A Fuzzy Based Verification Agent for the Persim Human Activity Simulator in Ambient Intelligent Environments

Amr Elfaham, Hani Hagras, *Senior Member, IEEE*, Sumi Helal, *Senior Member, IEEE*, Shantonu Hossain, Jae Woong Lee and Diane Cook, *Fellow, IEEE*

Abstract— The generation of useful sensory data from real-world deployments of Ambient Intelligent Environments (AIEs) is challenging because of the high cost, significant groundwork and lack of access to human subjects. This situation can be improved by providing efficient simulators that can produce realistic simulation of the data collection from AIEs. One of the main problems for developing AIE simulators lies in the ability to verify how close the simulated data are to the real world data. In this paper, we present a fuzzy based verification agent for Persim – an event driven simulator for human activities in AIEs. The employed fuzzy based verification agent builds a data model that mimics the operation of Persim which allows for the latter's objective and subjective verification. We have conducted the verification on real world data captured from an actual smart apartment deployment. The results show the effectiveness of the fuzzy based verification agent in analyzing and comparing the Persim simulated data with the real world collected data. We also demonstrate how the verification agent is able to pinpoint specific changes to the simulation model to increase the realism of the simulation.

I. Introduction

With the ever-increasing number of miniaturized and computerized artefacts and devices, information about state, location, roles and much more becomes transparent and with the pervasiveness of networks this information is made available to anyone, anywhere and at any time. These efforts of advancing technology to pervade everyday life and to foster wide availability and acceptance materialized in 1991 when Mark Weiser introduced his vision of ubiquitous computing in his famous seminal article "The Computer for the 21st Century" [24].

Ubiquitous computing aims to make computers aware of the needs of the user. In other words, the ubiquitous computing system can be regarded as a digital personal assistant equipped with some sort of intelligence to understand what the users are trying to accomplish, in order to determine how best to intervene and assist them.

This work was supported in part by the US National Institute of Health (NIH) Grant number1R21-DA024294-01.

A. ElFaham is with the German University in Cairo, Egypt.

H. Hagras is with the German University in Cairo, Egypt, he is also with the Computational Intelligence Centre, School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester, CO43SQ, UK. (e-mail: hani@essex.ac.uk).

S. Helal, S. Hossain and J.W. Lee are with the Computer and Information Science & Engineering Department, University of Florida, USA (e-mail: helal@cise.ufl.edu).

D. Cook is with the School of Electrical Engineering and Computer Science at Washington State University, USA (email: cook@eecs.wsu.edu).

Ambient Intelligence (AmI) is a new paradigm that puts forward the criteria for the design of smart spaces and ubiquitous computing environments [21]. In the AmI paradigm, intelligent computation will be invisibly embedded into our everyday environments through a pervasive transparent infrastructure (consisting of a multitude of sensors, actuators, processors and networks), which is capable of recognizing, responding and adapting to individuals in a seamless and unobtrusive way [9]. AmI offers great opportunities for an enormous number of applications such as health and elder care, the efficient use of energy resources in homes and public buildings, and in leisure and entertainment. However, there are many challenges facing the creation of Ambient Intelligent Environments (AIEs) which include:

- **Expensive Development.** In recent years it has become obvious that the increasing costs of building AIEs without a correct design and plan makes research initiation and progress in this area extremely difficult [12]. Not everybody has a large budget to build an AIE to test new algorithms and ideas.
- **Time Consuming Data Generation.** Even if budget is not an issue, it is usually very time consuming to generate adequate data for a meaningful collection of patterns or events. For instance some of the available data sets are useful to some researchers but not all, depending on the events captured and the specific sensors that were available in the AIE where the data was collected.
- **Scarce Human Resource.** Another difficulty is recruiting participants to test the AIE and to perform all of the activities under all possible conditions or contexts that a research team wishes to consider. Addressing human subject safety and guarding against abuse and exploitation, institutional review boards (IRBs) and many governmental agencies limit the length of time human subjects can be used in any research study. Although financial and human capital may be available, the range of data that could be collected would have to be restricted, leaving researchers with only a tiny fraction of the data they wish to collect.

Given the aforementioned challenges, it is necessary to look for alternative practical ways to experiment with AIE design and performance. Simulation is a promising and sensible alternative. It enables researchers to create focused synthetic replications of important events and activities.

Researchers could gain deep insight into the specifics of the AIE they may eventually build. Simulations can be easily changed and refined allowing the researchers to experiment, analyze and fine-tune their model and associated algorithms. Simulation also allows a wider community of researchers to engage and collaborate to solve a specific problem. Hence, a design based on preliminary simulation studies would most likely to be a more robust and inclusive design. Also, a simulation model that mimics an existing real world AIE is most likely to answer more questions (generate much more data) than the actual AIE. This early stage simulation can help researchers evaluate their ideas and algorithms quickly and with reasonable accuracy.

There is a major roadblock that impedes the use of simulation in AIE which is: How do researchers know that the simulation is a reasonable capture of the real AIE? Simulating an AIE is a complex task. It encompasses a capture of sensors, actuators, activities, behaviors, space elements and semantics, as well as a large collection of dependent and independent events. This complexity creates room for inaccuracies and errors in the simulation model. Hence checks and balances are needed to ensure that simulation-generated datasets are adequately realistic and comparable to real datasets that would result from the actual AIE they simulate.

In this paper, we present a fuzzy verification agent and a framework for verifying AIE simulators. Our verification agent is capable of analyzing and quantifying similarities between an AIE simulated dataset and the real world dataset that the simulator is emulating. We demonstrate our verification agent in the context of a case study to verify the design of an event-driven simulator called Persim [12], [13], [19]. We present experiments which verify Persim output in comparison with a real world dataset that was collected to capture information about human activities in a smart living space.

