International Journal of Computational Intelligence Systems
Vol. **12(2)**, 2019, *pp.* 1162–1172
DOI: https://doi.org/10.2991/ijcis.d.190912.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Fuzzy Systems-as-a-Service in Cloud Computing

Manuel Parra-Royon*, José M. Benítez

Department of Computer Science and Artificial Intelligence, DiCITS, DaSCI, IMUDS, University of Granada, ETS de Ingenerías Informática y de Telecomunicación, Granada, 18014, Spain

ARTICLE INFO

Article History

Received 24 May 2019
Accepted 17 June 2019

Keywords

Fuzzy system software
Cloud computing
Cloud platforms
Services deployment
Fuzzy model building

ABSTRACT

Fuzzy systems have become widely accepted and applied in a host of domains such as control, electronics or mechanics. The software for construction of these systems has traditionally been exploited from tools, platforms and languages run on-premise computing infrastructure. On the other hand, rise and ubiquity of the cloud computing model has brought a revolutionary way for computing services deployment. The boost of cloud services is leading towards increasingly specific service offering just as data mining and machine learning service. Unfortunately, so far, no definition for fuzzy system as service is available. This paper identifies this opportunity and focus on developing a proposal for fuzzy system-as-a-service definition. To achieve this, the proposal pursues three objectives: the complete description of cloud services for fuzzy systems using semantic technology, the composition of services and the exploitation of the model in cloud platforms for integration with other services. As an illustrative case, a real-world problem is addressed with the proposed specification.

1. INTRODUCTION

For over 30 years, fuzzy system software have been developed as the most direct and practical application of the theory of fuzzy sets [1]. These software-related systems have made possible scientists and engineers to deal with the representation of knowledge and reasoning subject to imprecision and uncertainty for the solution of very different kinds of problems. Fuzzy systems are extremely effective in modeling complex systems and have the capacity to incorporate human expert knowledge even affected by uncertainty. In this respect, fuzzy systems have been implemented and applied in different domains of application such as industry [2], electronics [3], control systems [4], computing or mechanics [5,6], among many others.

Along the years, several software tools have been developed with the aim of facilitating the modeling and rapid adoption of fuzzy systems within a broad set of domains and problems. Such applications allow researchers and scientists with basic knowledge of fuzzy logic to model a complete fuzzy systems with minimal effort and in a fully flexible fashion. Software for general purpose fuzzy systems, for specific applications and languages, is available as tools (XFuzzy [7],FuzzyStudio [8],...), platforms (KNIME [9], WEKA [10],...) and libraries (FRBS [11], pyFuzzy [12],...).

Over the last 10 years and with an increasing intensity, cloud computing is pushing the traditional on-premise model toward an on-demand service delivery model over the Internet [13]. In fact, cloud computing is rapidly becoming a generalized alternative to costly on-premise infrastructures with key aspects such as scalability, on-demand availability or pay-per-use. Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) are the three main abstract layers of services in cloud computing, where the computing resources from providers are deployed as services (storage, computation and communications) ready to be consumed by users as depicted in Figure 1.

The traditional solutions for the construction of fuzzy systems lack support for cloud computing. Thus, aspects such as flexibility,

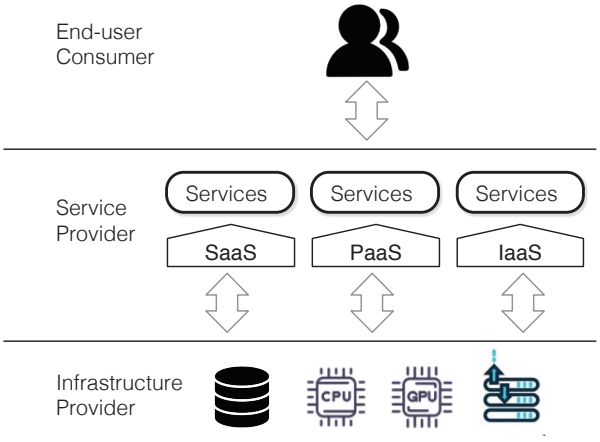

Figure 1 | Overall architecture and components for the cloud computing model.

*Corresponding author. E-mail: manuparra@gmail.com

portability, scaling-up of resources on demand or the discovery of services are not considered.

This way, a natural extension of the software for fuzzy systems would be the implementation, construction and deployment of fuzzy logic models as a service in cloud computing platforms, namely Fuzzy Systems-as-a-Service (FSaaS), with the benefits [14] of scalability, development speed, portability or interoperability, as well as several other features.

The aim of this work is to make an effective proposal for the construction of fuzzy models on cloud computing platforms as services, considering key aspects such as design, description (construction), deployment, composition and, finally, exploitation. To this end, semantic technology has been used to enable the creation and deployment of fuzzy systems as cloud computing services.

This work also considers aspects such as the composition and integration of fuzzy logic services with other services in cloud computing. The proposal is general enough to accommodate new models of service deployment such as serverless services or Function-as-a-Service (FaaS). This proposal seeks to address the problem of integrating tools and platforms for modeling fuzzy systems within native cloud computing environments, so that it endows all the characteristics inherent to the cloud model such as scalability, flexibility and integration, being a step forward in terms of transforming traditional tools to cloud computing services.

The paper is structured as follows: Section 2 compiles the state-of-the art of software fuzzy systems from different perspectives. Then, in Section 3 a definition for fuzzy systems-as-a-service based on semantic technology is detailed. Afterwards, one example of use case is defined in Section 4. In Section 5 all the advantages of deploying this software in cloud computing compared with traditional software tools are depicted. Conclusions and final remarks are described in the last section.

2. RELATED WORK

The great success of the use of fuzzy systems in a wide variety of fields such as decision making, control systems, image recognition, has motivated the advent of many software development tools for the construction of this type of systems. The develop of fuzzy systems has been done using languages, platforms, libraries or development environments, both in commercial or open source versions. The implementation for the subsequent use of the systems is always specific hardware devices or on-premise computers. Cloud Computing, however, allows for a new way of service deployment for computing tasks that will also result very beneficial for fuzzy systems. In the following paragraphs a study of the software systems for fuzzy systems is carried out.

The development of fuzzy models has been approached from different points of view. In [15] a taxonomy of fuzzy system software is made distinguishing between general purpose, and specific purpose and detailing the languages proposed for fuzzy systems modeling.

