A Fuzzy Based Verification Agent for PerSim - An Ambient Intelligent Environment Simulator

A. Elfaham, H. Hagras, *Senior Member, IEEE,* S. Helal, *Senior Member, IEEE*, S. Hossain,

Abstract— **The generation of useful sensory data from real-world deployments of Ambient Intelligent Environments (AIEs) is challenging because of the large 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 biggest problems for developing AIEs simulators lies in the ability to verify how close the simulated data are to the real world data. In this paper, we will report on the Persim platform which is a component based discrete-event simulator for AIEs. The paper will present a fuzzy based verification agent for the Persim platform. The employed fuzzy based verification agent will build from data a model that mimics the operation of Persim which will allow for its objective and subjective verification. We have conducted the verification on real world data captured from the smart three bedroom apartment in Washington State University. The results show the power of the fuzzy based verification agent in verifying the Persim platform as well as giving feedback to improve the simulation results.**

I. Introduction

With the ever-increasing number of miniaturised 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 yielded in 1991 when Mark Weiser introduced his vision of ubiquitous computing in his famous seminal article “The Computer for the 21st Century” [19].

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.

Ambient Intelligence (AmI) is a new paradigm that puts forward the criteria for the design smart spaces and ubiquitous computing environments [16]. 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 recognising, responding and adapting to individuals in a seamless and unobtrusive way [7]. AmI offers great opportunities for an enormous number of applications such as health care, the efficient use of energy resources, public buildings, and in leisure and entertainment. However, there are many challenges facing the creation of Ambient Intelligent Environments (AIEs) which posses AmI. Such challenges include

- **Heterogeneity:** Where AIEs are characterised by the ubiquitous presence of embedded computer artefacts which come in different sizes with different functionalities and most importantly, made by different vendors with different characteristics. Thus there is a need to research various middleware technologies which can facilitate the integration of these devices.
- **Transparency:** AIEs are characterized by seamlessly integrating the devices “invisibly” into the surrounding environment. This requires a careful design and integration of the technology from the design phase of the environment through to intelligent, intuitive, non-intrusive, and effective user-interfaces.
- **Limited Resources:** The limited hardware platforms and resources of the computational artefacts within AIEs form an additional challenge. Many of the devices and artefacts in AIEs are specific purpose devices and have commonly limited computational power, bandwidth, network connectivity, processing capability and memory storage, yet are required to operate economically and efficiently. One way around this can be achieved by employing advanced agents and intelligent techniques which could reduce network traffic, computing processing overheads as well as the storage of information in the devices.
- **Intelligence and Adaptation:** The dynamic and ad-hoc nature of AIEs means that the devices have to adapt to changing operating conditions and perform in a more efficient and effective fashion so that the changes do not cause a system failure. These and more intelligent attributes can be facilitated by embedding intelligent agents into the artefacts and devices with the capability to sense the environmental changes, reason about possibilities on how to act accordingly.

The aforementioned requirements AIEs are by no means comprehensive. Additional properties such as security, privacy, scalability and fault-tolerance should be considered

This work was supported in part by …..

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

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

at equal levels.

In order to be able to address the various challenges encountered in AIEs, there is a need to have either a physical test bed smart space or a realistic simulator for these AIEs. Having real AIEs testbed can proof to be expensive and difficult to construct for various institutions. In addition, the generation of useful sensory data from real-world deployments of AIEs is challenging because of the large cost, significant groundwork and lack of access to human subjects. Hence, there is a need to invest in the realization of simulation kits that can provide near real world testing for AIEs research.

In this paper, we will present a novel simulation platform called Persim where we will report on its initial phase to implement event driven simulator platform for AIEs. One of the main problems facing AIEs simulation platform is the ability to verify how much the simulation platform is giving close results to the real world AIEs. In this paper, we will present a fuzzy verification agent which is able to verify how well the AIE simulator is giving results similar to the real AIEs. We will present experiments which show the Persim results verified against real world data captured from the smart apartment in Washington State University (WSU). Section II will present background of the previous AIE simulation platforms. Section III will present an overview on the Persim platform. In Section IV, we will present the fuzzy based verification agent. Section V will present the experiments and results followed by conclusions and future work in Section VI.