The rest of the paper is organized as follows. Section II presents related work on verification of AIE simulators. Section III presents a brief overview on the Persim simulator used in this paper. In Section IV, we present the fuzzy based verification agent. Section V presents the experiments and results followed by conclusions and future work in Section VI.

II. Previous work in AIE Simulation Verification

In the fields of wireless sensor networks (WSN) and pervasive computing, many simulation concepts have been researched and various simulation tools have been built. But they differ in the focus and fields of application. SENSORIA [2] is a simulator focusing on traffic generation, energy consumption and inherent protocols of WSN. In [23], a detailed simulation model was presented which also focuses on accurate models for battery, processor power consumption and network traffic. In [1], the Discrete-Event system Specification (DEVS) was proposed to define asynchronous discrete-events occurring in WSN. None of the above mentioned simulators provide any verification of simulation output against real data.

On the other hand, there are some approaches such as [14], [17] which simulate a specific system at a very low level. For example, TOSSIM [17] simulates the TinyOS operating system in a WSN. TOSSIM emulates the operating system operation step by step. As such it embodies a self-validation by simply comparing the performance of its simulation traces with an actual WSN running TinyOS.

The DiaSim simulator [15] executes pervasive computing applications by using an emulation layer and developing simulation logic of the parts that needs to be simulated. Under DiaSim, a pervasive computing environment is described in terms of "stimulus" procedures (any change in the environment that can be consumed by sensors) and simulated services in a specification language called DiaSpec. Unlike Persim, DiaSim does not attempt to model an entire pervasive space for analysis and examination. It does emulate some of the pervasive system services (sensors and actuators) as part of a programming and development methodology. The programmers start off emulating almost all services and gradually, as they verify the correctness of application logic, replace emulated entities by actual devices. As such DiaSim's verification goals are different from Persim. It focuses on validating the application logic and making sure the pervasive application does what it is intended to do. Unlike DiaSim, Persim verification, which is introduced in this paper focuses on ensuring high realism of the simulation. Our approach uses a fuzzy based agent to directly measure the level of fidelity between the simulated system and its real counterpart in terms of their corresponding datasets.

III. Overview of the Persim Simulator Design

Persim is an event-driven simulator developed at the Pervasive Computing laboratory at the University of Florida. It focuses on simulating human activities [10] in ambient intelligent environments [12], [13], [19]. It is capable of capturing the physical elements of AIEs in terms of its sensors and user behaviors (activities). It allows researchers to simulate AIEs by first defining a target ambient space (e.g., Washington State's CASAS Smart Apartment Testbed [7] or the Gator Tech Smart House [4], [11]). The researcher adds a variety of sensors into the space. Activities of interest that can happen inside the space are then added. When the simulation design is complete, the researcher has the flexibility to define causal relationships between activities and sensors. In the last step, one has the choice to simulate the entire state space (similar to the dataset format of the iDorm AIE project of University of Essex [5]) or zoom into the space and simulate only activities of interest (similar to the CASAS dataset of WSU [7]). In Persim, researchers are empowered to design a simulation incrementally over multiple sessions. They may modify the design several times before a design is realized that adequately captures the target AIE. They can also modify the design to specialize the simulation for specific individual experiments. Persim

simulated data follow the Sensory Data Description Language (SDDL) standard [12], [19], which significantly enables sharing of the simulation results.

A. *Brief Description of the Simulation Model*

Persim is a component-based simulator. These components are used for defining (i) space to be simulated in terms of layout and sensors, (ii) activities to be simulated and (iii) simulation criteria/configuration. Each component is characterized by several attributes. The major components of the simulator are Space, Sensor, Activity, Activity-Sensor Mapper and Simulation Configurator. Persim supports two simulation modes – *Activity-driven* and *State-space*. In *Activity-driven* mode, a user can specify a set of activities, which would trigger a set of dependent sensors based on the *Activity-Sensor mapping*. On the other hand, in *State-space* mode, Persim generates a time-stamped sequence of events reporting on all sensor values, which are independent of any activity.

Persim adopts the discrete-event simulation model [16] using a next-event, time-advance approach to capture the dynamic nature of the AIEs, which evolve over time. According to this classical simulation technique, the target state space changes whenever any event (either activity-driven or time-driven) occurs and the system variables of the space are updated based on the simulation logic.

The flow chart of the Persim simulation algorithm is shown in Fig. 1. The simulation invokes the timing routine to get the most imminent event from an *event list*. Then it processes the event according to the type of the event, whether simulation mode is *Activity-driven* or *State-space*. It also advances the *simulation clock* to the time of occurrence of the current event. The simulation continues until the predefined value for the *simulation clock* is reached. Finally, it generates simulated data in SDDL form, which is a proposed standard format for representing sensory datasets [12], [19].

B. *Simulation Steps*

Fig. 2 shows a screen shot of the Persim parameter configuration interface. The user first defines the space to be simulated (e.g., kitchen, living room). Then she/he can add the desired sensors (e.g., motion sensor, light sensor) in each space area and configure the sensors with information such as the sensor name, id, type, value generation function, min/max value, and so forth. Then the user can add AIE activities (e.g. move to kitchen, clean dishes).

Next, the user needs to map each activity to a set of relevant sensors using the *Activity-Sensor Mapper*. While mapping, a user can specify the sequence of sensor-triggers for each activity. For example, if motion sensor M1 is assigned a sequence number 1 and motion sensor M2 is assigned a sequence number 2 for a specific activity, then M1 will trigger before M2 in the course of simulation of that activity.

