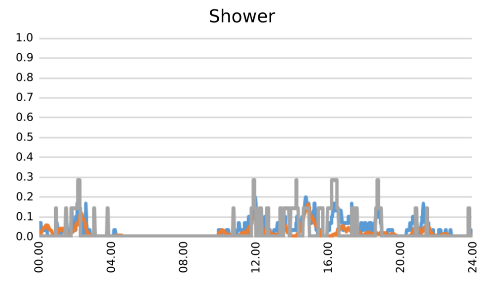
*Snack ADL is quite a random activity. Indeed, excluding the sleeping period, the dweller performs this ADL without showing any predominant trend.*



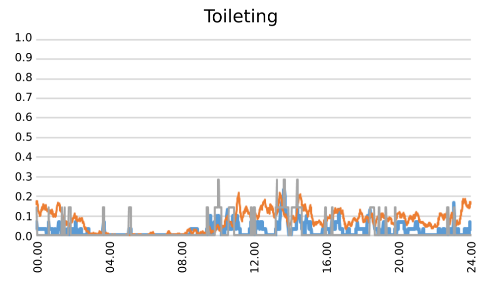
*Sleeping ADL has a very regular shape. Indeed, the training, the simulated and the test sets overlap with high accuracy.*



*Watching TV ADL has 5 peaks distributed during the day. The simulation follows quite good all these peaks.*

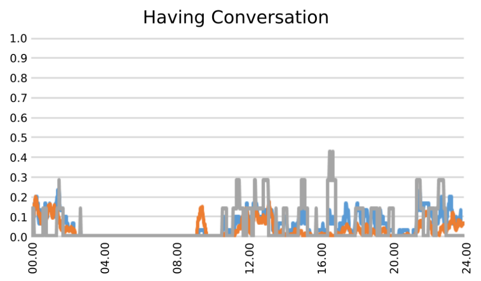


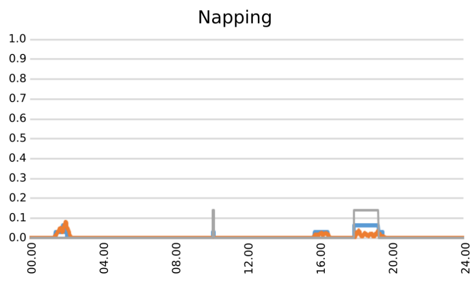
*Shower ADL presents peaks distributed over the day. Simulation overlaps quite well the original set by enhancing broadest peaks.*

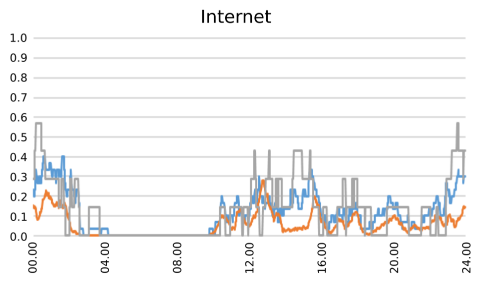


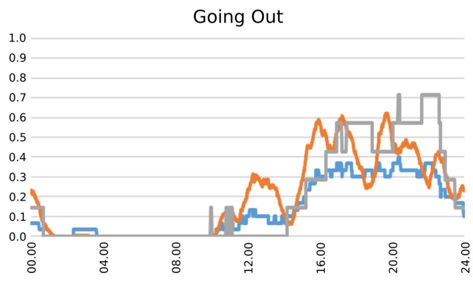
*Excretion ADL, as expected, has a quite random distribution. The simulation replicates the uniform distribution along the day, with almost no nightly episode.*

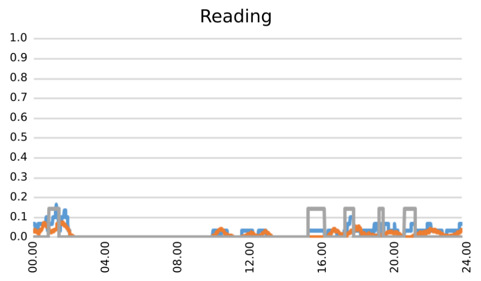
**Figure 4:** *ADL Time Distributions along the Day (1/2). Color codes are: orange synthetic, blue training, and gray validation data.*