II. Previous work in AIEs Simulation Platforms

In wireless sensor networks and pervasive computing, many simulation concepts have been researched and their tools have been built. Some good work has been done in the area of virtual sensor based simulation where [1], [2] suggested all component-based simulation approaches. SENSORIA [2] provided sensor node module, traffic generation module, energy model module and protocols modules. In [1], the Discrete-Event system Specification (DEVS) was proposed to define discrete-events rather than one big activity. However, they proposed a simulation platform for a wireless sensor network considering node mobility, power consumption, routing and data load. Eventually routing and communication are non-trivial factors for them. The proposed PerSim platform performs high-level processing on simulation and it concentrates on events of sensors for an activity. It generates a standardized output data upon activities. In implementing defined activities, the networking factors are trivial.

The DiaSim platform was presented in [10]. The difference from the previous simulators is that it has views of wireless sensor networks. For the new approach, it has unique design. It consists of DiaSpec, specification language, DiaGen, generator of simulating scenario, stimulus producers and simulation trigger. Also it divides the simulation environment into two layers: actual layer and emulation layer. The word actual layer here does not mean physical layer; it means simulated actual layer. What it does is to provide some events to scenario controller including DiaGen, as the events occur in actual environment in real environment. The support of actual layer benefits more accurate simulation. Its hierarchy services and DiaSpec are similar subactivities and simulation configuration respectively. Two types of stimulus, causal stimulus and simple stimulus are identical to our dependent sensors and independent sensors. The proposed PerSim platform works differently, however the DiaSim looks similar in two ways. First, DiaSim does not require an actual space and it can cause pure simulation. PerSim specifies an actual space and simulation occurs in the space. It can give us more realistic simulation. Second, DiaSim tests its simulation with simulated actuators in a loop inside the simulator. On the other hand, PerSim verifies itself as discussed in this paper via fuzzy based verification agents with well-defined output data which could come from previous simulation or data of other studying groups. The output data is called SDDL in terms of standardized dataset format. Through the SDDL, PerSim can verify its simulation better and cooperate with other researching groups.

On the other hand, there are some approaches [9], [13] which operate a specific system such as TinyOS that is an operating system running on sensor networks. TOSSIM [13] has been developed to make simulation more scalable and accurate and [9] suggested context-aware simulation. Because they are implemented based on TinyOS, some properties of TinyOS are inherited. For example, TinyOS's event-driven execution can map to discrete-event simulation simply in TOSSIM. However, they must be network-sensitive in simulation with TinyOS. On the other hand, PerSim is closer to service-oriented system such as Atlas [11]. For example, middleware layer in Atlas includes a layer, which defines services as PerSim defines activities. Therefore PerSim is referring to the architecture of Atlas.

In [18~~Varshney 20~~], a simulation platform was presented which asserts that its well-defined and faithful models can work well on simulation. However, too minute models could weaken flexibility in simulation. On the other hand, PerSim serves more widened and adaptable configuration and performs high-level simulation.

The simulators and their approaches are various. The simulation concepts depend on what the developers and researchers want to do. As will be presented below, PerSim is component-based and configurable for simulation, as so are several simulators. However, PerSim suggests some properties that donot exist in other simulators where the PerSim has well-defined and standardized output defined in the SDDL format. Because of the well-defined output, PerSim can verify itself easily, and because of the standardized format it makes itself sharable and collaborative. Second, PerSim is a web-based program. It is open to someone who wants to use and to share data. Also it is platform-independent and is easily operated in other machines. Thirdly, PerSim provides good flexibility through project-options. PerSim does not only generate output dataset, but also reuses configuration such as the number of spaces, location of sensors and activities.

III. Overview of PerSim Simulation Platform

PerSim is an event-driven in-home human activity simulator capable of capturing the physical elements of AIEs including its sensors and the behavior of the space users (activities). It allows to simulate AIEs by first defining the space similar to WSU Smart Apartment Testbed [5] or Gator Tech Smart House [20][21]. After defining the space, the researcher can add different type of sensors into the space. Then activities that can happen inside the space can be added. When the design is complete, researcher has the flexibility to define the causal relationship between activity and sensors. In the last step, one has the choice to simulate the entire state space similar to the dataset of iDorm of University of Essex [22] or zoom into the space and simulate only the activities that are being performed similar to the dataset of WSU Smart Apartment [5]. In PerSim, researcher has the power to design space, modify the design and simulate the space until it satisfies the requirement of the individual experiment. The researcher can also reuse generated dataset by modifying the design and fine tune the experiments to achieve advanced research-goal.

