A Fuzzy-based Brokering Service for Cloud Plan Selection

Sabrina De Capitani di Vimercati, *Senior Member, IEEE,* Sara Foresti, *Senior Member, IEEE,* Giovanni Livraga, *Member, IEEE,* Vincenzo Piuri, *Fellow, IEEE,* and Pierangela Samarati, *Fellow, IEEE.*
E-mail: *firstname.lastname*@unimi.it

***Abstract*—The current cloud market features a multitude of cloud services that differ from one another in terms of functionality or of security/performance guarantees. Users wishing to use a cloud service for storing, processing, or sharing their data must be able to select the service that best matches their desiderata. In this paper, we propose a novel, user-centric, brokering service for supporting users in the specification of requirements and enabling their evaluation against available cloud plans, assessing how much the different plans can satisfy the user's desiderata. Our brokering service allows users to specify their desiderata in an easy and intuitive way by using natural language expressions and high-level concepts. Fuzzy logic and fuzzy inference systems are adopted to quantitatively assess the compliance of cloud services with the users' desiderata, and hence to help users in the cloud service selection process.**

***Index Terms*—Cloud computing, cloud service selection, brokering service, natural language, fuzzy logic, fuzzy inference**

I. Introduction

THANKS to the undeniable benefits that data and applications outsourcing can bring to users, cloud computing has rapidly earned first-class citizenship in the current ICT panorama, and represents today a popular solution to store and analyze data and to deploy applications. As forecasted by analysts, this trend is expected to grow: a recent study by Gartner estimates that approximately 28% of the total market revenue for infrastructure, middleware, application and business processes will shift to the cloud by 2021 (from 17% in 2016) [1]. This scenario has fostered the appearance in the cloud market of a multitude of service plans, differing from one another in terms of functionality and/or security/performance guarantees, typically expressed via Service Level Objectives (SLOs) declared by providers in their Service Level Agreements (SLAs). As in any situation where a user can choose among alternatives offered by a rich and diversified market, selecting the right solution is a key requirement to ensure a satisfactory experience. Hence, the problem for users is to specify their desiderata and to map them onto the characteristics of cloud providers [2]. As testified by the growing academic and industrial efforts towards the creation of tools to enable cloud plan assessments (e.g., [2], [3], [4], [5], [6]), this problem is central to a wider adoption of the cloud. Research in this regard is however in its early stages, and the majority of existing solutions aim at a global assessment of cloud service plans compared to pre-defined baselines, while providing limited support for specific requirements by individual users [2], [6].

A simple approach to account for the specific needs of different users is to require users to look at the configuration parameters of the various cloud service plans and specify requirements on their values: for example, a user who needs to outsource mission-critical tasks can require a monthly uptime greater than 99.99%, hence ruling out all plans with lower uptime values. While this approach would certainly do, it has some drawbacks. First, operating directly with numbers (and crisp values in general) requires users to identify sharp boundaries between values that are acceptable (e.g., 99.99% uptime) and those that are not (e.g., a plan with 99.989% uptime would be discarded, although the difference with 99.99% can be - for many applications - negligible). Second, the features that characterize cloud service plans can be technically low-level and not immediately understandable to non-skilled users (e.g., the SLA of commercial providers can include terms such as 'API Error', 'Cluster', and 'Load Balancer' [7], which may not be understandable by users who do not have technical background). Third, different providers may use different terms to refer to similar/equivalent features, further complicating the scenario for users (e.g., both 'monthly uptime' in Amazon's Compute SLA [8] and 'monthly availability' in Rackspace's Cloud SLA [7] indicate, with a percentage, the time in which a plan is available in a month).

In this paper, we propose a fuzzy-based brokering service for cloud plan assessment with the specific goal of addressing and solving the three issues mentioned above. In particular, we address the first issue by allowing users to specify their needs by using natural language expressions (using terms such as 'high' and 'low' instead of crisp values) and fuzzy logic (in contrast to classical Boolean logic) to evaluate them, so to avoid sharp boundaries between acceptable and unacceptable values. We address the second issue by allowing users to operate on high-level concepts, easily usable and understandable also by non-skilled users, hence departing from the low-level technical details characterizing plans. We address the third issue by introducing a *broker*, acting as an intermediary between potential users and the cloud service plans, supporting users in the specification of high-level requirements and evaluating them against specific low-level plan characteristics. Our brokering service, allowing users to specify their requirements with natural language over high-level concepts and automatically comparing them against the

Sabrina De Capitani di Vimercati, Sara Foresti, Giovanni Livraga, Vincenzo Piuri, and Pierangela Samarati are with the Computer Science Department, Università degli Studi di Milano, Italy.

low-level characteristics of cloud plans, relieves the need for having technical skills at the user side. Our solution enables specification of desiderata also from non-skilled users, making the cloud a more appealing computing model to a wider audience.

The remainder of this paper is organized as follows. Section II presents the reference scenario. Section III illustrates an abstract model allowing users to easily express their desiderata using natural language expressions and high-level concepts. Section IV shows the semantics of parameters, concepts, and users' desiderata in terms of Fuzzy Set Theory. Section V describes how the cloud plan assessment works by relying on fuzzy inferences. Section VI discusses related work. Finally, Section VII concludes the paper.

II. Reference scenario

Our reference scenario is characterized by: *i) cloud providers* that offer service plans, *ii) users*, each in need to choose cloud service plans in line with her specific desiderata (e.g., a user may be interested in a plan that guarantees high throughput, while another user may prefer plans with a high replication factor), and *iii)* a *broker* that mediates the interaction between users and cloud providers. The brokering service offers an easy way for users to express their desiderata, and a mechanism for evaluating them against technical characteristics. In particular, the brokering service enables users to express their desiderata using natural language expressions without the need to refer to low-level technical characteristics of cloud service plans. It then evaluates such desiderata against the characteristics of available cloud service plans assessing how much each plan complies with the specific desiderata.

The plans offered by cloud providers are characterized by a set $\mathcal{P} = \{p_1, \ldots, p_n\}$ of configuration parameters, typically corresponding to Service Level Indicators (SLIs) used for the definition of SLOs. A parameter models a generic feature that can be measured/assessed and that characterizes the plans themselves (e.g., the reputation of a provider or its rating given by users). We consider as parameter any characteristic of interest for the users, provided that there is a method for verifying or measuring it. Each parameter $p \in \mathcal{P}$ is associated with a domain of crisp values, denoted $D(p)$, including all the values that p can assume. In our running example, we consider the following parameters (Figure 1, first and second column).

- `uptime` (the average percentage of time in a month in which a plan is available), with domain [96.00,99.99];
- `replicas` (the number of replicas guaranteed for the outsourced data), with domain [1,10] of integer values;
- `throughput` (rate of successful message delivery in Gbps), with domain [1,16];
- `bandwidth` (data transfer rate in Gbps), with domain [0.2,25];
- `reputation` (average rating of the plan given by users), with normalized domain [0,1].

A *cloud service plan*, denoted s, is characterized by a combination of values for the parameters in $\mathcal{P}$ and can be formally represented as a *specification vector* π_s with a cell for each parameter in $\mathcal{P}$, defined as follows.

