Developing a cloud-based service-oriented architecture for fuzzy logic systems

Bhavesh Pandya*, Amir Pourabdollah*, Ahmad Lotfi* and Giovanni Acampora†‡
* School of Science and Technology,
Nottingham Trent University, Napoli, Nottingham, United Kingdom,
Email: {bhavesh.pandya, amir.pourabdollah, ahmad.lotfi}@ntu.ac.uk
† Department of Physics "Ettore Pancini" University of Naples Federico II, Napoli, Italy
Email: giovanni.acampora@unina.it
‡Istituto Nazionale di Fisica Nucleare, Sezione di Napoli, 80126 Napoli, Italy
Email: giovanni.acampora@na.infn.it

***Abstract*—Fuzzy logic systems are customarily related to specific hardware or software systems. Nevertheless, it has been observed that distributed and cloud-based architectures of various intelligent systems are pouring intensifying attention. While the distributed architectures can potentially add values in developing fuzzy systems, a lack of standard methods and practices may limit their public use. This study aims to provide a standard solution for developing cloud-based service-oriented architectures for fuzzy logic systems, based on extending IEEE-1855 (2016) in the defining system and exchanging data. Experiments were performed employing simulation concerning collection, processing and monitoring of data in a distributed manner over the web. A real-time human activity recognition simulated scenario is also demonstrated through a cloud-based fuzzy system.**

***Index Terms*—Fuzzy logic, Fuzzy Markup Language, Cloud Computing**

I. Introduction

Fuzzy logic plays a crucial role in the fields of computer science, AI, control theory and mathematics. It is also used in businesses, medicinal and behavioural sciences and engineering. Utilising fuzzy logic has resulted in immense changes. The beginning of fuzzy logic has made several things uncomplicated, and this has led to the saving of not only time and cost but also energy. Moreover, fuzzy logic is used in intelligent problem-solving systems and applications. Fuzzy systems normally require large data to be processed, and this may not be possible by a single, small standalone system or sensors. Hence, such data relies on cloud computing for processing. The use of fuzzy logic with cloud for processing leads to the betterment of fuzzy logic systems in terms of their applications.

Due to a vast number of users gradually linked to the internet, cloud computing has lately drawn the interest of professionals, and academic communities [1]. Cloud users can access cloud resources from all over the world via web browsers and the internet if and when necessary. On-demand and in a modular manner, it delivers computational power [2]. This study makes use of the cloud for handling the processing power so that the state-of-the-art architecture can be developed and used in real-time with minimal use of hardware, which includes fuzzy logic. There are several problems in cloud computing that can be overcome with the aid of fuzzy logic [3]. The fuzzy markup language is a web communication language that is evolving as a basic method for the human-understandable and hardware-independent modelling of fuzzy logic controllers [4]. Fuzzy markup language enables fuzzy logic control designers to describe their structures independently of their legacy representations by means of these characteristics and provide them with with a set of facilities that speed up the entire process of creating a fuzzy structure [5], [6]. Nevertheless, the XML nature of the fuzzy markup language, in addition to the hardware independence function, makes it possible to incorporate fuzzy logic in computational scenarios marked by high degrees of omnipresence. [7].

The idea of developing fuzzy systems as a service on the cloud has been recently introduced in [8], [9]. A study on cloud-based service-oriented architecture for fuzzy logic systems for human activity recognition was conducted by [10]–[12]. This study aims to introduce a general standardised framework for fuzzy systems as-a-service (called FaaS).

Fuzzy logic systems (FLSs) have made known their competence in Ambient Intelligence (AmI) applications. Nonetheless, FLS deployment needs committed hardware/software systems. Sharing FLSs ability as web services permit flexibility, openness, load balancing, efficient resource distribution and value for money. Distributed architecture development for FLSs is a fairly new area with not much progress. FML is one of the standards that could be used to carry out communication between cloud and fuzzy systems. In other words, an internet-based language for FLS characterizations is the key pre-requisite for implementation/communication. Thus, the current standard is IEEE-1855 (2016) (FML), an XML-based markup language that facilitates an FLS to be construed as human comprehensible and hardware independent [1]. The use of fuzzy logic, cloud computing and fuzzy markup language will result in an appropriate AmI environment, and a suitable service-oriented architecture can be developed using fuzzy logic systems.

The main contribution of this paper is that for the first time, a distributed architecture or fuzzy logic system is proposed that includes the use of FML and cloud for developing an

SOA-based fuzzy system (i.e. fuzzy-as-a-service (FaaS)).

The rest of the paper is organised as follows; Sections II and III focus on background and literature review for performing this study, respectively. Section IV explains the methodology that has been used in this study. The experimental setup has been discussed in Section V. Section VI discusses the results and discussion based on the performed experiments. Section VII concludes the study with future scope.

II. Background

The intention behind developing a cloud-based service-oriented architecture for fuzzy logic systems using FML is partly based on the probable and increasing curiosity of the distributed use of fuzzy logic systems in an atmosphere of AmI environment [12].