A. *Organization of 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) events 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 will again trigger a set of dependent sensors based on *Activity-Sensor Mapper*. On the other hand, in *State-space* mode, simulator can generate time-driven events which are independent of any activity.

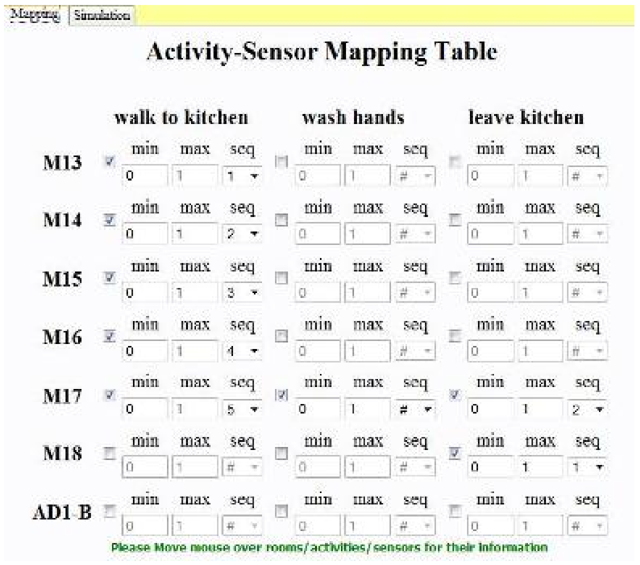
We have adopted discrete-event simulation model [12] using next-event time-advance approach to capture the dynamic nature of the AIEs which evolve over time. According this classical simulation technique, our target space changes whenever any event (either activity-driven or time-driven) occurs and the system variables of the space are updated based on the simulation logic. The simulation model consists of following major variables:

1) *Simulation Clock (simClk).* A variable that keeps track of the current value of the simulated time.

2) *List of Events (eventList).* A sequence of tuples of the form (e, t) where e denotes the event and t denotes the time of the occurrence of that event.

3) *Initialization Routine:* A sub-routine to initialize simClk and eventList before the simulation loop begins.

4) *Timing Rouitne:* A sub-routine that determines the next event from the *eventList* and advances the *simClk* to the time of occurance of the selected event.


Mapping | Simulation

Activity-Sensor Mapping Table

		walk to kitchen				wash hands				leave kitchen		
		min	max	seq		min	max	seq		min	max	seq
M13	✓	0	1	1	☐	0	1	#	☐	0	1	#
M14	✓	0	1	2	☐	0	1	#	☐	0	1	#
M15	✓	0	1	3	☐	0	1	#	☐	0	1	#
M16	✓	0	1	4	☐	0	1	#	☐	0	1	#
M17	✓	0	1	5	✓	0	1	#	✓	0	1	2
M18	☐	0	1	#	☐	0	1	#	✓	0	1	1
AD1-B	☐	0	1	#	☐	0	1	#	☐	0	1	#

Please Move mouse over rooms/activities/sensors for their information

Fig. . PerSim activity-sensor mapping table.

5) *Event Routine:* A sub-routine to schedule next event to occur. It also determines the time of occurrence of the event based on the library routine.

6) *Library Routine:* A sub-routine to generate random variates from probability distribution of inter-arrival time of an event and generation function of sensors.

7) *Simulation Loop:* Main program that invokes initialization routine, timing routine and event routine and propels the target space with respect to events.

B. *Simulation Algorithm*

The flow chart of the simulation algorithm is shown in Fig. . The simulation starts from the empty and idle state. Then it invokes the timing routine to get the most imminent event from the *eventList*. 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 *simClk* to the time of occurrence of the current event. The simulation continues until the *simClk* doesn't exceed the end time of the simulation loop. Finally, it generates simulated data in SDDL form which is a standardized format of representing sensory dataset [23][24].

C. *Simulation Steps*

Fig. shows a screen shot of configuring the simulation parameters in Persim. User first defines the space to be simulated e.g. kitchen or living room. Then he/she can put desired sensors e.g motion sensor, light sensor in each room and configure the sensors with information such as sensor name, id, type, value generation function, min value, max value etc. Then a user can add activities in the AIEs e.g move to kitchen, clean dishes etc.

