Fuzzy-as-a-Service for Real-Time Human Activity Recognition Using IEEE 1855-2016 Standard

Bhavesh Pandya, Amir Pourabdollah and Ahmad Lotfi
School of Science and Technology
Nottingham Trent University
Nottingham, United Kingdom
Email: {bhavesh.pandya, amir.pourabdollah, ahmad.lotfi} @ntu.ac.uk

***Abstract*—Fuzzy Logic Systems (FLSs) have shown their potentials in Ambient Intelligence (AmI) applications. However, the implementation of FLSs is typically linked to dedicated and non-scalable hardware/software systems. As a result, some specific AmI requirements such as web communications and Service-Oriented Architecture (SOA), which can be found in many modern systems, are rarely adapted for FLSs. Sharing FLSs accessibility as web services (called fuzzy-as-a-service), in which the service is developed independently from a specific FLS, allows for system autonomy, openness, load balancing, efficient resource allocation and eventually cost-efficiency, particularly for computationally intense FLSs. In a wider context, such features can open new dimensions for FLSs' applicability in Cloud Computing and Internet of Things devices. Recent advances in standardising Fuzzy Mark-up Language (IEEE 1855-2016) and its associated software libraries (such as JFML) has made this even more achievable. This paper proposed fuzzy-as-a-service architecture based on IEEE 1855-2016, JFML and SOA. Through a simulated experiment, this paper concerned the collection, processing and monitoring the distributed data over the web, thus a real-time human activity recognition simulated scenario using a rule based FLS is demonstrated.**

Keywords—fuzzy-as-a-service, ambient intelligence, human activity recognision, service-oriented architecture, internet of things, IoT, fuzzy markup language, FML

I. INTRODUCTION

Improvements in Ambient Intelligence (AmI) and the Internet of Things (IoT) have opened new research horizons for day-to-day applications such as environmental intelligence. The new emerging environments can ascertain from both environmental deviationsI and occupants' behavioural patterns. Collected data, including various attributes, such as the environmental changes and occupants' relations with the environment, are processed in order to formulate decisions or to predict the activities/benefits of occupants at atypical areas. However, most of the applied techniques are based on rigid observations of the environment and/or activities of the individuals, underestimating the value of the naturally embedded uncertainty elements in decision making processes in human activity related data. The methods of processing uncertain data in huge/complex settings would need high processing powers, something which is not achievable in sensors or stand-alone system.

Contemporary progress in cloud computing, IoT, and service-oriented architectures (SOAs) has caused a transition from traditional local or client–server paradigms to a service-oriented model for AmI [1]. The renowned fact is that FLSs are usually implemented on dedicated local hardware and software systems, and most of these contemporary systems do not comprise distributed, pervasive and scalable architectures.

By changing the classical approaches to FLSs software/hardware architecture, a fuzzy logic system can be perceived *as-a-service* that if maintained on the cloud, permits computing to be divested and concealed from devices in an intelligent environment, as well as being scalable. The initial idea, called fuzzy-as-a-service proposed in [1], is followed in this study. This paper proposed a development and standardization for the components of the underlying architecture and introduced a fully-functional service-oriented solution to make it practical and cost-effective to design and implement complex FLSs particularly in AmI environments [1].

The development of the distributed architectures for FLSs, as will be seen, is a relatively new field with very little progress. This study's innovation is to shift to an open service model. This openness can be accomplished by developing the system to be fully web-based and device self-sufficient, particularly by utilising standard formats for data exchange that are both consistent and readable as realistically as possible. For example, an individual client computer may outline its necessary FLS, an input data can be provided for the specified system by the same or other client computer(s), and lastly the same or other client(s) can repossess the calculated output [1]. The outline of a cloud-based model for a distributed FLS is shown in Fig. 1.

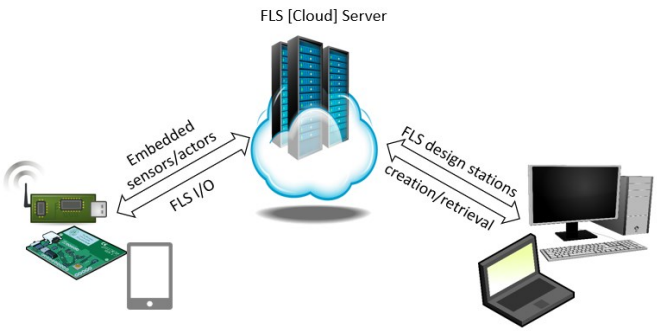


Fig. 1: The outline of a cloud -based model for a distributed FLS [1].

A web-based data language for FLS characterizations is the key criterion for implementing such an architecture. The current standard for this purpose is the IEEE-1855 (2016), also known as FML [3], an XML-based mark-up language allowing the human readable and hardware-independent definition of an FLS. FML and FML-compatible pieces of software such as JFML [4] are used as the basic design standard in this study, and the extensibility of this standard is a solution to architecture growth.

The remaining sections of the paper are as follows: Section II discusses the related work, section III reviews the proposed fuzzy-as-a-service architecture design including the design attributes and the new software components. Section IV explains an experiment of using the fuzzy as-a-service architecture for AmI applications, finally section V concludes the paper with an overview on the future work.