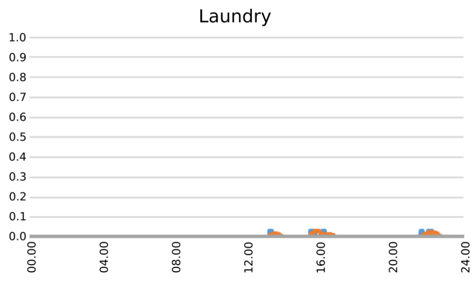

*Having Conversation ADL has a quite random distribution during the day.*

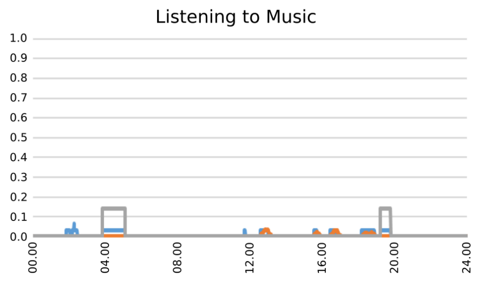

*Napping ADL consists of four isolated peaks, with very small magnitude.*


*Internet ADL is well spread during day and a before going to sleep. Also in this case the three sets overlap quite well.*


*Going Out ADL is characterized by a broader contribute during the late morning and during the afternoon: simulated data show an enhanced peak in the morning.*


*Reading ADL is uniformly distributed during the day excluding areas around 870 (2 P.M.).*


*Laundry ADL along the data set, is not very well represented. Indeed, in 23 days it has been performed rarely.*


*Music ADL is a quite rare activity. Indeed, there are only few peaks distributed along the whole day excluding the morning.*

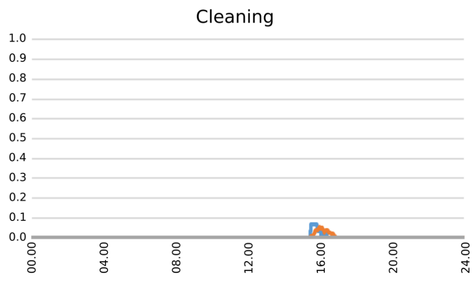

*Cleaning ADL is characterized by a single peak and it is quite rare on the analyzed time period.*

**Figure 5:** *ADL Time Distributions along the Day(2/2). Color codes are: orange synthetic, blue training, and gray validation data.*

## 7.1 Choice of ADLs for Methods Comparison

To design the simulator evaluation it is important to consider that each ADL has different characteristics in terms of repeatability and randomness in its execution. In particular, some ADLs have a strong procedural connotation, while others are made of events whose semantic is conserved even when their happening order changes. Since the two proposed methods are exploiting complementary approaches to reproduce repeatability and randomness, the choice of the ADLs to evaluate against was done to emphasize those aspects: *Cleaning* (where the sequence is almost random), *Lunch* (where several executions are different, but keeping an overall procedure), and *Having Shower* (where the procedural connotation is strong).

## 7.2 Experimental Settings

The setup of the experimental comparison hereby described requires a good quality real world dataset, annotated with ADLs, where a single person is monitored through a set of HA sensors: as already reported, none of the available datasets complied with such requirements off the shelf. Due to this, a filtered version of the ARAS dataset [21] was employed, where only one person is considered, and the sensors activations generated by the other person's actions are neglected. Moreover, as already exploited for the ADL simulation evaluation (Section 6), the reduced set of ADL labels was used.