An important problem when building fuzzy systems is that there is no standard for modeling this type of systems. Aspects such as terminology, methodology or compatibility would be resolved if a homogenization in the modeling was available. To try to mitigate this, different language proposals have been developed over the last years. Most of the language proposals are focused on providing environments to model fuzzy systems for both general purpose domains and specific fields. For example, Fuzzy Control Language (FCL [16]) offers a common basis for the development of fuzzy logic, through which to integrate fuzzy logic controllers. Languages such as Fuzzy Markup Language (FML [17]) offers an eXtensible Markup Language (XML)-based language for defining fuzzy systems or XFL [18] specification, a language specification for fuzzy modeling of general purpose systems used on the Xfuzzy [7] CAD platform. On the other hand, XFSML [19], based on XML and serves as a starting point for the definition of a standard modeling language, along with other more robust options such as IEEE P1855/D2.0, represent a clear commitment to standardization in fuzzy control modeling.

Another frequent approach is to use general purpose programming languages, complemented with specific libraries contributing fuzzy system functionality. Quite diverse libraries and packages have been developed for languages such as Matlab [20], C/C++ [21], Java [22], or R [11], among others. By using these libraries it is possible to build models of fuzzy systems within the general program that would otherwise be constructed. This choice easily enable the integration with functionalities supported by additional libraries, for example, visualization, data analytics, monitoring, etc.

On the other hand, software modeling for fuzzy systems has been widely integrated into visual applications for experimental modeling and data processing, such as KNIME [9] or WEKA [10]. These tools besides providing an engine for the work with Data Mining, also integrate modules for the modeling of fuzzy systems in a totally visual way, allowing the user to build the system through the interconnection of blocks, stages or operations.

Most of this traditional software for the modeling of fuzzy systems is designed to work on-premise. There is a minuscule portion of software for fuzzy systems deployed in web. With proposals like CAVUS [23], or FuzzyStudio [8] is possible to model a complete fuzzy systems using a web browser. However, these proposals are not cloud-native, so they cannot take advantage of the features that cloud-based solutions would offer. For this type of fuzzy systems to be deployed in the cloud, it is necessary to have the right tools to be able to define and describe services for cloud computing in an efficient and effective way, in order to capture its full potential.

Different proposals for the definition of services have been developed in both the syntactic and semantic fields. At syntactic level, part of the proposals are based on Services-Oriented-Architecture (SoA) and XML, in addition to other derived languages such as WSDL [24], WADL [25] or SoAML [26] dealing at syntactic level the technical aspects of services in general.

Proposals with semantic technology have an increasing added value, since they offer a much greater flexibility than syntactic languages, allowing to capture functional and non-functional features of the service. OWL-S [27] or WSMO [28] are some of the basic proposals for the development of semantic definitions in general. Others, like USDL [29], include many more aspects related to the service in cloud computing, such as entities, prices, composition or legal aspects. On the other hand Linked-USDL [30] offers a vocabulary and a complete scheme for the definition of generic services in cloud computing and includes modeling elements such as catalog, interaction or Service Level Agreement (SLA) among many others.

Following this line, our proposal tries to fill the existing gap in the description of services for fuzzy systems in cloud computing, through which it is possible to design, deploy and exploit fuzzy system software on cloud platforms, with all the advantages that this entails.

3. FUZZY SYSTEM MODELING IN CLOUD COMPUTING

3.1. Fuzzy Systems

Introduced by L. Zadeh in 1965, the theory of fuzzy sets emerged as an extension of the classical set theory, for the modeling of sets where their elements may belong to them with a degree between 0 and 1, and non-necessarily equal to the extreme values. This degree of membership indicates how close it is to being a member (1) or not being a member (0), while the intermediate values indicate a partial membership. The degree of membership is indicated by a function called membership function.

In addition many fuzzy concepts derive from the human language, which is eminently vague. The logic derived from fuzzy sets, fuzzy logic, becomes a very powerful tool to represent linguistic concepts, variables, and rules, necessary to represent human expert knowledge. A key concept is the linguistic variable that takes as values linguistic terms from a set whose is expressed as fuzzy sets.

The traditional concept of rule-based system is naturally extended to fuzzy rule-based system (FRBS), when some or all of its components are of a fuzzy nature. The most frequent case is when the rules are linguistic. In relation to the structure of the rule, there are two widely used models Mamdani and Takagi-Sugeno-Kang (TSK). Figure 2 represents the architecture of the Mamdani model, consisting of four key components: a) fuzzification: transforms the crisp inputs into fuzzy values, b) knowledge base: stores a database and a rules database, c) inference engine: performs reasoning operations on fuzzy rules and input data, d) defuzzification: produces crisp values from the linguistic values in order to produce an output result. In Mamdani model, rules both the antecedent and the consequent are composed of linguistic variables following this model:

$$\text{IF } X_1 \text{ is } A_1 \text{ and } \ldots X_n \text{ is } A_n \text{ THEN } Y \text{ is } B \qquad (1)$$

TSK rules differ from Mamdani's ones in that their consequent is a function of input variables and thus no defuzzifier is needed. In this way the construction of an FRBS requires the definition of each of the elements that composes it: variables, rules, inference engine, fuzzifier and defuzzifier modules. The most sensitive component of an FRBS is the knowledge base. There are two general approaches to produce it: a) derived from expert knowledge through some knowledge extraction processes, b) computed through learning methods. Furthermore, a number of hybrid proposals integrating the best properties of fuzzy systems with other computational intelligence techniques have been developed along the years. Clear examples of this hybridization are neuro-fuzzy systems an genetic-fuzzy systems. Both kinds of FRBS (i.e., Mamdani and TSK) as well as most commonly used hybrid systems should be considered when addressing the definition of a platform for FRBS development and exploitation.

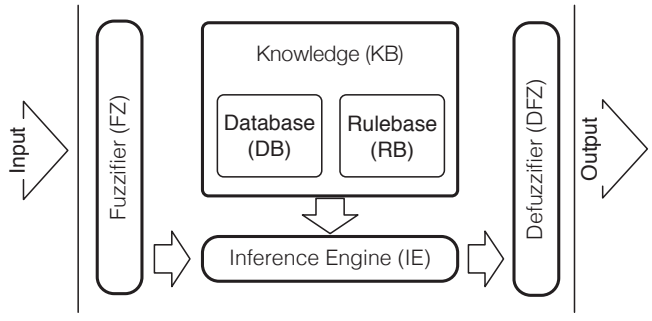

Figure 2 Components of the Mamdani model.

