Multi-Dimensional Certification of Modern Distributed Systems

Marco Anisetti, Claudio A. Ardagna, Nicola Bena

Abstract—The cloud computing has deeply changed how distributed systems are engineered, leading to the proliferation of ever-evolving and complex environments, where legacy systems, microservices, and nanoservices coexist. These services can severely impact on individuals' security and safety, introducing the need of solutions that properly assess and verify their correct behavior. Security assurance stands out as the way to address such pressing needs, with certification techniques being used to certify that a given service holds some non-functional properties. However, existing techniques build their evaluation on software artifacts only, falling short in providing a thorough evaluation of the non-functional properties under certification. In this paper, we present a multi-dimensional certification scheme where additional dimensions model relevant aspects (e.g., programming languages and development processes) that significantly contribute to the quality of the certification results. Our multi-dimensional certification enables a new generation of service selection approaches capable to handle a variety of user's requirements on the full system life cycle, from system development to its operation and maintenance. The performance and the quality of our approach are thoroughly evaluated in several experiments.

Index Terms—Assurance, Certification, Security, Service Selection

1 Introduction

Modern distributed systems are based on (micro)services developed using cloud-native technologies, composed at run time with orchestration platforms, and continuously monitored to ensure scalability and elasticity. At the same time, (micro)services coexist with legacy systems consisting of large and difficult-to-maintain codebases, on one side, and nanoservices consisting of few lines of codes, on the other side, substantially increasing systems' complexity [1]. This scenario radically changed the governance, risk, and compliance landscape, invading the safety and security of people. Distributed systems are in fact supporting virtually every transaction and process in our everyday life, calling for solutions properly assessing and verifying their behavior and, in turn, the impact on the individual's personal sphere.

In the last decade, security assurance has been widely accepted as a means to model and assess the behavior of a distributed system, to the aim of increasing its trustworthiness [2]. Assurance techniques have been consistently applied, with certification schemes adopted to drive the evaluation of non-functional (e.g., confidentiality, integrity) properties of a given target system (e.g., software, service) [3]–[6]. Notwithstanding the continuous research on certification [2], existing techniques are still inadequate and rarely applied to modern service-based scenarios. On the one hand, traditional schemes (e.g., Common Criteria) lack of flexibility to properly tackle the challenges of service-based systems, such as automatic service selection [7], [8]. On the other hand, service-specific schemes (e.g., [4]–[6]) fail to evaluate the services in their entirety. They in fact build their evaluation on the final software artifacts only (i.e., executable service), ignoring additional and relevant aspects of the target service, such as the development process [8]. This negatively affects the whole certification process, impairing existing approaches in retrieving a detailed and realistic description of the service and, in turn, reducing the accuracy of the certification results. This also negatively influences the life cycle management of distributed systems, since certified services are selected and composed on the basis of certified yet incomplete information.

Our paper aims to fill in the above gaps by defining a novel certification scheme that expands the scope of certification beyond the simple evaluation of software artifacts. It considers additional aspects related to, for instance, development and verification processes. Our scheme groups these aspects into coherent *dimensions*, permitting to retrieve a certified, complete, and well-structured picture of the target service, paving the way for a fully-informed and safe decision-making. Our scheme supports a new wave of certification, where *i)* services are certified by considering each dimension independently; *ii)* each dimension can be managed according to its peculiarities (e.g., its life cycle); *iii)* non-functional properties are verified by integrating the certification results in each dimension. Our certification scheme is finally integrated within the distributed system life cycle management to support a certification-based service selection, where functionally-equivalent services are ranked and selected according to non-functional requirements defined by the users.

The contribution of our paper is twofold. We first define a multi-dimensional certification scheme, where the state-of-the-art evaluation of non-functional properties focusing on software artifacts only is extended to consider additional dimensions, modeling relevant aspects that contribute to the quality of the certification results. To this aim, the proposed

• *Marco Anisetti, Claudio A. Ardagna, Nicola Bena are with the Department of Computer Science, Università degli Studi di Milano, Milan, Italy. E-mail: {firstname.lastname}@unimi.it*

scheme introduces a novel certification model as the collector of all activities driving a service certification, where all its building blocks (i.e., non-functional properties, target of certification, evidence) are based on dimensions to enable modular certification. Our scheme certifies the properties of a given target service composing the results of the evaluation in each dimension, and producing a more accurate certificate of the service behavior. We then implement a certification-based service selection process, where services are ranked according to the certified non-functional properties they hold and corresponding users' requirements, using a Multi-Criteria Decision-Making (MCDM) technique.