II. Related Work

AmI accumulates best outcomes from three key advances, i.e., omnipresent computing, omnipresent communication, and smart convenient interfaces. Useful and spatial distribution of activities is a characteristic advocated to utilize multispecialty worldview to structure and objectify AmI systems. A great deal of research has been performed for uncertain decision making in AmI environments, some of which focused on FLSs utilisation, and a few of which implemented standardized networked solutions.

Acampora et al. [2, 3] proposed using IEEE-1855 for developing fuzzy logic systems for controlling AmI environments. They focused on issues such as complexity and confusion of human activities with variability of hardware devices in Ambient Assisted Living (AAL) environments. Their proposed solution was implementing a revolutionary self-organizing feature map method to automatically identify a moving object's position in an enclosed environment and a stratagem to generate context aware FML services to optimize user comfort and hardware interoperability.

Soto-hidalgo et al. [4] proposed an open source Java library, i.e., JFML, which presented a complete implementation of the IEEE standard having a capacity to impart fuzzy systems as per different norms and programming. Herein, the authors offered three contextual examinations that delineated the capability of JFML in AmI, and the benefits of trading FML-aware FLSs along with accessible software.

Following the JFML introduction, some of its applications and follow-up implementations have recently emerged. Arcos et al. [5] developed an interoperability unit to design and run FLS for embedded systems in JFML, particularly for Arduino boards. Moreover, they defined a transmission protocol between JFML and Arduino boards, which removed regulated computing capability proffered by embedded systems. To delineate the competence of the new interoperability module, they exhibited contextual analysis by dividing fuzzy controller stamped by IEEE-1855 to handle mobile robots in different situations. The new module could effortlessly map a controller's input/output variables by means of distance sensors and motors, connect to an Arduino board's microcontroller, and manage the mobile robotic by conducting fuzzy inference from an outward PC that maintains communication via USB.

Alcala-Fdez et al. [6] expounded a Python wrapper for JFML (Py4JFML) that permits to use all JFML functionalities via Python programming language. By such a Java-Python bridging, the likelihood of using IEEE standard for denoting fuzzy systems was engorged to both developers and engineers with infinitesimal redundancy of code. The authors carried out some experiments that revealed complete interoperability between Python programs and JFML devoid of any concrete overhead.

Besides the above FML-aware solutions, there are related works on healthcare monitoring systems with rapidly growing applications in AAL environments [9, 10]. Simultaneously, the demand for wearable devices is also increasing keeping in mind cost effectiveness as well as user comfort [11, 12,]. Thato et al. [13] used latest virtual technologies to build a low-cost and customizable AAL system suited to the climate of South Africa. The authors conducted a survey of different AAL technologies and characteristics of the system, which were considered to be useful in defining the architecture.

Davide et al. [14] provided a comprehensive overview of AAL field, which comprised more than 10-year systematic study of pertinent information based on investor requirements, linking the gap in established assessments concentrated on technologies. The results of the review distinctly revealed that the view of entire AAL ecosystem has been ignored by the AAL community. Conversely, the proposed solutions tend to be more specific to existing technologies rather than meeting interests of numerous stakeholders. Another foremost shortcoming highlighted by the review was lack of adequate evaluation of different solutions.

Ashish and Jigarkumar [15] provided a review of diverse activity approaches and application of behavioural examination to recognize seventeen key problems associated with AALs. Their primary goal was to provide a standard to choose the paramount approach to regulate smart environment activity and human behaviour. Similarly, Hong et al. [16] examined the current status of AAL science, addressed AAL's promises and potential benefits, and specified obstacles that were faced to build real-world and competent AAL programs for the elderly.

Author [26], recommended the latest JFML the technique for both Arduino and Raspberry Pi that enables the automated development of runnable files for non-expert users without specific knowledge on embedded systems or sufficient programming capabilities. The author also explained how JFML maps FLC variables with sensors and actuators independently. This communication protocol and embedded systems are integrated with various communication mechanism such as (Wi-Fi, Bluetooth, and USB).

In Summary, the reviewed literature shows the adaptation of smart solutions, particularly using FLSs for AmI applications. The research gap seems to be the rarity of the standard FLS adaptations for addressing the growing requirements in AmI. Based on this, in the next section the proposed solution is explained.

III. Architecture Design

The inspiration following a service-oriented approach for FLSs is centred on the possibility of using FLSs in a decentralized AmI context. A number of general attributes that are considered in the architectural design are discussed in the first section, then a brief description of the utilized specific software components is presented as well as the implemented functionalities in order to address the described attributes.

A. Design Attributes

1) Distributed architecture: The elements of the FLS are to be tangibly distributed within the ambient intelligence model. Sensors collecting input data for the FLS, processors and running the FLS, and delivering the results to some output devices (such as actuators or monitoring stations) are distributed in such an environment. From the standpoint of efficiency, flexibility, and redundancy, it is advantageous not only to distribute input/output devices but also to distribute the necessary processing power to numerous servers. This model may require smart load-balancing that is not feasible with static hardware/software designs. This paper does not focus on an optimized load balancing method, however the possibility of achieving this is demonstrated through the experiment.