3.2. Semantic Technology

The design of services in cloud computing must focus on two aspects, the description of the service and the definition of the components. Most of the problem of service design is that there is no globally accepted standardization for this purpose. Moreover, since it is a service in the cloud, it must also consider the aspects related to the management of a service in this kind of platforms, as well as all the requirements for modeling fuzzy systems. The managing of services in cloud computing is both a need and an advantage with respect to on-premise alternative tools, as indicated in Section 2.

To deal with the definition of services in cloud computing, many proposals have been made since the early days of cloud computing. The proposals for the definition of services based on semantic technology are the most accepted for the modeling of services in cloud computing. Semantic technology provides an abstraction layer that connects data, content and processes. A key element are ontologies, which allow to define the formal specifications of terms, the relationships between them and the properties within a domain. Thanks to the use of semantic languages such as OWL [27], RDF [31] or Turtle [32], it is possible to capture the form, the format, the serialization or the schema of any existing data or system. With semantic languages it is possible to design new vocabularies by means of which concepts, terms and relations of an area of knowledge are represented by using semantic triples, which comprise a subject, a predicate and an object as depicted in Figure 3.

For the definition of fuzzy systems, we have developed `fsschema`, a vocabulary based on semantic technology. It includes all the elements and procedures required for a complete definition of fuzzy systems. In a complementary way, a vocabulary named `ccsschema` has been designed allowing the management aspects of a service in a cloud computing platform, such as authentication, interaction, SLA or pricing of services, among others.

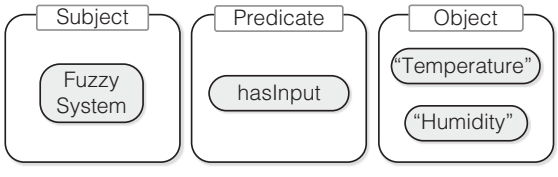

Figure 3 Triples structure of semantic knowledge.

In the following sub-sections we describe the modeling of FRBS as a service in cloud computing using semantic technology.

3.3. Definition and Description of a Fuzzy Systems Service in Cloud Computing

3.3.1. Semantics of fuzzy systems modeling

The main part of the design of a service for fuzzy logic focuses on the construction of the fuzzy systems and its components. After careful examination of widely accepted monographs and consulting some of the proposals referred to in Section 2, a semantic fuzzy system has been defined. The components and structure of the proposal are depicted in Figure 4. The more relevant components are detailed in the following:

- Inputs and outputs. They corresponds to the input and output variables of the fuzzy systems. Both types of variables need to be defined stating their name and specific characteristics thought their Type. For the semantic definition of both types of variables it is necessary to use `fsschema:hasInput` and `fsschema:hasOuput` both point respectively to `Input` and `Output` as definition classes for the types of variables. Both require an element `fsschema:type` and `fsschema:Membership` (not shown in the Figure 4).
- Definition of linguistic variable types. The type of a variable of the fuzzy systems is defined by the name of the type and universe of discourse, in conjunction with membership functions. The universe of discourse is identified with the values maximum, minimum and cardinality. Membership functions are contained within a more general class that agglutinates all the functions that the service is able to deploy for the construction of the fuzzy systems and the are used to describe language labels. To define each of the fuzzy model variable types it is necessary to instantiate `fsschema:hasType` and `fsschema:Type`. To fully specify the "Type" we use the properties `min`, `max`, `cardinality` and `name`, and also include all the required membership functions with `fsschema:hasMembershipF`. In the Figure 4 the set of functions that includes the `fsschema` within the class `fsschema:Function` is shown. For example, to instantiate a triangular membership function, use `fsschema:MembershipF` and the following properties: to define the name of the linguistic label `label` and `fsschema:hasProperties` of the `Function` class for `fsschema:Membership` of the `triangular` class with properties `a`, `b`, `c`, corresponding to the values of the triangular function, as illustrated in Figure 5. In this way with `fsschema` it is possible to collect all the usual concepts that are handled in the area of the fuzzy systems. Six types of functions are defined with `fsschema` (see Table 1): binary functions that can be used as T-norms, S-norms and implication functions; unary functions that are related with linguistic hedges; crisp functions that implement crisp blocks; membership functions that are used to define the semantic of linguistic labels; families of membership functions and defuzzification methods.
- Crisp Blocks. Crisp functions are used to describe mathematical operations between variables. These functions can be assigned to other crisp modules and included in the fuzzy systems composition. Table 1 shows some of the crisp functions included by `fsschema`.
- Sets of operators. A set of operators contains the mathematical functions that are assigned to each fuzzy operator. `fsschema:hasOperatorSet` and `fsschema:OperatorsSet` are necessary to describe the operators that will be used in the whole fuzzy systems.
- Rule and rule base. Rules are built with an antecedent and a consequent. The antecedent describes the relations between the input variables of the fuzzy systems and the consequent describes the assignment of a linguistic variable to an output variable, for instance in the Mamdani model. In TSK model,

Table 1 | Function groups and functions included in fsschema (not all of them are shown).

Unary	not,sqrt,square, cubic, sugeno, pow
Binary	max,min,prod,sum,zadeh,godel,...
Crisp	add,diff,prod,div,sample
Membership	triangle,trapezoid,slope,...
Defuzzification	mean,Gamma,TakagiSugeno,..

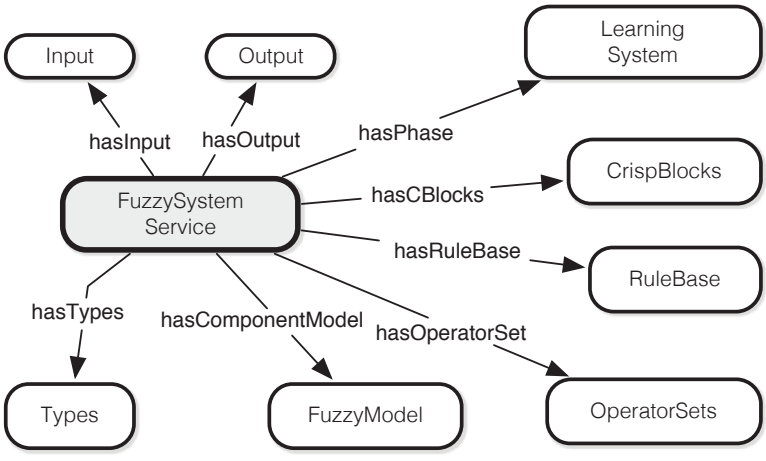

Figure 4 | Overall structure of the fuzzy systems modeling scheme with fsschema.

the consequent is represented by a function of input variables. For each rule, a specific weight or confidence value can be established. The instantiation is done with `fsschema:hasRuleBase` of the class `fsschema:RuleBase` and data as name of the rules base, input, output variables, the construction of the rules with `fsschema:Rule` and properties `fsschema:hasAntecent fsschema:hasConclusion` (see Figure 6).

