SN Computer Science (2023) 4:340
https://doi.org/10.1007/s42979-023-01707-2

ORIGINAL RESEARCH

Supporting User Protection Requirements in Cloud-Based Data Outsourcing

Sabrina De Capitani di Vimercati[1] · Sara Foresti[1] · Giovanni Livraga[1] · Pierangela Samarati[1]

Received: 25 November 2022 / Accepted: 15 January 2023

Abstract

The cloud is nowadays widely used for storing and managing data, and leveraging scalable and flexible IT infrastructures while guaranteeing continuous data and application availability from anywhere at any time. The cloud market is characterized by a rich and diversified offering that usually comes as predefined configurations (plans), which can be adopted to outsource data collections. Such plans exhibit different features and characteristics, which make different plans suitable to different scenarios. Seemingly a trivial problem, selecting a plan that responds well to the needs of a data owner is actually far from easy. In fact, the problem entails a number of challenges that need to be carefully addressed, ranging from representing and reasoning on plans' characteristics, to permitting data owners to formulate (and have enforced) expressive requirements to identify an optimal (combination of) plan(s) without requiring deep technical knowledge of the cloud technology and jargon. In this paper, we address this problem, discussing some of its main challenges, and illustrating some research directions and state-of-the-art solutions.

Keywords User protection requirements · Cloud plan selection · Data security and privacy · Requirements specification language · Cloud plan modeling

Introduction

The cloud has established itself as an effective solution for storing, managing, and sharing large data collections, as well as for executing and making available computationally-intensive applications. It permits individual users and companies to leverage the cutting-edge, fast, elastic and scalable IT infrastructures and services made available by cloud providers, without the need to own and maintain them. Moving data and applications to the cloud represents an ever increasing trend that has been constantly observable in the real world for years, and is expected to grow further in the coming years: Gartner forecasts that, in 2023, worldwide public cloud spending will grow 20.7% (reaching a total of US$591.8 billion, up from US$490.3 billion in 2022)[1] and, by 2025, enterprises will spend more on public cloud services than traditional IT solutions.[2] Fortunately, the cloud market is a diversified place, as cloud providers offer a rich panorama of solutions, usually characterized by predefined configurations (i.e., service plans, to which we refer for brevity as *plans*) that provide different features and guarantees. This makes such solutions suitable to different application scenarios, making—as an example—a plan offering strong security and privacy mechanisms more indicated for storing collections of sensitive data, and a plan guaranteeing high availability and low downtimes more indicated for sharing

This article is part of the topical collection "Advances on Information Systems Security and Privacy" guest edited by Steven Furnell and Paolo Mori.

✉ Pierangela Samarati
pierangela.samarati@unimi.it

Sabrina De Capitani di Vimercati
sabrina.decapitani@unimi.it

Sara Foresti
sara.foresti@unimi.it

Giovanni Livraga
giovanni.livraga@unimi.it

[1] Computer Science Department, Università degli Studi di Milano, via Celoria 18, 20133 Milano, Italy

[1] https://www.gartner.com/en/newsroom/press-releases/2022-10-31-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-600-billion-in-2023.

[2] https://www.gartner.com/en/newsroom/press-releases/2022-02-09-gartner-says-more-than-half-of-enterprise-it-spending.

Published online: 20 April 2023

public data. Indeed, moving data to the cloud requires their owner to entrust the cloud provider and its services for correctly managing them, responding to her needs (e.g., adequate security infrastructures or service availability levels). Resorting to the cloud raises a series of questions and concerns that need to be carefully investigated, to ensure that data owners outsourcing to the cloud their data enjoy the potential benefits that the elastic and performant cloud-based solutions can offer. These issues are multiple and diverse (e.g., [2, 15]), and range from having effective solutions for properly protecting data and applications security and privacy managing trust assumptions on the different providers, to ensuring adequate performance of the selected plans, to balancing the requested features and the economic costs charged by providers, to name a few.

A key aspect to be solved when moving to the cloud concerns the selection of a suitable (set of) plan(s) that responds well to the specific application scenario in which moving to the cloud is to be performed. The problem of selecting an optimal solution, among the diverse alternatives in the market, is complex for a multitude of reasons. Not only can different data owners have different needs, but the same owner may have different and dynamic needs for different application scenarios. A cloud plan, which may be a good fit for a certain owner in a specific scenario, may then be sub-optimal -or even detrimental- for another owner, or even for the same owner in a different scenario. The selection of a sub-optimal solution may negatively impact the adoption of the cloud. For example, outsourcing data for a mission-critical application to a cloud plan incurring frequent downtimes would be detrimental for the application itself, its owners, as well as its users. For these reasons, selecting the 'right' solution is a key requirement for ensuring more and more users can adopt, and hence benefit from, the cloud.

The goal of this paper is to present an overview of the main challenges that arise when data owners move their data to the cloud and need to identify the plan(s) that better suit their needs. We briefly illustrate these challenges and highlight recent research directions and state-of-the-art solutions that address them. The remainder of this paper is structured as follows—"Challenges in Outsourcing to the Cloud" overviews some of the main challenges to be addressed. Subsequent sections discuss research directions and state-of-the-art solutions addressing them, focusing on the modeling of cloud plans ("Modeling Cloud Plans"), on the specification of arbitrary requirements and preferences possibly using natural language and high-level abstractions ("Supporting Requirements and Preferences" and "Supporting Natural Language Desiderata"), and on the computation of optimal allocations in multicloud scenarios in full obedience of restrictions imposed by the owners of large data collections ("Supporting Requirements in Multicloud Scenarios"). Finally, "Conclusions" provides our conclusions.

Challenges in Outsourcing to the Cloud

We illustrate some of the main challenges to be investigated when data owners wish to outsource their data to the cloud. In particular, we focus our attention on the challenges connected to the problem of ensuring that the selected cloud plan(s) respond well to the needs and expectations of the data owners, possibly in obedience of protection requirements that owners may have on their data. Permitting owners to formulate, in a flexible and friendly yet rigorous way, the needs characterizing their data to be outsourced to the cloud (and defining solutions that enforce them suggesting which cloud plan, or combination thereof, better suits such needs) is central to empowering owners in maintaining control over their data. It is interesting to note that the importance of the problems connected to supporting users in cloud plan selection is recognized and addressed also by cloud providers themselves as well as by consulting and technological companies and organizations that, over the last years, have proposed models and approaches for guiding assessment of different plans, however typically according to pre-defined selection criteria and metrics (examples of such include, among others, the Cloud Decisions Tools by Gartner,[3] guidelines by Microsoft,[4] criteria from the Cloud Industry Forum[5] or, with a specific focus on security, by the Cloud Security Alliance[6]). The main challenges entailed by the problem of supporting users in selecting plans that are well aligned to their needs can be classified as follows.

- *Cloud plan modeling* ("Modeling Cloud Plans"): A first challenge connected to the problem of selecting a good plan for outsourcing concerns the definition of approaches for modeling, and subsequently evaluating, cloud plans. This requires to identify relevant features that characterize the different plans, and define metrics and techniques for assessing the plans based on their features, for example by scoring or ranking them. Early attempts in this regard have considered specific features (such as performance and costs) and proposed solutions based on, for example, benchmarking (e.g., [10, 14]). Recent lines of work have investigated the possibility of considering arbitrary features and properties that can be expressed in Service Level Agreements (SLAs) or that can be identified/measured/assessed (e.g., cloud providers along with their reputation, security infrastructures

[3] https://www.gartner.com/en/cloud-decisions.

[4] https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/choosing-a-cloud-service-provider.

[5] https://cloudindustryforum.org/8-criteria-to-ensure-you-select-the-right-cloud-service-provider/.

[6] https://cloudsecurityalliance.org/research/cloud-controls-matrix/.

and schemes, certifications) and have proposed ad-hoc evaluation approaches for assessing them against specific requirements owners may have (e.g., [8]).