2) High-power computation requirements: For FLS implementations, computational power is an identified bottleneck [21]. When the number of inputs, outputs and rules in an FLS are augmented, the system complexity may increase dramatically. Moreover, owing to their extraordinary computational complexity, the usage of more advanced FLSs, such as non-singleton or type-2 systems, becomes limited. A cloud-based approach is favoured to fixed hardware solutions that can animatedly assign memory and processors among the available resources.

3) Reuse of computation results: A server can record the inputs originating from and the output sent to different input/output devices for one single FLS in order to avoid repeated computations. For example, if a system inputs have been processed in the (recent) past, it is possible to reuse the saved outputs as a ready answer for future requests by the same or any other user having the same input parameters. In addition, in some other contexts, the concept of a single FLS for a specific application can be apportioned and reclaimed by numerous other applications. This can only be done if an FLS server can be probed for the definitions of FLSs and their history of input/output.

4) Openness and accessibility: These are conventional incentives for any form of FLS use, not limited to AmI applications. In fact, the currently available FLS computing tools are mostly suited for single-station applications, such as the special tools and libraries for fuzzy logic computing built in MATLAB and R software [2, 3]. A cloud-based solution where the FLS is accessible as a Web Service will eliminate the need of users/clients to have or learn any specific software, library or programming/scripting language. This will also facilitate users by system autonomy - specially pertinent for research or educational purposes where it is necessary to access and share software tools and data. This can also be extended to shared *FLS repositories* developed by the users' communities for different application domains.

5) Incorporating Cloud Computing and IoT: Cloud computing can be viewed as a valuable extension of the permanent hardware client–server architecture to make system resource allocation more versatile based on the computational power available for a given FLS. This will make it possible to access the FLS computation as a service from any computer. With this accessibility scope, any IoT system placed in any topographical location can use an FLS repositories in the sense of IoT.

B. Software Components of the Developed System

1) IEEE 1855-2016 (also known as Fuzzy Markup Language - FML), presented in [23] and standardized in [24-25], is an XML-based language that enables the modelling of an FLS in a human understandable, platform-independent and Web-compatible format. Before that, a portion of IEC61131 [7] devoted to fuzzy controllers could be used to describe a specific range of FLSs (known as FCL). IEEE-1855 has the benefit of being straightforwardly applicable to programming logic as much as server-side programming is concerned.

An FLS defined in IEEE-1855 can be immediately translated into numerous programming language codes (e.g., Java) [8] using an extensible stylesheet language translator (XSLT), so minimal server-side effort is needed to encode an FLS description in local program logic. In addition, IEEE-1855 permits different agents to monitor the same FLS, communicating with the environment from different locations, as shown in [7]. The details of this standard are out of the scope of this paper and can be found in [5]. Fig. 2 shows a fuzzy controller tree structure modelled in FML. This paper focused on the known capabilities of FML in describing an FLS by all the necessary parameters of its input fuzzy sets, rule-base, inference method, output fuzzy sets and defuzzification methods.

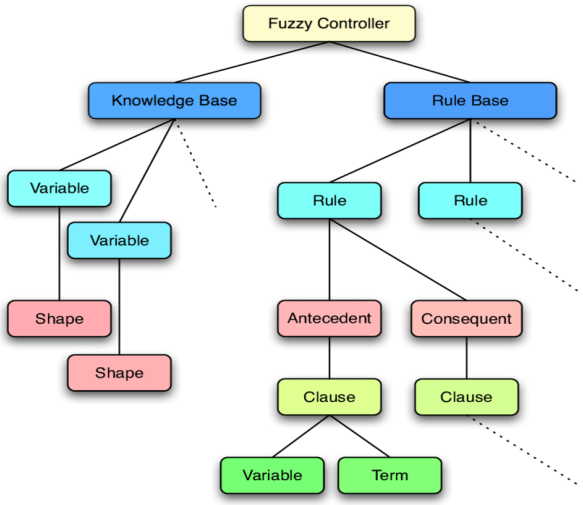


Fig. 2: The object model tree structure of FML in defining a Fuzzy controller [17].

2) SOA and distributed architectures: SOAs are characterized as a set of self-contained, networked, loosely coupled, and reusable software components generally used by customers with no or minimal reliance on their hardware or software platforms [10, 11]. The key resources are distributed in SOA for FLSs across one or more servers that several clients touch. The embedded autonomy of the SOA here means that (a) the Web service is developed independently of a specific FLS, if the FLS can be described by the standard format, and (b) that it allows for serving multiple FLSs simultaneously. The suggested architectures are moreover not based on a completely service-oriented model in the works described or are not based on a structured data exchange format.

3) Web Services: Web services is a standardized way or medium to propagate communication between clients and server applications on the World Wide Web. Some known implementation of web services include SOAP, WSDL, and REST. RESTful web services are designed to function best on the Cloud. REST determines constraints, such as the uniform interface, that when applied to a web service induces desirable properties, such as consistency and scalability, that allow services to work best on the Web. Data and functionality are called tools in the REST architectural style and are accessed via URIs, usually web links. Using a set of simple, well-defined operations, the resources are used. A web service invocation is configured for each function through an API that comprises a HTTP request and response. The developed API functionalities are listed later in this paper.