- Fuzzifier and defuzzifier. The fuzzifier module is in charge of converting a crisp input to a linguistic variable using the membership functions. The fuzzifier can also be defined as a mapping module from an observed input to a fuzzy set of labels in a universe of inputs specific to the discourse universe. On the other hand, defuzzifier convert the fuzzy output of the inference engine to crisp using membership functions similar to used by the fuzzifier. A natural and simple fuzzification example, is to convert a crisp input value into a fuzzy singleton, within the specified universe of discourse. For this purpose, the type `fsschema:Type` of the entry is specified as a membership function `fsschema:MembershipF`, the set of linguistic labels label and its membership `fsschema:Membership`, in this case `singleton`, is assigned with the definition value a.
- Learning system. Learning methods involve two parts: a) the identification of the structure, where a rule base is generated and b) the optimization of the estimation of the parameters of the membership functions. The structure that has been designed to describe the modeling of the fuzzy learning system part can be seen in Figure 7. For the instantiation of a learning phase `fsschema:hasPhase LearningM` is used. After that, for the description of `LearningM` it is necessary to indicate some entities, like the learning algorithm that will be used, `fsschema:hasAlg`. It is also necessary to define the training data set with `fsschema:hasTData` and the type of input and output (`fsschema:hasInput fsschema:hasOutput`) that will be used by the learning methods to build a rule-based structure. Learning system supports stand-alone procedures as well as various hybrid models such as Neuro-fuzzy systems.

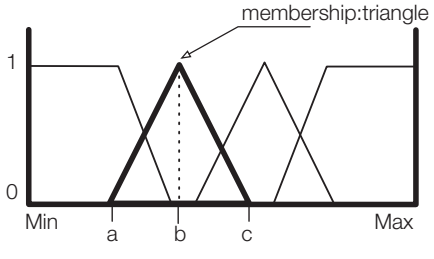

Figure 5 | Fuzzy modeling for a membership function with fsschema.

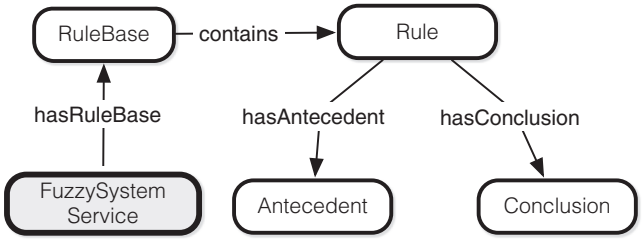

Figure 6 | Rule base definition schema.

3.3.2. Semantic of cloud computing management services for fuzzy systems

The definition of a service in cloud is not complete without taking into consideration the management elements that the cloud provider needs for the exchange of information with the cloud consumer user of services or other external entities. These elements are, for example, service pricing, authentication, SLA or interaction interfaces, among others.

Semantic definition of cloud computing services have the advantages of negotiation, composition and invocation, with a high degree of automation. This automation is fundamental because it allows services to be explored and discovered for exploitation by other entities. In this line, for example, other entities, such as cloud computing users or cloud brokers [33], could check the costs associated with the use of the fuzzy systems in cloud or to know which methods are necessary to authenticate in the service of the fuzzy systems if they are defined. Figure 8 shows the modeling diagram for the full service definition of fuzzy systems, including all the additional elements of service management in cloud computing such as authentication, catalog, entities, interaction, pricing and SLA.

The definition scheme for the management of a service that has been used is `ccsschema`.[1] It is a light version of the semantic scheme developed in `dmcc-schema` [34] and contains all the basic elements for its integration with a generic service in cloud computing. It allows to define in a simple way the management elements of the service:

- **Authentication.** Services require authentication for use and exploitation. For authentication the `waa` [35] definition scheme has been used, which supports the vast majority of new web authentication models such as Oauth or ApiKey among others. For example, once the fuzzy systems has been modeled it will be integrated with other applications and services in cloud computing, and it will be essential to provide the service with an authentication for the exploitation of the fuzzy systems by other entities.

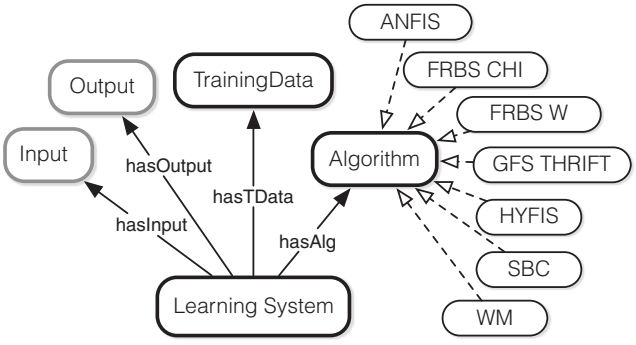

Figure 7 | Decomposition of the fuzzy learning system class.

[1] Cloud Computing schema: https://dicits.ugr.es/linkeddata/dmservices/

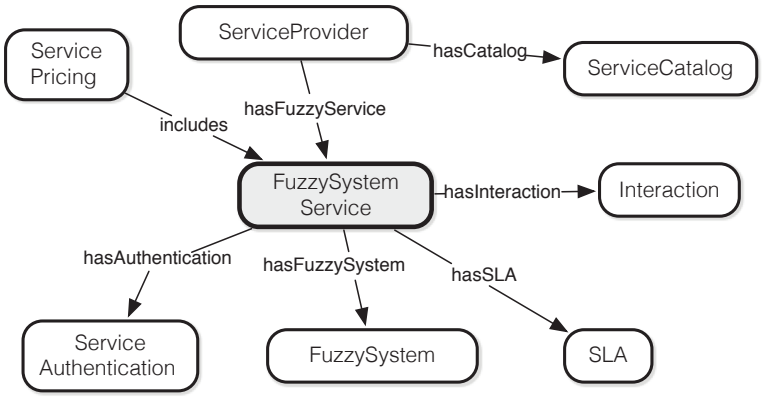

Figure 8 | Overall cloud computing semantic model including fuzzy systems and cloud management entities.