Finally, the user needs to define several simulation parameters as shown in Fig 2, such as simulation mode, activities to be simulated with an inter-arrival distribution function, and start time and end time of an activity. This completes all information required for the simulation of events in the space. Now the user can click the "Run Simulation" button to generate data from the space.

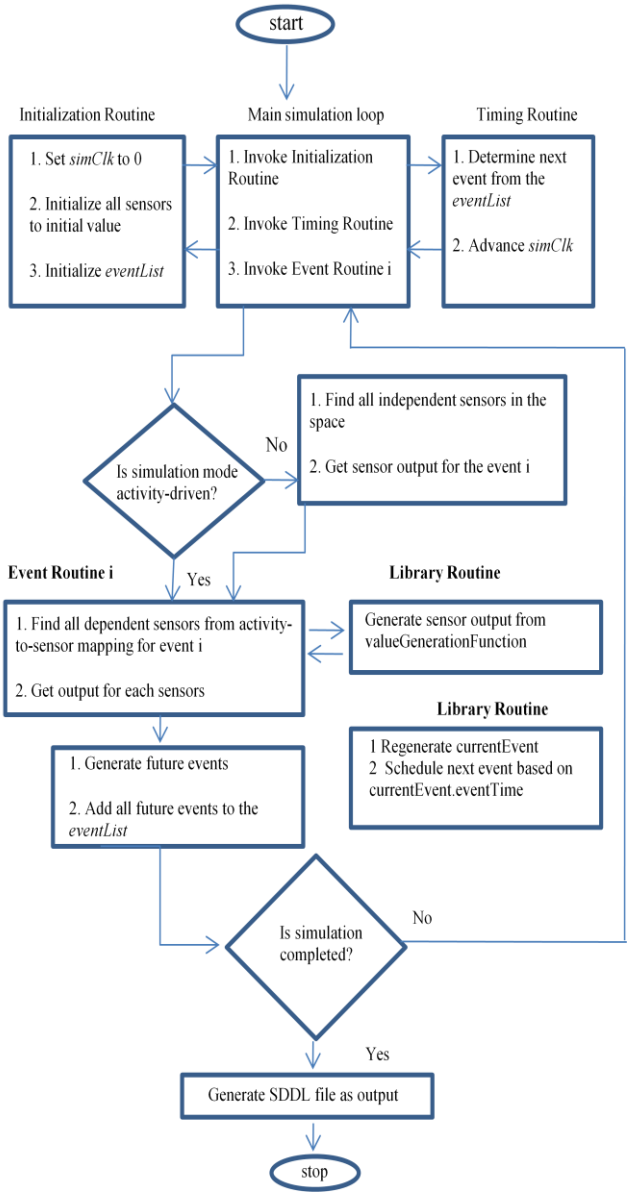


Fig.1. Flowchart of Persim event simulation.

IV. Overview of the Fuzzy Verification Agent

In order to verify the accuracy of the Persim simulator, we have followed a verification approach, which employs fuzzy logic based modeling.