Some of the aspects are specified below:

A. Need for distributed fuzzy system:

Fuzzy logic system components need to be substantially distributed in an AmI environment so that sensors can collect input data; processors can execute the system and actuators (i.e. output devices) can be distributed in the environment [11]. Nonetheless, in order to achieve both reliability and flexibility, it is necessary not only to distribute input and output equipment but also to distribute computing power between servers. Dynamic balancing of tasks between fuzzy logic device elements involves a need for an open system, which is difficult when permanent hardware and software designs are taken into account.

B. Open system:

This is a general motive not limited to AmI for some form of fuzzy logic system applications. Note that the tools available for computing fuzzy logic systems, e.g., some libraries/tools in MATLAB and R software tools are generated for fuzzy logic computation are typically fitted for a single purpose. When an independent architecture is developed, network connectivity to cloud service providers removes any need to install certain tools/libraries for fuzzy logic system computations [13], [14]. For academic and educational purposes, the platform-independent design is advantageous, where usability and dissemination of both resources and outcomes are essential. Moreover, such an architecture leads to the development of systems that can work in a cross-platform environment. The other benefits of developing a platform-independent architecture is that FLS computation accessible from any web-connected devices present at any geographical location.

C. Configuring the fuzzy systems from anywhere:

Cloud computing can be regarded as a valuable extension of the client–server architecture because it makes allocation of system resources more flexible based on required computational power for a specific FLS. The fuzzy systems can be configured remotely and independently from their working environment. For example, the system rule-base can be updated from any place.

D. Reuse of computational results:

To circumvent recurrent computations, a server can record inputs and produce output. For instance, if a specific device's inputs have been handled in the past, then the stored outputs can be used as a reply by means of a lookup table. A single FLS's definition for a specific application may be combined and reprocessed by several applications in dissimilar surroundings. This is possible only if FLS servers can be inquired for FLSs' definitions and their past input/output description.

E. Scalability of fuzzy logic system:

Fuzzy logic systems can be extended further and can be merged with artificial intelligence so that a smart system can be developed in an AmI environment with the help of a cloud-based service-oriented architecture. This possibility arises from the well-known Zadeh's Incompatibility Principle stating that when the complexity of a specific system rises, precise categorical statements about that system lose meaning and meaningful statements cease to be precise and categorical. The linguistic approach used by fuzzy logic allows to overcome this limitation and make fuzzy systems extremely scalable to be embedded in complex frameworks such as AmI environments [15]. In these scenarios, fuzzy rule bases semantically related to different aspects of the environment to control will be designed as standalone systems and plugged into the artificial intelligence ecosystem without any effort.

F. IEEE 1855: a standard language for fuzzy logic

Scalability is an important concept, but it could lose all its potential if not accompanied by a technology for data abstraction that allows to model computing with word in human-readable and hardware independent manner [16]. IEEE 1855-2016 also known as Fuzzy Markup Language(FML) is an XML based language achieving the above goal. This abstraction language, originally introduced as a tool for modelling the behaviour of devices in smart homes, has been used in different applications domains and, in 2016, it has been made an IEEE standard technology. The main benefits provided by IEEE 1855 are related to the advantage of being directly convertible to programming logic when it comes to server-side programming, so as to strongly minimising the development time of this kind of architectures. This aspect is particularly useful in AmI environments, where collection of heterogeneous devices need to be opportunely programmed and communicate among them [17].

III. Literature Review

There have been successful utilisation of fuzzy systems in countless fields like decision-making and image recognition, which has enthused the beginning of various tools for developing fuzzy systems. Moreover, fuzzy system development has been conducted using several programming languages, platforms, and libraries that are both commercial and open source.

The work reported in [18], developed an interoperability module specifically for Arduino boards to build and operate fuzzy logic systems for embedded systems in the Java fuzzy markup language. A communication protocol also was established between the Java fuzzy markup language and Arduino, which helped to eliminate the restricted computational capabilities provided by embedded systems. To demonstrate the interoperability module's capability, the authors considered a study of a wall-following fuzzy controller as per IEEE Std 1855TM-2016, which controls a mobile robot in two environments.

Java fuzzy markup language (JFML) presented in [19], which is an open-source GPLv3 certified Java library that is ready to not only build but also use fuzzy logic systems as per IEEE Std 1855-2016, was introduced by the authors. Note that JFML has proposed to fully incorporate four fuzzy inference systems (i.e. Mamdani, TSK, Tsukamoto and AnYa) used in the basic specification of the W3C XML schema. To demonstrate the potential of JFML and its benefits for the sharing of fuzzy logic structures through various software, the authors put forth three case studies. For the well-known tipper regression dilemma, the first case study was to develop a fuzzy logic system. The second case study focussed on designing a fuzzy logic system for controlling a robot's wall-following behaviour. The third case study focused on designing a preliminary fuzzy logic system for classification with MATLAB fuzzy logic toolbox. Later on, JFML was used to improve the design and estimate unidentified input values.