Parameter p	**Domain** $D(p)$	**Lingustic values** $\mathcal{L}(p)$
`uptime`	$[96.00, 99.90]$	{low,med,high}
`replicas`	$[1, 10]$	{scarce,many}
`throughput`	$[1, 16]$	{low,med,high}
`bandwidth`	$[0.2, 25]$	{small,large}
`reputation`	$[0, 1]$	{abad,avg,good}

Fig. 1. An example of configuration parameters

Definition II.1 (Specification Vector)**.** *Given a cloud service plan s, a set $\mathcal{P} = \{p_1, \ldots, p_n\}$ of configuration parameters, with $D(p_i)$ the domain of crisp values for p_i, $i = 1, \ldots, n$, the* specification vector π_s *of s is a vector of n elements where $\pi_s[i] \in D(p_i)$ is the value that cloud service plan s assumes for parameter p_i, $i = 1, \ldots, n$.*

For instance, a storage service plan s that guarantees 99.90% average monthly uptime, 10 replicas of stored data, 2.5Gbps throughput, 10Gbps bandwidth, and that has a reputation equal to 0.7 is represented by vector: π_s[`uptime`, `replicas`, `throughput`, `bandwidth`, `reputation`]=$[99.90, 10, 2.5, 10, 0.7]$.

III. Abstract model

Users, especially non technically skilled ones, may find it difficult to understand the meaning of low-level configuration parameters and therefore to specify precise constraints on their values that correspond to the needs of data and applications to be outsourced. Our approach to help users in formulating their desiderata is based on the idea that users can find it easier to use linguistic values (e.g., 'high' and 'low') instead of crisp parameter values and to operate on high-level properties (e.g., 'reliability' and 'performance') instead of low-level parameters. To this purpose, we introduce two constructs: *abstract parameters* and *abstract concepts*. Intuitively, abstract parameters allow users to express their requirements over the configuration parameters through natural language expressions, which can be used to define in a user-friendly and flexible manner the boundaries between preferred and non-preferred configurations (e.g., a user can ask for 'many' replicas, without specifying the exact number). Abstract concepts provide high-level abstractions over parameters (e.g., abstract concept `performance` can be defined over parameters `throughput` and `bandwidth`), expressing plan characteristics in a more intuitive way for users.

A. Abstract parameters and concepts

Abstract parameters allow users to express their desiderata without referring to specific crisp values of configuration parameters. This is done using linguistic values (e.g., low, med, high), which intuitively quantify in a less precise and more informal manner the value that configuration parameters can assume. An abstract parameter is then associated with (and can take values from) a set of linguistic values, as formally defined in the following.

Definition III.1 (Abstract Parameter)**.** *Given a configuration parameter $p \in \mathcal{P}$ with domain $D(p)$, an* abstract parameter

Concept c	Linguistic values $\mathcal{L}(c)$	Involved parameters $\phi(c)$
reliability	{low,med,high}	{uptime, replicas}
performance	{low,med,high}	{throughput, bandwidth}

Fig. 2. An example of concepts

is a triple $\langle p, D(p), \mathcal{L}(p)\rangle$ that extends p with a set $\mathcal{L}(p)$ of linguistic values for p.

For instance, parameter uptime can be associated with set $\mathcal{L}$(uptime)={low,med,high} of linguistic values. The last column in Figure 1 reports the set of linguistic values that can be associated with the configuration parameters of our running example. For simplicity, in the following, when clear from the context, we refer to abstract parameters simply using the term parameters.

Besides abstract parameters, our brokering service permits users to express their desiderata through abstract concepts. For instance, concept performance is an example of a high-level property that can characterize cloud service plans, with a semantics accessible also to non-skilled users. Each concept is associated with a set of linguistic values. Formally, a concept is defined as follows.

Definition III.2 (Abstract Concept)**.** *An* abstract concept *is a pair $\langle c, \mathcal{L}(c)\rangle$, where $\mathcal{L}(c)$ is a domain of linguistic values for c.*

Like for parameters, for simplicity, in the paper we use the term concept to denote abstract concepts. Concepts enable users to specify how much the properties (e.g., performance or reliability) they represent are relevant for their needs. In accordance to this, linguistic values for concepts correspond to levels of importance and we can assume them to be totally ordered (e.g., $\mathcal{L}(c)$={low,med,high}, with low lower that med and med lower than high).

Figure 2 (first and second column) illustrates an example of two concepts together with their sets of linguistic values. The definition of high-level concepts naturally depends on configuration parameters. Such a correspondence is however completely transparent to the user as it is mediated by the broker. For instance, with reference to the parameters in Figure 1, the performance (concept) of a plan depends on its guaranteed throughput and bandwidth (parameters). Users can express their desiderata through conditions on performance, without the need to know that the satisfaction of such conditions depends on the values that parameters throughput and bandwidth assume in the cloud service plan specification.

The relationship linking concepts and parameters is represented through a set of *implication rules*, dictating which values for parameters imply which values of high-level concepts. Intuitively, an implication rule identifies the combinations of values assumed by configuration parameters that imply a given linguistic value for a concept (e.g., performance is high if uptime is high or replicas is many). Note that each linguistic value $l \in \mathcal{L}(c)$ must be regulated by an implication rule for guaranteeing a correct and non-ambiguous interpretation of concepts by the broker (see Section V). Formally, the set of implication rules governing a concept c is defined as follows.

Implication rules for concept reliability

$\langle$uptime = high$\rangle \vee \langle$replicas = many$\rangle \Longrightarrow \langle$reliability = high$\rangle$
$\langle$uptime = med$\rangle \Longrightarrow \langle$reliability = med$\rangle$
$\langle$uptime = low$\rangle \vee \langle$replicas = scarce$\rangle \Longrightarrow \langle$reliability = low$\rangle$

Implication rules for concept performance

$\langle$throughput = high$\rangle \vee \langle$bandwidth = large$\rangle \Longrightarrow \langle$performance = high$\rangle$
$\langle$throughput = med$\rangle \Longrightarrow \langle$performance = med$\rangle$
$\langle$throughput = low$\rangle \vee \langle$bandwidth = small$\rangle \Longrightarrow \langle$performance = low$\rangle$

Desiderata of user u_1

$\langle$reliability = high$\rangle \vee \langle$reputation = good$\rangle \Longrightarrow \langle$satisfaction = high$\rangle$
$\langle$reliability = med$\rangle \vee \langle$reputation = avg$\rangle \Longrightarrow \langle$satisfaction = med$\rangle$
$\langle$reliability = low$\rangle \vee \langle$reputation = abad$\rangle \Longrightarrow \langle$satisfaction = low$\rangle$

Fig. 3. An example of implication rules for concepts and of user's desiderata

Definition III.3 (Implication Rules)**.** *Given an abstract concept $\langle c, \mathcal{L}(c)\rangle$, and a set $\mathcal{P}$ of abstract parameters, the set $R(c)$ of* implication rules *governing c is a set of rules of the form "*cond $\Longrightarrow \langle c = l\rangle$*", where $l \in \mathcal{L}(c)$ and* cond *is a monotonic Boolean formula over base clauses of the form $\langle p_i = l_j\rangle$ with $p_i \in \mathcal{P}$ and $l_j \in \mathcal{L}(p_i)$. There is a rule in $R(c)$ for each linguistic value l in $\mathcal{L}(c)$.*

Each rule in $R(c)$ identifies the configurations of parameter (linguistic) values that are associated with a linguistic value for concept c. Note that a single rule is sufficient to represent all the configurations of parameter values that imply c=l. Indeed, a single rule permits to express combinations of values as well as alternatives among them. For instance, since $\mathcal{L}$(performance) includes three linguistic values (i.e., low, med, high), the set R(performance) of rules regulating performance will include three implication rules (one for low, one for med, and one for high). Figure 3 illustrates the implication rules for our running example. For instance, the first rule in R(reliability) states that a high reliability is provided by plans that have either high uptime *or* many replicas. We expect the broker to rely on a pool of experts in the field for the identification of the implication rules governing the definition of concepts.