Considering the filtered ARAS dataset as a starting point, the configuration of MMRP is straightforward and it has been performed according to what has been described in Section 5.2.3. To configure properly the DAAP generator instead, a set of patterns based on the semantics of the ADLs executions is required as other characteristics of the data which are not available. The DAAP configuration was thus designed extracting the missing information through a classification-based procedure, pushing to the limit the characteristics of the DAAP model, intended to generate data rather than reproduce existing ones.

The two generated datasets were both obtained based on the filtered ARAS dataset ADL scheduling, so that the durations and the frequency of the ADL instances in all the data were the same. The comparison of the real and generated data was performed by comparing the probability distribution of the sensors activity along the ADL duration rescaled to 2048 time bins. Moreover, the distribution of the sensors uninterrupted active intervals was computed as well.

### 7.2.1 Results

Considering the plotted distributions of the sensors (Fig. 6,7,8) it appears clearly that the more random and stochastic activations of the original dataset are better reproduced by the MMRP model. Conversely, as expected, the deterministic patterns of DAAP permit to better emulate the execution of procedural ADLs.

Observing the *Having Shower* ADL sensors activity distributions in Fig. 6 it emerges how the data generated with DAAP have a more evident procedural semantic: the bathroom door is closed for the whole ADL duration, the shower cabinet is open at the

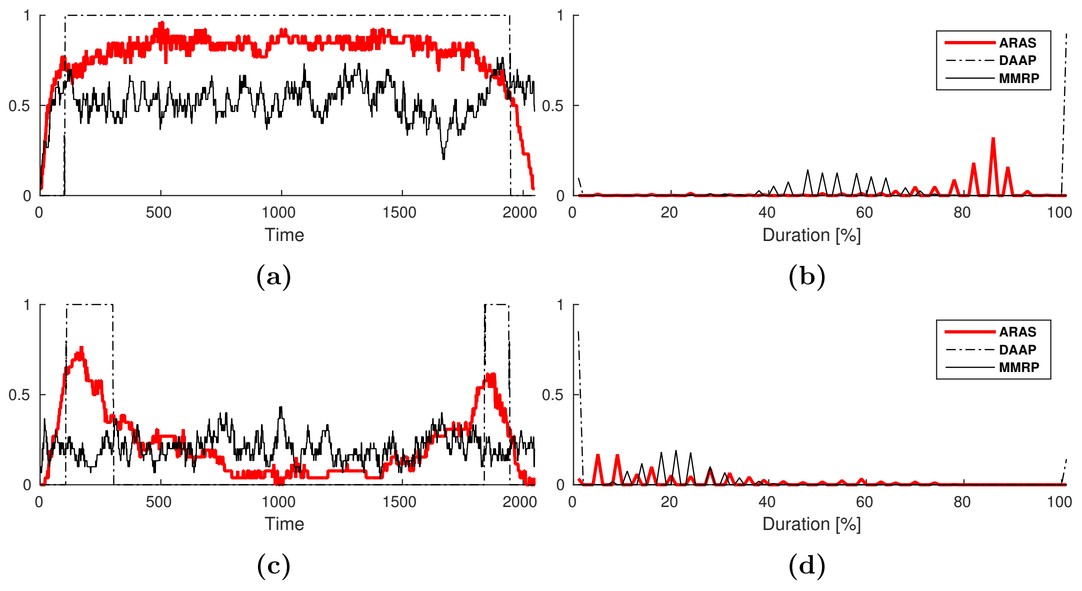


**Figure 6:** Having Shower *sensors activations distributions: on the left activation probability distribution along ADL duration, on the right sensors activation duration distribution with respect to the activity execution time. The top row (a,b) refers to the bathroom door sensor (active when closed), the bottom (c,d) to the shower cabinet door sensor (active when open)*