Next, user needs to map activity to a set of dependent sensors using *Activity-Sensor Mapper*, shown in Fig. . While mapping, a user can specify the sequence of sensor-trigger for each activity. For an example, if motion sensor M1 has sequence number 1 and motion sensor M2 has sequence number 2 for a specific activity, then M1 is will trigger before M2 in the simulation of that activity. If sequence number is set to ‘#’, it denotes that the sensor can trigger anytime during the simulation.

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

IV. Overview of the Fuzzy Verification Agent

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

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 the noise and uncertainties. Furthermore, FLSs employ linguistic IF-THEN rules which enable to represent 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 (wither 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.5 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 will presnet phases 1 and 2 where 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 the 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:

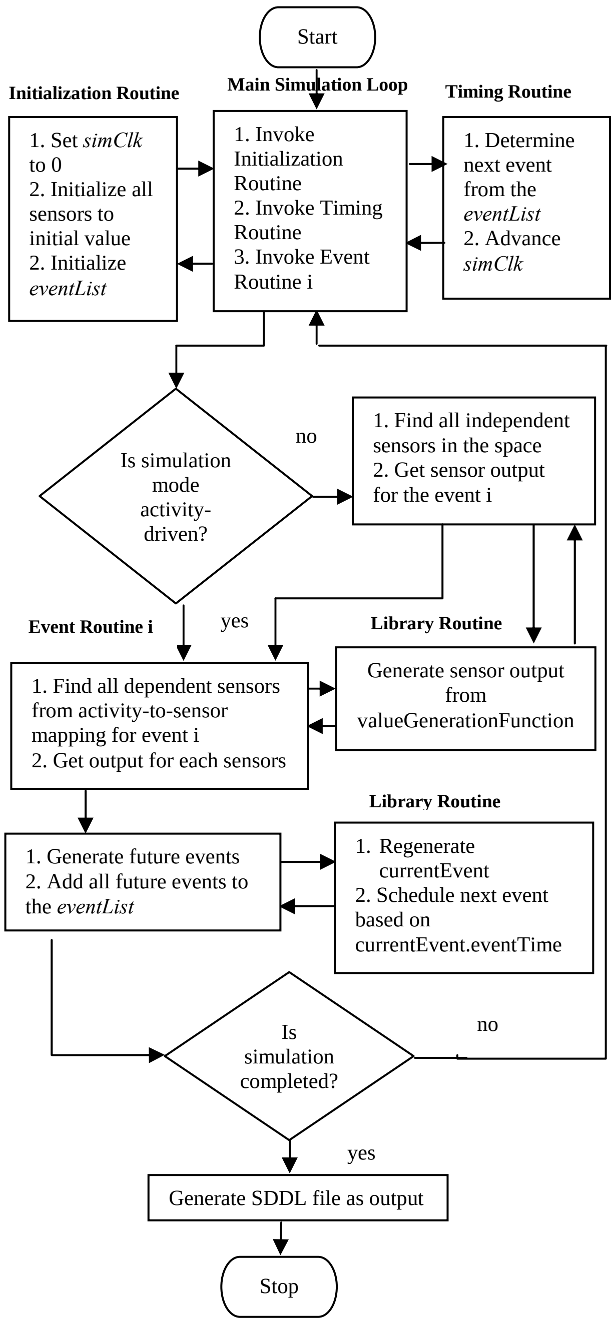
Consider a family of fuzzy sets $A_i, i = 1, 2, \dots, c$, as fuzzy c-partitions on the universe X. Fuzzy sets allow a degree of membership, hence, we can assign memberships to the various data in each fuzzy cluster and a single data point can have a membership to more than one cluster [3]. The membership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