- *Specification of arbitrary requirements and preferences* ("Supporting Requirements and Preferences"): A cloud plan can be more or less appealing to a data owner depending on specific protection needs for her data. For example, due to laws or regulations, an owner may want to outsource data only to cloud providers based on a specific geographical area. Plans of providers located outside that area, regardless of how performant and/or how economically convenient they may be, would then be of no interest to that specific owner for that data collection. Besides such hard protection requirements that must be satisfied, data owners may also have soft requirements that can make one plan more appealing over another. With reference to the example above, a data owner may favor, among the plans in the acceptable geographical area, those that have a certain security certification: a plan that does not have it would then be considered acceptable, but less appealing to the owner. Different owners may have different requirements, or even the same owner may have different requirements for different data collections, based on their specific needs and on the considered application scenario. A key challenge is therefore permitting owners to specify arbitrary requirements and preferences in an easy and flexible manner capturing, in an unambiguous way, the conditions that owners feel can make a plan acceptable/preferable for outsourcing.
- *Use of natural language and abstractions for requirement specification* ("Supporting Natural Language Desiderata"): Cloud plans are characterized by cutting-edge technologies, and non-technically skilled data owners may find difficult to identify and understand plan features, and express their needs based on these features. For example, SLAs can include terms, such as 'API Error', 'Data Plane', and 'Load Balancer' [18], with which owners without a technical background may not be familiar, and different cloud providers may adopt different terms to refer to a same feature, further complicating the scenario for non-skilled owners. A key challenge is then supporting *all* data owners in the specification of their needs, regardless of their technical/scientific background. It is therefore necessary to bridge the gap between the technicalities characterizing cloud plans and the data owners' expertise, supporting an easy formulation of arbitrary requirements without requiring deep technical knowledge. A promising direction is to permit owners to formulate their requirements using natural language expressions and high-level and easily accessible abstractions, so that it can be possible, for example, to require a plan that guarantees 'high security', delegating to some automated reasoning the mapping of such high-level requirement to the actual low-level characteristics of the plans.
- *Specification of requirements guiding multicloud allocations* ("Supporting Requirements in Multicloud Scenarios"): The cloud is a dynamic and evolving scenario, with new paradigms that can be beneficial to advanced applications. The multicloud paradigm concerns adopting more than one cloud plan at the same time to perform different tasks or to allocate data with different protection requirements. This permits to leverage multiple services (possibly offered by different providers) with benefits in terms of, for example, not being dependent on a single plan/provider, and leveraging, for each specific data collection, the strengths of the specific adopted plan. The multicloud paradigm is gaining high momentum at the time of writing, as testified by numbers and figures that show the vast majority of mid-to-large companies will have adopted a multicloud strategy by 2023.[7] A downside is clearly in terms of additional management overhead. Adopting a set of cloud plans requires establishing and maintaining different contracts and interacting with their providers, and requires the payment of the economic costs charged by the providers. A key challenge in this regard is connected to the specification and enforcement of requirements, encompassing both requirements modeling the protection needs of the different data collections, and global requirements governing the management of the overall data allocation (e.g., to avoid excessive data fragmentation among plans), while maintaining the overall economic costs under control and possibly minimized.

In the following sections, we discuss research directions under investigation and state-of-the-art solutions addressing the challenges illustrated above. When clear from the context, since owners of data moved to the cloud are users for the selected cloud plans, we use the terms owners and users interchangeably.

Modeling Cloud Plans

A first key challenge connected to supporting users in the cloud market concerns characterizing the providers and the plans they offer, in terms of their features and guarantees, so to enable reasoning about them. This is essential for permitting users to compare the available plans, possibly assessing their respondence to specific requirements ("Supporting Requirements and Preferences", "Supporting

[7] https://www.forbes.com/sites/bernardmarr/2022/10/17/the-top-5-cloud-computing-trends-in-2023/.

Attribute	P_1	P_2	P_3	P_4	P_5
provider	Alpha	Beta	Gamma	Delta	Delta
location	locB	locA	locB	locC	locB
encryption	AES	AES	AES	DES	3DES
certification	certA	certB	certB	certC	certB
audit	6M	12M	–	–	12M
bandwidth	25	20	15	10	25
throughput	15	10	10	5	15

Fig. 1 Abstract representation of five cloud plans

Natural Language Desiderata", and "Supporting Requirements in Multicloud Scenarios"). Acknowledging the centrality of the problem of supporting users in selecting the cloud plans that best suit their needs, different providers, companies and organizations offer support in suggesting prospective users some factors that should be taken into consideration when assessing candidates in the cloud market. Different subjects can however suggest different factors: for example, while Microsoft suggests to consider aspects related to business health and processes, administration support, technical capabilities and processes, and security practices[4] the Cloud Industry Forum suggests aspects related to certifications and standards, technologies and service roadmap, data security, data governance and business policies, service dependencies and partnerships, contracts, commercials and SLAs, reliability and performance, migration support, vendor lock in and exit planning, and business health and company profile.[5] In principle, any feature that characterizes a cloud plan can be used to model it, ranging from configuration parameters declared in the Service Level Agreements (SLAs) of plans, to arbitrary metadata and features of interest. A broad characterization of these characteristics and of how they can be derived is as follows.

- *Performance and costs*: A natural approach for characterizing cloud plans is based on their (promised) performance and charged costs. A possibility in this direction is to measure the elastic computing, persistent storage, and networking services offered by a plan and model their impact on the performance of the outsourced customer applications (e.g., [14]).
- *Objective assessments*: A second approach for characterizing cloud plans consists in leveraging standards and/or pre-defined metrics to 'quantify' a set of features of interest, such as accountability, agility, assurance of service, cost, performance, security and privacy, and usability (e.g., [1, 10]). In the context of cybersecurity, the Cloud Controls Matrix (CCM) together with its associated Consensus Assessment Initiative Questionnaire (CAIQ) by the Cloud Security Alliance (CSA) is a control framework (now at its fourth version) providing security concepts and principles to cloud providers, permitting users to assess the security risks associated with a provider [4].
- *Subjective assessments*: A third approach concerns the development of user-centric approaches, taking into consideration subjective assessments and past experiences of users (e.g., [9, 11, 16, 20]) as well as QoS values observed at the user side rather than those promised by the providers (e.g., [22]).

All the above approaches can be used to define properties of cloud plans. A cloud plan can then be described in terms of the different values it assumes for the different properties of interest (which we denote, for simplicity, as *attributes*). More precisely, a plan $\mathbf{P}$ can be formally represented as a tuple containing the values assumed by the attributes for $\mathbf{P}$. Figure 1 illustrates an example of five cloud plans $\mathbf{P}_1, \ldots, \mathbf{P}_5$, defined over a set of attributes modeling: the provider owning and offering the plan (`provider`); the geographical location of the servers of the plan (`location`); the encryption scheme adopted in the plan for protecting data at rest (`encryption`); the security certification awarded to the plan (`certification`); the frequency with which security audits are executed (`audit`); the maximum outbound bandwidth for the plan (`bandwidth`); the maximum throughput for the plan (`throughput`). Clearly, not all attributes must necessarily assume a specific value for each plan: for example, an attribute may not be relevant for a specific plan (e.g., the encryption scheme adopted for protecting data at rest when considering plans that only offer computational power) or may not be available. Special value '–' can be adopted in the plan specification to model the fact that, for a specific attribute, the value is unavailable/not relevant. As an example, plans $\mathbf{P}_3$ and $\mathbf{P}_4$ in Fig. 1 do not have a known value for attribute `audit`, meaning that the frequency of security auditing is unknown for them.

The characterization illustrated above can then be adopted to address different problems, ranging from resource allocation in the cloud (e.g., [12, 17, 21]), to the definition of approaches, possibly based on multicriteria decision making, for combining and evaluating user requirements (e.g., [3, 13]). Such a general plan modeling can support users in the formulation of both hard and soft requirements modeling their needs, possibly with the use of natural language expressions, also in multicloud scenarios. It also naturally fits scenarios characterized by the presence of a *broker* in charge of collecting user requirements, evaluating them, and assessing the degrees with which the different plans respond to them [19]. In the next sections, we illustrate in more details such problems, along with possible solutions.

Supporting Requirements and Preferences

A cloud plan can be more or less suitable for a user depending on how well it responds to the specific requirements of the user and of her data. A key challenge in supporting users to move their data to the cloud concerns therefore granting users the possibility of formulating expressive and arbitrary requirements, modeling their needs and preferences, in an easy and friendly—yet unambiguous—way. In this section, we illustrate a possible approach for supporting users in formulating arbitrary requirements and preferences through a flexible and expressive, yet easy to use, specification language [7]. We illustrate the rationale and main building blocks ("Rationale and Building Blocks"), and how requirements and preferences can be specified ("Specification") and assessed ("Assessment").

Rationale and Building Blocks

The two main concepts of *requirements* and *preferences* define hard and soft constraints that make, respectively, a plan *acceptable* (i.e., satisfying the hard requirements) and *preferable* (according to the degree with which it satisfies the soft preferences). The key idea behind the definition of such requirements and preferences is the specification of *conditions* over the values of the attributes characterizing the plans, identifying values that are acceptable or unacceptable (requirements) and that are to be preferred over other ones (preferences). As will be illustrated in the remainder of this section, preferences can also take into consideration the relative importance that the user assigns to the different attributes.

As for the specification of requirements, the main building block –upon which an easy yet flexible language ("Specification") is built– is the concept of *attribute term*. An attribute term t over an attribute a of a cloud plan permits to evaluate whether the value assumed by a for a plan belongs (or does not belong) to a given set of values. More precisely, a *positive* attribute term t of the form 'a IN $\{v_i, \ldots, v_j\}$' is satisfied if a has a value in $\{v_i, \ldots, v_j\}$, while a negative term t of the form 'a NOT IN $\{v_i, \ldots, v_j\}$' is satisfied if a does not have a value in $\{v_i, \ldots, v_j\}$. For example, with reference to the attributes in Fig. 1, positive term '`provider` IN {Alpha, Beta, Gamma}' is satisfied for a plan if its provider is Alpha, Beta, or Gamma. Negative term '`encryption` NOT IN {DES}' is satisfied for a plan if it does not adopt DES as an encryption scheme. In the following, for readability, we use notation $a(v_i, \ldots, v_j)$ ($\neg\, a(v_i, \ldots, v_j)$, respectively) as a shorthand for referring to positive (negative, respectively) attribute term a IN $\{v_i, \ldots, v_j\}$ (a NOT IN $\{v_i, \ldots, v_j\}$, respectively). Given an attribute a, its acceptable values are those that satisfy the hard requirements specified by the user.