4) JFML: It is a new open-source library for fuzzy logic computations based on FML (IEEE 1588-2016) data format [4]. Being developed in Java, JFML can act as a cross-platform back-end application server for the developed fuzzy-as-a-service. Moreover, JFML follows a strict object-oriented approach and a modular architecture based on the same tree structure that FML uses to describe FLSs, permitting developers to extend JFML devoid of modifying the grammar of the language. JFML permits the use of all standard fuzzy inference systems included in XSD, which also includes all membership functions, fuzzy operators, etc. (readers are referred to [4] for more information). Researchers may need to use other components, however, that are excluded in XSD's current definition. Thus, JFML offers custom methods for all the elements indicated in the XSD, providing a way to extend the library in compliance with this standard without having to change the grammar of the language.

C. API Functionalities

Several tasks are considered to be the key functional necessities for the solution creation. A web service invocation is configured for each function through an API that comprises a HTTP request and response. Fig. 3 shows a list of these invocations as a request/response series between the client and server sides. Since the parameters, inputs, and outputs of the FLS model are being reused in the process, they need to be stored on a database on the server side. It is also evident that the approach is presently restricted to singleton type-1 FLSs. As part of future research, the other forms of FLS will be considered.

IEEE standard 1855-2016 and its extension are used as the core schema in the API data exchange. The extension is added to the schema in order to support exchanging input/output values between the clients and the servers. Briefly, the "type" attribute determines if the HTTP packet is a request or a response, whereas the name of the requested functionality is encoded in "service" attribute of both request and response packets as shown in Fig. 3.

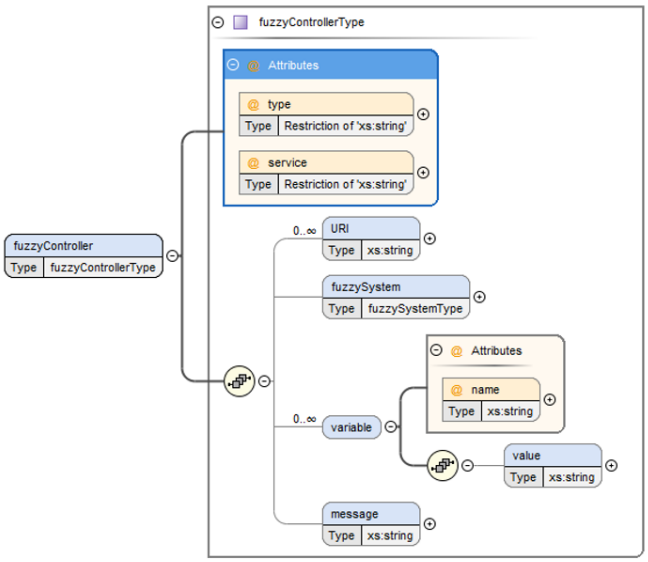


Fig. 3: Schema of request/response the designed fuzzy-as-a-service API. The fuzzySystem element is the core element that follows the IEEE1855-2016 standard, whereas the other elements are to be considered as extensions for the Web services.

Presenting the full details of the API input/output parameters is beyond the size of this paper. However, the main supported functionalities of the developed API are as summarized here.

1) Creation/Modification of a fuzzy logic system (createFLS/editFLS calls): This feature is provided for the client(s) responsible for the FLS project first creation and/or modification. IEEE 1855-2016 is used to describe the created/modified FLS, so the client machines compose a request that contains the FLS description in the standard format.

2) Querying an FLS (queryFLS call): The FLS status information stored on the server may need to be retrieved by different client forms. The FLS state includes its life and stored constraints or the latest input values.

3) Setting input (setInput call): For the FLS inputs, customers must be able to set individual values. It is also required that *any* number of inputs can be included in a single invocation by a specific client system. In a multi-input multi-output FLS, this balances data collection burden by allowing the system to capture individual or clustered outputs from various devices. Every collected input *can* be stored in the server, but during the execution of FLS, only the last collection will be used.

4) Getting output (getOutput call): The FLS output value(s) must also be calculated and obtained by the server on *the clients' requests*. It is favoured that the server(s) may return the requested output either as a fuzzy set (a data array) or as a defuzzified value, but for brevity, it is restricted the requisite to the defuzzified values. Particularly in a multi-input multi-output system, being able to request any number of output values means that the system can evade redundant and unsolicited output computation. The same FLS can be specified on several servers in a complex FLS computing situation, but each server can provide a part of the performance. The sensors can send their data to multiple servers in this scenario, but an output device can selectively request their required data from a single server while the other servers are busy offering data to other output devices. The server may also have access to a lookup table of the previous FLS runs, so it can avoid reiterating the calculation if in the past the same FLS has been running with the same inputs. The system will reply to the user if the requested output can be either measured or retrieved

5) Deletion of fuzzy logic system (deleteFLS call): Eventually, clients must be able to request to delete an FLS from the list of specified FLSs in the database if there is no longer any need for either the FLS description or its past history of input/output. Optionally, the device may be designed to remove the FLSs after a period of inactivity.