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

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

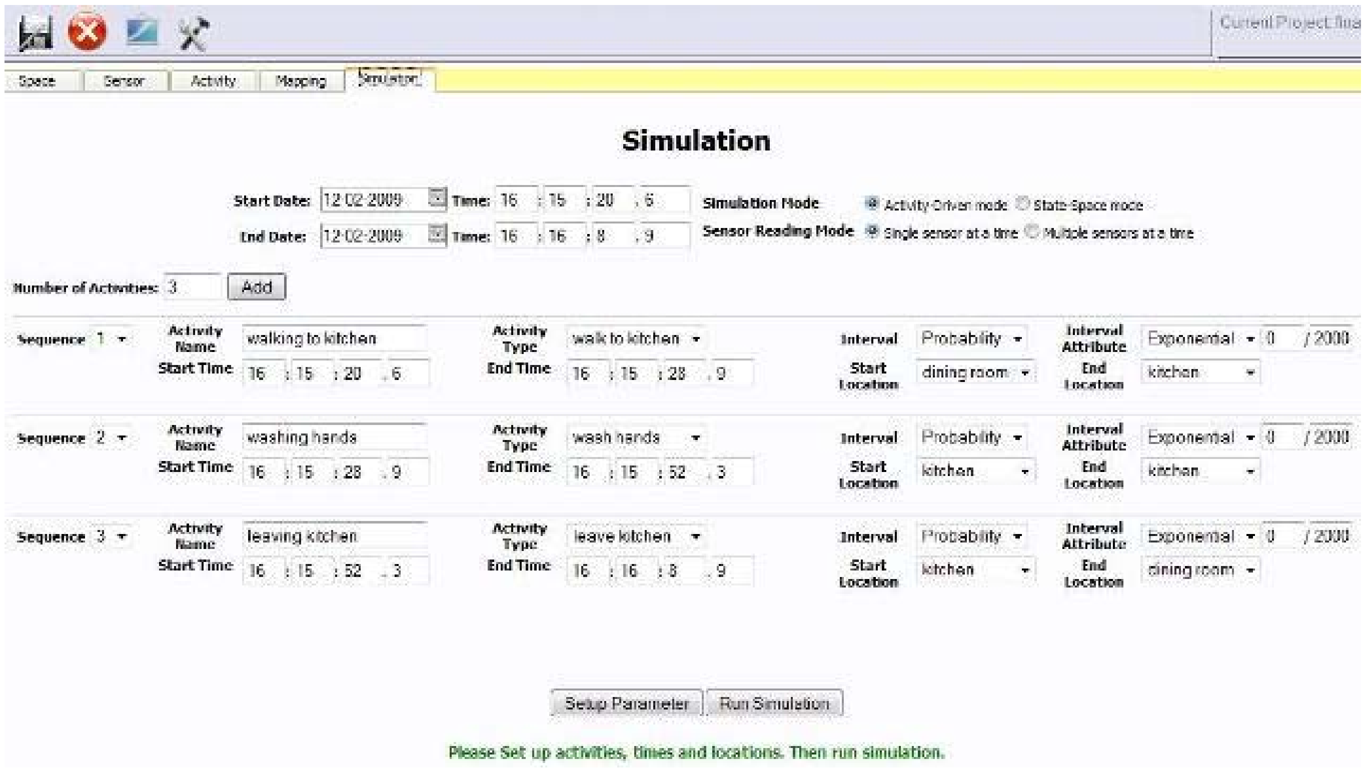
$$J_m(U, v) = \sum_{k=1}^{n}\sum_{i=1}^{c}(\mu_{ik})^{m'}(d_{ik})^2 \qquad (2)$$

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.



Start
Initialization Routine
Main Simulation Loop
Timing Routine
1. Set *simClk* to 0
2. Initialize all sensors to initial value
2. Initialize *eventList*
1. Invoke Initialization Routine
2. Invoke Timing Routine
3. Invoke Event Routine i
1. Determine next event from the *eventList*
2. Advance *simClk*
Is simulation mode activity-driven?
no
1. Find all independent sensors in the space
2. Get sensor output for the event i
yes
Event Routine i
Library Routine
1. Find all dependent sensors from activity-to-sensor mapping for event i
2. Get output for each sensors
Generate sensor output from valueGenerationFunction
Library Routine
1. Generate future events
2. Add all future events to the *eventList*
1. Regenerate currentEvent
2. Schedule next event based on currentEvent.eventTime
Is simulation completed?
no
yes
Generate SDDL file as output
Stop

Fig. . Flow chart of PerSim event simulation.