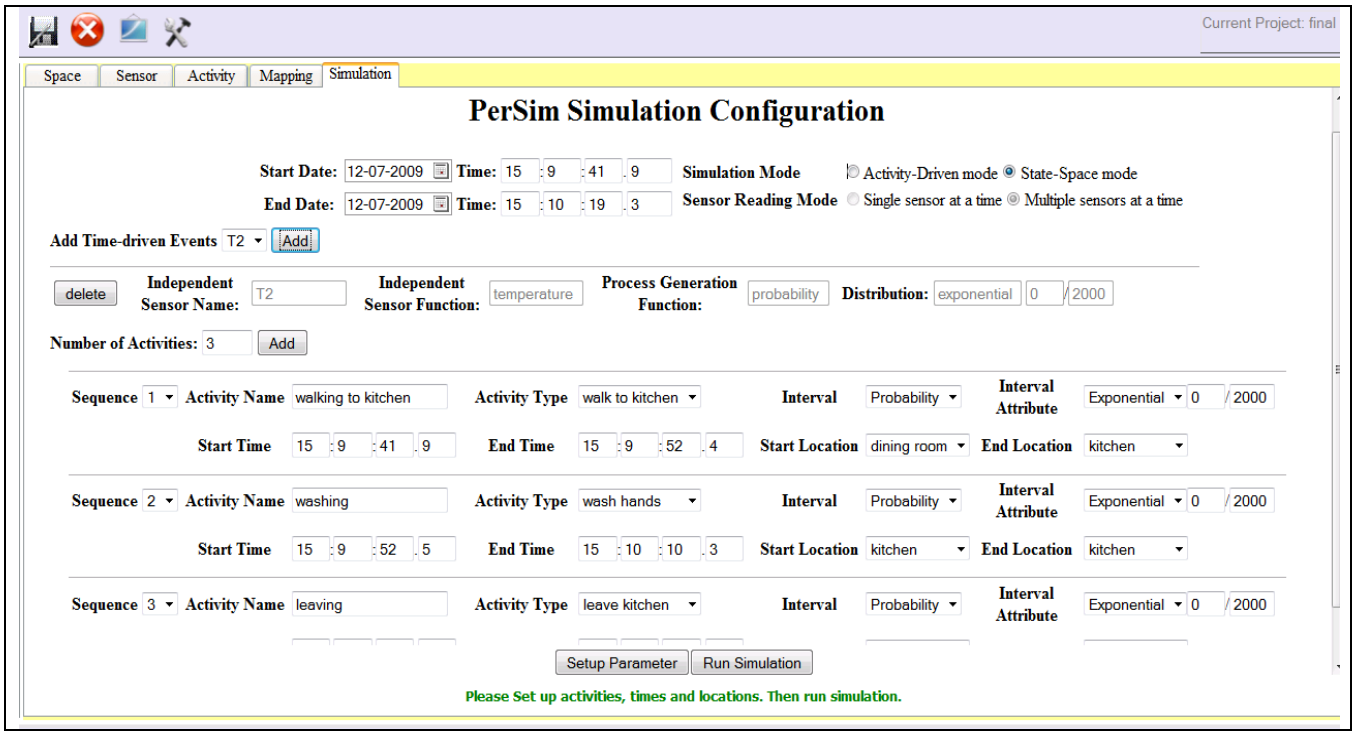


Fig.2. Persim simulation configuration.

Fuzzy Logic Systems (FLSs) attempt to mimic the way of human thinking to reason in an approximate way rather than a precise way. The smooth transition between the fuzzy sets will give a good decision response when facing noise and uncertainties. Furthermore, FLSs employ linguistic IF-THEN rules, which enables a representation of the control information in a human readable form.

The employed verification approach uses data to construct a fuzzy logic based system that models the given process and gives mapping from the data (real or simulated) to the correct activities. The motivation behind this approach is that just by using data, we can generate fuzzy models, which could be easily read and interpreted by the user.

Fig. 3 shows an overview of the fuzzy based verification agent. The agents start in ***Phase 1*** by aggregating the simulation and real data logs to generate the fuzzy sets for the simulated and real FLSs, respectively. In ***Phase 2***, a sliding window approach is employed to generate the rulebases of the simulated and real FLSs respectively. In ***Phase 3***, the agent perfoms a numerical verification by feeding the real data to the simulation FLS and then verifying how close are the outputs of the simulated FLS to the real data. In ***Phase 4***, the agent perfoms a linguistic verification by comparing the fuzzy sets and rule bases of the real and simulated FLSs. In the following subsections, we presnet phases 1 and 2 whereas phases 3,4 will be further illustrated in the experiments section.

A. Phase 1: Learning the Fuzzy Sets of Simulation and Real FLSs

Phase 1 deals with learning fuzzy sets for the simulation and real FLSs from the accumuated simulation and real logs, respectively. Both the simulated and real data logs share the same time frames. We employ a Fuzzy C-means Clustering approach to learn the numerical values associated with the various fuzzy sets as follows:

Consider a family of fuzzy sets $A_i, i = 1, 2, \dots, c$, as fuzzy c-partitions on the universe X. Fuzzy sets allow a degree of membership, hence, we can assign memberships to the various data in each fuzzy cluster and a single data point can have a membership to more than one cluster [3]. The membership value of the kth data point in the ith cluster is described as follows:

$$\mu_{ik} = \mu_{A_i}(x_k) \in [0,1] \tag{1}$$

The objective function for optimal fuzzy c-partition can be written as follows:

$$J_m(U, v) = \sum_{k=1}^{n} \sum_{i=1}^{c} (\mu_{ik})^{m'} (d_{ik})^2 \tag{2}$$

Where $i = 1,2, \dots, c$ and $k = 1, 2, \dots, n$, where c is the number of centres or fuzzy sets and n is the number of data points. Where

$$d_{ik} = d(x_k - v_i) = \left[\sum_{j=1}^{m} (x_{kj} - v_{ij})^2 \right]^{\frac{1}{2}} \tag{3}$$

The membership μ_{ik} is the membership of the kth data point in the ith cluster. The goal of the objective function is to get the best clustering. The parameter $m'$is the weighting parameter which has a range $m' \in [0, \infty)$. This parameter controls the amount of fuzziness in the classification process and usually takes the values between $[1.25, 2]$. In our

experiments m' was set to 2. The vector for a cluster center $v_i = \{v_{i1}, v_{i2}, \ldots, v_{im}\}$ is calculated as follows:

$$v_{ij} = \frac{\sum_{k=1}^{n} \mu_{ik}^{m'} x_{kj}}{\sum_{k=1}^{n} \mu_{ik}^{m'}} \quad (4)$$

Where j is the variable on the feature space, i.e. $j = 1, 2, .. m$. The Fuzzy C-means clustering operate as follows:

1. Fix c $(2 \le c < n)$ and select a value for parameter m'. Initialize the partition matrix $U^{(r)}$, $r = 0,1,2, \ldots$
2. Calculate the c center vectors $\{v_i^r\}$ for each step.
3. Update $U^{(r)}$, the partition matrix in the rth step; calculate the updated membership function matrix as follows:

$$\mu_{ik}^{(r+1)} = \left[\sum_{j=1}^{c} \left(\frac{d_{ik}^{(r)}}{d_{jk}^{(r)}} \right)^{\frac{2}{m'-1}} \right]^{-1} \quad for\ I_k = \emptyset \quad (5)$$

$$\mu_{ik}^{(r+1)} = 0 \quad for\ all\ classes\ i\ where\ i \in \tilde{I}_k \quad (6)$$

$I_k = \{i \,|2 \le c < n\,; d_{ik}^{(r)} = 0\}, \tilde{I}_k = \{1, 2, \ldots, c\} - I_k$.

4. If $\left\| U^{(r+1)} - U^{(r)} \right\| \le \varepsilon$ (tolerance level) STOP, otherwise set $r = r + 1$ and return to step 2.