- **Catalogue.** Cloud providers have a catalogue of services through which both users and other entities (cloud brokers, for example) can query the services that the provider makes available. From the catalogue, it is possible to view descriptions, prices, parameters, Application Programming interface (API), SLA, etc., so that it is the key starting point for the discovery of services on cloud platforms.
- **Interaction.** To interact with a service it is necessary to expose an API that allows consumers to communicate with the service in some way. This communication can be done in different ways such as an API RESTful, WSDL, or any other option available for web services.
- **Prices.** The services offered by cloud providers are subject to costs based on use of the service, specifications or time, among others. We use `ccprice`[2] scheme to cover the most common types of pricing; on the one hand, it allows us to expose the costs of use based on the cost of computing instances and resources for the modeling of fuzzy systems. On the other hand, it allows us to price a service according to the use and exploitation, not based on computing time, but on calls to the fuzzy systems built.

This section can be summarized stating that a complete proposal has been made for the definition of fuzzy system-as-a-service: a service in cloud computing for a fuzzy systems that unifies all the key aspects that this type of service should have.

4. SAMPLE CASE OF USE

To confirm the validity of `fsschema` and `ccsschema` as tools for the definition of fuzzy systems services in cloud computing, a real case study will be developed, in which a complete fuzzy systems service is described, including all the aspects of the fuzzy model, as well as all the elements related to the management of the fuzzy service in cloud computing.

For reasons of space we will not detail all the data or attributes in depth and will only consider what is most important for the basic specification of the service and his comprehension. All detailed information on each component, examples and other fuzzy systems are available on the complementary information website[3].

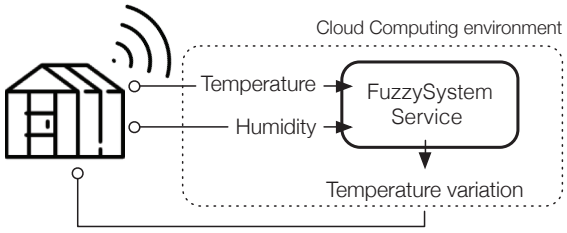

Figure 9 | Example of a greenhouse element and its connection to a fuzzy systems in a cloud computing environment.

For this case of use we want to model a fuzzy systems for the control of temperature and humidity in a network of greenhouses, where cherry tomatoes are cultivated (see Figure 9). In this way, the overall exploitation of the system has to be managed as a cloud service, considering some management aspects such as pricing, authentication in the service or interaction, as well as the complete modeling of the control system.

The deployment of a fuzzy system on the cloud as a solution for this problem is motivated by the advantages of cloud computing. On the one hand, the size and configuration of the set of greenhouses may change easily: through the addition of new greenhouses or removal of others no longer in use or according to the harvest schedules. The data is gathered through a network of sensors, most of which adapts to the Internet of Things device philosophy. Then the adaption of fuzzy logic controllers for each independent greenhouse should be tuned to its particular features. Thus the resource assignment, for communication, storage and computing should be scalable and provided on-demand [36]. For the modeling, semantic schemes `fsschema` and `ccsschema` will be used, to cover both the description of the cloud computing service management and the description of the fuzzy systems

The definition of the elements and components of the fuzzy systems with `fsschema` is composed of several levels of abstraction, allowing to specify from the most general entities to the final details. The

[2] Cloud Computing pricing schema: https://dicits.ugr.es/linkeddata/dmservices/#ccpricing

[3] Semantics of fuzzy system modeling: https://dicits.ugr.es/linkeddata/fuzzyservices/

particular definition of the main of these elements is made, along with their attributes and relationships. Code Listing 1 shows each of the elements that will be included in the modeling, such as the inputs (`fsschema:hasInput`) `_:temperature`, `_:humidity`, the output (`fsschema:hasOutput`) `_:tempvariation`, the definition of the types of variables (`fsschema:hasType`), the operators (`fsschema:hasOperatorSet`) and the rule base (`fsschema:hasRuleBase`) `_:rulesbase`.

Listing 1 : Main components of the fuzzy system description.

```
<http://dicits.ugr.es.com/ld/FSGRH>
 a fsschema:FuzzySystem;
   rdfs:label
    "Greenhouse FuzzySystem"@en;
   fsschema:hasInput
    _:temperature,_:humidity;
   fsschema:hasOutput
    _:tempvariation;
   fsschema:hasTypes
     _:types;
    fsschema:hasOperatorSet
     _:opset;
    fsschema:hasRuleBase
     _:rulebase;
    .
```

The first part to be described are the inputs and output of the fuzzy systems. Table 2 describes each of the input variables (Temperature and Humidity), the labels and the values that define the membership function. Table 3 the output of the fuzzy system is described.

To model, for example, the input corresponding to `_:temperature`, it is necessary to indicate the variable and the parameters of the universe of discourse such as Max `fsschema:max "50"`, Min `fsschema:min "-5"` or cardinality `fsschema:cardinality` (not indicated), together with their corresponding membership functions and labels `_:TVeryLow`, `_:TLow`, `_:TNormal`, `_:THigh`, `_:TVeryHigh` as shown in code Listing 2. Then, code Listing 3 defines in detail each of the types of membership functions for the input `_:temperature`.

Table 2 Labels, membership function and values for the input variables (Temperature and Humidity).

Label and Membership Function	Temperature Values (C)	Humidity Values (%)
Very low (trapezoidal)	-5,-2.10,15	0,0,10,20
Low (triangular)	10,15,20	10,25,50
Normal (triangular)	18,20,22	30,40,50
High (triangular)	20,25,30	40,55,70
Very high (trapezoidal)	25,30,40,50	60,70,100,100

Table 3 Labels, membership function (all Triangular) and values for the output variable.

Label and Membership Function	Control Variation of Temperature
Large descent (LD)	-15, 10.5, -7.5
Normal descent (ND)	-10.5, 5.5, -2.5
Small descent (SD)	-7.5, -2.5, 0
No action - Maintain (M)	-1.5, 0, 1.5
Small rise (SR)	0, 2.5, 7.5
Normal rise (NR)	2.5, 5.5, 10
High rise (HR)	7.5, 10.5, 15

Listing 2 : Definition of the input `temperature`.

```
_:temperature a fsschema:Input;
 fsschema:inputName "Temperature";
 fsschema:min "-5"^^xsd:decimal;
 fsschema:max "50"^^xsd:decimal;
 fsschema:name "Temp Type";
 fsschema:hasMembershipF [
  a fsschema:MembershipF;
    fsschema:hasProperties
     _:TVeryLow,_:TLow,
      _:TNormal,_:THigh,
      _:TVeryHigh; ]
 .
```

To specify the types of labels, it is necessary to indicate the type of the function and the associated parameters. For example, code Listing 3, for a `_:TVeryLow` label you define the trapezoid type with `fsschema:trapezoid` and the values that define the trapezoid `param_a`, `param_b`, `param_c`, `param_d`.