6) Serving different FLSs simultaneously: Each FLS is given a unique identifier once it is created. This is called URI, as shown in Fig. 3. Each following API request/response includes the URI as a mandatory parameter. As a result, the same Web server serves multiple FLSs.

The above architectural components and API functionalities collectively deliver the so-called fuzzy-as-a-service. Obviously, this architecture is be itself an empty platform that must be populated by different FLSs for serving different applications. In the next section, the design of FLS is explained for human activity detection scenario and its configuration to act as a real-time Web service.

IV. Developing A Fuzzy-As-A-Service for Human Activity Recognistion

A. The Experiment Settings

To demonstrate the utility of the described architecture, a fuzzy-as-a-service solution for a human activity classification scenario is developed. To validate the system, a dataset of accelerometer/gyroscope measurements coming from body sensors of individuals and labelled with their walking/running status is used. A fuzzy rule-based system is designed, in which the rules are trained by sample input/output pairs. Then the designed FLS is set up on the Web server in order to classify the individual´s current status as running or walking. All the FLS definitions, input data collection, data processing and output classification (running/walking) data are being carried out in real-time purely via API calls.

The experiment and its findings were confined to a particular sample scenario and has been used in this paper as a proof-of-concept, so that the suggested approach will be implemented in future works in a practical problem where more rigorous research will be conducted for real-world scenarios and actual sensors/output devices are established.

In a real-world scenario, the setInput API calls should be sent by Web-connected sensors, or through some Web-connected interfaces. In our simulation, a client-side computer program is developed that reads the sensory data from the dataset and send them back-to-back to the Web server in the same rate that they were originally produced by the sensors. Another program (running on a different client PC) is also written that simulates a "monitoring station", in which the runner/walker status is requested from the Web server in some different rate from that f the data collection (by sending back-to-back getOutput API calls). On each getOutput request, the server runs the necessary FLS calculation and delivers the real-time running/walking status back to the monitoring station. As a result, the human activity is detected in real-time. This process is illustrated in Fig. 4.

A Web application server is developed to serve the necessary HTTP messages to/from clients following the above SOA model. JFML [4] is used for fuzzy logic computing within the developed application server. The server is using Apache Tomcat as a generic Java servlet implementation. This setting on the server side can fit into numerous applications, specifically where the client-side agents are sensors and/or actuators/monitoring stations are distributed in a smart environment.

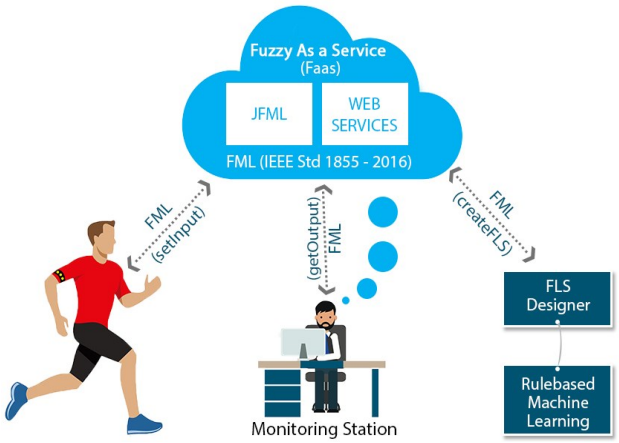


Fig. 4: The elements of the carried-out experiment for human activity monitoring

B. Dataset

To evaluate the performance of the proposed system, a dataset taken from Kaggle.com open repository called "Run or Walk". The dataset contains 88588 sensor data samples from accelerometer and gyroscope collected from smart phone located on a person's wrist and an average of about 5 seconds frequency. This dataset contains six attributes including accelerometer and gyroscope sensor data, each in 3 dimensions, along with their timestamps. The dataset is also labelled with the actual status where 1 indicates that person is running and 0 indicates person is walking.

The dataset includes about 90K data samples where about 60K samples were used for training and the remaining data was used for testing. A sample part of the Run or Walk dataset is shown in Fig. 5.

date	time	activity	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z
30/06/2017	13:55:45	0	0.2927	-0.8271	0.0291	-0.5193	0.5369	-1.3963
30/06/2017	13:55:45	0	0.4243	-1.0259	-0.2221	0.8334	-0.0653	1.7977
30/06/2017	13:55:46	0	0.3722	-0.6749	-0.1641	0.7345	0.238	1.2111
30/06/2017	13:55:46	0	0.5202	-1.2726	-0.3422	-0.3252	-0.2746	0.501
30/06/2017	13:55:46	0	0.5255	-1.0243	-0.1791	-0.7732	0.2769	-1.2629
30/06/2017	13:55:46	0	0.2717	-0.7515	0.0187	-0.544	-0.1414	-1.817
30/06/2017	20:33:44	1	1.2842	0.0526	-0.2362	-1.9543	2.4977	1.3038
30/06/2017	20:33:44	1	0.3057	0.4504	-0.1	1.473	-0.5391	-2.603
30/06/2017	20:33:44	1	1.0601	-0.9857	-0.0732	1.43	-0.9917	-2.3499
30/06/2017	20:33:44	1	-0.1065	-0.7203	0.2003	-0.4013	0.424	3.0591
30/06/2017	20:33:45	1	1.0069	0.3441	-0.276	-1.6086	2.3222	1.4736
30/06/2017	20:33:45	1	0.0997	0.4844	-0.0588	1.0809	-0.6441	-2.7383
30/06/2017	20:33:45	1	1.5466	-1.1082	-0.2509	0.8609	-0.4994	-1.6458

Fig. 5: A sample parts of the run/walk dataset used in this experiment for both training and testing.