In the following, $\phi(c)$ is used to denote the set of parameters influencing concept c, that is, those parameters that appear in at least a rule in $R(c)$. Clearly, a parameter can influence different concepts. Also, concepts can be defined on top of other concepts, thus forming a hierarchy of concepts where high-level concepts depend on low-level concepts. This would be possible by allowing concepts in the definition of the implication rules governing other (higher-level) concepts. For simplicity, but without loss of generality, in the following we will consider simple concepts directly depending on parameters only, while noting that our approach can be easily extended to the consideration of a hierarchy of concepts.

B. User's desiderata

The desiderata of a user express her preferences with respect to the configurations of cloud service plans that suit her needs. Intuitively, user's desiderata reflect the different levels of satisfaction (e.g., high, med, low) of the user with respect to different cloud service plan configurations. In line with our

goal of supporting users in the definition of their desiderata using natural language expressions, the broker allows users to use linguistic values for specifying levels of satisfaction. We then use a variable, called `satisfaction`, which can take values from a set $\mathcal{L}$(`satisfaction`) of linguistic values, reflecting different levels of satisfaction. Desiderata are expressed by users via rules declaring the level of satisfaction implied by a combination of parameters and concepts. For generality, accounting for scenarios where users may want to express preferences also over low-level configuration parameters and/or using crisp values, we include the possibility of referring to them in such rules.

Formally, the set of rules modeling a set of user's desiderata is defined as follows.

Definition III.4 (User's Desiderata)**.** *Given a set $\mathcal{P}$ of abstract parameters and a set $\mathcal{C}$ of concepts, the set of* user's desiderata *governing* `satisfaction` *is a set R(`satisfaction`) of implication rules of the form "*cond $\Longrightarrow \langle$`satisfaction` $= l\rangle$*", where $l \in \mathcal{L}$(`satisfaction`) and* cond *is a monotonic Boolean formula over base clauses of the form: $\langle p_i = l_j\rangle$, with $p_i \in \mathcal{P}$ and $l_j \in \mathcal{L}(p_i)$; $\langle p_i = v_j\rangle$, with $p_i \in \mathcal{P}$ and $v_j \in D(p_i)$; or $\langle c_i = l_j\rangle$, with $c_i \in \mathcal{C}$ and $l_j \in \mathcal{L}(c_i)$. There is a rule in R(`satisfaction`) for each linguistic value l in $\mathcal{L}$(`satisfaction`).*

Intuitively, each rule in R(`satisfaction`) identifies the configurations that are associated by the user with level l of satisfaction. Similarly to what already noted for concepts, one rule is sufficient to express all the configurations providing the same level of satisfaction. Figure 3 illustrates an example of a set of user's desiderata, defined over the parameters and concepts in Figures 1 and 2. In this example, rule $\langle$`reliability=high`$\rangle \vee \langle$`reputation=good`$\rangle \Longrightarrow \langle$`satisfaction=high`$\rangle$ states that a cloud service plan that guarantees high reliability or has good reputation is considered highly satisfactory by the user.

In the following, we use ϕ(`satisfaction`) to denote the set of parameters and concepts influencing `satisfaction`, that is, those parameters and concepts that appear in at least a rule in R(`satisfaction`). With reference to our running example, ϕ(`satisfaction`)={`reliability`, `reputation`}.

IV. Fuzzy modeling

Our brokering service enables users to express their desiderata through conditions over parameters and/or concepts using linguistic values. Clearly, to evaluate such conditions, the broker needs to establish a mapping between concepts and parameters (according to implication rules) and between crisp and linguistic values. In this respect, the broker can operate in different ways. An intuitive approach for translating crisp values into linguistic values consists in partitioning the domain $D(p)$ of crisp values of each parameter p in a set of intervals, and map each interval to a linguistic value l in $\mathcal{L}(p)$. For instance, the domain of parameter `bandwidth` could be partitioned in two intervals, where range [0.2,1] corresponds to small and range (1,25] corresponds to large. Such a translation, while intuitive, has the disadvantage of maintaining sharp boundaries between the intervals of crisp values that correspond to linguistic values. Hence, while `bandwidth`=1Gbps is considered small, `bandwidth`=1.01Gbps is considered large, even if these values are quite close. The transition between two linguistic values is instead usually perceived by the user as smooth (e.g., there is not a sharp boundary between values of `bandwidth` that are considered small and those that are considered large) and crisp values can correspond, to a different extent, to more than one linguistic value. A `bandwidth` of 1Gbps can then be considered small, but not as much as 0.2Gbps, but it can also be considered large, but not as much as 25Gbps. To better model the understanding of users when expressing their preferences in terms of linguistic values, we interpret parameters and concepts as *fuzzy variables*. Fuzzy variables can assume crisp as well as linguistic values, which are interpreted as fuzzy sets. A fuzzy set is a set whose elements have *degrees* of membership. In contrast to the classical set theory, where an element either belongs or does not belong to a set, fuzzy set theory permits a gradual assessment of the membership of an element to a set through a membership function. In our context, each pair $\langle x, l\rangle$ (with x either a parameter, a concept, or `satisfaction`, and $l \in \mathcal{L}(x)$) corresponds to a fuzzy set and the membership function for it maps crisp values of $D(x)$ to their degree of membership. The degree of membership is a value in the interval [0,1], with 0 corresponding to no membership and 1 to full membership.

In the remainder of this section, we discuss the interpretation, according to fuzzy logic, of parameters, concepts, and user's desiderata.

A. Fuzzy parameters

The fuzzy variable modeling a parameter is called *fuzzy parameter* and enriches an abstract parameter p with a fuzzy interpretation regulating the relationship between crisp values and linguistic values through membership functions. According to fuzzy set theory, each linguistic value l in $\mathcal{L}(p)$ represents a fuzzy set with a membership function $\mu_{p,l}$:$D(p)\rightarrow$[0,1], assessing the degree of membership of a crisp value v in $D(p)$ to the fuzzy set represented by the linguistic value l. Figures 4(a)-(e) illustrate an example of membership functions for the parameters in Figure 1. Each subfigure reports the functions for the different linguistic values of the considered parameter. For simplicity, we refer our examples to triangular membership functions. Our approach is however general and does not restrict membership functions to be defined in a particular way nor to assume specific shapes, and other functions (e.g., trapezoidal, Gaussian, or sigmoidal) can be used.