Pourabdollah et al. [8] used structured FML (IEEE-1855) and recommended extensions to design web servers for computing the fuzzy logic system. They adopted the novel approach to combine multiple elements of the fuzzy logic paradigm into a single web server framework using a well-specified language via HTTP requests/responses for communication over the web. Due to the design of fuzzy logic structures in AmI settings, the effectiveness of this method was demonstrated.

How the native extensibility functionality of IEEE Std 1855 allowed Arduino architectures to build fuzzy rule-based systems in an interoperable manner was shown in [20], [21]. In addition, this feature helped programmers to concentrate on fuzzy ideas without considering the specifics of hardware and software belonging to a particular Arduino device.

A proposal on developing a fuzzy-as-a-service system. Their proposal consisted of three major objectives: a description of cloud services for fuzzy systems using semantic technologies, the composition of services, and the exploitation of cloud computing model in cloud platforms for integration with other services [9].

A new open-source fuzzy system software was implemented by the authors of [22] that could allow the modelling of fuzzy systems as per IEEE Std 1855, which improved Java Fuzzy Markup Language via a GUI-based visual framework. According to their analysis, VisualJFML provided considerable feedback as it allowed designers without programming expertise to model shareable fuzzy structures. Through a case study that dealt with the iris classification issue, they demonstrated their user-friendly interface supported by VisualJFML.

A new Java fuzzy markup language module in their analysis that helped developers design and deploy fuzzy ruled-based frameworks for open embedded hardware systems developed in [23]. For Arduino and Raspberry Pi, the new module was ready, but it was easily extendable to other hardware architectures. The new module was able to generate running files on Arduino or Raspberry Pi automatically to assist non-expert users (i.e. those who do not possess any programming skills). The authors explained their new module by means of two case studies.

Using the wearable accelerometer and gyroscope sensors with the aid of fuzzy logic web providers, a study was carried out real-time dropping detection. Wearable sensors were used to demonstrate human behaviour tracking using a rule-dependent fuzzy logic system in their research. They proposed an algorithm to determine the occurrence of a fall and non-fall and attained an accuracy of 90% along with the sensitivity of 88.89% and specificity of 91.67% [11].

A fuzzy-as-a-service architecture based on IEEE Std 1855-2016, Java fuzzy markup language and service-oriented architecture was proposed in [12]. Simulation experiments were performed over the web that involved steps like data collection, processing and monitoring. Therefore, with the aid of the rule-based fuzzy logic system, the authors were able to show a simulated scenario of real-time human activity detection.

A comparative study of real-time fall detection using fuzzy logic web services and machine learning techniques was performed in [10] and determined which one is better for real-time fall detection. Their study showed that the proposed fuzzy-as-a service in a real-time setting with a precision of 90% was in a better position to distinguish between fall and non-fall occurrences. In comparison, the authors attained a precision of 99.19% when the random forest machine-learning algorithm was used.

This study aims to develop a cloud-based SOA for fuzzy logic systems. Herein, fuzzy logic, FML and cloud are used, thereby creating a distributed architecture wherein processing will be carried on cloud and FML will be used to carry out communication between the devices from where data is obtained (i.e. smart watches, mobile phones and web connected devices) and FaaS. The main advantage of this system is that any system that is connected to such type of service will be able to access data, which implies openness and accessibility.

IV. Proposed Architecture

In a decentralized AmI context, the inspiration behind a service-oriented approach to FLSs is focused on the possibility of using FLSs. In the first section, a Hardware architecture is discussed, then a brief description of the specific software components used and the features implemented to address the described attributes is presented.

A. Hardware architecture

Web connected devices are used in our proposed system to develop a cloud-based service-oriented architecture for fuzzy