C. Rule-base Training

Rule-base training was carried out using the training set part of the dataset, based on the known Wang-Mendel's "learning from example" algorithm [20, 27]. This algorithm was independently carried out prior to the main Web services experiment. For each input data, 5 evenly distributed triangular fuzzy sets were used, whereas the binary output data was represented as two singleton fuzzy sets located at 0 and 1.

As a result, a total of 247 rules were created based on the six inputs and one output. The number of rules is relatively high, so it leads to a computationally intense FLS. Achieving a real-time data processing with such a system might to respond in real-time if implemented locally in the environment using low processing power of the embedded devices. Therefore, this could be a suitable use case for checking if the Web service can provide a real-time processing.

List 1: Partial KB and RB of the adopted FML

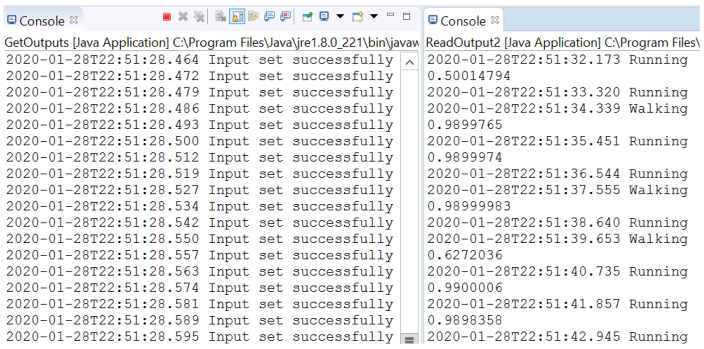
```
<?xml version="1.0" encoding="UTF-8" ?>
  <fuzzyController type="request" service="createFLS">
      <URI>FLS7</URI>
      <service>createFLS</service>
    <fuzzySystem name="Bhavesh">
       <knowledgeBase>
         <fuzzyVariable name="ax" scale="" domainleft="0.0" domainright="2.5"
type="input">
   <fuzzyTerm name="1" complement="false">
      </fuzzyVariable>
            .....
     <fuzzyVariable name="gz" scale="" domainleft="0.0" domainright="2.5"
type="input">
       <fuzzyTerm name="1" complement="false">
         <triangularShape param1="-0.625" param2="0.0" param3="0.625"/>
        </fuzzyTerm>
        </fuzzyVariable>
       <fuzzyVariable name="y" scale="" domainleft="-5.0"   domainright="5.0"
type="output" accumulation="MAX" defuzzifier="COG" defaultValue="0.0">
        <fuzzyTerm name="0" complement="false">  </fuzzyTerm>
              <fuzzyTerm name="1" complement="false">  </fuzzyTerm>
         </fuzzyVariable>
         </knowledgeBase>
<mamdaniRuleBase activationMethod="MIN" andMethod="MIN" orMethod="MAX"
networkAddress="127.0.0.1">

  ……..                                  |  …….
  <rule name="1" andMethod="MIN"         |  <rule name="247" andMethod="MIN"
  orMethod="MAX" connector="AND"         |  orMethod="MAX" connector="AND"
  weight="1.0"                           |  weight="1.0"
  networkAddress="127.0.0.1">            |  networkAddress="127.0.0.1">
            <antecedent>                 |            <antecedent>
             <clause>                    |             <clause>
               <variable>ax</variable>   |               <variable>ax</variable>
               <term>1</term>            |               <term>4</term>
             </clause>                   |             </clause>
             ........                    |             ......
             <clause>                    |             <clause>
               <variable>gz</variable>   |               <variable>gz</variable>
               <term>1</term>            |               <term>1</term>
             </clause>                   |             </clause>
            </antecedent>                |            </antecedent>
            <consequent>                 |            <consequent>
             <then>                      |             <then>
               <clause>                  |               <clause>
                <variable>y</variable>   |               <variable>y</variable>
                 <term>1</term>          |                 <term>1</term>
               </clause>                 |               </clause>
             </then>                     |             </then>
            </consequent>                |            </consequent>
          </rule>                        |    </rule>

      </mamdaniRuleBase>
    </fuzzySystem>
</fuzzyController>
```

D. FLS Definition

The outcome of the training was used to compose the FLS description in FML format. As shown in List 1, the same triangular fuzzy sets for inputs and singleton fuzzy sets for output was embedded in the FLS description. Other parameters

set for the FLS include centroid defuzzification and Mamdani inference with min and max operators for the t-norm and t-conorm respectively. Finally, the FLS description was composed in FML format and was sent to the server via a single API call from a simulated "FLS designing station" client machine as shown in Fig. 4. It is noticeable that the same machine can later modify the FLS parameters dynamically. It can even intervene in the middle of an experiment, by resending an updated FLS description via another API call, in which the FLS outputs will be automatically adjusted in real-time according to the new FLS parameters. It is an important feature to be considered in adaptable and dynamically trained FLSs.