The remainder of this paper is organized as follows. Section 2 discusses the state of the art and our motivations. Section 3 introduces our approach at a glance. Section 4 defines the certification model driving all certification activities, whose execution is described in Section 5. Section 6 details the methodology for ranking and selection of certified services. Section 7 presents an extensive experimental evaluation of the proposed certification scheme. Finally, Section 8 draws our conclusions.

2 Background and Motivations

Certification schemes aim to verify whether a given system supports one or more non-functional properties and behaves as expected. Schemes are based on a *certification model* that specifies all the activities that have to be executed on the *target of certification* (e.g., a software or a service) to collect the *evidence* proving a given *non-functional* (e.g., confidentiality, integrity, reliability) *property*. If the collected evidence supports the given non-functional property, the certification scheme triggers the release of a *certificate* for the service, which is further used to comparatively select services according to non-functional requirements of end users. Certification schemes, in fact, have been commonly adopted to drive service ranking and selection according to certified non-functional properties and corresponding evidence in certificates [9].

Existing certification schemes followed the evolution of ICT systems [2], first focusing on the certification of traditional software-based systems (e.g., [10]) and then focusing on the certification of service-based and cloud-based systems in virtually any domains (e.g., [3], [4], [6], [11]). Two main approaches to evidence collection have been proposed to support existing certification schemes: *i) test-based evidence collection*, where evidence is collected as the result of testing activities performed by the Certification Authority (CA) on the target of certification [3], [4], [6]; *ii) monitor-based evidence collection*, where evidence is collected in the form of metrics retrieved by monitoring service execution [11]–[19]. These approaches have been then applied in additional scenarios, including compliance with Service-Level Agreements [13], [16], [17], [20]–[23], verification-as-a-service in grid computing [24], behavioral analysis of Network Virtualization Functions [25], as well as data integrity in cloud-edge datastores [26]. Peculiar schemes also include other means for evidence collection, such as Trusted Platform Modules [5], and formal methods for certification of *functional* properties (i.e., correctness) [24], [27]. Certification schemes also support self-adaptive systems, ensuring the compliance of a system in any of its states and driving the system adaptations according to properties in the certificates [28], [29].

The undebatable advantages brought by certification schemes conflict with some strong assumptions that limit their applicability and quality as follows.

A1) Benign behavior of all involved parties. Service providers, end-users, and certification authorities follow the certification scheme and its rules, resulting in certificates properly modeling the properties of the certified services. We note that some peculiar schemes relax this assumption [3], [26], [30], though they are out of the scope of this paper.

A2) Chain of trust rooted at the CA. The CA is trusted by design, meaning that it correctly produces and executes certification models driving the certification activities. The trust is propagated from the CA to the certificates, and is guaranteed by cryptographic signatures [6].

A3) Certificates awarded to services according to their software artifacts only. The certification scheme builds its evaluation, and corresponding certificate award, on evidence collected by analyzing the software artifacts of the target of certification [3], [4], [6], [11]. In other words, evidence is collected by testing and monitoring the software artifacts of the target service, ignoring additional information coming from, for example, the development process.

A4) Software artifacts sufficient for optimal selection. Service ranking and selection are built on certified non-functional properties. The latter are evaluated according to corresponding evidence on software artifacts [9], [13], [16], [20]–[23], [31] stored in certificates. Though partial, the evidence is assumed to be complete and support accurate ranking.

Assumptions A1) and A2) limit the applicability of certification schemes to scenarios where all parties are benign and the chain of trust is built on a trusted party (i.e., the CA), while posing no limits to the quality of retrieved certificates. By contrast, assumption A3) reduces the quality of the certificates, by limiting the amount of evidence that can be collected. Services are in fact evaluated and certified on the basis of partial information describing their final software artifacts only, while discarding all evidence related to, for instance, how the software artifacts have been implemented, which can still provide relevant insights [8]. This lack of information degrades the quality of decisions based on certificates [32] directly impairing assumption A4), which requires complete information to retrieve an accurate ranking for optimal service selection.

As a result, the effectiveness and usefulness of current certification schemes is strongly impaired by assumptions A3) and A4); this results in scenarios where the selected services can exhibit a suboptimal behavior once provisioned, thus impacting on the users' trust in service providers.