We perform the above algorithm to get the cluster centres and then each data point is matched against the various cluster centres to form the shape of the given fuzzy set as shown in Fig.4. The shapes of the fuzzy sets are then smoothed by piecewise linear membership function as shown in Fig.4.

B. Phase 2: Learning the Fuzzy Rule Bases of the Simulated and Real FLSs

We employ a sliding window approach as shown in Fig.5 to pass through the data and generate fuzzy rules. Each training data pattern ($x^{(t)}$;$y^{(t)}$) will consist of the $x^{(t)}$ for the given sensors within the given window and the associated activity $y^{(t)}$. In case of Boolean sensors like motion sensors, the value of $x^{(t)}$ will be a binary value indicating if the relevant sensor has been triggered within the window or not, in relation to the sensor changing from OFF to ON indicating that someone has crossed the relevant area.

The fuzzy rule extraction approach used in this paper is similar to the AOFIS system reported in [8]. In the following steps we summarize the different steps involved in rule extraction:

Step 1: For a fixed input-output pair ($x^{(t)}$;$y^{(t)}$) in the dataset (t=1,2, ...,N), compute the membership $\mu_{A_s^q}(x_s^{(t)})$ values q=1,2, ...,V and for each input variable s (s=1,2, ...,n) find $q^* \in \{1, \ldots V\}$, such that

$$\mu_{A_s^{q^*}}(x_s^{(t)}) \ge \mu_{A_s^q}(x_s^{(t)}) \quad (7)$$

Let the following rule be called the rule generated by ($x^{(t)}$;$y^{(t)}$):

IF x_1^t is $A_1^{q^*}$ *and ... and* x_n^t is $A_n^{q^*}$, *THEN* y *is activity* $y^{(t)}$ (8)

For each input variable x_s there are V fuzzy sets $A_s^q, q = 1, \ldots, V$, to characterize it; so that the maximum number of possible rules that can be generated is V^n. However given the dataset only those rules among the V^n possibilities whose dominant region contains at least one data point will be generated. In step 1, one rule is generated for each input–output data pair, where for each input the fuzzy set that achieves the maximum membership value at the data point is selected as the one in the IF part of the rule. This however is not the final rule, which will be calculated in the next step. The weight of the rule is computed as:

$$w^{(t)} = \prod_{s=1}^{n} \mu_{A_s^q}(x_s(t)) \quad (9)$$

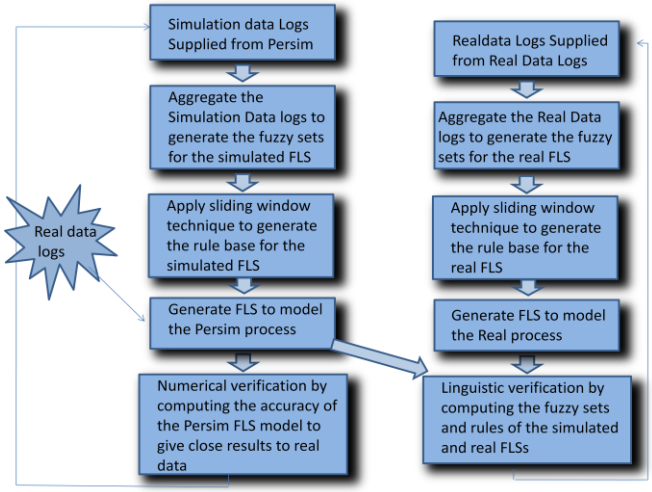


Fig. 3. An overview of the fuzzy based verification agent approach.

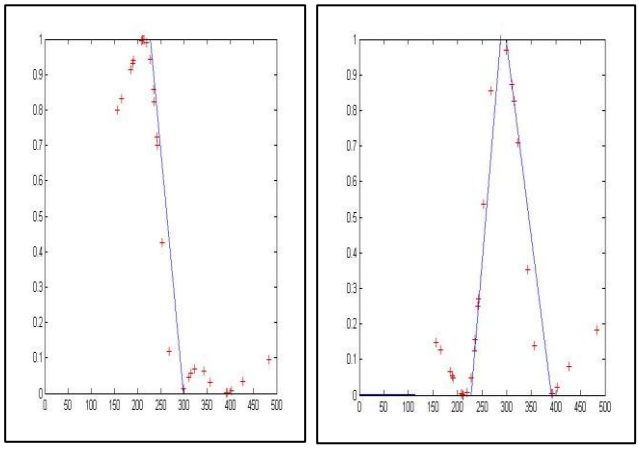


Fig. 4. The clustered points against the cluster centers and the smoothed fuzzy set.

Step 2: Step 1 is repeated for all the t data points from 1 to N to obtain N data generated rules. Due to the fact that the number of data points is quite large, many rules are generated in step 1, that all share the same IF part and are conflicting, i.e. rules with the same antecedent membership functions and different consequent activities. In this step

rules with the same *IF* part are combined into a single rule.

The *N* rules are therefore divided into groups, with rules in each group sharing the same IF part. Within each group of rules sharing the same antecedents, the rule with the highest value $w^{(t)}$ will select its consequent to be overall rule consequent. If more than one rule share the same $w^{(t)}$, then the consequent which appears with higher frequency will be selected.