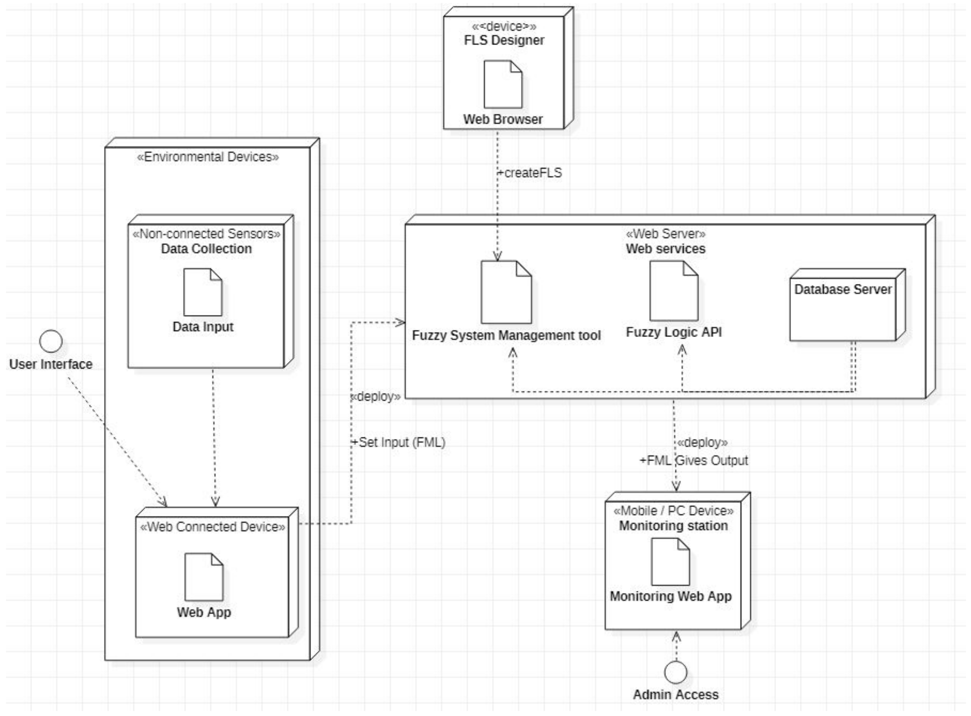

Fig. 1. The deployment diagram of client-server communication through the API.

logic systems. The significant benefit of using such sensors is that they can be connected to any device such as IoT devices [24], [25]. The other benefit is that real-time data can be obtained from these sensors, which can be used for prediction purposes. On the other hand, for server-side architecture, we made use of Amazon web services. The advantage of using Amazon web services is that real-time data is obtained from the sensors and processed using a cloud service, which gives the desired results instantaneously.

B. *Fuzzy logic processing*

A fuzzy logic system defined in IEEE-1855 can be converted into numerous programming languages such as Java [26] without any delay using an extensible stylesheet language translator, and so, minimal attempts from the server are required to encode a fuzzy logic system's description into a local program logic. IEEE-1855 also authorises various agents to monitor the same fuzzy logic system that communicates with the environment from dissimilar locations as illustrated in Fig.1. This study aimed to focus on fuzzy mark-up language's known capabilities in defining a fuzzy logic system by considering essential parameters such as input fuzzy sets, rule base, inference method, output fuzzy sets and defuzzification [27].

An open-source library for fuzzy logic computations based on FML (IEEE 1588-2016) data format is JFML [19]. JFML is developed using Java and can work as a cross-platform backend application server for the developed fuzzy-as-a-service. JFML adheres to an object-oriented technique and a modular architecture based on a similar tree structure that is used by FMLs to describe FLSs. This enables developers to extend JFML without changing the grammar of the language. JFML facilitates the use of standard fuzzy inference systems that are present in XML schema definition which includes membership functions and fuzzy operators. Other components could be needed for further use by researchers; however, those components are not included in XSD's current definition. Therefore, JFML makes available custom methods for all elements specified in XSD offering a method to expand the library in fulfilment with this standard devoid of any changes in the grammar of the language.

C. *Fuzzy system management tool*

Web interface is a software that people can design FLS as per their requirements. People can add/remove the system and anybody can use it at anytime from anywhere using http request and response. We developed a special web version to promote web services in a a homogeneous manner or medium to propagate communication between clients and server applications (via FML) on the world wide web to make it simpler for people by means of cloud computing. SOAP, WSDL, and REST are examples of web service applications. The best way to use RESTful web resources is through the cloud. When a web service uses REST, it identifies limitations such as a single interface that give rise to desired characteristics that allow services to work effectively on the web. In the REST architectural style, data and features are referred to as resources and can be accessed via URIs, which are typically web links. Using numerous well-defined operations, the resources will be used. An API consisting of an HTTP request and response is used to configure a web service invocation for each feature.

The functionalities of the API that have been established are described later in this paper.

D. Extending IEEE-1855 Schema

Note that the request and response of the XML schema are different from that of the IEEE-1855 schema. From the schema, it is evident that the recently added XML elements lie within two major elements, i.e. FLSRequest and FLSResponse. This has been done intentionally and placed within the IEEE-1855's root element (FuzzyController) [1]. In other words, FLSRequest, FLSResponse and their sub-elements are candidate extensions of the original IEEE-1855 schema. The other interpretation here would be that all responses and requests from or to the server can be substantiated through a single schema. There could be a possibility of producing an extended schema when the schema for the newly required elements and the standard IEEE-1855 schema are combined. The extended schema would have the same root element from the original schema. Moreover, the original sub-elements, i.e. KnowledgeBase and RuleBase, [23], [20] would be added along with two additional complex-type elements for handling FLS requests and responses under the root element. The standard IEEE-1855 has been opportunely designed to enable a simple and direct extension to allow the original language to be adapted to different application scenarios.

In the next subsection, the schema of the extension and the API design are explained using examples.

E. API design.

1) createFLS: The client machines generate a request that includes the FLS definition in IEEE 1855-2016 standard format which contains information and values for fuzzy variables with a unique URI ID.

Create an FLS request consists of URI and fuzzy system tags as input parameters and "createFLS" as a service attribute. The fuzzy system consists of two major elements, i.e. knowledge base and rule base. The knowledge base consists of fuzzy variables such as pulse and SpO_2 with values such as critical, alert, low and normal. There are sixteen rules in the rule base (considered as per the selected example), depending on which we can decide if a patient is COVID-19 critical or not.