As for the specification of preferences, the main building block is the concept of *preferable value*. Given the acceptable values for an attribute a, the user can specify her preferences that make one value preferable to another. For example, considering the values assumed by the plans in Fig. 1 for attribute `provider`, assume that values Alpha, Beta, and Gamma satisfy all the requirements specified by the user. On these values, the user can specify preferences modeling which values are preferable to which other values, for example stating that Alpha is preferable to Beta, which is in turn preferable to Gamma ("Specification").

Specification

We now illustrate how requirements and preferences can be specified by users.

Requirements: Requirements can be distinguished in *base* and *complex* requirements. A base requirement corresponds to an attribute term, be it positive (a IN $\{v_i, \ldots, v_j\}$) or negative (a NOT IN $\{v_i, \ldots, v_j\}$). Complex requirements, on the other hand, permit to capture and model more articulate needs, such as alternatives or conditional requirements among attribute terms. More precisely, the specification language permits to express the following requirements.

- A *base requirement* is of the form $r = t$, with t an attribute term. It restricts the values that can be assumed, for the attribute over which t is defined, by a plan to be considered acceptable. Requirements r_1 and r_2 in Fig. 2

$$
\begin{aligned}
r_1 &: \texttt{provider}(\text{Alpha}, \text{Beta}, \text{Gamma})\\
r_2 &: \neg\texttt{encryption}(\text{DES}, -)\\
r_3 &: \text{ANY}(\{\texttt{certification}(\text{certA}, \text{certB}, \text{certC}), \texttt{audit}(6\text{M}, 12\text{M})\})\\
r_4 &: \text{ALL}(\{\neg\texttt{throughput}(5, -), \texttt{bandwidth}(15,\ 20,\ 25), \neg\texttt{certification}(-), \neg\texttt{location}(-)\})\\
r_5 &: \text{IF ALL}(\{\texttt{encryption}(3\text{DES})\})\ \text{THEN ANY}(\{\texttt{certification}(\text{certA}), \texttt{audit}(6\text{M})\})\\
r_6 &: \text{FORBIDDEN}(\{\texttt{certification}(\text{certC}), \texttt{audit}(-)\})\\
r_7 &: \text{AT\_LEAST}(2, \{\texttt{location}(\text{locA},\text{locB}), \texttt{encr}(\text{AES}), \texttt{audit}(6\text{M},\ 12\text{M})\})\\
r_8 &: \text{AT\_MOST}(2, \{\texttt{audit}(-), \texttt{location}(\text{locC}), \texttt{provider}(\text{Gamma})\})
\end{aligned}
$$

Fig. 2 An example of a set of requirements for the plans in Fig. 1

are two examples of base requirements for the plans in Fig. 1, stating that the plans considered acceptable are only plans with provider Alpha, Beta, or Gamma (r_1), and which guarantee data encryption with a scheme that is different from DES, and has been declared (as modeled by the inclusion of the special value '−' in the negative term in r_2).

- An ANY *requirement* is of the form $r = \text{ANY}(\{t_1, \ldots, t_n\})$, with $\{t_1, \ldots, t_n\}$ a set of attribute terms. It models *alternatives* among attribute terms, and requires that *at least one* among $t_1, \ldots, t_n$ be satisfied by a plan to be considered acceptable. For example, requirement r_3 in Fig. 2 requires plans to have a security certification certA, certB, or certC, or to be audited every 6 or 12 months.
- An ALL *requirement* is of the form $r = \text{ALL}(\{t_1, \ldots, t_n\})$, with $\{t_1, \ldots, t_n\}$ a set of attribute terms. It requires that *all* the attribute terms $t_1, \ldots, t_n$ be satisfied by a plan to be considered acceptable. For example, requirement r_4 in Fig. 2 requires plans to ensure a (specified) throughput different from 5, a bandwidth equal to 15, 20, or 25, and to specify an explicit value for the security certification and for the server location.
- An IF-THEN *requirement* is of the form $\text{IF ALL}(\{t_1, \ldots, t_k\})$ $\text{THEN ANY}(\{t_{k+1}, \ldots, t_n\})$, with $\{t_1, \ldots, t_n\}$ a set of attribute terms. It models *conditional* requirements, and requires that if all attribute terms $t_1, \ldots, t_k$ appearing in the premise are satisfied by a plan, then—to be considered acceptable—at least one among terms $t_{k+1}, \ldots, t_n$ in the consequence must also be satisfied. For example, requirement r_5 in Fig. 2 requires plans that encrypt data with 3DES to have a security certification certA, or to be audited every 6 months.
- A FORBIDDEN *requirement* is of the form $\text{FORBIDDEN}(\{t_1, \ldots, t_n\})$, with $\{t_1, \ldots, t_n\}$ a set of attribute terms. It models forbidden configurations and requires that at least one among $t_1, \ldots, t_n$ be not satisfied by a plan to be considered acceptable. For example, requirement r_6 in Fig. 2 requires plans *not* to have a security certification certC *and* an unspecified value for the auditing frequency.
- An AT_LEAST *requirement* is of the form $\text{AT\_LEAST}(m, \{t_1, \ldots, t_n\})$, with $\{t_1, \ldots, t_n\}$ a set of attribute terms, and $m \leq n$ an integer value. It requires that at least m of the attribute terms appearing in the requirement be satisfied by a plan to be considered acceptable. For example, requirement r_7 in Fig. 2 requires plans to satisfy at least two among (i) having servers located in locA or locB; (ii) encrypting data with AES; and (iii) being audited every 6 or 12 months.
- An AT_MOST *requirement* is of the form $\text{AT\_MOST}(m, \{t_1, \ldots, t_n\})$, with $\{t_1, \ldots, t_n\}$ a set of attribute terms, and $m \leq n$ an integer value. Similarly to AT_LEAST requirements, it requires that at most m of the attribute terms appearing in the requirement be satisfied by a plan to be considered acceptable. For example, requirement r_8 in Fig. 2 requires plans to satisfy at most two among (i) not having a specified value for the auditing frequency; (ii) having servers located in locC; and (iii) being offered by provider Gamma.

We note that the different forms of requirements supported by the language permit a user to formulate her needs in different ways. For example, an ALL requirement over a set of n attribute terms can also be formulated as a set of n base requirements, one for each attribute term in the ALL requirement. To illustrate, requirement r_4 in Fig. 2 could also have been expressed with a set of four base requirements restricting the values assumed by attributes `throughput`, `bandwidth`, `certification`, and `location` as per the corresponding attribute terms in r_4. An $\text{AT\_MOST}(m, (t_1, \ldots, t_n))$ requirement such that $m = n$ corresponds to an $\text{ALL}(t_1, \ldots, t_n)$ requirement covering all the involved attribute terms. A base requirement over an attribute term t may be formulated as an ALL (t) requirement, or even as an ANY (t) requirement. The possibility to formulate requirements in different manners demonstrates that the language provides for great flexibility and user-friendliness, permitting users to capture and formulate their needs freely, in what they feel is the most convenient way.

Preferences: Preferences can be specified on attribute values (modeling a preference relationship among values), and on attribute themselves (modeling the relative importance given to the attributes).

- Preferences on *attribute values* specify that some values are preferable to other ones (clearly, preferences apply only to acceptable attribute values). The specification of such preferences can rely on different approaches. An intuitive and user-friendly approach is based on the definition of a total order relationship among values (or, more generally, among sets of equally acceptable values). A graphical representation (e.g., as a hierarchy where values in higher positions are preferable to values in lower positions) can further help users in visualizing and specifying these preferences. Figure 3 illustrates an example of hierarchies over attributes representing the preferences for the attributes of the plans in Fig. 1. For example, the preferences specified for the values of attribute `provider` state that provider Alpha is preferable to Beta, in turn preferable to Gamma. Note that these are the acceptable values for `provider` (i.e., they satisfy the requirements in Fig. 2). Note also that attribute `audit` is the only attribute for which special value '−' is in the preference hierarchy, since it is the only attribute for which the requirements in Fig. 2 do not exclude this possibility (for every other attribute a, this value is ruled

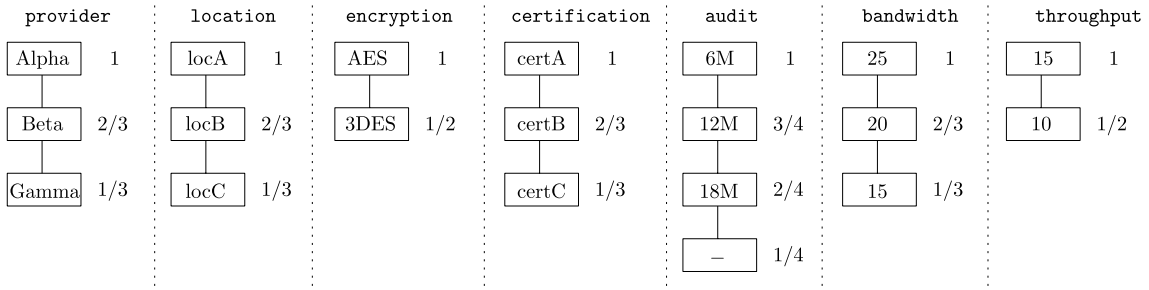

Fig. 3 An example of preferences on attribute values for the plans in Fig. 1

out by either a negative attribute term $\neg a(-)$ or by a positive attribute term $a(v_1, \ldots, v_k)$ with $- \notin \{v_1, \ldots, v_k\}$).