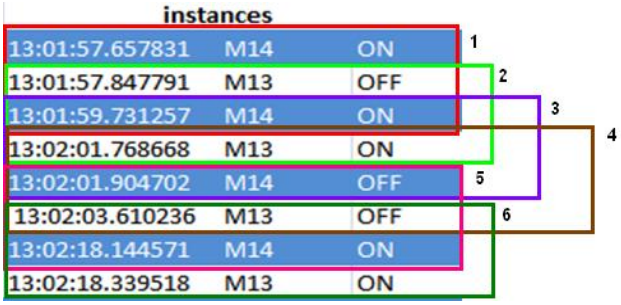


Fig. 5. The sliding window for forming data for rule extraction

V. Experiments and Results

This section reports on the verification of the Persim simulator using data which were obtained from the smart three-bedroom apartment (shown in Fig.6) on the Washington State University campus that is part of the CASAS smart home project [7] The CASAS project treats environments as intelligent agents, where the status of the residents and their physical surroundings are perceived using sensors and the environment is acted upon using controllers in a way that improves the comfort, safety, and/or productivity of the residents. The smart apartment testbed includes three bedrooms, one bathroom, a kitchen, and a living / dining room. The apartment is equipped with motion sensors distributed approximately 1 meter apart throughout the space. In addition, digital sensors provide ambient temperature readings and analog sensors provide readings for hot water, cold water, and stove burner use. VOIP captures phone usage and contact switch sensors are used to monitor door closure as well as usage of the phone book, a cooking pot, the medicine container, and key cooking ingredients in the apartment. Sensor data is captured using a customized sensor network and is stored in a SQL database.

The data were collected to mimic the Activities of Daily Living (ADLs), which are activities to be identified by assistive technologies as desired most by family caregivers of Alzheimer's disease patients. Hence, this data mapped raw sensor values to ADL activities. In this paper we focus only on three of these ADL activities, which are:

- *Making a Phone Call* (*T1*): Participants are asked to look up a specified number in a phone book, call the number, and write down the cooking directions given on the recorded message. The phone book, notepad and telephone were located on the dining room table.
- *Hand Washing (T2)*: Participants were told to wash their hands in the kitchen sink using the soap and paper towels provided.
- *Cleaning (T3)*: This activity required participants to clean the dishes and put the medicine bottle and other materials back in the cabinet.

The selected activities include both instrumental and basic ADLs. These ADLs are typically found in clinical questionnaires assessing everyday functional activities [18], [20] and deficits in these ADLs can help identify individuals who are having difficulty living independently at home [22]. In addition, poor performance for these activities has been associated with greater use of health care services and increased risk for institutionalization [6].

In order to evaluate the accuracy of the produced model, we have provided test data to measure the accuracy for the simulation FLS. The fuzzy simulation models predicted T1 with an accuracy of 99.8% while predicting T2 with an accuracy of 98.7% and predicting T3 with an accuracy of 99.8%. Hence the simulation FLS provided a very reliable FLS to validate Persim.

In order to perform the verification, the real world data were fed to the FLS-simulated model. As the FLS model provides a near-accurate approximation of the Persim generated dataset, we can predict Persim accuracy based on how accurate the FLS maps the real world sensors values to the correct activities.

From this numerical analysis, it has been seen that when the simulation FLS was fed by time stamped sensors data, the following accuracies were achieved for the associated activities: 66.7% accuracy for predicting T1 and 95.8% for predicting T2 and 29.2% for predicting T3. When dealing with time independent real sensor data the following accuracies were achieved for the associated activities: 70.83% accuracy for predicting T1, 100% accuracy for predicting T2 and 41.67% accuracy for predicting T3. Another metric we employed to judge the accuracy of the simulation was by calculating R, which is the set of sensors triggered within the window of instances in real data and calculating S, which is the set of sensors triggered within the same window size in the simulated data. We define window percentage to be $W = |\frac{|R|-|S|}{|R|}|$. W presents a measure for the similarity between the simulated and real data in terms of comparing the number of sensor triggers in each of the real and simulated windows. For T1, W= 0.87, for T2, W=0.89 and for T3 W= 0.71. This analysis again shows that more accuracy was achieved for T2 than for T1 or for T3.

The advantage of FLS is that it provides both linguistic rules and fuzzy sets which are easily readable allowing us (through analyzing the rules and fuzzy sets) to find why there are differences in the accuracy of predicting the various activities. For example, Fig. 7 shows the extracted fuzzy sets (and the associated smoothed piece-wise linear fuzzy sets) from the real data for the Time variable, which looks very similar to the extracted fuzzy sets from the simulated data in Fig. 8. For T3 where the accuracy of the simulation is not as high, it is obvious that the extracted fuzzy sets for the variables are different as shown in Fig. 9a

and Fig.9b which shows the extracted fuzzy sets for the Water_B sensor from the real and simulated data in Fig. 9a and Fig. 9b respectively. This difference can be then fed back to the simulator to help to adjust the sensor ranges in simulation to be in line with those used in the real data

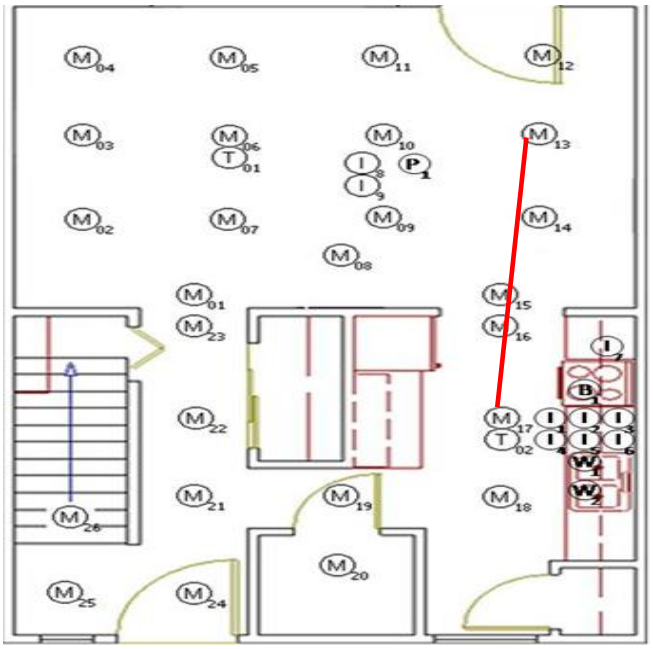


Fig. 6. The kitchen/living room area and the sensor distribution in the smart apartment in WSU. (Sensors include motion sensors (M), temperature sensors (T), water sensors (W), phone sensors (P) and item sensors (I).