As already noted, a crisp value v in $D(p)$ can have membership greater than 0 for more than one linguistic value. In fact, a value in $D(p)$ can belong to different fuzzy sets with a different membership degree. For instance, according to the functions in Figure 4, value `uptime`=99% is considered med with membership $\mu_{\texttt{uptime},\mathsf{med}}(99.2\%)$=0.326, and high with membership $\mu_{\texttt{uptime},\mathsf{high}}(99.2\%)$=0.531. We can then say that 99% is perceived as both med and high, with the second

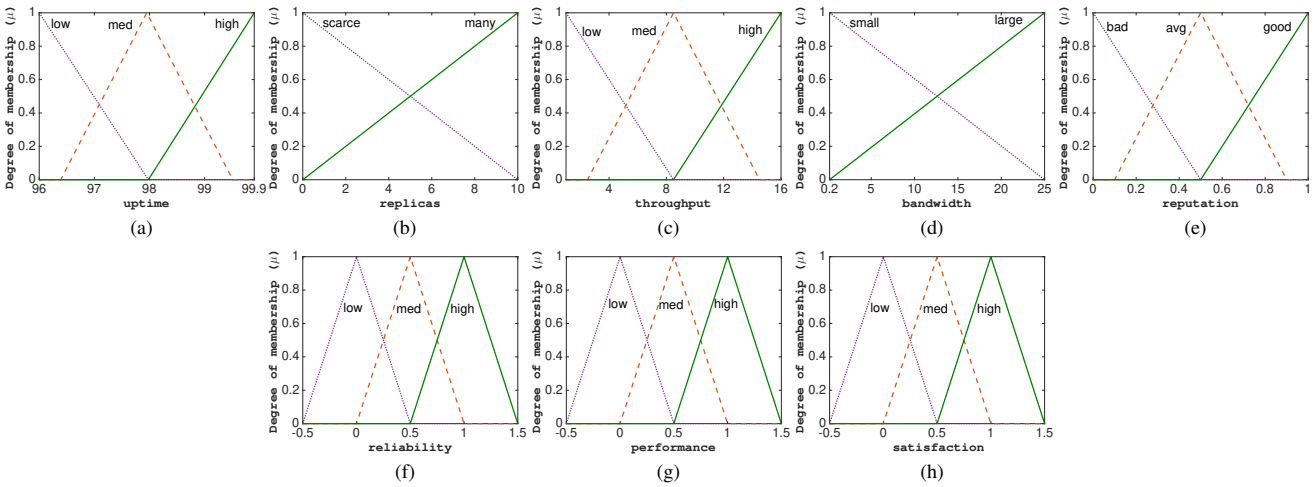


Fig. 4. An example of membership functions for the parameters in Figure 1 (a)-(e), for concepts in Figure 2 (f)-(g), and for `satisfaction` (h)

linguistic value being *more representative* than the first one. This flexibility provided by the adoption of fuzzy variables permits to model user's reasoning on the relationship between crisp and linguistic values without the need of setting sharp boundaries in the mapping of crisp values to linguistic values. Formally, a fuzzy parameter is defined as follows.

Definition IV.1 (Fuzzy Parameter)**.** *Given an abstract parameter $p \in \mathcal{P}$ (Def. III.1), a* fuzzy parameter *is a quadruple $\langle p,D(p),\mathcal{L}(p),M(p)\rangle$ that extends the abstract parameter p with a set $M(p)$ of membership functions such that $M(p)=\{\mu_{p,l}:D(p)\rightarrow[0,1] \mid l \in \mathcal{L}(p)\}$.*

For instance, abstract parameter `uptime` in Figure 1 is represented as a fuzzy parameter $\langle \texttt{uptime}, [96.00, 99.90], \{\text{low}, \text{med}, \text{high}\}, \{\mu_{\texttt{uptime},\text{low}}, \mu_{\texttt{uptime},\text{med}}, \mu_{\texttt{uptime},\text{high}}\}\rangle$, with $\mu_{\texttt{uptime},\text{low}}$, $\mu_{\texttt{uptime},\text{med}}$, and $\mu_{\texttt{uptime},\text{high}}$ the membership functions illustrated in Figure 4(a).

The membership functions of a fuzzy parameter are provided by the broker and depend on the specific application scenario. We expect the broker to rely on experts in the field for the definition of the set of linguistic values, and the corresponding membership functions, for each configuration parameter. Linguistic values and membership functions are made available to the final user, who can take them into consideration when formulating her desiderata.

B. Fuzzy concepts

The fuzzy variable modeling a concept is called *fuzzy concept* and extends a concept c with a domain $D(c)$ of crisp values (which is not naturally associated with an abstract concept) and a set of membership functions. Intuitively, the crisp values of a concept should quantify how much a cloud service plan is compliant with the concept itself (e.g., how much a plan guarantees `performance`), which is based on the values of the configuration parameters governing the concept (i.e., parameters in $\phi(c)$). To enable quantification of compliance with a concept, we use the continuous interval [0,1] as domain for concept c, where higher values represent higher compliance with the concept (value 0 models non compliance with the concept at all, while value 1 models full compliance with the concept). As already discussed, linguistic values correspond to different degrees of compliance for the considered concept and are ordered.

In the interpretation of a concept c as a fuzzy concept, each linguistic value $l \in \mathcal{L}(c)$ represents a fuzzy set and is associated with a membership function. Figures 4(f)-(g) illustrate an example of membership functions for abstract concepts `reliability` and `performance`. Note that, in the definition of the membership functions, we assume the domain to be a superset of [0,1] to guarantee that the centroid of any possible area defined by a membership degree over the different membership functions covers the whole interval [0,1]. The reason for this is that, as it will be clear in the following section, for concepts both fuzzification and defuzzification (which we perform taking an area's centroid) need to be computed. To guarantee such conditions, the extreme membership functions (i.e., the membership functions with maximum equal to 0 and 1) must be designed in such a way that their centroids correspond to 0 and 1, respectively. Concretely, this means that the domain of these functions must be larger than [0,1]. In our example, the membership functions corresponding to fuzzy sets low and high of concepts `reliability` and `performance` are defined over the range $[-0.5, 1.5]$. Formally, a fuzzy concept is defined as follows.