- Preferences on *attributes* specify the perceived relative importance of different attributes. Intuitively, this can be specified by assigning weights to the different attributes, with higher weights corresponding to higher importance perceived by the user. For example, a user considering the throughput more important than other properties of cloud plans may assign a higher weight to attribute `throughput`, and lower weights to the remaining attributes.

Assessment

Based on the evaluation of the requirements specified by the user, a plan can be classified in acceptable/unacceptable, depending on whether the plan satisfies such requirements. Acceptable plans can then be ranked according to the extent to which they satisfy user preferences.

Acceptable plans: As illustrated in "Rationale and Building Blocks", requirements model the conditions that a plan must satisfy to be considered acceptable. In particular, given a set of requirements and a set of available plans, only those plans that satisfy *all* the requirements formulated by the user can be considered acceptable to the user. To this end, a Boolean interpretation of the requirements and of the attribute values that characterize cloud plans can be adopted, with the added benefit that such a Boolean interpretation makes it possible to identify whether there are conflicting requirements that would inevitably result in an empty set of acceptable plans. In a nutshell, the attribute terms appearing in the set of requirements are interpreted as Boolean variables. Each plan is interpreted as a truth assignment to attribute terms: given a term t over an attribute a, a plan evaluates 1 for t if the value it assumes for a satisfies t. For example, consider the plans in Fig. 1, and an attribute term $t =$ `provider`(Alpha,Beta,Gamma). Term t evaluates to 1 according to plan $\mathbf{P}_1$, since the value (Alpha) assumed by `provider` satisfies t. On the contrary, t evaluates to 0 according to plan $\mathbf{P}_4$, since value Delta does not satisfy t. Requirements are then interpreted as Boolean formulas over the Boolean variables modeling attribute terms, and a plan satisfies a requirement if it satisfies the Boolean formula modeling it. The Boolean interpretation of the different kinds of requirements clearly depends on their formulation [7]. For example, the Boolean interpretation of requirement ANY($\{t_1, \ldots, t_n\}$) corresponds to disjunction $b_1 \vee \cdots \vee b_n$, with b_i the Boolean variable modeling t_i. Consider requirement r_3 in Fig. 2, which includes attribute terms `certification`(certA,certB,certC) and `audit`(6 M, 12 M), and plan $\mathbf{P}_3$ in Fig. 1. The truth value assigned by $\mathbf{P}_3$ to `certification`(certA,certB,certC) is 1, while the value assigned to `audit`(6 M,12 M) is 0 as the plan does not guarantee a security audit every 6 or 12 months. Since $1 \vee 0 = 1$, r_3 is satisfied by $\mathbf{P}_3$. Given a set of requirements, a plan is acceptable if it satisfies all Boolean formulas resulting from the translation of the requirements. With reference to the plans in Fig. 1 and the requirements in Fig. 2, plans $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ are acceptable, while $\mathbf{P}_4$ and $\mathbf{P}_5$ are not acceptable as they do not satisfy, respectively, requirements r_1, r_2, r_4, r_6, and r_7, and requirements r_1 and r_5.

Preferred plans: Once acceptable plans have been identified, they can be ranked according to the preferences set by the user. Different solutions can be adopted to rank plans. A first approach is based on the classical notion of Pareto dominance, according to which a plan $\mathbf{P}_a$ is ranked higher than (i.e., it is preferable to) plan $\mathbf{P}_b$ if, for each attribute characterizing them, $\mathbf{P}_a$ has a value that is preferred or equal to that of $\mathbf{P}_b$ and, for at least one attribute, a value that is preferred to that of $\mathbf{P}_b$. Considering the acceptable plans $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ in Fig. 1, for example, $\mathbf{P}_1$ is preferable to $\mathbf{P}_3$ (they assume equal values for attributes `location` and `encryption`, and for all other attributes the values in $\mathbf{P}_1$ are preferable to those in $\mathbf{P}_3$). Similarly, $\mathbf{P}_2$ is preferable to $\mathbf{P}_3$ (they assume the same values for `encryption`, `certification`, and `throughput`, and for all other attributes the values in $\mathbf{P}_2$ are preferable to those in $\mathbf{P}_3$). However, nothing can be said for the relationship between $\mathbf{P}_1$ and $\mathbf{P}_2$: for example, for `provider` the value in $\mathbf{P}_1$ is preferable to that in $\mathbf{P}_2$ but, on the contrary, for `location` the value

in $\mathbf{P}_2$ is preferable to that in $\mathbf{P}_1$. Adopting a Pareto-based approach, therefore, plans $\mathbf{P}_1$ and $\mathbf{P}_2$ are incomparable. Figure 4a illustrates the Pareto-based ranking for the acceptable plans in Fig. 1.

A different approach, which has the advantage of enabling a total ranking among plans (and can also accommodate the preferences on the attributes), is based on the computation of a distance between each acceptable plan and an *ideal* plan. The ideal plan is a (possibly non-existing) plan which assumes, for all attributes, the preferred value. Intuitively, the closer a plan is to such ideal plan, the more it satisfies user preferences. The idea is then to consider plans as points in a m-dimensional space, with m the number of attributes. The coordinate for an attribute in a plan is obtained by associating the value of the attribute with a number (score) reflecting its position in the preference hierarchy. With reference to the preferences of our example, Fig. 3 reports, for each attribute a and each value v, the score associated with v, computed as follows: given k the number of partitions in which acceptable values are grouped (i.e., the number of elements in the hierarchy representing the preferences for the values of a), and starting from the least preferred value for which the score is $1/k$, at each step in the hierarchy the score of the associated values increases of $1/k$. Clearly, the top element will have score 1. For example, attribute `encryption` has $k = 2$ partitions (one for AES and one for 3DES). The score for 3DES, being the least preferred value, is 1/2, and the score for AES is therefore $1/2 + 1/2 = 1$. Given a plan $\mathbf{P}$, its coordinates are then the scores of its attribute values in the preference hierarchies. For example, consider the preferences in Fig. 3: plan $\mathbf{P}_1$ in Fig. 1 will be represented as vector [1 2/3 1 1 1 1 1]. Indeed, the ideal plan will be represented by a point having value 1 for each coordinate (i.e., for each attribute). With this spatial representation of plans, the distance between a plan and the ideal plan can be simply assessed through the evaluation of the Euclidean distance between the points representing them. The Euclidean distance between two points (one representing the plan under assessment with the coordinates illustrated above, and the other representing the ideal plan with coordinate 1 for each attribute) is simply given by the square root of the sums of the squares of the difference between the coordinates, coordinate-wise (i.e., attribute-wise). To illustrate, consider—for simplicity—two points with coordinates [1 1 1] and [1 2/3 3/4]: their Euclidean distance is simply computed as $\sqrt{(1-1)^2 + (1-2/3)^2 + (1-3/4)} = \sqrt{(0)^2 + (1/3)^2 + (1/4)^2} = 0.42$. Figure 4 graphically illustrates the ranking induced over the acceptable plans of our running example (where each plan reports, besides its attribute values, also the related scores, modeling the coordinates in the space), where the Euclidean distance from the ideal plan is reported in boldface on the right-hand side of each plan.

It is interesting to note that such distance-based ranking can also easily enforce preferences on attributes, by simply considering the weights associated with attributes to scale the corresponding dimensions (i.e., weight the corresponding coordinates) accordingly.

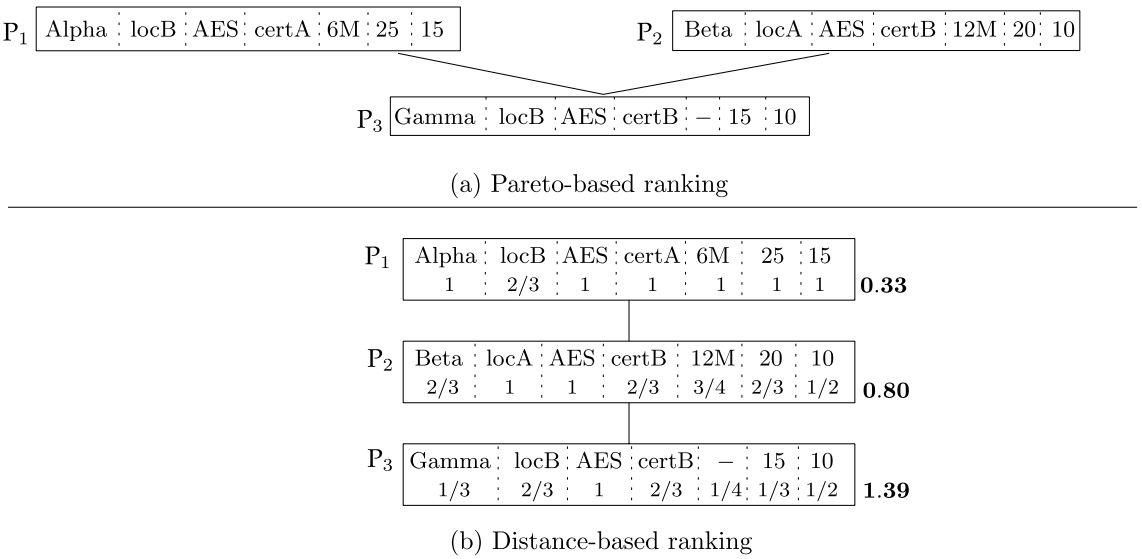

Fig. 4 Rankings of plans $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ in Fig. 1 according to the preferences in Fig. 3

Supporting Natural Language Desiderata

The approach illustrated in "Supporting Requirements and Preferences", while effectively supporting users in specifying arbitrary requirements and preferences, requires users to reason—and hence understand—low-level parameters characterizing cloud plans, for identifying acceptable values to be used in the specification language. In this section, we illustrate a possible approach to support users in the adoption of natural language expressions and high-level concepts in the formulation of their desiderata [5]. We discuss the rationale and main building blocks ("Rationale and Building Blocks"), and how desiderata can be specified ("Specification") and assessed ("Assessment").