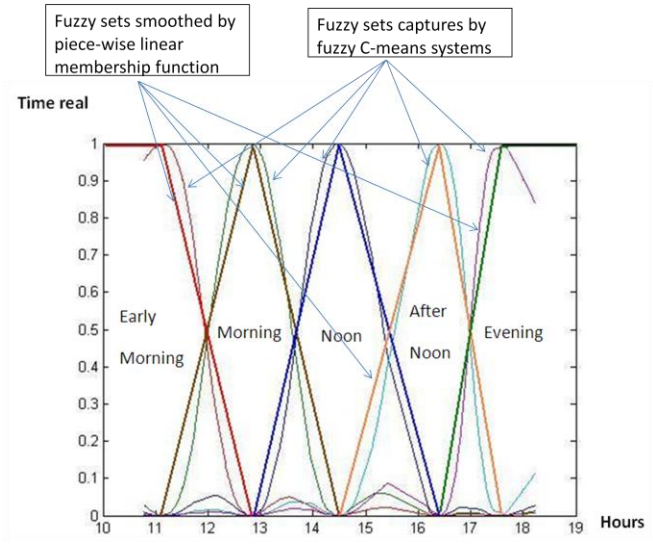


Fig. 7. The extracted fuzzy sets by the Fuzzy C-means and the associated smoothed piece-wise linear membership structure of the Fuzzy Logic System (FLS) for the time variable for the real data.

More insights about the differences between the simulation and the real data can be discovered when investigating the rule bases of the FLS generated from the real world data and the FLS generated from the simulated data. For example for T3, when investigating the rules of the simulated and real FLSs, it was discovered that as shown in Fig. 8, the user follows the red line of going through the motion sensor M13 to M14 to M15 to M16 to M17 where the user reaches the wash basin area. It was noted that the real FLS has shown this sequence. However, in simulation FLS has shown a sequence of M13 to M14 to M17, which is physically not possible without also triggering M15 and M16. Hence, this information was fed back to the simulator to improve its performance or those activities where the simulation did not perform well, as in the case of T3.

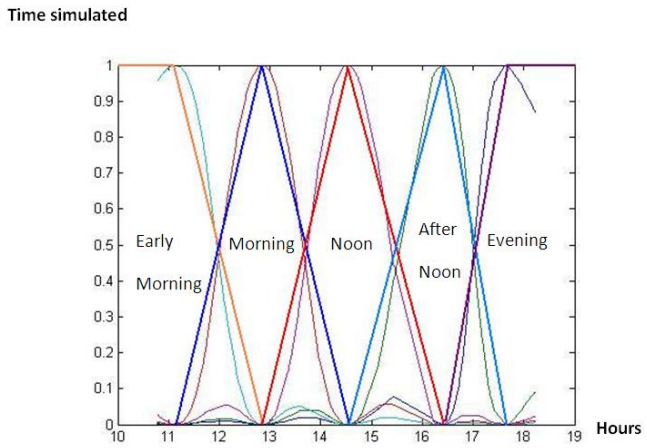
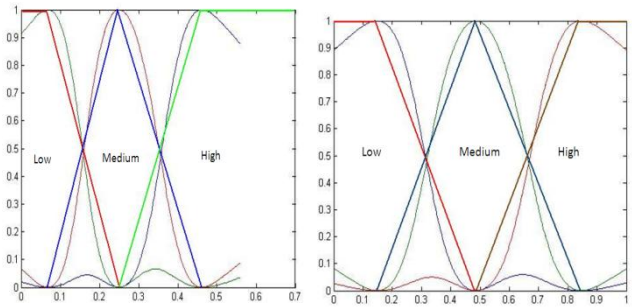


Fig. 8. The extracted fuzzy sets by the Fuzzy C-means and the associated smoothed piece-wise linear membership structure of the Fuzzy Logic System (FLS) for the time variable for the simulated data.

.



(a) (b)

Fig. 9. The extracted fuzzy sets by the Fuzzy C-means and the associated smoothed piece-wise linear membership structure of the Fuzzy Logic System (FLS) for the water sensors for (a) Real data (b) Simulated data.

VI. Conclusions and Future Work

This paper presents a fuzzy based agent approach for the verification of an AIE simulator entitled Persim. We propose a fuzzy verification agent which operates in four phases. It starts in phase 1 by aggregating the simulation and real data logs to generate the fuzzy sets for the simulated and real FLSs, respectively. In phase 2, a sliding window approach is employed to generate the rulebases of the simulated and real FLSs respectively. In phase 3, the agent perfoms a numerical verification by feeding the real data to the simulation FLS and then verifying how close are the outputs of the simulated FLS to the real data.

In phase 4, the agent perfoms a linguistic verification by comparing the fuzzy sets and rule bases of the real and simulated FLSs. The fuzzy based agent verification approach has been applied for the verification of a Persim simulation results against real data obtained from the smart apartment in WSU. It was shown that the fuzzy based verification agent can verify the Persim simulator both numerically and linguistically. Our verification approach helps simulator designers in debugging their developemnt and in assessing the adequacy of "reality capture" in their simulation loops.