Simulation

Start Date: 12-02-2009 Time: 16 : 15 : 20 . 6 Simulation Mode [✓ Activity-Driven mode] State-Space mode
End Date: 12-02-2009 Time: 16 : 16 : 8 . 9 Sensor Reading Mode [✓ Single sensor at a time] Multiple sensors at a time

Number of Activities: 3 Add

Sequence 1 Activity Name: walking to kitchen Activity Type: walk to kitchen Interval: Probability Interval Attribute: Exponential 0 / 2000
Start Time: 16 : 15 : 20 . 6 End Time: 16 : 15 : 28 . 9 Start Location: dining room End Location: kitchen

Sequence 2 Activity Name: washing hands Activity Type: wash hands Interval: Probability Interval Attribute: Exponential 0 / 2000
Start Time: 16 : 15 : 28 . 9 End Time: 16 : 15 : 52 . 3 Start Location: kitchen End Location: kitchen

Sequence 3 Activity Name: leaving kitchen Activity Type: leave kitchen Interval: Probability Interval Attribute: Exponential 0 / 2000
Start Time: 16 : 15 : 52 . 3 End Time: 16 : 16 : 8 . 9 Start Location: kitchen End Location: dining room

Setup Parameter Run Simulation

Please Set up activities, times and locations. Then run simulation.

Fig. . PerSim simulation configuration.

Fig.5. An overview of the fuzzy based verification agent approach.

Where

–

The membership is the membership of the th data point in the th cluster. The goal of the objective function is to get the best clustering. The parameter is the weighting parameter which has a range . This parameter controls the amount of fuzziness in the classification process and usually takes the values between . In our experiments was set to 2. The vector for a cluster center is calculated as follows:

Where is the variable on the feature space, i.e. . The Fuzzy C-means clustering operate as follows:

1. Fix and select a value for parameter . Initialize the partition matrix ,
2. Calculate the center vectors for each step.
3. Update , the partition matrix in the th step; calculate the updated membership function matrix as follows:

4. If (tolerance level) STOP, otherwise set 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.6. The shapes of the fuzzy sets are then smoothed by piecewise linear membership function as shown in Fig.6.

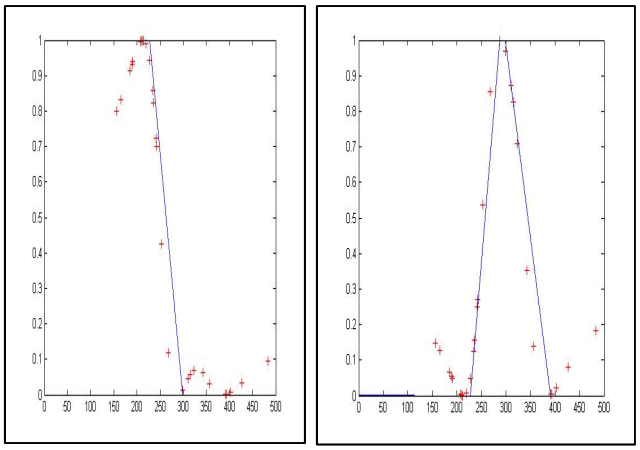


Fig.6. The clustered points plotted against the cluster centres and the smoothed fuzzy set.

B. *Phase 2: Learning the Fuzzy Rules Bases of the of the Simulation and Real FLSs*

We employ a sliding window approach as shown in Fig.7 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 if the relevant sensor has been triggered or no 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 [6]. In the following steps we will summarise the different steps involved in rule extraction:

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

$$\mu_{A_s^{q^*}}(x_s^{(t)}) \geq \mu_{A_s^q}(x_s^{(t)}) \qquad (7)$$

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

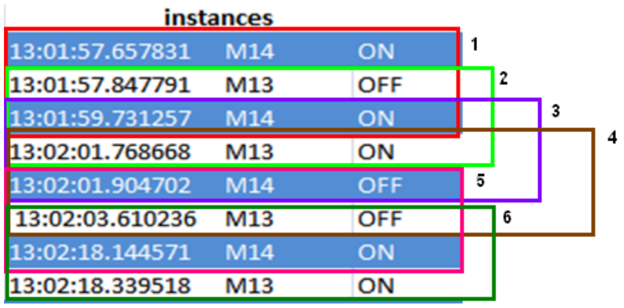
IF x_1^t is $A_1^{q^*}$ *and ... and* x_n^t is $A_n^{q^*}$, *THEN* y *is activity* $y^{(t)}$ (8)