Rationale and Building Blocks

To support users in easily formulating their desiderata, a possible approach builds on two main building blocks: *abstract parameters* and *abstract concepts*.

- *Abstract parameters*: Abstract parameters model the attributes that characterize cloud plans (e.g., those on which the requirements and preferences illustrated in "Supporting Requirements and Preferences" are formulated) and permit to formulate requirements using natural language expressions (i.e., linguistic labels such as high or low). To illustrate, consider the plans and attributes in Fig. 1: since attribute `bandwidth` is used to model and characterize plans, an abstract parameter for `bandwidth` will be defined. Rather than specifying requirements directly on the `bandwidth` (crisp) values that make a plan acceptable (or more or less preferable) as illustrated in "Supporting Requirements and Preferences", its abstract interpretation permits users to adopt in their requirements natural language expressions, stating, for example, that they are interested in plans with high `bandwidth`. In this way, users can more easily specify requirements on the characteristics of the different plans, without using crisp values of the attributes modeling them. Abstract parameters can then be used by users whenever they are unsure about the specific crisp value they are requesting for an attribute, but are aware of the attribute semantics and are able to linguistically specify, with periphrases or adjectives, a requirement for it.
- *Abstract concepts*: Abstract parameters already provide user-friendliness in terms of the possibility of adopting natural language expressions, but map directly to the specific attributes of plans and hence still require a certain degree of understanding. Abstract concepts represent higher-level abstractions of (sets of) attributes, with a semantics that can be more easily understandable also to users who may not have sufficient technical background to fully understand the semantics of low-level attributes. `Performance` is an example of an abstract concept, representing an intuitive high-level abstraction of a series of attributes (e.g., `bandwidth` and `throughput` in our running example). Like for abstract parameters, also abstract concepts can be used to specify requirements with natural language expressions. For example, users can require a plan that exhibits high `performance`.

Both abstract parameters and concepts require the definition of a set of linguistic labels, which are used in requirement specification ("Specification") and evaluation ("Assessment"). Such linguistic labels are associated with abstract parameters and concepts, and can be arbitrarily defined, with the aid of domain experts, possibly by the users themselves to quantify parameters and concepts.

The relationship existing between abstract parameters and abstract concepts is modeled through a set of *implication rules*, which govern the implications between a combination of linguistic values for a set of abstract parameters and a linguistic value for an abstract concept. Intuitively, each label defined for an abstract concept should be associated with an implication rule, providing for a complete and clear interpretation of abstract concepts. To illustrate, consider abstract concept `performance` as an abstraction over abstract parameters `throughput` and `bandwidth`, and assume it is associated with three linguistic labels low, med and high. Figure 5a illustrates an example of implication rules defining abstract concept `performance`. These rules state that: (i)

$$\langle \texttt{throughput} = \mathsf{high} \rangle \wedge \langle \texttt{bandwidth} = \mathsf{large} \rangle \implies \langle \texttt{performance} = \mathsf{high} \rangle$$
$$\langle \texttt{throughput} = \mathsf{med} \rangle \implies \langle \texttt{performance} = \mathsf{med} \rangle$$
$$\langle \texttt{throughput} = \mathsf{low} \rangle \vee \langle \texttt{bandwidth} = \mathsf{small} \rangle \implies \langle \texttt{performance} = \mathsf{low} \rangle$$

(a) Implication rules

$$\langle \texttt{performance} = \mathsf{high} \rangle \implies \langle \texttt{satisfaction} = \mathsf{high} \rangle$$
$$\langle \texttt{audit} = \mathsf{med} \rangle \implies \langle \texttt{satisfaction} = \mathsf{med} \rangle$$
$$\langle \texttt{performance} = \mathsf{low} \rangle \vee \langle \texttt{security} = \mathsf{low} \rangle \implies \langle \texttt{satisfaction} = \mathsf{low} \rangle$$

(b) Desiderata

Fig. 5 An example of implication rules for the definition of abstract concept `performance` (**a**) and of user desiderata (**b**)

if a plan guarantees high bandwidth and high throughput, then its performance is high; (ii) if a plan guarantees medium bandwidth, then its performance is medium; and (iii) if a plan guarantees low bandwidth or low throughput, then its performance is low.

Specification

Abstract parameters and abstract concepts can be used by users, as a sort of easy-to-use vocabulary, for formulating their requirements. The idea is to support users in specifying their degree of satisfaction given by different combinations of conditions on abstract parameters and/or abstract concepts. The intuition is to permit users to specify a set of desiderata, stating—for example—that a plan providing 'high' `security` and 'high' `performance` is highly satisfactory, while a plan providing 'low' `security` or 'low' `performance` is less satisfactory. In other words, the specification of user desiderata corresponds to a set of rules that specify how a certain combination of linguistic values for abstract parameters and/or abstract concepts is satisfactory. More concretely, user desiderata can be formulated as a set of if-then rules, similarly to the implication rules governing the definition of abstract concepts, with expressions over abstract parameters and abstract concepts (and their linguistic labels) in the premise, and a level (again expressed with a linguistic label) for an ad hoc variable `satisfaction` in the consequence, modeling the overall user satisfaction. Figure 5b illustrates an example of user desiderata, defining plans with high performance as highly satisfactory, plans with a medium frequency of security auditing (with `audit` an abstract parameter) as satisfactory to a medium extent, and plans with low performance or low security (with `security` an abstract concept similarly to `performance`) as satisfactory to a low extent.

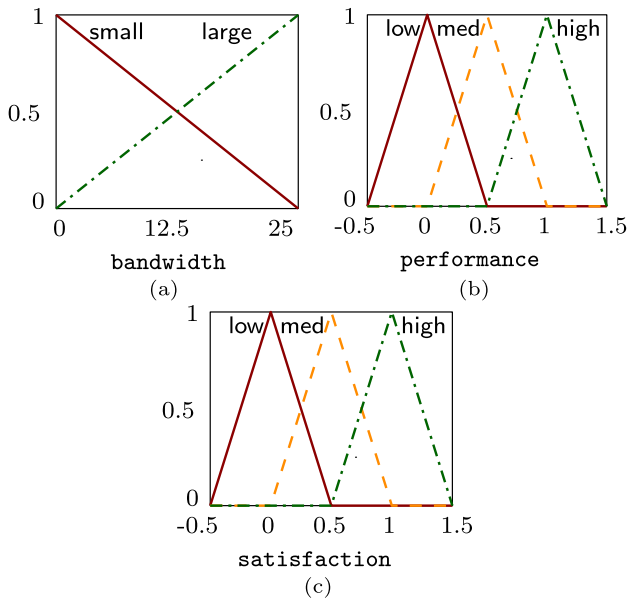

Fig. 6 An example of membership functions for abstract parameter `bandwidth` (**a**), abstract concept `performance` (**b**), and the ad-hoc variable `satisfaction` (**c**). The degree μ of membership is on the y-axis

Assessment

We now illustrate how linguistic values can be mapped to the attribute values characterizing cloud plans, and how the abstract concepts and the user desiderata can be quantified and assessed.

From crisp values to linguistic values: Considering that desiderata are expressed with linguistic values, and possibly on abstract concepts, it is necessary to reason on how such desiderata can map to the actual (crisp) parameters characterizing cloud plans. To map linguistic labels to the actual crisp parameters, an intuitive approach could associate predefined sets of crisp values to the different linguistic labels. For example, assume that the domain of crisp values that can be assumed by parameter `bandwidth` is the continuous interval [0, 25]Gb/s, and that two labels small and high are to be mapped to it. The domain could be partitioned in two disjoint intervals, with one label each. For example, small could be associated with values in the [0, 10)Gb/s interval, and high with values in the [10, 25]Gb/s interval. This approach would certainly do, but it would create sharp boundaries between pairs of adjacent values that are associated to different labels. With reference to the `bandwidth` domain partitioning, a sharp boundary is created around value 10 Gb/s: value 9.999Gb/s would be considered small, while the—almost equal—value 10Gb/s would be considered large.