Although the fuzzy based verification agent was applied to event driven simulators, it can be easily applied to any kind of AIE simulator which will be the subject of future work. Also type-2 fuzzy systems will be investigated in generating the fuzzy model due to their abilities to better handle the uncertainties than their type-1 counterparts.

REFERENCES

[1] T. Antoine-Santoni, J. Santucci, E. De Gentili and B. Costa, "Modelling & simulation oriented components of wireless sensor network using DEVS formalism," In *SpringSim '07: Proceedings of the 2007 spring simulation multiconference*, March 2007, pp. 299-306.

[2] J. Al-Karaki and G. Al-Mashaqbeh "SENSORIA: A new Simulation platform for wireless sensor networks," In *SENSORCOMM '07: Proceedings of the 2007 International Conference on Sensor Technologies and Applications*, October 2007, pp. 424-429.

[3] J. Bezdek, "Pattern Recognition with Fuzzy Objective Function Algorithms," Kluwer Academic Publishers, Norwell, USA, 1981.

[4] R. Bose, J. King, H. El-zabadani, S. Pickles, and A. Helal, "Building Plug-and-Play Smart Homes Using the Atlas Platform," *Proceedings of the 4th International Conference on Smart Homes and Health Telematic* (ICOST), Belfast, the Northern Islands, June 2006.

[5] V. Callaghan, J. Woods, S. Fitz, T. Dennis, H. Hagras, M. Colley, I. Henning, "The Essex iDorm: A Testbed for Exploring Intelligent Energy Usage Technologies in the Home", *Proceedings of the International Conference on Intelligent Green and Energy Efficient Building and Technologies*, Beijing, China, April 2008.

[6] K. Cameron, K. Hughes, and K. Doughty, "Reducing fall incidence in community elders by telecare using predictive systems," In *Proceedings of the International IEEE-EMBS Conference*, October 1997, pp. 1036-1039.

[7] D. Cook and M. Schmitter-Edgecombe, "Activity profiling using pervasive sensing in smart homes," *IEEE Transactions on Information Technology for Biomedicine*, 2008.

[8] F. Doctor, H. Hagras, and V. Callaghan, "A type-2 fuzzy embedded agent to realise ambient intelligence in ubiquitous computing environments," *Journal of Information Sciences*, vol. 171, no. 4, pp. 309-334, May 2005.

[9] K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten, and J. Burgelman, "Scenarios for Ambient Intelligence in 2010," IST Advisory Group Final Report, European Commission, February 2001.

[10] E. Kim, A. Helal and D. Cook, "Human Activity Recognition and Pattern Discovery," *IEEE Pervasive Computing Magazine*, December 2009.

[11] A. Helal, W. Mann, H. Elzabadani, J. King, Y. Kaddourah and E. Jansen, "Gator Tech Smart House: A Programmable Pervasive Space", *IEEE Computer magazine*, pp 64-74, March 2005.

[12] S. Hossain, A. Helal and H. Hagras "Persim – Pervasive System Simulation," *Proceedings of the 2010 International Conference on Pervasive Computing, (PERVASIVE 2010), Poster track*, Helsinki, Finland, May 2010.

[13] S. Hossain, A. Helal, J.W. Lee, H. Hagras, A. Elfaham, H. Gabr and D. Cook, "Persim – A Simulator for Human Activities in Pervasive Spaces," *Submitted to the 2010 International Conference on Ubiquitous Computing (Ubicomp 2010)*, September 2010, Copenhagen, Denmark.

[14] M. Huebscher and J. McCann. "Simulation model for self-adaptive applications in pervasive computing", In *DEXA '04: Proceedings of the Database and Expert Systems Applications, 15th International Workshop*, August 2004, pp. 694-698

[15] W. Jouve, J. Bruneau and C. Consel. "DiaSim: A parameterized simulator for pervasive computing applications," In *PERCOM '09: Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications*, March 2009, pp.1-3.

[16] A. Law and W. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, 2nd edition, 1997.

[17] S. Nath, P. B. Gibbons, S. Seshan and Z. Anderson, "TOSSIM: accurate and scalable simulation of entire tinyos applications," In *Transactions on Sensor Networks (TOSN), Volume 4 Issue 2*. ACM, pp. 126-137, March 2008.

[18] M. Patterson and J. Mack, "The Cleveland scale for activities of dailyl living (CSADL): Its reliability and validity," *Journal of Clinical Gerontology*," pp. 15-28, November 2001.

[19] Persim – A simulator for human-activities in Pervasive Spaces. Project web site: http://www.icta.ufl.edu/projects_persim.

[20] B. Reisberg and S. Finkel, "The Alzheimer's disease activities of daily living international scale (ADL-IS)," *International Psychogeriatrics*, vol. 13, no. 2, pp. 163-181, 2001.

[21] P. Remagnino and Gian Luca Foresti, "Ambient Intelligence: A New Multidisciplinary Paradigm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35, no. 1, pp.1-6, January 2005.

[22] M. Schmitter-Edgecombe, E. Woo, and D. Greeley, "Memory deficits, everyday functioning, and mild cognitive impairment", *Proceedings of the Annual Rehabilitation Psychology Conference*, Tucson, Arizona, 2008.

[23] M. Varshney and R. Bagrodia, " Detailed models for sensor network simulations and their impact on network performance," In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, ACM, October 2004, pp. 70-77

[24] M. Weiser, "The Computer for the 21st Century", *Scientific American*, Vol. 265, no. 3, pp. 66-75, September 1991.