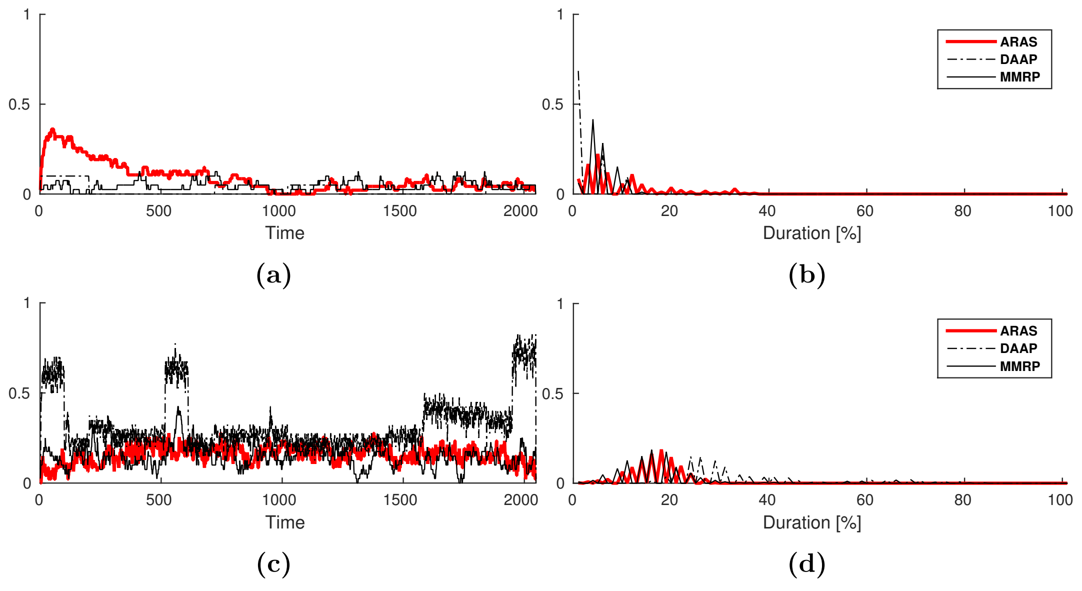


**Figure 7:** Lunch *sensors activations distributions: on the left activation probability distribution along ADL duration, on the right sensors activation duration distribution with respect to the activity execution time. The top row (a,b) refers to the fridge door sensor (active when open), the bottom (c,d) to the kitchen motion sensor*