The certification scheme in this paper addresses these gaps departing from assumptions A3) and A4). It extends the evaluation of target services beyond software artifacts (assumption A3)) at the basis of an accurate service selection (assumption A4)), by integrating relevant aspects of the target services influencing their certificates.

Table 1
Non-functional attributes of services s_1–s_5 grouped in the three dimensions.

Service	Dimension Artifacts D_{art}			Dimension Development D_{dev}					Dimension Evaluation D_{eval}	
	Repl.	**Repl. Zones**	**HA Prot.**	Prog. Lang.	Dev. Proc.	Type	State Mgmt.	Code Review	Trust. Contr.	When
s_1	**3**	**3**	**Managed**	Python	Waterfall	Monolith	Stateful	No	No	After
s_2	**3**	**2**	**Custom**	Java	Spiral	Microservice	Stateless	No	No	During
s_3	**1**	**1**	**Custom**	Java	Prototype	Monolith	Stateless	Yes	Yes	After
s_4	**3**	**3**	**Managed**	Rust	DevSecOps	Microservice	Stateless	Yes	Yes	During
s_5	**2**	**3**	**Managed**	Python	DevOps	Microservice	Stateless	No	Yes	After

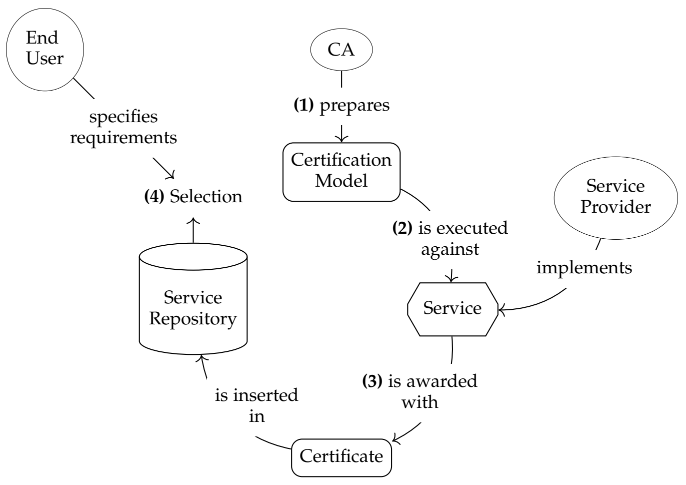


Figure 1. Overview of our approach.

3 Our Approach

Our reference scenario in Figure 1 is a cloud environment where services are first certified and then selected according to their non-functional properties to ensure stable quality of service. It includes the following main parties: *i) service provider* that implements and distributes certified services; *ii) end user* that selects and integrates certified services within its system according to certified non-functional properties; *iii) certification authority* (CA) that defines and executes a certification scheme proving non-functional properties on services. Our certification scheme is based on the novel concept of *dimension* (Section 3.1), departing from the state of the art and positively impacting on all the involved parties (Section 3.2).

3.1 Dimension-Based Certification

The certification scheme in this paper implements a flow of activities composed of three steps: *i) certification model definition* (Section 4 and step (1) in Figure 1), where the CA describes the activities to be executed on the target of certification to collect evidence proving the support of a non-functional property; *ii) certification model execution* (Section 5.1 and step (2) in Figure 1), where the CA executes, with the help of its verification labs, the activities in the certification model; *iii) certificate award* (Section 5.2 and step (3) in Figure 1), where the CA awards a certificate proving a given non-functional property to the target of certification according to the collected evidence. End users finally select and compose services with certified behavior on the basis of their certificates and corresponding properties (Section 6 and step (4) in Figure 1).

Our scheme radically changes the definition and execution of the certification model by modifying the definition of non-functional property in literature (e.g., [6]). Our definition of property, which is formalized in Section 4, models different aspects influencing the evaluation of the service behavior beyond simple software artifacts. New attributes (see Table 1) organized in *dimensions* are added, where each dimension describes a particular aspect of the non-functional property, as follows.

Definition 1** **(D). A dimension D is a set $\{(a_1, v_1), \ldots, (a_n, v_n)\}$, where each attribute is a pair (a_i, v_i) with

- a_i the attribute name;
- $v_i \in V_{a_i}$ the value for a_i, denoted as $a_i.v_i$.