Request:

```
<fuzzyController type="request" service="createFLS">
    <URI>Covid_FLS</URI>
    <fuzzySystem>...</fuzzySystem>
</fuzzyController>
```

Response:

```
<fuzzyController type="response" service="createFLS">
    <URI>Covid_FLS</URI>
    <message>System created successfully.</message>
</fuzzyController>
```

The above response's XML schema (as well as the responses in the following functions) varies from the IEEE-1855 schema. The schema extension mentioned in the following subsection would explain how the newly introduced features are dealt.

2) setInput: After the backend server has created the FLS file, values need to be set by the client in that fuzzy system for different variables, so that the system can provide the desired output. The service "setInput" is used for this reason. Herein, the XML request service attribute is "setInput", and the input parameter is a variable name. These crisp values are saved at the backend server so that the fuzzy system can evaluate the end result. The sample code are as follows:

Request:

```
<fuzzyController type="request" service="setInput">
    <URI>Covid_FLS</URI>
    <variable name="pulse">
        <value>50</value>
    </variable>
    <variable name="spo2">
        <value>80</value>
    </variable>
</fuzzyController>
```

Response:

```
<fuzzyController
    type="response" service="setInput">
    <URI>Covid_FLS</URI>
    <message>Input(s) set successfully.</message>
</fuzzyController>
```

The Client can set data repeatedly into the database but the latest data can be used for evaluating the result.

3) getInput: This service is used to retrieve the latest information about variables. Values are retrieved for variable names provided in the request under the fuzzy controller type. A specific URI file is searched at the backend server and tried to match with the variable names passed in the request by iterating the entire file.

Request:

```
<fuzzyController type="request" service="getInput">
    <URI>Covid_FLS</URI>
    <variable name="pulse" />
    <variable name="spo2" />
</fuzzyController>
```

As shown in the sample code, the XML request service attribute is "getInput", and in the request, the unique "URI" is passed, and only those variable names whose latest information is required is taken into account. Values for matched variable names are sent as a response back to the client or else the error response sent is "Input variables does not exist".

Response:

```
<fuzzyController type="response" service="getInput">
    <URI>Covid_FLS</URI>
    <variable name="pulse">
        <value>50</value>
    </variable>
    <variable name="spo2">
        <value>80</value>
    </variable>
    <message>Input retrieved successfully</message>
</fuzzyController>
```

4) getOutput: In getOutput service, the result/output of data provided in the createFLS and setInput operation is determined. The system evaluates all input values for the unique URI, and based on the rule base; it decides the final outcome. The service attribute used is “getOutput”, and in a request, the “variable name” and “URI” are passed. The output acquired is COVID-19 critical, alert, low or normal. In case the variable values are unsent using the “setInput” service, the output is “Input fields not set”. Also, if the variable does not exist at the fuzzy backend system, then the output is “Output variable does not exist”.

Request:

```
<fuzzyController type="request" service="getOutput">
    <URI>Covid_FLS</URI>
    <variable name="covid" />
</fuzzyController>
```

Response:

```
<fuzzyController type="response" service="getOutput">
    <URI>Covid_FLS</URI>
    <variable name="covid">
        <value>critical</value>
    </variable>
    <message>Output calculated successfully</message>
</fuzzyController>
```

5) queryFLS: Various clients may need access to information about the status of a FLS that is stored on the server. The stored FLS will be retrieved from backend server based on the URI passed in the request. The requested URI is searched in the system, and if found, the XML is converted to a Java object and sent as a response, which includes the complete fuzzy system object (i.e. variable names, rule base and knowledge base).

Request:

```
<fuzzyController type="request" service="getFLS">
    <URI>Covid_FLS</URI>
</fuzzyController>
```

The service attribute used is “queryFLS”, and in a request, only the “URI” is passed. If the client requested FLS with a unique URI is not present, then the response will be displayed as “System does not exist” and if the URI is found in the database then the server response will be “System URI ID retrieved successfully”. The server response includes the FLS definitions as mentioned earlier in IEEE-1855 schema as createFLS and latest value which is set by using setInput.

Response:

```
<fuzzyController type="response" service="getFLS">
    <URI>FS1</URI>
    <fuzzySystem>...</fuzzySystem>
    <message>System retrieved successfully</message>
</fuzzyController>
```

6) deleteFLS: Finally, clients should be able to request that a FLS be deleted from the database’s list of specified FLSs if the FLS description or past input/output history are no longer required. This service is used to delete the FLS that was generated using the “createFLS” service. In deleteFLS service, the file at the backend with the given URI is searched. If the file is found, then it is deleted, and a successful response is sent. However, if the file is not found, then the error response “System does not exist” is sent back to the client. The service attribute used is “deleteFLS”, and client need to pass ”URI” with type of request to get response from the server.

Request:

```
<fuzzyController type="request" service="deleteFLS">
    <URI>Covid_FLS</URI>
</fuzzyController>
```

Response:

```
<fuzzyController type="response" service="deleteFLS">
    <URI>Covid_FLS</URI>
    <message>System deleted successfully</message>
</fuzzyController>
```

V. Experiments

As shown in Fig. 2, three devices were used on client-side hardware architecture for developing a fuzzy logic system, namely smartwatch, mobile and oximeter.