A less strict interpretation where the same crisp value could be mapped, with different degrees, to different linguistic labels would be more in line with the uncertainty and imprecision of the natural language expressions used in desiderata. To this end, a fuzzy-based modeling can be employed, interpreting abstract parameters and concepts as *fuzzy variables*, and the linguistic labels (adopted in users' desiderata as well as in implication rules) as *fuzzy sets*. In a nutshell, a fuzzy variable is a variable that can assume crisp as well as linguistic values. A fuzzy set is a set in which, in contrast to the classical set theory where an element either belongs or does not belong to a set, elements have *degree of memberships*. The degree μ with which an element belongs to a fuzzy set is regulated by the definition of a *membership function*. Membership functions can be defined with different shapes (e.g., triangular, trapezoidal, sigmoidal) and permit a gradual assessment of the membership of values to fuzzy sets. Assuming that the labels (i.e., the fuzzy sets) that can be associated with `bandwidth` are small and large, Fig. 6a illustrates an example of two membership functions regulating the membership of crisp `bandwidth`

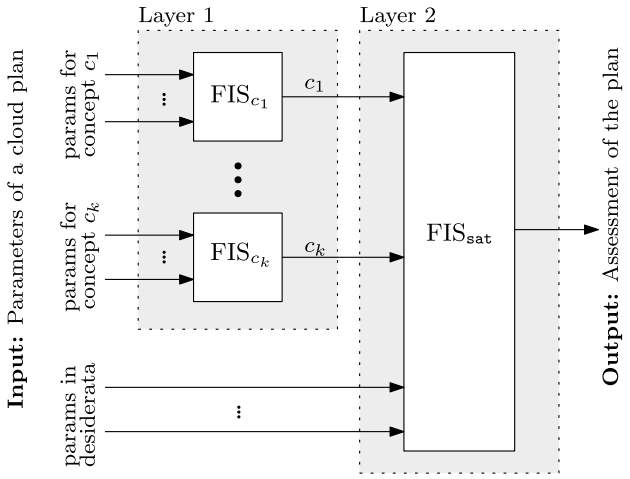

Fig. 7 Two-layers architecture for evaluating user desiderata

values to the fuzzy sets representing linguistic labels small and large. The functions operate over the domains of crisp values that can be assumed by `bandwidth`, and dictate how a crisp value 'belongs' to the set: in other words, they permit to assess how much a crisp value is 'representative' of the linguistic label interpreted as the fuzzy set. It is interesting to note that the same value can belong, possibly with different membership degrees, to different fuzzy sets (and hence be representative, up to different degrees, of different linguistic labels). For instance, consider a crisp `bandwidth` value v and the membership functions in Fig. 6a: the more v grows, the more it belongs (i.e., the higher its degree μ of membership) to the fuzzy set large, and the less it belongs (i.e., the lower its degree μ of membership) to the fuzzy set small.

Membership functions then establish a correspondence between the crisp values of the low-level attributes and the linguistic labels of the corresponding abstract parameters. The same approach can also be used to reason on the linguistic values assumed by abstract concepts and by the ad-hoc variable `satisfaction`. Being however abstract concepts and `satisfaction` arbitrary abstractions, they do not have a naturally associated domain of crisp values, which is however needed to quantify how much a plan is compliant with such abstractions (e.g., how much a plan guarantees `performance` and how much it is satisfactory). Any domain of crisp values could be defined, such as in the continuous interval [0, 1]. Figure 6b–c illustrate examples of membership functions for the abstract concept `performance` and for the ad hoc variable `satisfaction`. Note that the domains for the membership functions have been defined as $[-0.5, 1.5]$ to guarantee that the centroid of any area defined by a membership degree over the different membership functions covers the whole interval [0, 1].

Abstract concepts and desiderata quantification: Fuzzy logic, besides providing a means for interpreting linguistic labels, permits also the evaluation of how much a cloud plan satisfies user desiderata, using fuzzy inferences. A fuzzy inference takes as input a set of crisp values, interprets such values with a fuzzy modeling as illustrated above (i.e., evaluating them against membership functions characterizing the fuzzy sets representing the linguistic labels), evaluates a set of if–then rules based on such fuzzy modeling obtaining a (fuzzy) result, transforms such fuzzy result into a crisp value, and returns it. Our user desiderata can then be fully evaluated with fuzzy inferences, with a two-layers approach.

- The first layer of inferences is in charge of 'quantifying' values for the abstract concepts used in the desiderata. It does so by mapping the crisp values of the attributes characterizing cloud plans to their associated abstract concepts appearing in the desiderata. The quantification of such concepts leverages the implication rules governing their definition.
- The second layer of inferences is in charge of 'quantifying' the `satisfaction`. It reasons over the abstract parameters and abstract concepts used in the desiderata and, for the concepts, it leverages the quantification returned by the first layer.

The first layer adopts a set of Fuzzy Inference Systems (FISs), with a FIS for each abstract concept appearing in the desiderata (to quantify it). The second layer adopts a FIS, quantifying user's satisfaction based on the evaluation of her desiderata. Figure 7 graphically illustrates such an architecture.To illustrate the working of the fuzzy inference process, consider the first desideratum in Fig. 5b "$\langle \texttt{performance} = \textsf{high} \rangle \implies \langle \texttt{satisfaction} = \textsf{high} \rangle$". According to the inference rules in Fig. 5a, abstract concept `performance` is governed based on abstract parameters `throughput` and `bandwidth`. Plans (e.g., see Fig. 1) are characterized by the values they assume for such attributes, and not directly by their `performance` (abstract concept), which should then be quantified. A first inference process to quantify the performance of plans would then operate as follows: (i) the crisp values for `throughput` and `bandwidth` (e.g., Fig. 1) are taken as input; (ii) the inference rules defining concept `performance` (e.g., Fig. 5a) are translated into if-then rules and applied, as rulebase, to the (fuzzified) input values; (iii) a quantification of `performance` based on `bandwidth` and `throughput` is returned. With such assessment for `performance`, a second inference process would then be executed to quantify the satisfaction for the user formulating the desiderata. This inference process operates like the first one, with the difference that it takes as input the quantification of `performance` (and of the other abstract concepts appearing in the desiderata) and operates

on it (them) considering, as rulebase, the user's desiderata, translated in if-then rules. If the desiderata include (also) conditions on abstract parameters (e.g., the second rule in Fig. 5b, operating on abstract parameter `audit`), these are directly evaluated in the second layer since, for these, no abstract concept quantification is needed from the first layer. The second layer reasons on a rulebase including the desiderata, and returns a quantification for `satisfaction`, hence assessing the degree with which a plan satisfies the user desiderata.

Supporting Requirements in Multicloud Scenarios

The approaches illustrated in the previous sections permit to empower users to specify requirements and preferences, which are then evaluated against a set of cloud plans to assess how well each plan responds to such needs. When resorting to the cloud for the storage and management of large and heterogeneous data collections, the scenario can become more complicated, as different datasets may have different (and even possibly contrasting) needs, which may be difficult—if at all possible—to be formulated as a single set of requirements/preferences. The selection of a single cloud plan may, in these scenarios, not be an optimal strategy: a single plan satisfying the diverse requirements of all datasets may not exist, or may be too costly (e.g., fulfilling the requirements of the most critical dataset(s)). In these scenarios, a more promising solution could be that of selecting a *set of cloud plans*, adopting the *multi-cloud* paradigm: the joint adoption of multiple cloud plans/services to optimize the satisfaction of a set of disparate needs ("Challenges in Outsourcing to the Cloud"). A key requirement in these scenarios is to empower users who wish to outsource a heterogeneous data collection with the possibility of specifying arbitrary requirements that can guide the allocation of the different datasets to different plans. Such an approach should be carefully designed, as the adoption of multiple plans can cause an increase in the management overhead and in the economic costs to be sustained for establishing multiple contracts with different providers. Given a collection of datasets and a set of candidate cloud plans, the goal is therefore that of finding an *optimal allocation* of datasets to (a subset of) plans, so to ensure that the specific needs of each dataset be properly satisfied by the plan selected for its outsourcing, trying to balance the satisfaction of requirements and the economic costs entailed by outsourcing.

In this section, we illustrate a possible approach for supporting owners of collections of a datasets, with diverse and possibly contrasting requirements for different datasets, to determine an optimal allocation of such datasets to cloud plans [6]. We illustrate the rationale and main building blocks ("Rationale and Building Blocks"), and how requirements can be specified ("Specification") and assessed ("Assessment").

Rationale and Building Blocks

Considering the peculiarities of the problem of allocating different datasets to a set of plans, users should be supported in the specification (and enforcement) of two main kinds of requirements:

- *Protection requirements*, which permit to easily model the specific protection needs of the different datasets in the collection to be outsourced; and
- *Global requirements*, which are not related to single datasets but model additional restrictions (e.g., on the number of plans to be adopted in the allocation) that users may wish to impose on how the datasets are allocated to the plans.

In this section, we illustrate the main building blocks on which protection requirements for datasets can be formulated. Global requirements, being more immediate in their formulation, will be covered in "Specification".