We note that V_{a_i} is a totally ordered set according to total order relationship $>_{a_i}$ defined by experts (e.g., the CA). $>_{a_i}$ orders attribute values on the basis of their effect on the strength of the corresponding non-functional property, such that, for any pairs of attribute values $a_i.v_j, a_i.v_k \in V_{a_i}$, $a_i.v_j >_{a_i} a_i.v_k$ *iff* $a_i.v_j$ increases the property strength more than $a_i.v_k$.

While being generic and extensible, the approach in this paper considers three dimensions as presented in Table 1: *i)* D_{art} that includes attributes describing the software artifacts of the target (i.e., the attributes considered in the state of the art); *ii)* D_{dev} that includes attributes describing the development process used to implement the target, such as attributes *Prog. Lang.*, *Dev. Proc.*, *Type*, *State Mgmt.*, and *Code Review*;[1] *iii)* D_{eval} that includes attributes describing the verification process at the basis of the target certification, such as attributes *Trust. Contr.* and *When*.

3.2 Impact and Reference Example

Let us consider a service directory with 5 functionally-equivalent services s_1–s_5, to be certified for property reliability, whose non-functional attributes and corresponding dimensions are shown in Table 1. For instance, service s_4 is a microservice operating in 3 replicas spread in 3 zones, indirectly managing the application state, and implemented following a DevSecOps methodology. Following our scheme, the CA certifies each service s_1–s_5 for property reliability according to three dimensions D_{art}, D_{dev}, and D_{eval} in the certification model (Section 4), whose execution (Section 5.1) triggers the release of the corresponding certificate (Section 5.2). Each certificate departs from assumption A3), expanding the service behavior beyond dimension D_{art}, and supporting a certification scheme that

1. We that note the development process has a substantial impact on the resulting software and, in turn, on the non-functional properties it holds [8], [33].

is *i)* fine-grained, since it models service behavior according to several detailed attributes, *ii)* dimension-aware, since it organizes attributes in dimensions influencing the non-functional property [8]. Certificates are then matched and ranked against user's requirements to support service selection (Section 6). This addresses the shortcomings of assumption A4), grounding selection on more complete and accurate multi-dimensional information on service behavior.

The proposed scheme provides benefits for all the involved parties as follows.

Service provider retrieves certificates better reflecting the behavior of its services [30], [32], [34]. For instance, the certificate of s_4 models the corresponding development process, which outperforms the development process of other services in Table 1, being built on a DevSecOps approach and microservices with code review.

End user selects services according to more accurate certificates, supporting fully-informed and safe decisions. The accuracy of properties in certificates, as well as the structure and content of certificates are in fact fundamental in decision-making [32]. For instance, when evaluated according to D_{art}, s_1 or s_4 are equivalent among them and clearly outperform the remaining ones. However, when considering the additional dimensions in our scheme, it is clear that s_1 is largely unacceptable and s_4 is the best solution. In fact, s_1 is a legacy service developed following a Waterfall process, has not been validated using code review, and directly manages application state, a practice which is not recommended.

CA improves the quality and in turn the trustworthiness of its certification scheme, providing higher accuracy with a marginal increase in overhead.

4 Certification Model Definition

Our certification scheme defines a certification model detailing all activities required to certify a given target against a non-functional property (step (1) in Figure 1).

***Definition 2** ($\mathcal{M}$).* A certification model is a tuple of the form $\mathcal{M}$=$\langle p, ToC, \mathcal{E}, \mathcal{F}\rangle$, where

- p is the non-functional property in Definition 3;
- ToC is the target of certification in Definition 4;
- $\mathcal{E}$ is the evidence collection model in Definition 5;
- $\mathcal{F}$ is the evaluation function in Definition 7, determining the final outcome of the certification model execution.

The certification model follows the state of the art in Section 2 and specifies the non-functional property to be certified on a target service according to an evidence collection process. It is prepared and cryptographically signed by the certification authority, and trusted by service providers and end users according to the chain of trust in Section 2 [6]. Hereafter we detail the different components of the certification model.

4.1 Non-Functional Property and Target of Certification

A non-functional property p describes the non-functional behavior of a target of certification, as follows.

***Definition 3** (p).* A non-functional property p is a pair $(\hat{p}, \{D_1, \ldots, D_n\})$, where $\hat{p}$ is an abstract property (i.e., the property name) and D_i is a dimension organizing non-functional attributes as described in Definition 1.