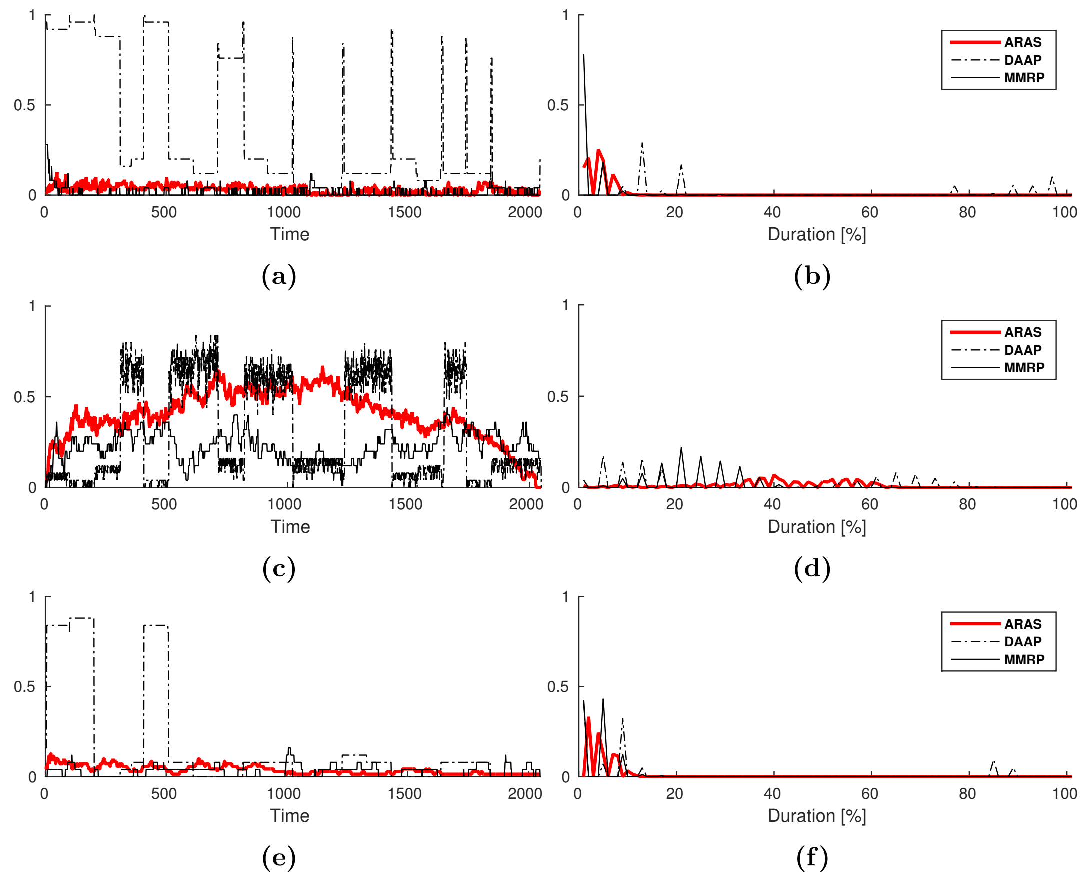


**Figure 8:** Cleaning *sensors activations distributions: on the left activation probability distribution along ADL duration, on the right sensors activation duration distribution with respect to the activity execution time. The top row (a,b) refers to the hall motion sensor, the central row (d,f) to the kitchen motion sensor, the bottom (e,f) to the fridge door sensor (active when open)*

**Table 3:** *Quantitative results of the sensors activation distribution comparison. All the results refer to BD of the involved sensor (where applicable) with respect to the ARAS dataset distributions: smaller values represent closer distributions.*

| **ADL** | **Lunch** | | **Shower** | | **Cleaning** | |
|---|---|---|---|---|---|---|
| **DATASET** | **DAAP** | **MMRP** | **DAAP** | **MMRP** | **DAAP** | **MMRP** |
| **Couch** | 0.34 | 0.06 | - | - | 0.79 | 0.43 |
| **Chair 1** | 0.38 | 0.29 | - | - | - | - |
| **Chair 2** | 0.25 | 0.47 | - | - | 0.47 | 0.59 |
| **Fridge** | 0.66 | 0.41 | - | - | 0.54 | 0.54 |
| **K. Drawer** | 0.74 | 0.60 | - | - | - | - |
| **B. Door** | - | - | 0.16 | 0.11 | - | - |
| **Shower** | - | - | 0.63 | 0.34 | - | - |
| **Hall** | 0.83 | 0.89 | - | - | 0.36 | 0.77 |
| **K. Mov.** | 0.22 | 0.20 | - | - | 0.33 | 0.21 |
| **Tap** | - | - | - | - | 0.94 | 0.54 |
| **K. Temp.** | 0.18 | 0.19 | - | - | - | - |
| **Average** | **0.45** | **0.39** | **0.40** | **0.22** | **0.57** | **0.51** |

beginning and at the end, when the person gets in and out of it. The MMRP generated data conversely show better results in modeling the ADL *Cleaning* (Fig. 8) where the randomness of the execution pattern has more relevance.

To give a quantitative evaluation of the generated data, it was also computed the Bhattacharyya distances between the sensor activation distribution in the ARAS dataset and both DAAP and MMRP for each of the involved sensor, as reported in Table 3. In particular it is worth noting how the scores, in many occasions widely below 0.5, confirm the adequate reproduction of the sensors activations in both cases. However, the different approaches of the two models are visible also through the numeric results, where too random or too deterministic activations are characterized by higher distance values.