Definition IV.2 (Fuzzy Concept)**.** *Given an abstract concept $\langle c,\mathcal{L}(c)\rangle$ (Def. III.2), a* fuzzy concept *is a quadruple $\langle c,D(c),\mathcal{L}(c),M(c)\rangle$ that extends the abstract concept with a domain $D(c) = [0,1]$ of crisp values and a set $M(c)$ of membership functions such that there is a membership function $\mu_{c,l}$ in $M(c)$ for each linguistic value l in $\mathcal{L}(c)$.*

For instance, abstract concept `reliability` in Figure 2 is represented as a fuzzy concept $\langle \texttt{reliability}, [0,1], \{\text{low}, \text{med}, \text{high}\}, \{\mu_{\texttt{reliability},\text{low}}, \mu_{\texttt{reliability},\text{med}}, \mu_{\texttt{reliability},\text{high}}\}\rangle$, with $\mu_{\texttt{reliability},\text{low}}$,

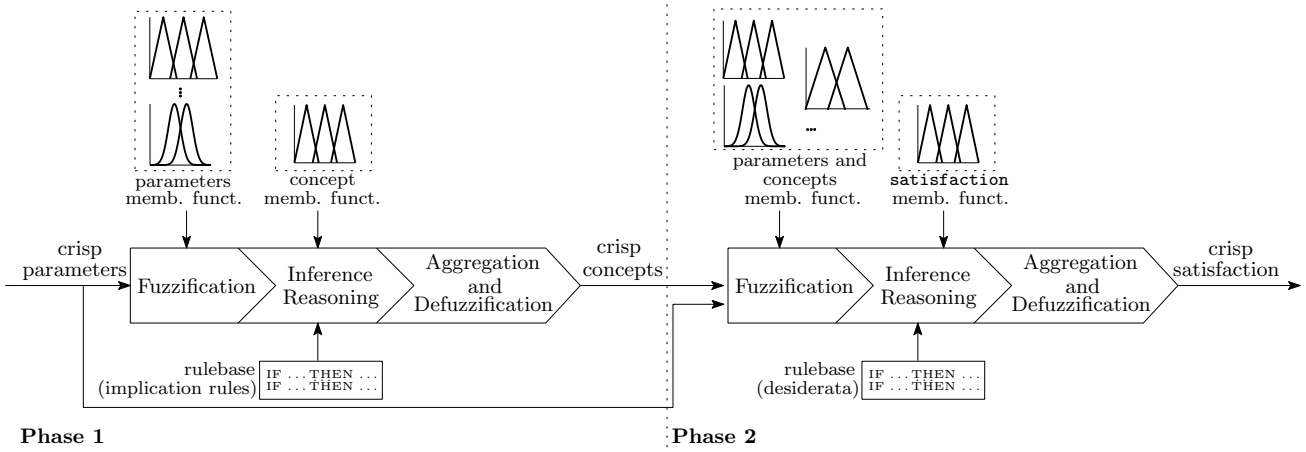


Fig. 5. Graphical representation of the steps for cloud plan assessment

$\mu_{\texttt{reliability,med}}$, and $\mu_{\texttt{reliability,high}}$ the membership functions in Figure 4(f).

C. Fuzzy user's desiderata

The degree of satisfaction of user's desiderata is modeled as a fuzzy variable defined similarly to fuzzy concepts.

Definition IV.3 (Satisfaction)**.** *Given variable* `satisfaction` *modeling the satisfaction of a user along with the corresponding set* $\mathcal{L}(\texttt{satisfaction})$ *of linguistic values, the* fuzzy user's desiderata *is a quadruple* $\langle \texttt{satisfaction}, D(\texttt{satisfaction}), \mathcal{L}(\texttt{satisfaction}), M(\texttt{satisfaction})\rangle$ *where* $D(\texttt{satisfaction}) = [0,1]$ *and* $M(\texttt{satisfaction})$ *is a set of membership functions such that there is a membership function* $\mu_{\texttt{satisfaction},l}$ *in* $M(\texttt{satisfaction})$ *for each linguistic value* l *in* $\mathcal{L}(\texttt{satisfaction})$.

The domain $D(\texttt{satisfaction})$ of crisp values for `satisfaction`, necessary to quantify the degree of satisfaction, is the interval [0,1], with higher values representing higher levels of satisfaction. Figure 4(h) illustrates an example of membership functions for `satisfaction`, modeling the smooth transition between linguistic values. Like for concepts, the domains of these functions are designed in such a way that the centroid of any area defined by a membership degree over the different membership functions covers the whole [0,1] interval.

V. Cloud plan assessment

The broker evaluates the available cloud service plans against users' desiderata to assess the level of satisfaction in the different cloud service plans for each user. The brokering process operates in two phases, the first one abstracting to concepts, and the second one performing assessment. Each phase relies on a set of Fuzzy Inference Systems (FISs) (Figure 5). Phase 1 provides reasoning on fuzzy parameters to derive concepts. It takes as input the specification of cloud service plans (Def. II.1), the specifications of fuzzy parameters (Def. IV.1) and fuzzy concepts (Def. IV.2), and the implication rules for concepts (Def. III.3) and produces as output a (crisp) value for each concept for each cloud service plan. Phase 2 evaluates concepts and parameters of the different cloud service plans to determine the user's satisfaction in the cloud service plans. It takes as input the specification of cloud service plans (Def. II.1), the crisp values of concepts computed in Phase 1, the specification of parameters and concepts (Def. IV.1 and Def. IV.2) as well as the specification of satisfaction provided by users (Def. IV.3) and the rules establishing desiderata (Def. III.4), and produces as output a (crisp) value of `satisfaction` for each user and each plan. Note that, since the specification of fuzzy parameters and concepts is provided by the broker itself with the help of a pool of experts, Phase 1 is user-independent, while Phase 2, evaluating satisfaction of individual users (and hence regulated by different rules for each user) is user-dependent.

The rulebases input to the FISs are obtained by translating the rules in input (implication rules for concepts for Phase 1 and user's desiderata for Phase 2) in the form of "IF-THEN" rules. More precisely, each rule "$cond \implies f$", with f either $\langle p{=}l\rangle$ or $\langle c{=}l\rangle$, is transformed into an equivalent rule "IF $cond'$ THEN f'" where $cond'$ is obtained substituting $\wedge$ and $\vee$ with *and* and *or* in *cond*, respectively, and f' is obtained substituting = with 'is' in f. Figure 6 reports the translation of the implication rules and desiderata (for user u_1) in Figure 3.

For the design of the FISs we use the Mamdani's method [9], since it is one of the most common methods adopted in decision support systems [10]. Accordingly, the output of each rule is interpreted as a fuzzy set.

Each of the two phases, while operating on fuzzy specifications and inferences, has crisp values as input and as output, and comprises the following three steps (Figure 5).

- *Fuzzification.* It maps input crisp values into degrees of membership for each fuzzy set of the fuzzy parameters/concepts to which the input value refers, by evaluating the corresponding membership functions.
- *Inference reasoning.* It evaluates the different rules with respect to the fuzzy values computed in the previous (fuzzification) step.