Similarly, the target of certification is defined as a set of mechanisms that are logically grouped according to dimensions, as follows.

***Definition 4** (ToC).* A target of certification ToC is a set $\{\Theta_{D_1},\ldots,\Theta_{D_n}\}$, where Θ_{D_i}=$\{\theta_1,\ldots,\theta_m\}$ is a set of non-functional mechanisms θ_j describing the target according to dimension D_i in Definition 1. A non-functional mechanism θ_j is a pair $(\hat{\theta}, A_\theta)$, where $\hat{\theta}$ is a mechanism type and A_θ is a set of values refining it [6].

We note that non-functional mechanisms are the means by which the target supports a non-functional property.

Example 1. Let us consider our reference example in Section 2. Property reliability is defined as p_{rel}=$(\hat{p}_{rel}$, $\{D_{art}$, D_{dev}, $D_{eval}\})$. Attribute values are ordered according to $>_{a_i}$ following their position in the corresponding definition. For instance, attribute values of attribute *Prog. Lang.* in D_{dev} are ordered as [Rust $>_{a_{pl}}$ Java $>_{a_{pl}}$ Python]. The certification model for s_4, defined as $\mathcal{M}_{s_4}$, includes s_4 as ToC=$\{\Theta_{D_{art}}$,$\Theta_{D_{dev}}$, $\Theta_{D_{eval}}\}$, where $\Theta_{D_{art}}$=\{*Replica Manager*=Kubernetes\}, $\Theta_{D_{dev}}$=\{*Pipeline*=File Content, *Source Code*=Rust, *Code Review Document*=File Content\}, and $\Theta_{D_{eval}}$=\{*Certification Framework*=Trusted-and-Continuous\}. More in detail, s_4 is deployed in Kubernetes, is written in Rust and has a code review document ($\Theta_{D_{dev}}$), and is certified by means of a trustworthy and continuous certification framework ($\Theta_{D_{eval}}$).

4.2 Evidence

The certification scheme collects evidence to prove that a target of certification holds a non-functional property. Evidence can be collected according to testing, monitoring, or formal proofs, and is bound to a subset of the ToC it insists on. For simplicity but no lack of generality, we consider test-based evidence, where the execution of testing activities permits to collect evidence on the service behavior. Evidence is collected according to an *evidence collection model* $\mathcal{E}$=$\mathcal{E}_{D_{art}}\cup\mathcal{E}_{D_{dev}}\cup\mathcal{E}_{D_{eval}}$, where $\mathcal{E}_D$ is the evidence collection model detailing the testing activities in dimension D, as follows.

***Definition 5** ($\mathcal{E}_D$).* An evidence collection model $\mathcal{E}_D$ for dimension D is a set $\{\{(\theta_1,t_1),\ldots,(\theta_n,t_n)\}\}$, where each $\{(\theta_1,t_1),\ldots, (\theta_n,t_n)\}$ is a single test case. Each test case consists of several steps (θ_i,t_i), where $\theta_i\in\Theta_D$ is the portion of the target the evidence collection model insists on, and t_i is a step of the test case used to verify such target.

In other words, the evidence collection model $\mathcal{E}_D$ is a sequence of test cases. Each test case verifies a specific (set of) non-functional mechanisms in Θ_D to prove a non-functional property in dimension D. It is a sequence of steps t_i specifying all inputs, preconditions, and postconditions for its execution, as well as the expected output, at design

Table 2
Evidence collection model in $\mathcal{M}_{s_4}$.

D	$\mathcal{E}_D$
D_{art}	{{(*Replica Manager*=Kubernetes, Get-Orchestrator), (*Replica Manager*=Kubernetes, Check-Replicas), (*Replica Manager*=Kubernetes, Check-Zones)}}
D_{dev}	{{(*Pipeline*=File Content, Dry-Run)}, {(*Source Code*=Rust, Check-Code)}, {(*Code Review Document*=File Content, Check-Review-Document)}}
D_{eval}	{{(*Certification Framework*=Trusted-and-Continuous, Check-Document)}}

time [6]. The result of the execution of an evidence collection model $\mathcal{E}_D$ is a set of evidence $\{ev\}_D$ defined as follows.

***Definition 6** (ev).* An evidence ev is a set $\{(to_1, tr_1), \ldots, (to_n, tr_n)\}$ describing the result of the execution of a single test case in $\mathcal{E}_D$. It consists of several pairs (to_i, tr_i), where to_i is the output of the execution of test step t_i, and tr_i is either *Success* or *Failure* indicating whether the execution of t_i is successful, that is, output to_i matches the expected output in t_i.