Listing 3 : Membership function and values of definition for `TVeryLow` label.

```
_:TVeryLow a fsschema:Membership,
   fsschema:trapezoid;
 fscchema:label "Very Low";
 fsschema:param_a "-5";
 fsschema:param_b "-2";
 fsschema:param_c "10";
 fsschema:param_d "15";
 .
```

The process of describing this output "Temperature variation" is done in the same way as for `temperature` and `humidity` inputs, but specifying that it is an output variable `fsschema:Output`.

Once the inputs and outputs of the problem are defined, the rule base is created (see Table 4). The definition of the rule base consists of the specification of the rule set.

Listing 4 : Definition of rules base.

```
 _:rulebase a fsschema:RuleBase;
 rdfs:label "GreenHouse FRBS"@en ;
 fsschema:name "FRBS"@en ;
 fsschema:contains
   _:R1,_:R2,_:R3;
 .
```

The code Listing 4 shows the instantiation of the rule base (`_:rulebase`), with the name and the list of rules that it contains

Table 4 Rules base for temperature and humidity.

T./H.	VL	L	N	H	VH
VL	SR	SR	HR	HR	HR
L	M	M	SR	SR	NR
N	M	M	M	M	SD
H	M	M	SD	SD	ND
VH	SD	ND	ND	LD	LD

for its later definition. The definition of each rule is formed by three elements, the premise (`:hasPremise`), the conclusion (`:hasConclusion`) and the confidence (`:confidence`) or weight of the rule, as can be seen in code Listing 5.

Listing 5 : Definition of the rule R1.

```
_:R1 a fsschema:Rule;
  fsschema:hasPremise _:P1
  fsschema:hasConclusion _:C1;
  fsschema:confidence "1.0";
.
```

In this first part, the fuzzy systems for the control of temperature and humidity has been defined, but the definition of the fuzzy logic modeling service in cloud computing is not complete without the description of the cloud computing management elements associated with the service itself. Regarding the aspects related to the management of the cloud service, only the SLA and a pricing model will be detailed. For example, for the SLA (schema ccsla[4]), there will be a credit bonus credits when the following cases happen: a) if the Monthly Uptime Percentage (MUP) of the service is between 99.9 and 99.0 (see Listing 6), it will be compensated with 10 credits, and b) if the percentage is between 99.0 and 95.0, it will be compensated with 30 credits. In order to reflect this example, code Listing 6 defines how one of the terms of the agreement (`_:SLADefinition_A`) is defined as well as its range `s:maxValue` and `s:minValue`, which has a value corresponding to the compensation.

Listing 6 : SLA term definition for fuzzy systems service.

```
 _:SLADefinition_A a ccsla:Definition;
ccsla:hasDefinitionValue [
a s:structuredValue;
s:value [
 a s:QuantitativeValue;
 s:maxValue 99.99;
 s:minValue 99.00;
 s:unitText "Percentaje";
 ];
];
.
```

For the pricing of the service there are multiple possible options, for this example we have decided that the service has two types of costs, one related to the development of the fuzzy systems and the second based on the exploitation of the system. In this case for the development, costs are a function of the time and the features of instance used for the modeling of the fuzzy systems. On the other hand, for exploitation, where once the model of the fuzzy systems has been made, it is directly usable by another service, by an user or an application in cloud, so that for this mode of price charging a cost per volume of calls to the service of the fuzzy system has been established as follows: a) first 1000 calls without cost (see Listing 7), and b) from 1000 each block of 5000 calls is charged with $0.50. Listing 7, the first pricing model is described (schema `ccp`[5]).

Listing 7 : Free plan for fuzzy systems explotation until 1000 calls.

```
 _:MaxUsageFree a
  gr:PriceSpecification,
  gr:Offering;
gr:max 0.00;
gr:priceCurrency "USD";
gr:includesObject [
 a gr:TypeAndQualityNode;
 gr:amountOfThisGood "<1000";
 gr:hasUnitOfMeasurement "units";
];
.
```

Finally, we join all the pieces that have been defined separately to integrate them in the definition part of the full service. It contains part of the aspects represented in the example for cloud computing management, as well as the elements of the definition of the construction of the fuzzy systems. Listing 8 defines the general aspects of the service, such as the details of the provider (`gr:name`), contact `s:serviceLocation` and `s:contactPoint`), and the services offered `ccsschema:hasFuzzyService _:FuzzySystem;`.

Listing 8 : Fuzzy system in cloud computing.

```
_:FuzzyProvider a
  ccsschema:ServiceProvider;
rdfs:label "FS Provider"@en ;
dc:description
  "DICITS FuzzySystem SP"@en ;
gr:name "DITICS FS Provider";
gr:legalName "U. of Granada";
gr:hasNAICS "541519";
s:url <http://www.dicits.ugr.es>;
s:serviceLocation
 [ a s:PostalAddress;
   s:addressCountry "ES";
   s:addressLocality "Granada";
 ] ;
 s:contactPoint
 [
  a s:ContactPoint;
  s:contactType "Costumer Service";
  s:availableLanguage [
    a s:Language;
    s:name "English";];
  s:email "fuzzy@dicits.ugr.es";
 ];
ccsschema:hasFuzzyService
  _:FuzzySystem;
.
```

Then, code Listing 9 specifies the entire structure of the service in cloud, with all the entities, interaction point (`ccsschema:hasInteraction`), SLA `ccsschema:hasSLA`, authentication `ccsschema:hasAuthenticacion` and pricing `ccsschema:hasPricing`. Afterwards, each of the elements must be described in detail.

Listing 9 : Components of a fuzzy systems service in cloud computing.

```
 _:FuzzySystem a
  ccsschema:FuzzySystemService;
```

[4] https://lov.linkeddata.es/dataset/lov/vocabs/ccsla

[5] Cloud Computing pricing schema: https://lov.linkeddata.es/dataset/lov/vocabs/ccp

```
3 rdfs:label
4    "Fuzzy Service dicits.ugr.es"@en ;
5 dc:description
6    "DICITS Fuzzy Service"@en ;
7 ccsschema:hasInteraction
8    _:FSServiceInteraction;
9 ccsschema:hasSLA
10   _:FSSLA;
11 ccsschema:hasFuzzySystem
12   _:FuzzySystemService;
13 ccsschema:hasAuthentication
14   _:FSServiceAuth;
15 ccsschema:hasPricingPlan
16   _:FSServicePricing;
17  .
```

The use case exposed in this section illustrates how `fsschemacan` be used to describe a complete service of fuzzy system, which can be deployed, discovered, composed and exploited as a cloud computing service.