(a)



(b)

Fig. 6: Realtime human activity classification process using HTTP request/responses to the developed Web Services. (a) A sensor client console view: sending back-to-back XML request to the server to set the input variables (i.e., the accelerometer and gyroscope data); (b) Consoles views of two monitoring clients: On the right side the server acknowledgements per request are received; On the right side, the output values (i.e., the activity classification) per requests are coming from the server (the client's requests for getting the output values are not shown here).

E. Running the Experiment

Once the FLS is created on the server side, the fuzzy-as-a-service system is ready to serve setInput and getOupput requests coming from different clients simultaneously.

Two client programs independently started to send back-to-back setInput and getOutput requests to the server in different frequencies. The highest testing rate was two simultaneous requests every one second - whereas the original data collection frequency was about 5 seconds. In this case, the server was able to calculate the results based on the latest coming input and to send the defuzzified output back within the 1-second window, without missing any of the coming requests. This shows the ability of processing data in a computationally intense FLS over the developed Web services in a more sampling rate that would be necessary in the real-world scenario. Fig. 6 shows a sample running of the system where it serves the requests of the both clients in parallel.

The accuracy of the activity classification performed by this FLS Web Service is also shown in Table 1. This shows a noticeably high accuracy processed in real-time. Although the classification performance of the FLS is provided here, it is noticeable that the purpose of this experiment is not to check if the designed FLS can do any more or less accurate than other classifiers. Instead, this should be considered as a proof-of-concept for achieving a real-time FLS execution over using Web services, which does not have any similar implementation as far as we are aware. A comparative study between the performance of this system against a non-Web solution is our subject of a future work.

Table 1: Human activity monitoring result.

Data samples	**Type**	**Accuracy**
59058	Training	97.23%
29530	Testing	97.42%
88588	Overall	97.30%

V. CONCLUSION AND FUTURE WORK

As an example of implementing the recently accepted IEEE-1855-2016 standard, this study offers a novel approach for a web-based service-oriented FLS architecture. It is shown through experimenting with human activity monitoring datasets, that the proposed service-oriented architecture is capable of undertaking real-time data processing using a complex fuzzy rule-based system. The accuracy and the response time of the developed system was shown to be relatively high.

Although the architecture is viewed in the sense of AmI environments, it can be extended to a much broader application field, i.e., wherever an FLS's storage logic requires to be abstracted from its logic of information and presentation. The main motivation is to allow the versatile delivery from the clients to dedicated servers of potentially complex computation needed for FLSs. Unambiguously, the use of virtualized cloud services provides the system with elasticity, essentially allowing universally accessible fuzzy-as-a-service. The other benefits of such an architecture are network share, hardware/software autonomy, reuse of existing data, balancing the load between FLS devices, and cost-efficiency.

The technical standpoint of the proposed system has several facets of extension. The FLS community are encouraged to participate and provide input on the design in terms of feature prioritization, as well as supporting the collective development effort in deployable applications. This refines the proposed schema and/or invocation formats of the API and offer more detailed specialized feedback for other architecture implementation. Note that this study only discusses a specific case of fuzzy logic systems (i.e., the rule-based systems). Other fuzzy services can be considered in the near future, such as fuzzy querying about fuzzy databases or fuzzy ontologies. In addition, expanding web services to the Semantic Web (e.g., developing cloud-based searchable repositories of FLSs) will be a noteworthy choice due to the close relationship between FML and fuzzy ontologies.

REFERENCES

[1] Pourabdollah, A., Wagner, C., Acampora, G., & Lotfi, A. (2018, July). Fuzzy Logic As-a-Service for Ambient Intelligence Environments. In 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) (pp. 1-7). IEEE.

[2] Acampora, G., & Loia, V. (2005, May). Using fuzzy technology in ambient intelligence environments. In The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05. (pp. 465-470). IEEE.

[3] Acampora, G., Di Stefano, B., & Vitiello, A. (2016). IEEE 1855: The First IEEE Standard Sponsored by IEEE Computational Intelligence Society [Society Briefs]. IEEE Computational Intelligence Magazine, 11(4), 4-6.

[4] Soto-Hidalgo, J. M., Alonso, J. M., Acampora, G., & Alcala-Fdez, J. (2018). JFML: a java library to design fuzzy logic systems according to the IEEE std 1855-2016. IEEE Access, 6, 54952-54964.

[5] Arcos, F. J., Soto-Hidalgo, J. M., Vitiello, A., Acampora, G., & Alcala-Fdez, J. (2018, July). Interoperability for Embedded Systems in JFML Software: An Arduino-based implementation. In 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) (pp. 1-8). IEEE.