For each input variable x_s there are V fuzzy sets $A_s^q, q = 1,..., V$, to characterise 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:

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

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 have 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.



instances		
13:01:57.657831	M14	ON
13:01:57.847791	M13	OFF
13:01:59.731257	M14	ON
13:02:01.768668	M13	ON
13:02:01.904702	M14	OFF
13:02:03.610236	M13	OFF
13:02:18.144571	M14	ON
13:02:18.339518	M13	ON

Fig.7. The sliding window for forming data for rule extraction.

V. Experiments and Results

This section reports on the verification of the Persim platform using data which were obtained from the smart three-bedroom apartment (whose layout is shown in Fig. 8) in Washington State University (WSU) campus that is part of the ongoing CASAS smart home project [5]. 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, there are digital sensors to provide ambient temperature readings, and analog sensors to provide readings for hot water, cold water, and stove burner use. VOIP captures phone usage and contact switch sensors are used to monitor 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 have focused only on three of these ADL activities which are

- *Making a Phone Call* (*T1*): where the 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)*: For this activity, participants were told to wash their hands in the kitchen sink using the soap and paper towels provided.
- *Cleaning (T5)*: The cleaning 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 [14], [15] and deficits in these ADLs can help identify individuals who are having difficulty living independently at home [17]. In addition, poor performance for these activities has been associated with greater use of health care services and increased risk for institutionalization [4].

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 T5 with an accuracy of 99.8 %. Hence the simulation FLS provided a very reliable FLS to validate the Persim platform.

In order to verify the Persim platform, the real world data were fed to FLS simulated model. As the FLS model provides a near accurate approximation of the Persim platform then we can predict the accuracy of the Persim platform 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 T5. 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 T5. 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 simulated data. We define window percentage to be W= abs(|R| -|S|) / |R|. W presents a measure for the similarity between the simulated and real data in terms of comparing the number of sensors triggers in each of the real and simulated windows.

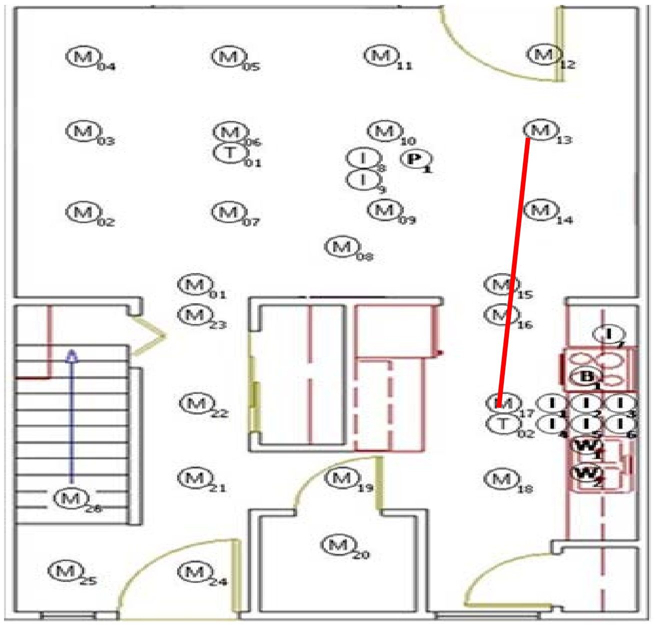


Fig. 8. The Kitchen/living room area and the sensor distribution in the smart apartment in WSU. (Sensors include motion sensors (M), temperature sensors (T), water sensors (W), burner sensors (B), phone sensors (P), and item sensors (I))

For T1, W= 0.87, for T2, W=0.89 and for T5 W= 0.71. This analysis again shows that more accuracy was achieved for T2 more than T1 more than T5.

The advantage of the FLS that it provides linguistic rules and fuzzy sets which are easily readable which enables 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. 9 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. 10. For T5 where the accuracy of the simulation is not quite high, it is obvious that the extracted fuzzy sets for the variables are different as shown in Fig. 11a and Fig. 11b which shows the extracted fuzzy sets for the Water_B sensor from the real and simulated data in Fig. 11a and Fig. 11b respectively. This difference can be then fed back to the simulator to help to adjust the sensor ranges in simulation to be inline with those used in the real data.