5. ADVANTAGES OF THE FSaaS PROPOSAL

A breakdown of the key advantages to the adoption of fuzzy systems within cloud is presented hereafter.

- Standarization of fuzzy modeling in cloud computing environments. One of the outstanding features of the use of semantic technology for the definition of services in cloud computing is that it allows to capture in fine detail all the components of a service (both from the point of view of the cloud model, and from the description of the modeling). The idea with cloud services is that the service definition can be uniform between different cloud providers, achieving a high degree of portability in the construction and modeling of fuzzy systems between cloud computing entities. This means that from a single service description, the deployment of a complete system of fuzzy logic in cloud does not depend on the underlying architecture, the language implemented or other variables that do concern on-premise systems.
- Synthesis and exploitation of the service. The on-premise tools have modules for the synthesis and exploitation of the built model to languages such as C/C++, Python, Java or VHDL (for FPGA [37]), which allow it to be integrated into other applications and systems. Nevertheless, when we deal with fuzzy systems in cloud computing, this mode of exploitation does not make sense due to cloud nature. Cloud computing takes advantage of new service exploitation models by offering the capacity for interoperability between services. With this, for example, the output of a fuzzy systems service in Cloud can be used to offer computational intelligence to another service that demands it. This entails that the model built as a service can be fully integrated with other services in cloud computing .
- Composition of services. This advantage is fully exploitable when modeling a fuzzy systems in cloud computing. For example, when building a fuzzy model, certain functions or systems such as the management of the fuzzy rules base can be implemented in other services, so that the modeling itself can make use of them by performing a composition of services at the end. This means great flexibility for the use of resources as services offered by cloud computing providers.
- Serverless model. Serverless computing is a cloud computing execution model in which the cloud provider dynamically manages the allocation of computing resources [38].The serverless model is the ideal companion for deploying fuzzy systems. When the fuzzy systems is designed, it can be integrated into a serverless service, which means that much of the problem of scaling or distribution of computing resources lies in the provider. This service functionality can be integrated into other tools that are implemented in cloud, facilitating the deployment and use of models from any other service.
- BigData fuzzy model scaling. When we deal with problems of modeling fuzzy systems in cloud, they need a computing infrastructure to support all parts of it such as construction, validation and exploitation of the designed system. For each of these stages, the cloud model can provide computing resources [39]. In the same way, thanks to this feature, for example, when there is a rule base of a fuzzy systems model, with a massive number of rules, it is necessary to use the innate capacities of distributed computing in both processing and storage that cloud computing offers in a transparent way.
- Portability and reproducibility of fuzzy models. Another of the great advantages of using cloud computing for the modeling of fuzzy systems is that thanks to the use of semantic technology for the definition of services, the models that are built are completely portable and agnostic with respect to cloud computing providers. In this way it is possible to work with the same definition of modeling from different providers, so the reproducibility [40] of the experimentation with fuzzy systems in cloud computing is assured. It would also improve the competitiveness between providers that accept this type of services in their catalog to take advantage of their infrastructure.
- Digital transformation. The cloud is the key enabler of digital transformation projects and provides the scale and speed necessary for enterprises to focus on transformation. In this context, the development of fuzzy systems in cloud computing would allow progress in terms of the digital transformation of processes and systems that until now were only focused on traditional fuzzy logic modeling tools. Having this model of construction of fuzzy systems services in cloud, it offers the possibility of moving the existing fuzzy systems models to models cloud-based like FSaaS.

6. CONCLUSIONS AND FINAL REMARKS

Cloud computing is displacing the traditional on-premise model of consuming Information and Communication Technology (ICT) services by users, toward a model based on Internet services, where not only are there generic storage services, or computing resources but also more specific services such as Data Mining in cloud computing. Taking advantage of this increasing trend, the idea of covering the existing gap to build a scheme for the definition and modeling of fuzzy systems in cloud computing, has led us to

propose in this article a scheme and complete vocabulary based on semantic technology called `fsschema`.

This scheme allows in a compact and direct way to describe models of fuzzy systems as services in cloud computing. On the one hand it allows to define all the components of an FRBS (inputs, outputs, rule base, membership functions, operators, learning elements, . . .), and, on the other hand, it also contemplates all the aspects of cloud computing management, which a cloud provider has to take into account for the deployment of the service and its subsequent exploitation by consumer users.

In this way, `fss-cc` collects in a single definition all the elements that allow to describe a complete service in cloud computing, which is denied by other proposals for the modeling of these systems.

The scheme is based on the semantic web and the use of ontologies and vocabularies, which at a practical level on the one hand ensures that the modeling of fuzzy systems can be easily extended and completed, and on the other hand confirms the capabilities related to the portability of fuzzy systems services between different cloud providers.

Possibilities such as the virtually infinite scaling of the infrastructure, the integration of fuzzy systems services within the ecosystem of a cloud provider, the capabilities to deliver and exploit the fuzzy service from APIs or as serverless functions independent of resources, as well as the imminent exploitation of IoT in which fuzzy systems for devices will be more than relevant, highlight the importance of the adoption of fuzzy systems modeling in cloud computing.

Finally, the applicability of fsschema to real-world problems have been illustrated through the development of a case study.

ACKNOWLEDGMENTS

Manuel Parra-Royon holds a ”Excelencia” scholarship from the Regional Government of Andalucía, Spain. This work was supported by the Research Projects P12-TIC-2958 and TIN2016-81113-R (Ministry of Economy, Industry and Competitiveness - Government of Spain).