Implication rules for concept `reliability`

IF ⟨`uptime` is high⟩ or ⟨`replicas` is many⟩ THEN ⟨`reliability` is high⟩
IF ⟨`uptime` is med⟩ THEN ⟨`reliability` is med⟩
IF ⟨`uptime` is low⟩ or ⟨`replicas` is scarce⟩ THEN ⟨`reliability` is low⟩

Implication rules for concept `performance`

IF ⟨`throughput` is high⟩ or ⟨`bandwidth` is large THEN ⟨`performance` is high⟩
IF ⟨`throughput` is med⟩ THEN ⟨`performance` is med⟩
IF ⟨`throughput` is low⟩ or ⟨`bandwidth` is small THEN ⟨`performance` is low⟩

Desiderata of user u_1

IF ⟨`reliability` is high⟩ or ⟨`reputation` is good⟩ THEN ⟨`satisfaction` is high⟩
IF ⟨`reliability` is med⟩ or ⟨`reputation` is avg⟩ THEN ⟨`satisfaction` is med⟩
IF ⟨`reliability` is low⟩ or ⟨`reputation` is abad⟩ THEN ⟨`satisfaction` is low⟩

Desiderata of user u_2

IF ⟨`reliability` is high⟩ THEN ⟨`satisfaction` is high⟩
IF ⟨`reliability` is med⟩ THEN ⟨`satisfaction` is med⟩
IF ⟨`reliability` is low⟩ THEN ⟨`satisfaction` is low⟩

Desiderata of user u_3

IF ⟨`reputation` is good⟩ and (⟨`reliability` is high⟩ or ⟨`performance` is high⟩) THEN ⟨`satisfaction` is high⟩
IF ⟨`reputation` is good⟩ and (⟨`reliability` is med⟩ or ⟨`performance` is med⟩) THEN ⟨`satisfaction` is med⟩
IF ⟨`reputation` is good⟩ and (⟨`reliability` is low⟩ or ⟨`performance` is low⟩) THEN ⟨`satisfaction` is low⟩
IF ⟨`reputation` is avg⟩ THEN ⟨`satisfaction` is low⟩
IF ⟨`reputation` is low⟩ THEN ⟨`satisfaction` is low⟩

Fig. 6. Rules for concepts `performance` and `reliability`, and for the desiderata of users u_1, u_2, and u_3

			Cloud service plans				
			s_1	s_2	s_3	s_4	s_5
Parameters	uptime	crisp	99	99.9	99	97	96
		high	0.531	1	0.531	0	0
		med	0.326	0	0.326	0.391	0
		low	0	0	0	0.494	1
	replicas	crisp	5	10	0	2	1
		many	0.500	0	1	0.800	0.900
		scarce	0.500	1	0	0.200	0.100
	throughput	crisp	14	8.5	16	8.5	8.5
		high	0.733	0	1	0	0
		med	0.083	1	0	1	1
		low	0	0	0	0	0
	bandwidth	crisp	18.5	12.5	25	12.5	12.5
		large	0.738	0.496	1	0.496	0.496
		small	0.262	0.504	0	0.504	0.504
	reputation	crisp	1	0.5	1	0.5	0.25
		good	1	0	1	0	0
		avg	0	1	0	1	0.375
		abad	0	0	0	0	0.500
Concepts	reliability	crisp	0.509	1	0.435	0.319	0.159
		high	0.018	1	0	0	0
		med	0.982	1	0.870	0.638	0.318
		low	0	0	0.130	0.362	0.682
	performance	crisp	0.671	0.498	1	0.498	0.498
		high	0.342	0	1	0	0
		med	0.658	0.996	0	0.996	0.996
		low	0	0.004	0	0.004	0.004
Sat.	satisfaction	u_1	0.669	0.750	0.685	0.325	0.265
		u_2	0.513	1	0.416	0.306	0.175
		u_3	0.669	0	0.685	0	0

Fig. 7. Parameters of five service plans in the example, corresponding concepts, and resulting users' satisfaction. (Gray cells denote crisp values for parameters and concept quantifications)

- *Aggregation and defuzzification.* It combines the results of the different rules to determine a single (crisp) value as a result.

We now describe the working of the two phases. In the discussion, we consider the cloud service plans in Figure 7 the fuzzy values and membership functions for parameters and concepts in Figures 4(a)-(g), and the implication rules for concepts and users' desiderata in Figure 6. Note that our approach is general and does not require the adoption of specific and pre-defined methods for executing the different operations needed in the inference process (e.g., the adoption of the centroid method for defuzzification) [11]. The methods illustrated in the discussion (i.e., *min* for computing the *and* among values and for evaluating rule implication; *max* for computing the *or* among values and for evaluating rules aggregation; and *centroid* for defuzzification) are chosen as an example, among the possible ones, due to their intuitive interpretation.

Phase 1. It provides reasoning for mapping parameters to concepts. The FIS operates on each concept as follows.

- *Fuzzification.* It fuzzifies the parameters involved in the concept's definition. Input to a concept's FIS are the cloud service plans (more precisely, crisp values of the parameters that are involved in the concept). For instance, for concept `reliability`, input crisp values will be the values of `uptime` and `replicas`. Figure 7 reports the result of the fuzzification step for the different parameters. For instance, for service plan s_1, fuzzification of `uptime` produces $\mu_{\texttt{uptime,high}}(99)$=0.531, $\mu_{\texttt{uptime,med}}(99)$=0.326, and $\mu_{\texttt{uptime,low}}(99)$=0.
- *Inference reasoning.* Reasoning works on the rulebase corresponding to the implication rules defining the concept (for `reliability` the first set of rules in Figure 6). For each rule, the antecedent is evaluated by considering, for each individual clause in it, the corresponding membership degree, and enforcing *and* (*or*, resp.) combination of clauses by taking their *min* (*max*, resp.) value. The final result is, for the rule consequent, the area truncated at the resulting degree of membership on its fuzzy set. Figure 8(a) graphically reports the evaluation of the implication rules for concept `reliability` for service plan s_1, illustrating both the evaluation of the antecedents of the rules and the output truncated fuzzy sets. For instance, for the first rule: $\mu_{\texttt{uptime,high}}(99)$=0.531 and $\mu_{\texttt{replicas,many}}(5)$=0.5, corresponding to the yellow (light gray in b/w printout) areas in the membership function. Their *or* combination, that is, their maximum value, is then 0.531 for the rule consequent (i.e., `reliability` is high) producing the truncated fuzzy set as in the figure.
- *Aggregation and defuzzification.* The results of all the rules referring to the concept are aggregated by taking the union of the areas corresponding to the truncated fuzzy sets produced by each rule. Defuzzification is computed by taking the centroid of the area. Figure 8(a) graphically reports such a step for the rules on concept `reliability` for service plan s_1. The aggregation of the different rules produces the area at the very bottom, whose centroid is 0.509, which then corresponds to the quantification of concept `reliability` for cloud service plan s_1. Note that the reason for producing a crisp value for the concepts (to be fed to Phase 2 on which they will be fuzzified) is that the result of rule aggregation is a truncated fuzzy set, having a shape that does not

correspond to any membership function for the concept. Also, passing through crisp values, the evaluation of concepts is completely independent from the evaluation of the satisfaction of cloud service plans.

The evaluation of the other cloud service plans works in an analogous way, producing the values for concepts `reliability` and `performance` reported in Figure 7.