Evidence collection model $\mathcal{E}_D$ and the corresponding set of evidence $\{ev\}_D$ provide the trust anchor of our scheme, binding certificates on concrete evidence retrieved from the execution of the test cases against the targets.

Example 2. Following Example 1, we present an excerpt of the evidence collection model $\mathcal{M}_{s_4}.\mathcal{E}_{D_{art}}$={{(*Replica Manager*=Kubernetes, Get-Orchestrator), *Replica Manager*=Kubernetes, Check-Replicas), *Replica Manager*=Kubernetes, Check-Zones)}}. It contains one test case consisting of three steps insisting on the same mechanism *Replica Manager*. It first retrieves the orchestrator checking whether it is a HA-enabled Kubernetes cluster (step "Get-Orchestrator"); it then verifies the number of replicas checking whether it is compatible with the expected number of replicas (step "Check-Replicas"); it finally verifies the number of zones (e.g., data centers) where the replicas are spread checking whether it is compatible with the expected number of zones (step "Check-Zones"). We note that the details of each step (e.g., preconditions, inputs) are omitted for brevity. Table 2 shows the complete $\mathcal{M}_{s_4}.\mathcal{E}$.

4.3 Evaluation Function

The last component of the certification model is evaluation function $\mathcal{F}$. It determines the outcome (success or failure) of evidence collection and, if positive, enables certificate release. It follows the concept of dimension and is modeled as a sequence of Boolean functions, each retrieving the outcome of evidence collection in a specific dimension. Functions are then combined using a Boolean operator AND as follows.

***Definition 7** ($\mathcal{F}$).* The evaluation function $\mathcal{F}$ is a Boolean expression $\mathcal{F}_{D_1} \wedge \ldots \wedge \mathcal{F}_{D_n}$, where $\mathcal{F}_{D_i} : \{ev\}_{D_i} \rightarrow \{\top, \bot\}$ is a function returning the outcome of the certification model execution within a dimension D_i according to the collected evidence $\{ev\}_{D_i}$.

According to the dimensions in Section 3, $\mathcal{F}$ is defined as $\mathcal{F}_{D_{art}} \wedge \mathcal{F}_{D_{dev}} \wedge \mathcal{F}_{D_{eval}}$. Each function takes value: *i)* $\top$, in case of success, allowing certificate release, or *ii)* $\bot$, in case of failure, preventing certificate release. We note that the evaluation function can support complex rules determining the result of the certification, beyond the "all-or-nothing" in this paper where all evidence must be successfully collected or all dimensions must be successfully certified.

Example 3. Following Example 2, the certification model for s_4 is defined as $\mathcal{M}_{s_4}$=$\langle p_{rel}, ToC, \mathcal{E}, \mathcal{F}\rangle$, where p_{rel} and ToC are defined in Example 1; $\mathcal{E}$=$\mathcal{E}_{D_{art}} \cup \mathcal{E}_{D_{dev}} \cup \mathcal{E}_{D_{eval}}$, with $\mathcal{E}_{D_{art}}$ defined in Example 2; $\mathcal{F}$ is defined as $\mathcal{F}_{D_{art}} \wedge \mathcal{F}_{D_{dev}} \wedge \mathcal{F}_{D_{eval}}$.

5 Certification Model Execution and Certificate Award

Our certification scheme executes the certification model in Section 4 and, if successful, awards a certificate to the ToC (steps (2) and (3) in Figure 1). The soundness of the entire scheme is built on the *well-formedness* of the certification model defined by the CA.

Definition 8. A certification model $\mathcal{M}$ is *well-formed* iff $\forall\ \theta_i \in \mathcal{M}.ToC\ \exists\ (\theta_j, t_j) \in \mathcal{M}.\mathcal{E} \mid \theta_i = \theta_j$.

The above definition requires that, for each mechanism θ_i forming the ToC, there exists at least one test step (θ_j, t_j) in the sequence of test cases $\mathcal{M}.\mathcal{E}$ verifying θ_i. In other words, each mechanism in the target of certification must be verified by the test cases defined in the certification model.

In the following of this section, we present certification model execution (Section 5.1) and certificate award (Section 5.2), completing our certification scheme. The pseudocode of these steps is reported in Figure 2.