A possible approach for supporting users in formulating the protection requirements of their datasets builds on the concept of *security property*, a high-level concept that can be used to easily capture the protection needs of the different datasets. For simplicity, classical properties can include Confidentiality, Integrity, and Availability, but different properties can also be considered. Such properties can be associated with *domains of labels*, used to 'quantify' them. To illustrate, consider two properties **C**onfidentiality and **A**vailability, where **C**onfidentiality is associated with two labels modeling high confidentiality (HC) and low confidentiality (LC) and, similarly, **A**vailability is associated with two labels modeling high availability (HA) and low availability (LA). Intuitively, the labels associated with a property p are totally ordered through a total order relationship $\succ^p$, with higher labels representing a larger quantification. For example, with respect to property **C**onfidentiality, it holds that $\mathsf{HC} \succ^{\mathbf{C}} \mathsf{LC}$, meaning that high confidentiality HC dominates low confidentiality LC. The mapping between labels and plans is based on the definition of expressions (e.g., Boolean formulas, or more in general modeling some form of (fuzzy) reasoning) over the attributes characterizing the cloud plans. A default label $\bot$, common to all properties, can be used when a property is of no interest. Figure 8 illustrates an example of labels HC, LC, HA, and LA (high and low **C**onfidentiality and **A**vailability). These expressions (as well as the properties themselves) can be defined with the support of domain experts, or could be specified by skilled users. In

the example, HA corresponds to requiring encryption with AES and certA security certification.

For the definition (and satisfaction) of requirements, *security classes* are defined as vectors of labels, with a label for each relevant property. For example, [HC, HA] is a security class defined over the two properties (**C** and **A**) in Fig. 8. Since the labels of each property are totally ordered, the security classes combining the labels of different properties are partially ordered, and thus form a *lattice* of security classes. Security classes are characterized by a dominance relationship $\succeq$ according to which a security class c_1 dominates another class c_2 (denoted $c_1 \succeq c_2$) iff the dominance relationship holds for each of its components (i.e., labels in the tuple). For instance, [HC, HA]$\succeq$[LC, LA], since HC$>^{\mathbf{C}}$LC and HA$>^{\mathbf{A}}$LA. Figure 9 illustrates the lattice defined over the security classes induced by the properties and labels in Fig. 8.

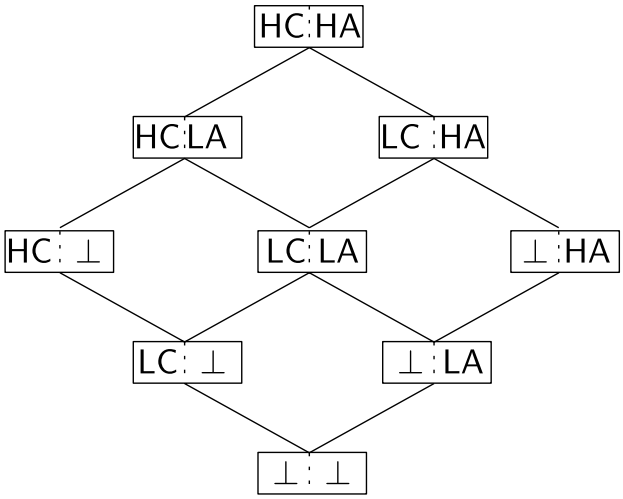

Fig. 9 Security lattice induced by the security properties and labels in Fig. 8

Specification

We now illustrate how a user wishing to allocate a collection of datasets to a set of plans can specify protection requirements for the datasets, and global requirements guiding the overall allocation.

Protection requirements: Security classes are used as building bocks for specifying the protection requirements for the datasets to be outsourced. Intuitively, security classes specify the minimum guarantees to be provided to the datasets for the considered properties: datasets cannot be outsourced to a plan that does not provide at least such guarantees. In the computation of an allocation, it has however to be considered that a dataset can be outsourced in encrypted form, wrapped in a layer of encryption administered by its owner. Intuitively, this provides an additional protection layer to the dataset (especially when confidentiality is a property of interest), since it can be accessed only by authorized/trusted subjects who know the encryption key. It is then to be expected that the protection requirement of a dataset may be impacted (and hence be different) by the (plaintext/encrypted) format in which the dataset will be outsourced. For example, with reference to the properties and labels in Fig. 8, a high confidentiality (HC) protection requirement for a plaintext dataset may be lowered (e.g., to LC) if the dataset is encrypted before outsourcing, accounting for the extra-layer of protection ensured by owner-side encryption.

Property	Label	Expression
C	HC	`encr`$=$AES $\wedge$ `cert`$=$certA
	LC	`audit`$=$12M
	$\bot$	`true`
A	HA	`bandwidth`$=25$
	LA	`throughput`$=5 \vee$ `throughput`$=10$
	$\bot$	`true`

Fig. 8 An example of security properties with labels and expressions

Given a collection of datasets and a set of security classes, the data owner can then easily formulate protection requirements for the datasets by associating with each dataset d one or two security classes $\lambda^{\circ}(d)$ and $\lambda^{\bullet}(d)$, for the plaintext (λ°) and/or encrypted ($\lambda^{\bullet}$) representation of d. Intuitively, the class $\lambda^{\circ}(d)$ ($\lambda^{\bullet}(d)$, respectively) specified for the plaintext (encrypted, respectively) representation of dataset d denotes, as mentioned above, the minimum guarantees to be provided for d in case d is outsourced in plaintext (encrypted, respectively). To illustrate, consider a company wishing to allocate to cloud plans a collection composed of datasets `projects` and `past_projects`, including all data related to the current and past projects of the company; `admin`, including all data related to the company administration; and `archive`, including data to be archived. Figure 10 illustrates an example of protection requirements for these datasets, where symbol '–' denotes the fact that no specific requirement is formulated for the plaintext/encrypted representation of a dataset. Note that the possibility of formulating a requirement for one (plaintext/encrypted) representation only nicely models the possibility to force the consideration of only one format (either plaintext or encrypted) for a dataset: the one for which the requirement is specified. With reference to the datasets in Fig. 10, assume that fast retrieval is needed for data related to the current and past projects: since encryption and decryption inevitably incur additional latency, a possibility is to avoid considering owner-side encryption, and therefore specify a protection requirement only for the plaintext versions of datasets `projects` and `past_projects`. The administrative dataset `admin`, on the other hand, has two different protection requirements, meaning it could be outsourced in plaintext or in encrypted form. Depending on whether `admin` will then be outsourced plaintext/encrypted, one of the two protection

requirements will be enforced. "Assessment" will illustrate how such protection requirements can be satisfied in the definition of an allocation of datasets to cloud plans.

Global requirements: As mentioned in "Specification", considering the peculiarities of the multicloud scenario and of the problem of allocating different datasets to different plans, there is the need to support users in specifying global requirements on the overall allocation. The rationale is that resorting to a set of plans, while certainly a promising strategy for accommodating the needs of heterogeneous datasets, inevitably brings an additional overhead given by the need to start and manage multiple contracts with the involved providers. Hence, to guide the overall allocation, the global requirements that may be specified demand that: (i) a certain set of datasets should be outsourced to the same plan (*co-location* requirement), to model the fact that such data are expected to be frequently accessed together and hence it can be more convenient to allocate them to the same plan; (ii) a certain set of datasets should not be outsourced in plaintext to the same plan (*separation* requirement), to impede joint visibility over these datasets in their entirety when this can disclose sensitive information; (iii) a maximum number of plans should be selected for the allocation (*max_plans* requirement), to avoid excessive fragmentation of the datasets; and (iv) a minimum storage occupation should be used for each selected plan (*min_storage* requirement), to ensure that the inevitable overhead given by the adoption of the different plans is compensated by the fact that every plan is used to store at least a reasonable amount of data. The specification of global requirements simply demands the definition of the datasets to be co-located or separated, and of two thresholds for the number of plans and minimum storage. Figure 11 illustrates an example of global requirements for our running example. They state that: (i) datasets `projects` and `past_projects` should be allocated to the same plan; (ii) datasets `projects` and `admin` should not be allocated in plaintext to the same plan; and (iii) the maximum number of plans selected for the allocation and the minimum storage at each plan are, respectively, 3 and 30GB.

Dataset	λ°	$\lambda^{\bullet}$
`projects`	[HC, HA]	–
`past_projects`	[HC, LA]	–
`admin`	[HC, HA]	[LC, HA]
`archive`	–	[⊥, LA]

Fig. 10 An example of protection requirements for a collection of datasets

co-location	{{`projects`,`past_projects`}}
separation	{{`projects`,`admin`}}
max_plans	3
min_storage	30GB

Fig. 11 An example of global allocation requirements for the datasets in Fig. 10

Assessment

The protection and global requirements illustrated in "Specification" can be used to restrict the allocation of a dataset to a plan based on whether such allocation respects all specified requirements. Indeed, multiple allocations satisfying all requirements may exist, possibly entailing different economic costs depending on the selected plans. Different strategies could be adopted to select one allocation over another one, and a natural solution is to compute an allocation that: (i) satisfies all the requirements; and (ii) minimizes the economic costs of the overall allocation. In other words, the allocation should select a (optimal) combination of plans that satisfies all constraints, while ensuring no different allocation could satisfy all requirements at a lower cost.