A. Fuzzy-as-a-Service for Real-Time Human Activity Recognition Using IEEE 1855-2016 Standard [12].

A new approach to a web-based FLS service-oriented architecture was presented by this study. It was shown that the proposed service-oriented architecture was able to perform instantaneous data processing via a dynamic fuzzy rule-based framework by experiments with human behaviour tracking data sets. The accuracy of the developed system and the response time were found to be relatively high. Although the architecture was presented in the context of AmI environment, it can be expanded further, i.e. wherever a fuzzy logic system’s storage logic needs to be conceptualised from its logic of presentation. The key purpose was to facilitate the flexible distribution of theoretically complex computation necessary for fuzzy logic systems from clients to dedicated servers. The use of virtualized cloud resources unambiguously gave the device elasticity and made the fuzzy-as-a-service widely accessible efficiently. Network sharing, hardware/software autonomy, reuse of existing files, load balancing between fuzzy logic machine computers, and cost-efficiency were the other advantages.

B. Fuzzy logic web services for real-time fall detection using wearable accelerometer and gyroscope sensors [11].

A new fuzzy logic algorithm was applied to detect a fall based on wearable sensors and controlled noise sensor data to monitor human behaviours for their everyday tasks. This approach was one of a kind where it implemented a web-based fuzzy logic framework. Three major parameters were derived in this system, i.e. SVM_A, D_A, and SVM_G, using wearable sensors to classify hand movements during a crash event. Using a real-time algorithm with fuzzy logic to treat sensor information with noise was the specific aspect of the analysis. Fuzzy-as-a-service and real-time tracking operations that were not traditionally carried out on wearable devices were also used. Experiments showed that the proposed algorithm could effectively discern between instances of decline and non-fall.

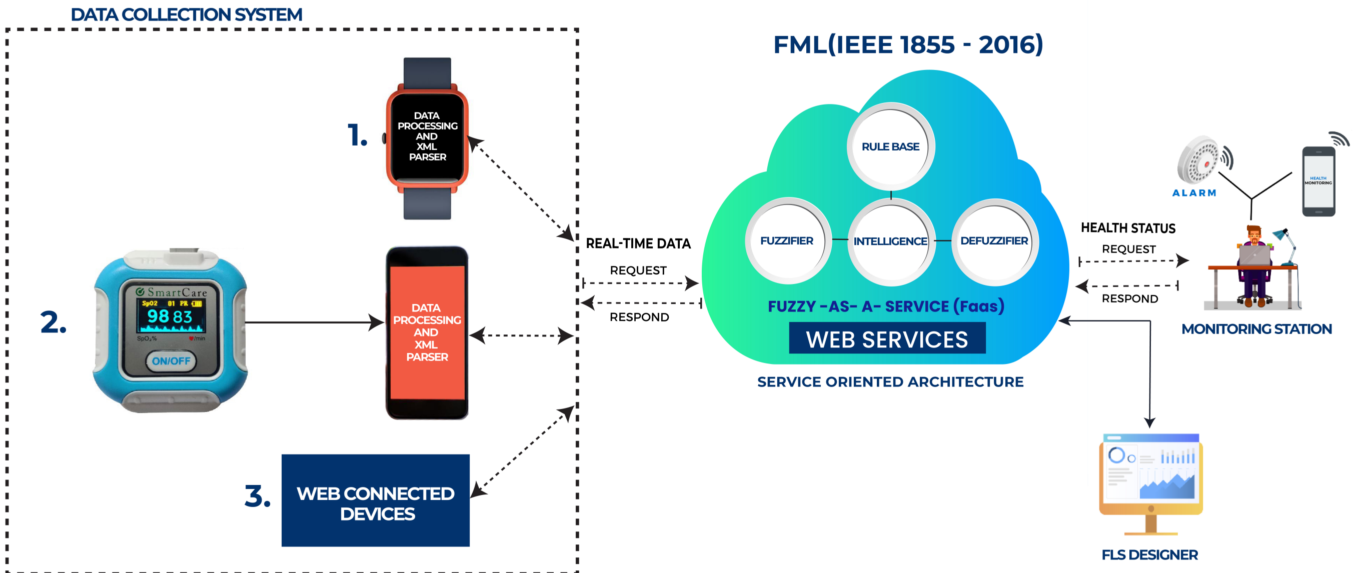

Fig. 2. The components of the conducted human activity monitoring experiment.

C. A cloud-based pervasive application for monitoring oxygen saturation and heart rate using fuzzy-as-a-service.

A novel fuzzy logic algorithm was used to determine a human's medical condition with a real-time wearable sensor and cloud-based web services. The sensor tracked noisy information to detect the day-to-day behaviours or tasks of a person. SpO_2 and pulse rate were the two major factors that were considered from wearable sensors. For the control of sensor knowledge with noise, the analysis used a real-time algorithm and fuzzy logic. The research also involved fuzzy-as-a-service and real-time tracking of operations not traditionally undertaken on wearable sensors. Better sensitivity and above average precision values could be obtained by the analysis.