REFERENCES

[1] L.A. Zadeh, Fuzzy sets, Inf. Control 8 (1965), 338–353.
[2] R.-E. Precup, H. Hellendoorn, A survey on industrial applications of fuzzy control, Comput. Ind. 62 (2011), 213–226.
[3] D. Kannan, A.B. Jabbour, C.J. Chiappetta Jabbour, Ana Beatriz de and Charbel José Jabbour. Selecting green suppliers based on GSCM practices: using fuzzy TOPSIS applied to a Brazilian electronics company, Eur. J. Oper. Res. 233 (2014), 432–447.
[4] A. Özlem, E. Erdem, The control of greenhouses based on fuzzy logic using wireless sensor networks, Int. J. Comput. Int. Syst. 12 (2018), 190–203.
[5] N. Pandeeswari, G. Kumar, Anomaly detection system in cloud environment using fuzzy clustering based ANN, Mobile Netw. Appl. 21 (2016), 494–505.
[6] V. Venkatesa Kumar, K. Dinesh, Job scheduling using fuzzy neural network algorithm in cloud environment, Bonfring Int. J. Man Mach. Interface 2 (2012), 01–06.
[7] F.J. Moreno Velo, M.I. Baturone Castillo, S. Sánchez Solano, Á. Barriga Barros, Sánchez and Ángel Barriga Xfuzzy 3.0: a development environment for fuzzy systems, in Proceeding of the International Conference in Fuzzy Logic and Technology, Leicester, 2001.
[8] M. de Souza, F. dos Santos, A.R. de Soto, A. Vahldick, Fuzzystudio: A web tool for modeling and simulation of fuzzy systems, in 2014 Brazilian Conference on Intelligent Systems, Sao Paulo, 2014, pp. 306–311.
[9] M.R. Berthold, N. Cebron, D. Fabian, T.R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, B. Wiswedel, KNIME-the Konstanz information miner: version 2.0 and beyond, SIGKDD Explor. 11 (2009), 26–31.
[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, SIGKDD Explor. 11 (2009), 10–18.
[11] L. Riza, C. Bergmeir, F. Herrera, J.M. Benítez, frbs: fuzzy rule-based systems for classification and regression in R, J. Stat. Softw. 65 (2015), 1–30.
[12] J. McCulloch, Fuzzycreator: A python-based toolkit for automatically generating and analysing data-driven fuzzy sets, in 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Naples, 2017, pp. 1–6.
[13] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, J. Internet Serv. Appl. 1 (2010), 7–18.
[14] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, A. Ghalsasi, Cloud computingThe business perspective, Decis. Support Syst. 51 (2011), 176–189.
[15] J. Alcal-Fdez, J. Alonso, A survey of fuzzy systems software: taxonomy, current research trends, and prospects, IEEE Trans. Fuzzy Syst. 24 (2016), 40–56.
[16] M. Tiegelkamp, K.-H. John, IEC 61131-3: programming Industrial Automation Systems, Springer, Berlin, 1995.
[17] G. Acampora, V. Loia, C.-S. Lee, M.-H. Wang, On the Power of Fuzzy Markup Language, 2013.
[18] D. López, F.J. Moreno, A. Barriga, S. Sánchez-Solano, A and S XFL: a language for the definition of fuzzy systems, in Proceedings of 6th International Fuzzy Systems Conference, Barcelona, 1997, vol. 3, pp. 1585–1591.
[19] F. Moreno-Velo, A. Barriga, S. Sánchez-Solano, I. Baturone, XFSML: an XML-based modeling language for fuzzy systems, in 2012 IEEE International Conference on Fuzzy Systems, Brisbane, 2012, pp. 1–8.
[20] Z.C. Johanyák, D. Tikk, S. Kovács, K. Wai Wong, Fuzzy rule interpolation Matlab toolbox-FRI toolbox, in 2006 IEEE International Conference on Fuzzy Systems, 2006, pp. 351–357.
[21] J. Rada-vilela, Fuzzylite a fuzzy logic control library in C++, in Proceeding of the Open Source Developers Conference, Academic Press, Auckland, 2013.
[22] C. Wagner, Juzzy-a java based toolkit for type-2 fuzzy logic, in 2013 IEEE Symposium on Advances in Type-2 Fuzzy Logic Systems (T2FUZZ), Singapore, 2013, pp. 45–52.
[23] N. Cavus, The evaluation of learning management systems using an artificial intelligence fuzzy logic algorithm, Adv. Eng. Softw. 41 (2010), 248–254.
[24] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, Web service definition language (WSDL), Technical Report, World Wide Web Consortium, Sophia-Antipolis, 2001.
[25] M.J. Hadley, Web application description language (WADL), Technical Report, Mountain View, 2006.

[26] B. Elvesaeter, D. Panfilenko, S. Jacobi, C. Hahn, Aligning business and IT models in service-oriented architectures using BPMN and SoaML, in Proceedings of the First International Workshop on Model-Driven Interoperability, Oslo, 2010, pp. 61–68.

[27] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, *et al.*, OWL-S: Semantic markup for web services, W3C Member Submission 22 (2004), 2007–04.

[28] J. Domingue, D. Roman, M. Stollberg, Web service modeling ontology (WSMO) - An ontology for semantic web services, 2005.

[29] S. Kona, A. Bansal, L. Simon, A. Mallya, G. Gupta, T.D. Hite, USDL: aservice-semantics description language for automatic service discovery and composition, Int. J. Web Serv. Res. 6 (2009), 20.

[30] C. Pedrinaci, J. Cardoso, T. Leidig, Linked usdl: a vocabulary for web-scale service trading, in European Semantic Web Conference, Crete, 2014, pp. 68–82.

[31] G. Klyne, J.J. Carroll, Resource description framework (RDF): concepts and abstract syntax, Technical Report, W3C, Sophia-Antipolis, 2004.

[32] D. Beckett, Turtle-terse RDF triple language, 2008. http://www.ilrt.bris.ac.uk/discovery/2004/01/turtle/

[33] S. Alptekin, G. Alptekin, A fuzzy quality function deployment approach for differentiating cloud products, Int. J. Comput. Intell. Syst. 11 (2018), 1041–1055.

[34] M. Parra-Royon, G. Atemezing, J.M. Benítez, Semantics of data mining services in cloud computing, abs/1806.06826, 2018. http://arxiv.org/abs/1806.06826

[35] M. Maleshkova, C. Pedrinaci, J. Domingue, G. Alvaro, I. Martinez, Using semantics for automating the authentication of web APIs, in The Semantic Web–ISWC, Shanghai, 2010, pp. 534–549.

[36] S. Gong, B. Yin, Z. Zheng, K. yuan Cai, An adaptive control method for resource provisioning with resource utilization constraints in cloud computing, Int. J. Comput. Intell. Syst. 12 (2019), 485–497.

[37] P.T. Vuong, A.M. Madni, J.B. Vuong, VHDL implementation for a fuzzy logic controller, in 2006 World Automation Congress, Budapest, 2006, pp. 1–8.

[38] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, B. Recht, Occupy the cloud: Distributed computing for the 99%, in Proceedings of the 2017 Symposium on Cloud Computing, 2017, pp. 445–451.

[39] P.M. Mell, T. Grance, The NIST definition of cloud computing, Technical Report, Gaithersburg, 2011.

[40] R.D. Peng, Reproducible research in computational science, Science 334 (2011), 1226–1227.