## 7.3 Discussion

In the second part of this work we propose two approaches for the generation of HA data. The DAAP method is similar to the state of the art, comprising the design of the agent actions and environment response. Moreover, it is possible to configure different implementations of the same ADL, among which the agent is going to chose, based on a configurable ratio. This approach thus permits the creation of a virtual environment from scratch, without any ground truth data or initial dataset.

However, in some settings it would be advisable to extend an existent HA dataset, or to add some modification in the inhabitant behavior. MMRP makes it possible to implement these features, exploiting a model that can be trained from real world data.

It is worthy to note that having both DAAP and MMRP makes it possible to generate also hybrid datasets, where activities are either implemented by the deterministic agent or through the stochastic approach.

## 8 Conclusions and Future Works

In this work we have presented a quantitative method to reproduce behaviours based on two main factors: the temporal dependency (routines, habits and conventions) and the personal needs evolution during the day. The relationship between the two is mathematically described through a score function, representing the probability of an action to be performed in that moment of the day. The resulting method is employed by SHARON (Simulator of Human Activities, ROutines and Needs), which can be employed to replicate humans ADLs routines. To validate the presented method the ARAS dataset was used, considering 23 days as training and the latter 7 as validation. Based on the training set, 300 days were simulated and their ADLs distribution was compared with one extracted from validation set, using three different metrics. The obtained results show a good match with the original distributions especially when looking at the more regular ADLs.

Moreover, to complete SHARON functionality, two models are proposed to produce synthetic Home Automation data: DAAP generates data based on the response of a synthetic environment stimulated by the execution of ADL-patterns by an agent, MMRP leverages an ad-hoc designed stochastic model to be trained on a dataset. Experimental results show the diversity of the two approaches, confirming how DAAP is more suitable to design manually semantic-based patters, while MMRP better describes and reproduces randomness and stochastic executions.

SHARON and both the HA simulation engines are also available for download on the ATG website: `http:\\atg.deib.polimi.it`

Concerning future developments, the research will focus on the modeling of multiple simultaneous actions, to ensure realistic simulations, introducing the ADLs overlapping (e.g. watch TV during a meal). Moreover, it would be interesting to consider multiple users simulation: human behaviour is characterized also by the interaction among multiple agents, thus it would generate particular ADLs patterns (e.g., alternation, cooperation, exclusion, etc.). Allowing inhabitants to interact with other agents (other people or pets) the simulator realism will increase considerably.

## References

[1] Augusto JC, Callaghan V, Cook D, Kameas A, Satoh I. Intelligent environments: a manifesto. Human-Centric Computing and Information Sciences. 2013;3(1):1–18.

[2] Fuchsberger V; ACM. Ambient assisted living: elderly people's needs and how to face them. 2008;p. 21–24.

[3] Macal CM, North MJ. Agent-based modeling and simulation. In: Winter simulation conference. Winter Simulation Conference; 2009. p. 86–98.

[4] Camazine S, Deneubourg J, Franks N, Sneyd J, Theraulaz G. E. Bonabeau (2001) Self-Organization in Biological Systems. Princeton University PressCamazineSelf-organization in biological systems2001. 2001;.

[5] Casti J. Would-be worlds: how simulation is changing the world of science. New York: Wiley; 1997.

[6] Jennings NR. On agent-based software engineering. Artificial intelligence. 2000;117(2):277–296.

[7] Macal CM, North MJ. Tutorial on agent-based modeling and simulation. In: Proceedings of the 37th conference on Winter simulation. Winter Simulation Conference; 2005. p. 2–15.

[8] Bower J, Bunn DW. Model based comparison of pool and bilateral markets for electricity. The energy journal. 2000;p. 1–29.

[9] NetLogo source code; 2015. https://github.com/NetLogo/NetLogo.

[10] NetLogo Website; 2015. https://ccl.northwestern.edu/netlogo/.

[11] SCALA Homepage; 2015. http://www.scala-lang.org/.

[12] Lotka AJ. Analytical note on certain rhythmic relations in organic systems. Proceedings of the National Academy of Sciences. 1920;6(7):410–415.