VI. Results and Discussion

Managing data in an intense FLS on top of developed web services with more sampling rate is essential in real-world situations. Experiment 1 showed a sample state execution of the system where it served parallelly both client requests. The developed fuzzy logic web service system attained an accuracy of 97.23% and 97.42% for training and testing, respectively. This showed an outstandingly high accuracy processed in real time. Experiment 2 comprised wearable sensors (i.e. accelerometers and gyroscopes) that promoted human monitoring of activities using a rule-dependent FLS. Re- search findings exhibited that the proposed method could effortlessly differentiate between fall and non-fall occurrences with an accuracy, sensitivity and specificity of 90%, 88.89% and 91.67%, respectively. Experiment 3 included the use of the wearable sensor BM2000A wrist pulse oximeter to gather real-time data. The data was passed to the server through an Android application that evaluated the results based on fuzzy rule set. The health status was considered to be critical when SpO_2 value was 88% and heart rate was 55 bpm, which is very low as per fuzzy calculation. The health status was treated normal when SpO_2 value was 98% and heart rate was 90 bpm as per fuzzy calculation. An alert was displayed for a health status when SpO_2 value was 91% and heart rate was 108 bpm, which was obtained as per fuzzy calculation. Even though the classification performance of FLS is mentioned here, the idea of conducting this experiment is to expedite whether the designed FLSs are relatively accurate than other classifiers. The performed experiment is just a proof-of concept that aims to achieve an instantaneous FLS execution via web services. Note that, to the best of out knowledge, the performed experiment does not have any similar implementation. A comparative study between the performance of the developed system against that of a non-web system would be the future scope.

VII. Conclusion and Future Work

This research presents a novel approach towards a web-based service-oriented FLS architecture as an example of the implementation of the recently agreed IEEE 1855-2016 standard. Based on the above experiments, it can be concluded that the proposed SOA can perform real-time data processing by means of cloud computing with human activity monitoring datasets using a complex fuzzy rule-based framework through FML. The efficiency and response time of the developed system were considered to be comparatively high. Even though the architecture is specified in terms of AmI environments, it can be expanded to a wider area. This study aimed to facilitate the flexible delivery of the relatively complex computing required for FLS from clients to dedicated servers. The use of virtualized cloud services provided distinctive elasticity to the device. Fundamentally allowing universally accessible FaaS, other advantages of such architecture included network sharing, hardware/software control, data reuse, load balancing amongst FLS devices and cost-efficiency.

The unique feature of the study was the use of an IEEE 1855-2016 algorithm in real-time, in addition to fuzzy logic, for data sensor management. The examination also required the usage of FaaS and real-time activity tracking, which previously were impossible with wearable sensors. The practical perspective of the proposed system has many aspects of expansion. The FLS Group is motivated to take part in the design process as well as to provide feedback on feature prioritisation and collaborative growth activities in deployable applications. This reinforces the proposed API schema and/or invocation formats as well as offering more advanced input for other architecture implementations. The study only looked at one type of FLS (i.e. rule-based systems). Other fuzzy services, such as fuzzy querying of fuzzy databases or fuzzy ontologies, may be estimated in the future. Furthermore, caused by the close relationship between FML and fuzzy ontologies, expanding web services to semantic web services (e.g. developing cloud-based searchable FLS repositories) will make a significant choice. In the future, the proposed architecture will be also used as a main framework to design an artificial intelligence system for the analysis of crime scenes and the automatic reconstruction of crime dynamics, so as to highlight the role of fuzzy logic and IEEE-1855 in a critical real world scenario, such as that of forensic sciences.

Acknowledgement

This work was partly funded by the Nottingham Trent University's Internal Postgraduate Scholarship (B11) and partly by the FRONTIER project (CUP: E69C20000380005) funded by Finanziamento della Ricerca di Ateneo (FRA) 2020 of University of Naples Federico II.

References

[1] G. Acampora, B. Di Stefano, and A. Vitiello, "Ieee 1855™: The first ieee standard sponsored by ieee computational intelligence society [society briefs]," *IEEE Computational Intelligence Magazine*, vol. 11, no. 4, pp. 4–6, 2016.

[2] M. Mittal, V. E. Balas, L. M. Goyal, and R. Kumar, *Big data processing using spark in cloud*. Springer, 2019.

[3] M. I. Tariq, S. Tayyaba, M. W. Ashraf, M. Imran, E. Pricop, O. Cangea, N. Paraschiv, and N. Ali Mian, "An analysis of the application of fuzzy logic in cloud computing," *Journal of Intelligent & Fuzzy Systems*, no. Preprint, pp. 1–15, 2020.

[4] G. Acampora and V. Loia, "Using fuzzy technology in ambient intelligence environments," in *The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05*. IEEE, 2005, pp. 465–470.

[5] M. Arif, G. Wang, V. E. Balas, and S. Chen, "Band segmentation and detection of dna by using fast fuzzy c-mean and neuro adaptive fuzzy inference system," in *International Conference on Smart City and Informatization*. Springer, 2019, pp. 49–59.