[6] Alcala-Fdez, J., Alonso, J. M., Castiello, C., Mencar, C., & Soto-Hidalgo, J. M. (2019). Py4JFML: A Python wrapper for using the IEEE Std 1855-2016 through JFML. In IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2019) (pp. 1-6). IEEE

[7] Lee, C. S., Wang, M. H., Wang, C. S., Teytaud, O., Liu, J., Lin, S. W., & Hung, P. H. (2018). PSO-based fuzzy markup language for student learning performance evaluation and educational application. IEEE Transactions on Fuzzy Systems, 26(5), 2618-2633.

[8] Lee, C. S., Wang, M. H., Ko, L. W., Tsai, B. Y., Tsai, Y. L., Yang, S. C., ... & Shuo, N. (2019). PFML-based Semantic BCI Agent for Game of Go Learning and Prediction. arXiv preprint arXiv:1901.02999.

[9] Ramos, F. B. A., Lorayne, A., Costa, A. A. M., de Sousa, R. R., Almeida, H. O., & Perkusich, A. (2016, July). Combining Smartphone and Smartwatch Sensor Data in Activity Recognition Approaches: An Experimental Evaluation. In SEKE (pp. 267-272).

[10] Marques, G., & Pitarma, R. (2018, November). Smartwatch-based application for enhanced healthy lifestyle in indoor environments. In International Conference on Computational Intelligence in Information System (pp. 168-177). Springer, Cham.

[11] Jiménez, A. R., Seco, F., Peltola, P., & Espinilla, M. (2018, September). Location of persons using binary sensors and BLE beacons for ambient assitive living. In 2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN) (pp. 206-212). IEEE.

[12] Ahire, S. B., & Khanuja, H. K. (2015). HealthCare recommendation for personalized framework. International Journal of Computer Applications, 110(1).

[13] Foko, T.E., Dlodlo, N. and Montsi, L., 2013. An integrated smart system for ambient-assisted living. In Internet of Things, Smart Spaces, and Next Generation Networking (pp. 128-138). Springer, Berlin, Heidelberg.

[14] Calvaresi, D., Cesarini, D., Sernani, P., Marinoni, M., Dragoni, A.F., & Sturm, A. (2017). Exploring the ambient assisted living domain: a systematic review. *Journal of Ambient Intelligence and Humanized Computing, 8*, 239-257.

[15] Patel, A., and Shah, J. "Sensor-based activity recognition in the context of ambient assisted living systems: A review." Journal of Ambient Intelligence and Smart Environments 11, no. 4 (2019): 301-322.

[16] Sun, Hong, Vincenzo De Florio, Ning Gui, and Chris Blondia. "Promises and challenges of ambient assisted living systems." In 2009 Sixth International Conference on Information Technology: New Generations, pp. 1201-1207. IEEE, 2009.

[17] Acampora, Giovanni. (2007). Transparent Fuzzy Agents in Ambient Intelligence Environments. 10.13140/RG.2.1.1249.6401.

[18] Acampora, G., Appiah, K., Hunter, A., & Vitiello, A. (2014). Interoperable services based on activity monitoring in Ambient Assisted Living environments. *2014 IEEE Symposium on Intelligent Agents (IA)*, 81-88.

[19] Sun, H., Florio, V.D., Gui, N., & Blondia, C. (2009). Promises and Challenges of Ambient Assisted Living Systems. *2009 Sixth International Conference on Information Technology: New Generations*, 1201-1207.

[20] Wang, L.X. and Mendel, J.M., 1992. Generating fuzzy rules by learning from examples. IEEE Transactions on systems, man, and cybernetics, 22(6), pp.1414-1427.

[21] World Health Organization (2011) Global health and ageing. World Health Organization, Geneva.

[22] Vimarlund V, Wass S. Big data, smart homes and ambient assisted living. Yearb Med Inform. 2014;9(1):143–149. Published 2014 Aug 15. doi:10.15265/IY-2014-0011.

[23] G. Acampora, V. Loia, A. Vitiello, Distributing fuzzy reasoning through fuzzy markup language: an application to ambient intelligence, in: A. Giovanni, L. Vincenzo, L. Chang-Shing, W. Mei-Hui (Eds.), On the Power of Fuzzy Markup Language, Springer,Berlin, 2013, pp. 33-50.

[24] IEEE-SA Standards Board, IEEE standard for fuzzy markup language, IEEE Std. (2016), 1855-2016.

[25] G. Acampora, B. di Stefano, A. Vitiello, IEEE 1855TM: the first IEEE standard sponsored by IEEE Computational Intelligence Society, IEEE Comput. Intell. Mag. 11(4) (2016), 4-6.

[26] J.M. Soto-Hidalgo, A. Vitiello, Jose M. Alonso, G. Acampora, and J. Alcala-Fdez. "Design of Fuzzy Controllers for Embedded Systems with JFML" in International Journal of Computational Intelligence Systems, vol. 12(1), pp. 204-214, 2019.

[27] Z. Chi, H. Yan, T. Pham. Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition. World Scientific, 1996.