As for the enforcement of the protection requirements, it is first necessary to determine, for each dataset, the set of candidate plans satisfying protection requirements that may be selected for the allocation. To this end, it is possible to reason over the security classes (used for specifying protection requirements) and their associated expressions (e.g., those in Fig. 10 for our running example), evaluated against the attribute values of the available plans. A possible approach is to determine the security class of each plan, defined as the highest security class c_{max} (in the lattice) for which its attributes satisfy the expression characterizing c_{max}. To illustrate, consider the plans in Fig. 1 and the lattice in Fig. 9. Security class [LC, HA] is satisfied by $\mathbf{P}_5$, since its attribute values satisfy the expressions associated with with LC and HA. Since $\mathbf{P}_5$ does not satisfy other classes dominating [LC, HA], then [LC, HA] is the class of $\mathbf{P}_5$. Figure 12 illustrates the lattice in Fig. 9 reporting also, on the right-hand side of each class c, the plans having c as their class. Intuitively, a plan can be a candidate for a dataset d only if its security class is equal to, or dominates, the class of d's protection requirement: indeed, any plan that satisfies a security class equal to or dominating d's protection requirement provides *at least* the protection guarantees requested for d. Figure 12 illustrates, on the left-hand-side of each class c, the datasets that have c as protection requirement. When a class is a requirement for the encrypted representation of a dataset, we denote the dataset with a gray background. Note that, whenever two different protection requirements are specified for a dataset d depending on its plaintext/encrypted representation, d may have two different sets of candidate plans. To illustrate, consider the lattice in Fig. 12. Dataset `archive`, which can only be outsourced in encrypted form with a

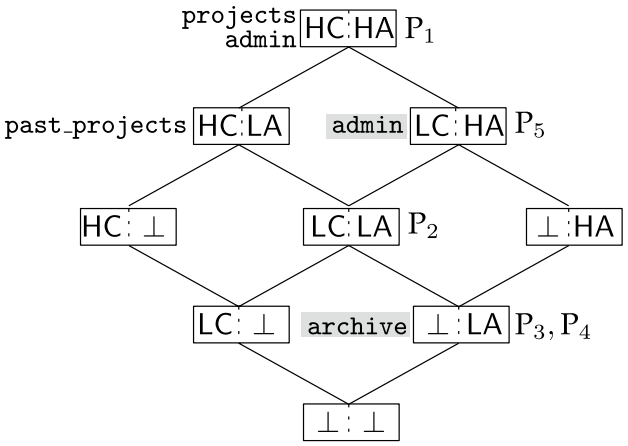

Fig. 12 Security lattice of Fig. 9 with the classification of the plans in Fig. 1 and datasets with the protection requirements in Fig. 10

protection requirement [⊥, LA] (Fig. 10) could be allocated to $\mathbf{P}_3$ or $\mathbf{P}_4$ (which have the same class as `archive`), as well as to $\mathbf{P}_2$, $\mathbf{P}_5$, and $\mathbf{P}_1$ whose classes appear higher in the lattice. Dataset `admin`, which can be outsourced in plaintext of encrypted, has two sets of candidate plans: if outsourced plaintext, it could only be allocated to $\mathbf{P}_1$ while, if outsourced in encrypted form, it could be allocated also to $\mathbf{P}_5$ (which has the same class as that of encrypted `admin`), besides $\mathbf{P}_1$ (whose class is higher in the lattice).

In principle, given the set of candidate plans for a dataset, any of them could be finally selected for storing the dataset without violating any protection requirement. Aiming at minimizing the overall cost entailed by the allocation, it is necessary to compute an optimal allocation, selecting a plan for each dataset in such a way that the selected set of plans satisfy all global requirements and no other allocation would entail lower costs while satisfying all constraints. To this end, the problem can be translated into a binary programming problem, aimed at minimizing an objective function that models the economic costs of the allocation. The problem of computing an optimal allocation satisfying arbitrary user requirements in multicloud scenario can then be easily solved leveraging off-the-shelf solvers.

Conclusions

We discussed the problem of supporting and guiding data owners in adopting cloud-based services for storing and managing their data in the cloud. We illustrated some of the main challenges that characterize the problem, and illustrated research directions that address these challenges. We focused on presenting solutions for: (i) modeling cloud plan characteristics, (ii) supporting users in specifying (and have enforced) arbitrary requirements and preferences, possibly leveraging natural language expressions and high-level and easy-to-understand abstractions, and (iii) computing optimal allocations in multicloud scenarios in obedience of protection requirements while minimizing economic costs. The challenges and solutions discussed are central for empowering data owners in maintaining control over their data and applications while resorting to the cloud, ultimately facilitating an even wider adoption of the cloud paradigm.

Acknowledgements This work was supported in part by the EC under projects MARSAL (101017171), GLACIATION (101070141), EdgeAI (101097300), by the Italian MUR under PRIN project HOPE, and by project SERICS (PE00000014) under the MUR NRRP funded by the EU - NextGenerationEU.

Funding Open access funding provided by Università degli Studi di Milano within the CRUI-CARE Agreement.

Data availability No data have been generated or analyzed.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Bartolini C, Lenzini G, Robaldo L. The data protection regulation compliance model usage control on cloud systems. IEEE Secur Priv. 2019;17(6):37–45.
2. Carniani E, D'Arenzo D, Lazouski A, Martinelli F, Mori P, et al. Usage control on cloud systems. Future Gener Comput Syst. 2016;63:37–55.
3. Chauhan SS, Pilli ES, Joshi RC. BSS: a brokering model for service selection using integrated weighting approach in cloud environment. J Cloud Comput. 2021;10(1):1–14.
4. Cloud Security Alliance. Cloud Controls Matrix and CAIQ v4. https://cloudsecurityalliance.org/artifacts/cloud-controls-matrix-v4/.
5. De Capitani di Vimercati S, Foresti S, Livraga G, Piuri V, Samarati P. A fuzzy-based brokering service for cloud plan selection. IEEE Syst J. 2019;13(4):4101–9.
6. De Capitani di Vimercati S, Foresti S, Livraga G, Piuri V, Samarati P. Security-aware data allocation in multicloud scenarios. IEEE Trans Dependable Secur Comput. 2021;18(5):2456–68.

7. De Capitani di Vimercati S, Foresti S, Livraga G, Piuri V, Samarati P. Supporting user requirements and preferences in cloud plan selection. IEEE Trans Serv Comput. 2021;14(1):274–85.
8. De Capitani di Vimercati S, Foresti S, Livraga G, Samarati P. Towards owner-controlled data sharing. In: Nicopolitidis P, Misra S, Yang LT, editors., et al., Advances in computing, informatics, networking and cybersecurity. Springer; 2022.
9. Ding S, Wang Z, Wu D, Olson DL. Utilizing customer satisfaction in ranking prediction for personalized cloud service selection. Decis Support Syst. 2017;93:1–10.
10. Garg SK, Versteeg S, Buyya R. A framework for ranking of cloud computing services. Future Gener Comput Syst. 2013;29(4):1012–23.
11. Ghosh N, Ghosh SK, Das SK. SelCSP: a framework to facilitate selection of cloud service providers. IEEE Trans Comput. 2015;3(1):66–79.
12. Guo Y, Mi Z, Yang Y, Ma H, Obaidat MS. Efficient network resource preallocation on demand in multitenant cloud systems. IEEE Syst J. 2019;13(4):4027–38.
13. Hussain A, Chun J. Cloud service scrutinization and selection framework (C3SF): a novel unified approach to cloud service selection with consensus. Inform Sci. 2022;586:155–75.
14. Li A, Yang X, Kandula S, Zhang M. CloudCmp: comparing public cloud providers. In: Proc. of the 10th ACM Internet Measurement Conference (ACM IMC): Australia, Melbourne; 2010.
15. Ouedraogo M, Mignon S, Cholez H, Furnell S, Dubois E. Security transparency: the next frontier for security research in the cloud. J Cloud Comput. 2015;4(1):1–14.
16. Qu L, Wang Y, Orgun MA, Liu L, Liu H, Bouguettaya A. CCCloud: context-aware and credible cloud service selection based on subjective assessment and objective assessment. IEEE Trans Serv Comput. 2015;8(3):369–83.
17. Piuri V, Jhawar R, Santambrogio M. Fault tolerance management in cloud computing: a system-level perspective. IEEE Syst J. 2013;7(2):288–97.
18. Rackspace cloud service level agreement. https://www.rackspace.com/information/legal/cloud/sla.
19. Sundareswaran S, Squicciarini A, Lin D. A brokerage-based approach for cloud service selection. In: Proc. of the 5th IEEE International Conference on Cloud Computing (IEEE CLOUD). 2012; Honolulu, HI, USA.
20. Tang M, Dai X, Liu J, Chen J. Towards a trust evaluation middleware for cloud service selection. Future Gener Comput Syst. 2017;74:302–12.
21. Xie Y, Guo Y, Mi Z, Yang Y, Obaidat MS. Loosely coupled cloud robotic framework for QoS-driven resource allocation-based Web service composition. IEEE Syst J. 2020;14(1):1245–56.
22. Zheng Z, Wu X, Zhang Y, Lyu MR, Wang J. QoS ranking prediction for cloud services. IEEE Trans Parallel Distrib Syst. 2013;24(6):1213–22.