[13] Volterra V. Variazioni e fluttuazioni del numero d'individui in specie animali conviventi (Variations and fluctuations of the number of individuals in animal species living together). C. Ferrari; 1927.

[14] SWARM project homepage; 2015. http://www.swarm.org.

[15] Repast Simphony project homepage; 2015. http://repast.sourceforge.net/.

[16] Grids ABMS Comparison; 2015. http://www.grids.ac.uk/Complex/ABMS/.

[17] Mason project website; 2015. http://cs.gmu.edu/∼eclab/projects/mason.

[18] Martin M, Nurmi P. A generic large scale simulator for ubiquitous computing. In: Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on. IEEE; 2006. p. 1–3.

[19] Armac I, Retkowitz D. Simulation of smart environments. In: Pervasive Services, IEEE International Conference on. IEEE; 2007. p. 257–266.

[20] Van Nguyen T, Kim JG, Choi D. Iss: the interactive smart home simulator. In: Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on. vol. 3. IEEE; 2009. p. 1828–1833.

[21] Alemdar H, Ertan H, Incel OD, Ersoy C. ARAS human activity datasets in multiple homes with multiple residents. In: Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2013 7th International Conference on. IEEE; 2013. p. 232–235.

[22] Helal S, Mann W, El-Zabadani H, King J, Kaddoura Y, Jansen E. The gator tech smart house: A programmable pervasive space. Computer. 2005;38(3):50–60.

[23] Kientz JA, Patel SN, Jones B, Price E, Mynatt ED, Abowd GD. The georgia tech aware home. In: CHI 08 extended abstracts on Human factors in computing systems. ACM; 2008. p. 3675–3680.

[24] Intille SS, Larson K, Beaudin J, Nawyn J, Tapia EM, Kaushik P. A living laboratory for the design and evaluation of ubiquitous computing technologies. In: CHI 05 extended abstracts on Human factors in computing systems. ACM; 2005. p. 1941–1944.

[25] ARAS dataset project website; 2015. http://netlab.boun.edu.tr/WiSe/aras/.

[26] CASAS dataset website; 2015. http://ailab.wsu.edu/casas/datasets/ :.

[27] Van Kasteren T, Noulas A, Englebienne G, Kröse B. Accurate activity recognition in a home setting. In: Proceedings of the 10th international conference on Ubiquitous computing. ACM; 2008. p. 1–9.

[28] Tim van Kasteren Dataset website; 2015. https://sites.google.com/site/tim0306/datasets.

[29] Barbu VS, Limnios N. Semi-Markov chains and hidden semi-Markov models toward applications: their use in reliability and DNA analysis. vol. 191. Springer Science & Business Media; 2009.

[30] Grinstead CM, Snell JL. Introduction to probability. American Mathematical Soc.; 2012.

[31] Jarvis RA, Patrick EA. Clustering using a similarity measure based on shared near neighbors. IEEE Transactions on Computers. 1973;100(11):1025–1034.

[32] Serfozo R. Basics of applied stochastic processes. Springer Science & Business Media; 2009.

[33] Wilson DH, Atkeson C. Simultaneous tracking and activity recognition (STAR) using many anonymous, binary sensors. In: Pervasive computing. Springer; 2005. p. 62–79.

[34] Applegate D, Dasu T, Krishnan S, Urbanek S. Unsupervised clustering of multidimensional distributions using earth mover distance. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2011. p. 636–644.

[35] Jonker R, Volgenant T. Improving the Hungarian assignment algorithm. Operations Research Letters. 1986;5(4):171–175.

[36] Bhattacharyya A. On a measure of divergence between two multinomial populations. Sankhyā: The Indian Journal of Statistics. 1946;p. 401–406.

[37] Hitchcock FL. The distribution of a product from several sources to numerous localities. J Math Phys. 1941;20(2):224–230.

[38] Kullback S, Leibler RA. On information and sufficiency. The annals of mathematical statistics. 1951;p. 79–86.