5.1 Certification Model Execution

A well-formed certificate model $\mathcal{M}$ enables a modular execution, building on the concept of *view* as the portion of $\mathcal{M}$ induced by a dimension D∈$\mathcal{M}.p$. A view contains all relevant information to evaluate ToC against the non-functional property in D. It is formally defined as follows.

Definition 9 (View). Let $\mathcal{M}$ be a certification model and D a dimension in $\mathcal{M}.p$. A view induced by D on $\mathcal{M}$ is a tuple of the form $\mathcal{V}$=$\langle D, \Theta_D, \mathcal{E}_D, \mathcal{F}_D\rangle$, where

- D is the dimension;
- Θ_D is the portion of the ToC corresponding to dimension D;
- $\mathcal{E}_D$ is the evidence collection model of dimension D;
- $\mathcal{F}_D$ is the portion of the evaluation function $\mathcal{F}$ returning the outcome of the certification model execution in the current view.

According to the three considered dimensions in Section 3, three views are induced on the certification model: $\mathcal{V}_{art}$, $\mathcal{V}_{dev}$, and $\mathcal{V}_{eval}$. Each view is evaluated in two steps as follows:

1) **Evidence Collection.** The certification process collects evidence $\{ev\}_D$ by executing the test cases in $\mathcal{V}.\mathcal{E}_D$ against the portion Θ_D of ToC.

```
INPUT
$\mathcal{M}$: Certification model

OUTPUT
$\mathcal{C}$: Certificate

CERTIFICATION MODEL EXECUTION AND
CERTIFICATE AWARD

/* View Computation */
forall $D_i \in \mathcal{M}.p$;
    /* For each dimension D build the corresponding view*/
    $\mathcal{V}_i \leftarrow$ build_view ($D_i$);
endfor;

/* Certification Model Execution */
forall $\mathcal{V}_i \in \{\mathcal{V}_1, \dots, \mathcal{V}_n\}$
    /* (1) Evidence Collection*/
    $\{ev_i\}, \{(m_i, v_i)\} \leftarrow$ execute_test_cases($\mathcal{V}_i.\mathcal{E}$);
    /* (2) Individual View Evaluation*/
    $r_i \leftarrow \mathcal{V}_i.\mathcal{F}_D(\{ev_i\})$;
endfor;

/* Result Aggregation*/
/* Evaluate overall result using Evaluation Function $\mathcal{F}$*/
res $\leftarrow \mathcal{F}(\{r_i\})$;

if res $= \top$ then
    /* build certificate if evaluation is successful*/
    $\mathcal{C} \leftarrow$ build_cert($\mathcal{M}$, $\{evid_i\}$, $\{(m_i, v_i)\}$);
    return $\mathcal{C}$;
else return $\perp$;
endif;
```

Figure 2. Certification process: pseudocode.

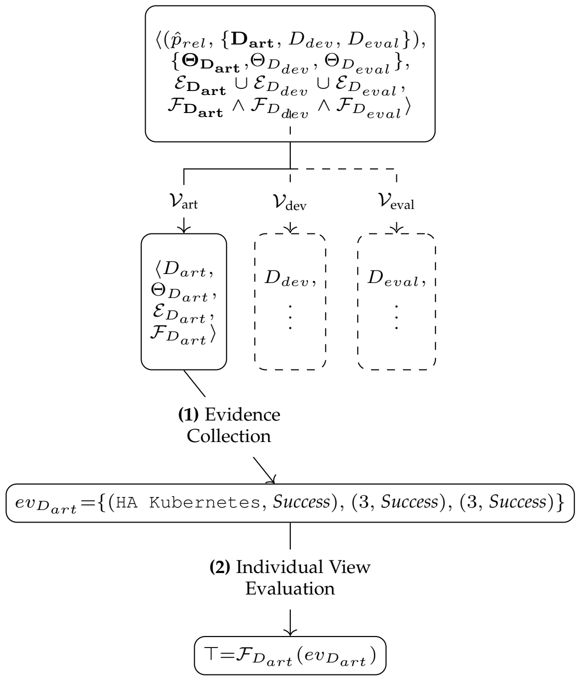
2) **Individual View Evaluation.** Evaluation function $\mathcal{F}_D \in \{\mathcal{F}_{D_{art}}, \mathcal{F}_{D_{dev}}, \mathcal{F}_{D_{eval}}\}$ determines the result ($\top$ or $\perp$) of evidence collection at step 1) in the corresponding dimension D_{art}, D_{dev}, D_{eval}. In particular, $\mathcal{F}_D$ is a Boolean expression over the collected evidence, requiring each evidence $ev \in \{ev\}_D$ to be successful (denoted as *Succ*$(ev)=\top$). An evidence is successful if all the test steps therein return *Success* (Definition 6). In other words, evaluation function $\mathcal{F}_D$ is successful ($\top$) iff $\forall ev \in \{ev\}_D$, *Succ*$(ev)=\top$. We note that different domain-specific functions can refine the notion of *successful evidence* with no impact on the overall process.