[6] V. E. Balas, J. L. Hong, J. Gu, and T.-C. Lin, "Special issue on fuzzy theoretical model analysis for signal processing," *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 4, pp. 4407–4411, 2019.

[7] G. Acampora, V. Loia, and A. Vitiello, "Distributing fuzzy reasoning through fuzzy markup language: An application to ambient intelligence," in *On the Power of Fuzzy Markup Language*. Springer, 2013, pp. 33–50.

[8] A. Pourabdollah, C. Wagner, G. Acampora, and A. Lotfi, "Fuzzy logic as-a-service for ambient intelligence environments," in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2018, pp. 1–7.

[9] M. Parra-Royon and J. M. Benítez, "Fuzzy systems-as-a-service in cloud computing," *International Journal of Computational Intelligence Systems*, vol. 12, no. 2, pp. 1162–1172, 2019.

[10] B. Pandya, A. Pourabdollah, and A. Lotfi, "Comparative analysis of real-time fall detection using fuzzy logic web services and machine learning," *Technologies*, vol. 8, no. 4, p. 74, 2020.

[11] Pandya, Bhavesh and Pourabdollah, Amir and Lotfi, Ahmad, "Fuzzy logic web services for real-time fall detection using wearable accelerometer and gyroscope sensors," in *Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, 2020, pp. 1–7.

[12] Pandya, Bhavesh and Pourabdollah, Amir and Lotfi, Ahmad, "Fuzzy-as-a-service for real-time human activity recognition using ieee 1855-2016 standard," in *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2020, pp. 1–8.

[13] J. R. Jang, *MATLAB: Fuzzy logic toolbox user's guide: Version 1*. Math Works, 1997.

[14] C. Wagner, S. Miller, and J. M. Garibaldi, "A fuzzy toolbox for the r programming language," in *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*. IEEE, 2011, pp. 1185–1192.

[15] G. Acampora and A. Vitiello, "Interoperable neuro-fuzzy services for emotion-aware ambient intelligence," *Neurocomputing*, vol. 122, pp. 3–12, 2013, advances in cognitive and ubiquitous computing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231213005766

[16] G. Acampora and V. Loia, "Fuzzy control interoperability and scalability for adaptive domotic framework," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 2, pp. 97–111, 2005.

[17] G. Acampora, M. Gaeta, V. Loia, and A. V. Vasilakos, "Interoperable and adaptive fuzzy services for ambient intelligence applications," *ACM Trans. Auton. Adapt. Syst.*, vol. 5, no. 2, May 2010. [Online]. Available: https://doi.org/10.1145/1740600.1740604

[18] F. J. Arcos, J. M. Soto-Hidalgo, A. Vitiello, G. Acampora, and J. Alcalá-Fdez, "Interoperability for embedded systems in jfml software: An arduino-based implementation," in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2018, pp. 1–8.

[19] J. M. Soto-Hidalgo, J. M. Alonso, G. Acampora, and J. Alcalá-Fdez, "Jfml: a java library to design fuzzy logic systems according to the ieee std 1855-2016," *IEEE Access*, vol. 6, pp. 54 952–54 964, 2018.

[20] G. Acampora and A. Vitiello, "Extending ieee std 1855 for designing arduino™-based fuzzy systems," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2017, pp. 1–6.

[21] G. Acampora, K. Appiah, A. Hunter, and A. Vitiello, "Interoperable services based on activity monitoring in ambient assisted living environments," in *2014 IEEE Symposium on Intelligent Agents (IA)*. IEEE, 2014, pp. 81–88.

[22] G. Acampora, J. Alcala-Fdez, R. Siciliano, J. M. Soto-Hidalgo, and A. Vitiello, "Visualjfml: A visual environment for designing fuzzy systems according to ieee std 1855-2016," in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2019, pp. 1–6.

[23] J. M. Soto-Hidalgo, A. Vitiello, J. M. Alonso, G. Acampora, and J. Alcalá-Fdez, "Design of fuzzy controllers for embedded systems with jfml," *International Journal of Computational Intelligence Systems*, vol. 12, no. 1, pp. 204–214, 2019.

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

[25] T. E. Foko, N. Dlodlo, and L. Montsi, "An integrated smart system for ambient-assisted living," in *Internet of Things, Smart Spaces, and Next Generation Networking*. Springer, 2013, pp. 128–138.

[26] C.-S. Lee, M.-H. Wang, L.-W. Ko, B.-Y. Tsai, Y.-L. Tsai, S.-C. Yang, L.-A. Lin, Y.-H. Lee, H. Ohashi, N. Kubota *et al.*, "Pfml-based semantic bci agent for game of go learning and prediction," *arXiv preprint arXiv:1901.02999*, 2019.

[27] C.-S. Lee, M.-H. Wang, C.-S. Wang, O. Teytaud, J. Liu, S.-W. Lin, and P.-H. Hung, "Pso-based fuzzy markup language for student learning performance evaluation and educational application," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 2618–2633, 2018.