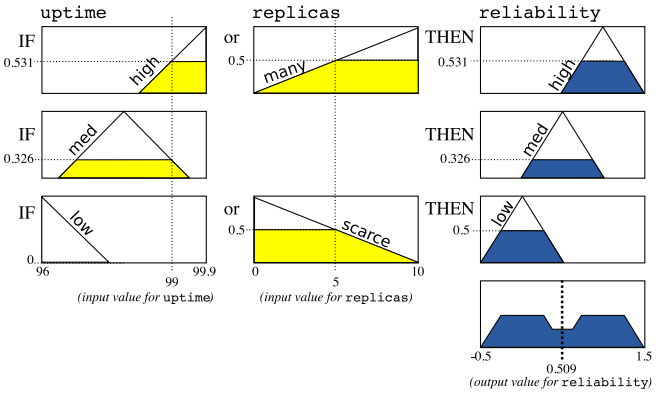
Phase 2. It provides reasoning on users' desiderata to produce, for each user, the `satisfaction` for the different cloud service plans. The FIS operates on each user as follows.

- *Fuzzification.* It fuzzifies, for each plan, the parameters and concepts involved in user's desiderata. Figure 7 reports the result of the fuzzification step for the different parameters and concepts. For instance, for user u_1, for whom concept `reliablity` and parameter `reputation` need to be considered, fuzzification for service plan s_1 produces for `reliability`: $\mu_{\texttt{reliability,high}}(0.509)$=0.018, $\mu_{\texttt{reliability,med}}(0.509)$=0.982, $\mu_{\texttt{reliability,low}}(0.509)$=0.
- *Inference reasoning.* Fuzzy reasoning works on the rule-base corresponding to the rules representing the desiderata of the user. Rule evaluation operates like for the first phase: the antecedent is evaluated by considering for each individual clause in it the corresponding membership degree,[1] and enforcing *and* (*or*, resp.) combination of clauses by taking their *min* (*max*, resp.) value. The final result is, for the rule consequent, the area truncated at the resulting degree of membership on its fuzzy set. Figure 8(b) graphically reports the evaluation of the implication rules for the desiderata of user u_1, illustrating both the evaluation of the antecedents of the rules and the output truncated fuzzy sets. For instance, $\mu_{\texttt{reliability,high}}(0.509)$=0.018, and $\mu_{\texttt{reputation,good}}(1)$=1, corresponding to the yellow (light gray in b/w printout) areas in the membership functions. Their *or* combination, that is, their maximum value, is then 1 for the rule consequent (i.e., `satisfaction` is high) producing the truncated fuzzy set as in the figure.
- *Aggregation and defuzzification.* The results of all the rules representing the desiderata of the user are aggregated by taking the union of the areas corresponding to the truncated fuzzy sets produced by each rule. Defuzzification is computed by taking the centroid of the area. Figure 8(b) graphically reports such a step for the rules on the desiderata of u_1 computing `satisfaction` in service plan s_1. The aggregation of the different rules produces the area at the very bottom, whose centroid is 0.669, which then corresponds to the quantification of the `satisfaction` of uses u_1 for cloud service plan s_1.

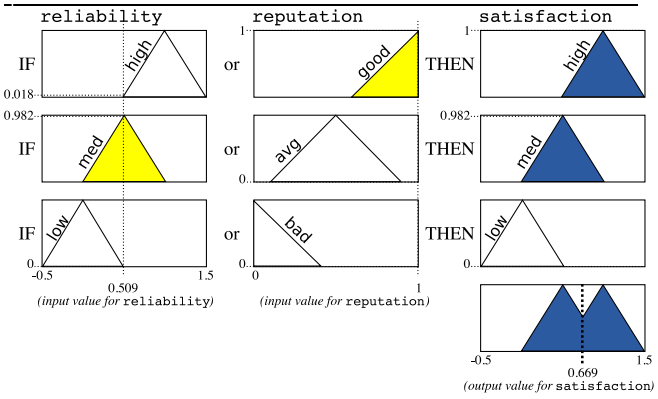
Evaluation of other plans and for other users works in an analogous way, producing the results in Figure 7.

Interpreting the results obtained for the five cloud service plans reported in Figure 7 and for the three users with desiderata reported in Figure 6, we can comment as follows

[1]For the evaluation of clauses of the form $\langle p_i{=}v_j\rangle$, we define a membership function having value 1 for v_j, and value 0 for any other value in $D(p_i)$.



(a) Inferences for `reliability`



(b) Inferences for `satisfaction`

Fig. 8. Evaluation of concepts `reliability` (a) and `satisfaction` (b) for service plan s_1 and user u_1

the resulting values of `satisfaction`. For user u_1, for whom `reliability` (which is a concept based on `uptime` and `replicas`) or `reputation` are important, the best matching plan is s_2, with s_1 and s_3 less preferred but still probably acceptable, while s_4 and s_5 are not satisfactory. For user u_2, where `reliability` is the only deciding factor, service plan s_2 (providing the maximum number of replicas) is fully satisfactory while the others (providing limited number of replicas) are much less satisfactory. For user u_3, for whom satisfaction depends on `reputation` of service plan being good and then on `reliability` or `performance`, service plans s_2, s_4, and s_5 are not satisfactory, while service plans s_1 and s_3 provide some (comparable) satisfaction.

VI. Related work

The problem of selecting a cloud provider/service considering user requirements has been recently widely investigated [12]. The line of work closest to ours provides support for the use of fuzzy logic and/or natural language expressions [13]. The solution in [6], while sharing with our work the use of fuzzy inferences to assess cloud services, specifically focuses on storage services. Also, our solution offers specific support for the definition of desiderata on abstract concepts. The approach in [5] allows users to specify

desiderata with linguistic labels to reflect the importance of the different parameters characterizing plans, while we support more general requirements based on parameters and concepts. In [14], the authors propose a framework for cloud service composition under fuzzy user preferences: besides pursuing a different goal (determining optimal compositions of cloud services), user preferences model different desiderata (properties that service compositions should satisfy). The fuzzy AHP-based framework in [15] supports imprecise user requirements expressed through linguistic labels over attributes organized in a fixed hierarchy. In [16], the authors present a multi-criteria decision making approach for cloud service selection that relays on the definition of a fuzzy ontology. While the ontology might resemble our relationship between parameters and concepts, our approach explicitly aims to map high-level concepts accessible to users to low-level parameters characterizing cloud services.

Our work shows similarities with other approaches that provide support for crisp user requirements in cloud provider/service selection (e.g., [17], [18], [19], [20]), which however address complementary problems. The approach in [2] proposes a language for expressing user requirements and preferences, a formal model for reasoning on them, and different strategies for ranking acceptable services. This work differs from ours as it manages requirements and preferences on crisp values, while we operate with linguistic expressions and high-level concepts.

Another related line of work deals with the general problem of cloud provider/service assessment (e.g., [21], [22], [23], [24]). Existing solutions, however, typically aim at identifying KPIs (Key Performance Indicators) [21] and operate on pre-defined metrics [22].