More insight about the difference 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 T5, 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 like T5.

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 Time variable for the real data.

VI. Conclusions and Future Work

This paper presented a fuzzy based agent approach for the verification of a simulation platform for AIEs entitled Persim. The Persim is an event driven simulation platform. The proposed fuzzy verification agent operate through four phases where 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.

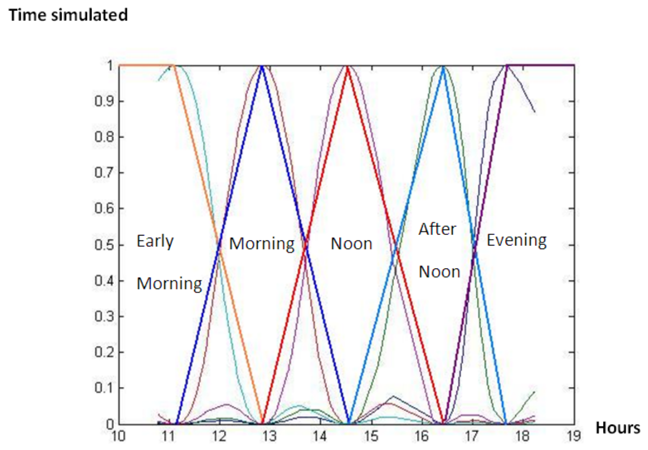
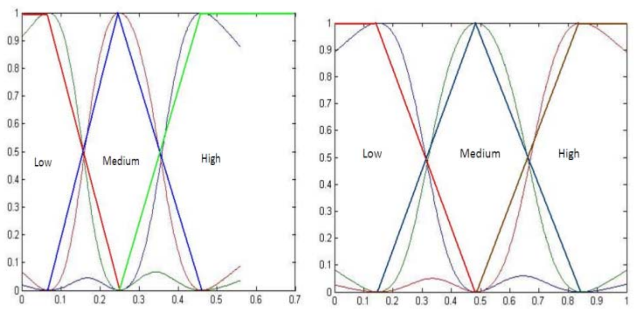


Fig. 10. The extracted fuzzy sets by the Fuzzy C-means and the associated smoothed piece-wise linear membership structure of the Fuzzy Logic System (FLS) for the Time variable for the simulated data.



(a) (b)

Fig. 11. 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_B sensor for (a) Real data. (b)Simulated 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 the Persim simulation results against real data obtained from the smart apartment in WSU. It was shown that the the fuzzy based verification agent can verify the Persim simulator both numerically and linguistically. This helps the simulators designers to gain more understanding of the diffrenece between the simulation platform and the real envirnments which will help to improve the performace of the simulation.

Although the fuzzy based verification agent was applied to event driven simulators, it can be eaily applied to any kind of AIE simulators which will be the subject of future work. Also type-2 fuzzy systems will be investogated in generatung the fuzzy model due to their abilities to better handle the unceratainties 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] 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.

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

[6] 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.

[7] 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.

[8] A. Helal, A. Mendez-Vazquez and S. Hossain, “Specification and synthesis of sensory datasets in pervasive spaces,” *IEEE Symposium on Computers and Comm.*, Sousse, Tunisia, July 2009, pp. 920-925.

[9] 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*, IEEE Computer Society, August 2004, pp. 694-698

[10] 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.

[11] J. King, R. Bose, H. Yang, S. Pickles and A. Helal. Atlas, “A Service-Oriented Sensor Platform,” In *Proceedings of the first IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*. Tampa, Florida, November 2006, pp. 2627-2648.

[12] A.. Law and W. Kelton, “Simulation Modeling and Analysis”, 2nd edition, 1997, pp 1-103.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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

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

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

[21] 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.

[22] 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.

[23] A. Helal, A. Mendez-Vazquez, S. Hossain, “Specification and Synthesis of Sensory Datasets in Pervasive Spaces,” *IEEE Symposium on Computers and Communications*, Sousse, Tunisia, July 5-8, 2009.

[24] “Sensory Dataset Description Language (SDDL) Specification”, Technical Report, Mobile and Pervasive Computing Laboratory, University of Florida, Florida.