VII. Conclusions

In the rapidly evolving cloud scenario, selecting the right cloud service plan is an important yet complex task that users need to face. We have proposed a novel, user-centric, brokering service for assessing cloud service plans according to the desiderata of users. To facilitate users, our brokering service supports desiderata expressed using natural language, possibly with high-level concepts easily accessible also to non-skilled users. Such desiderata are automatically mapped by the broker onto the parameters characterizing plans. Our broker relies on fuzzy logic and fuzzy inferences to assess the extent to which a candidate plan complies with a set of user's desiderata. The fuzzy reasoning accommodates for flexibility in the specification and enforcement of desiderata, and can be further extended to membership functions by adopting a type-2 fuzzy system. This can be an interesting direction to investigate as future work.

Acknowledgments

This work was supported in part by the EC within the H2020 program under grant agreement 825333.

References

[1] Gartner, Inc., "Gartner Forecasts Worldwide Public Cloud Services Revenue to Reach $260 Billion in 2017," https://www.gartner.com/newsroom/id/3815165, 2017.

[2] S. De Capitani di Vimercati, S. Foresti, G. Livraga, V. Piuri, and P. Samarati, "Supporting user requirements and preferences in cloud plan selection," *IEEE Transactions on Services Computing*, 2017 (preprint).

[3] Cloud Security Alliance – Consensus Assessments Initiative, https://cloudsecurityalliance.org/group/consensus-assessments/.

[4] S. Garg, S. Versteeg, and R. Buyya, "SMICloud: A framework for comparing and ranking cloud services," in *Proc. of IEEE UCC 2011*, Victoria, Australia, December 2011.

[5] L. Qu, Y. Wang, M. A. Orgun, L. Liu, H. Liu, and A. Bouguettaya, "CCCloud: Context-aware and credible cloud service selection based on subjective assessment and objective assessment," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 369–383, May-June 2015.

[6] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2348–2362, August 2016.

[7] Rackspace Cloud Service Level Agreement, https://www.rackspace.com/information/legal/cloud/sla.

[8] Amazon Compute Service Level Agreement, https://aws.amazon.com/ec2/sla/.

[9] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, January 1975.

[10] A. Rikalovic, I. Cosic, V. Piuri, and R. Donida Labati, "A comprehensive method for industrial site selection: The macro location analysis," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2971–2980, December 2017.

[11] T. Nguyen, V. Lee, A. Khosravi, D. Creighton, and S. Nahavandi, "Solving fuzzy programming with a consistent fuzzy number ranking," in *Proc. of IEEE SMC 2014*, San Diego, CA, USA, October 2014.

[12] S. De Capitani di Vimercati, S. Foresti, G. Livraga, V. Piuri, and P. Samarati, "Supporting users in cloud plan selection," in *From Database to Cyber Security: Essays Dedicated to Sushil Jajodia on the Occasion of his 70th Birthday*, P. Samarati, I. Ray, and I. Ray, Eds. Springer, 2018.

[13] S. Foresti, V. Piuri, and G. Soares, "On the use of fuzzy logic in dependable cloud management," in *Proc. of IEEE CNS 2015*, Florence, Italy, September 2015.

[14] A. V. Dastjerdi and R. Buyya, "Compatibility-aware cloud service composition under fuzzy preferences of users," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 1–13, January-March 2014.

[15] I. Patiniotakis, S. Rizou, Y. Verginadis, and G. Mentzas, "Managing imprecise criteria in cloud service ranking with a fuzzy multi-criteria decision making method," in *Proc. of ESOCC 2013*, Malaga, Spain, September 2013.

[16] L. Sun, J. Ma, Y. Zhang, H. Dong, and F. K. Hussain, "Cloud-FuSeR: Fuzzy ontology and MCDM based cloud service selection," *Future Generation Computer Systems*, vol. 57, pp. 42–55, April 2016.

[17] A. Arman, S. Foresti, G. Livraga, and P. Samarati, "A consensus-based approach for selecting cloud plans," in *Proc. of IEEE RTSI 2016*, Bologna, Italy, September 2016.

[18] R. Jhawar, V. Piuri, and P. Samarati, "Supporting security requirements for resource management in cloud computing," in *Proc. of IEEE CSE 2012*, Paphos, Cyprus, December 2012.

[19] S. De Capitani di Vimercati, G. Livraga, V. Piuri, P. Samarati, and G. Soares, "Supporting application requirements in cloud-based IoT information processing," in *Proc. of IoTBD 2016*, Rome, Italy, April 2016.

[20] S. De Capitani di Vimercati, G. Livraga, and V. Piuri, "Application requirements with preferences in cloud-based information processing," in *Proc. of IEEE RTSI 2016*, Bologna, Italy, September 2016.

[21] S. K. Garg, S. Versteeg, and R. Buyya, "A framework for ranking of cloud computing services," *Future Generation Computer Systems*, vol. 29, no. 4, pp. 1012–1023, June 2013.

[22] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: Comparing public cloud providers," in *Proc. of ACM SIGCOMM IMC 2010*, Melbourne, Australia, November 2010.

[23] N. Ghosh, S. K. Ghosh, and S. K. Das, "SelCSP: A framework to facilitate selection of cloud service providers," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 66–79, January-March 2015.

[24] P. Pawluk, B. Simmons, M. Smit, M. Litoiu, and S. Mankovski, "Introducing STRATOS: A cloud broker service," in *Proc. of IEEE CLOUD 2012*, Honolulu, HI, USA, June 2012.

Sabrina De Capitani di Vimercati is a professor at the Computer Science Department, Università degli Studi di Milano, Italy. Her research interests are in data security and privacy. She has published more than 210 papers in journals, conference proceedings, and books. She has been a visiting researcher at SRI International, CA (USA), and George Mason University, VA (USA). She chairs the IFIP WG 11.3 on Data and Application Security and Privacy.
http://www.di.unimi.it/decapita

Sara Foresti is a professor at the Computer Science Department, Università degli Studi di Milano, Italy. Her research interests are in data security and privacy. She has published more than 90 papers in journals, conference proceedings, and books. She has been a visiting researcher at George Mason University, VA (USA). She has been serving as General chair and PC chair of several conferences. She chairs the IEEE CS Italy Chapter.
http://www.di.unimi.it/foresti

Giovanni Livraga is an assistant professor at the Computer Science Department, Università degli Studi di Milano, Italy. His research interests are in data privacy and security in emerging scenarios. His PhD thesis received the ERCIM STM WG 2015 award. He has been a visiting researcher at SAP Labs, France and George Mason University, VA (USA). He has been serving as PC member for several conferences.
http://www.di.unimi.it/livraga

Vincenzo Piuri is a professor at the Computer Science Department, Università degli Studi di Milano, Italy. His main research interests are in signal and image processing, biometrics, intelligent systems, digital and signal processing architectures. He has published more than 350 papers in journals, conference proceedings, and books. He has been named IEEE Fellow (2001) and ACM Distinguished Scientist (2008).
http://www.di.unimi.it/piuri

Pierangela Samarati is a professor at the Computer Science Department, Università degli Studi di Milano, Italy. Her main research interests are in data protection, security, and privacy. She has participated in/coordinated several projects. She has published more than 270 papers in journals, conference proceedings, and books. She has been named ACM Distinguished Scientist (2009) and IEEE Fellow (2012).
http://www.di.unimi.it/samarati