Evaluation function $\mathcal{M}.\mathcal{F}$ finally aggregates the Boolean results of the individual view evaluations and retrieves the final result of certification model execution. Results of view evaluations are aggregated according to an AND operator (Definition 7), meaning that evidence collection must be successful in any views and, in turn, in any dimensions.

Example 4. Following Example 3, Figure 3 shows an example of certification model execution in dimension D_{art}. The components in bold in the certification model become the components of view $\mathcal{V}_{\text{art}}$, where evidence is collected according to $\mathcal{V}_{\text{art}}.\mathcal{E}_{D_{art}}$ in Table 2 (step (1)). All evidence $ev_{D_{art}}$ is successful, therefore the view is evaluated $\top$ (step (2)).

5.2 Certificate Award

A successful certification model execution triggers the release of a certificate. It includes three main components:



$\langle(\hat{p}_{rel}, \{\mathbf{D_{art}}, D_{dev}, D_{eval}\}),$
$\{\mathbf{\Theta_{D_{art}}}, \Theta_{D_{dev}}, \Theta_{D_{eval}}\},$
$\mathcal{E}_{\mathbf{D_{art}}} \cup \mathcal{E}_{D_{dev}} \cup \mathcal{E}_{D_{eval}},$
$\mathcal{F}_{\mathbf{D_{art}}} \wedge \mathcal{F}_{D_{dev}} \wedge \mathcal{F}_{D_{eval}}\rangle$
$\mathcal{V}_{\text{art}}$
$\mathcal{V}_{\text{dev}}$
$\mathcal{V}_{\text{eval}}$
$\langle D_{art}, \Theta_{D_{art}}, \mathcal{E}_{D_{art}}, \mathcal{F}_{D_{art}}\rangle$
$D_{dev}, \vdots$
$D_{eval}, \vdots$
(1) Evidence Collection
$ev_{D_{art}}$={(HA Kubernetes, Success), (3, Success), (3, Success)}
(2) Individual View Evaluation
$\top = \mathcal{F}_{D_{art}}(ev_{D_{art}})$

Figure 3. Example of certification model execution in dimension D_{art} (excerpt).

i) the certification model, *ii)* the set of collected evidence, *iii)* a set of test metrics describing the evidence collection performance. A certificate is formally defined as follows.

Definition 10 ($\mathcal{C}$). A certificate is a tuple of the form $\mathcal{C}=\langle\mathcal{M}, \{ev\}, \{(m_1, v_1), \dots, (m_m, v_m)\}\rangle$, where

- $\mathcal{M}$ is the certification model in Definition 2;
- $\{ev\}=\{ev\}_{D_1} \cup \dots \cup \{ev\}_{D_n}$ is the set of collected evidence proving $\mathcal{M}.p$ on $\mathcal{M}.ToC$ in each dimension;
- $\{(m_i, v_i)\}$ is the set of metrics resulting from the execution of the evidence collection model $\mathcal{M}.\mathcal{E}$.

Metrics $\{(m_i, v_i)\}$ describe the performance of the evidence collection model, where m_i is a metric class and v_i its normalized value; notation $m_i.v$ denotes the value of metric m_i. Table 4 reports the considered metrics.

We note that the process for certificate awarding can be replicated by executing $\mathcal{E}$, and its correctness evaluated against $\mathcal{E}$, $\mathcal{F}$, and $\{ev\}$.

Example 5. Following Example 4, views $\mathcal{V}_{art}$, $\mathcal{V}_{dev}$, $\mathcal{V}_{eval}$ are retrieved from the certification model. Evidence is collected in each view and, assuming these activities to be successful according to evaluation functions, a certificate $\mathcal{C}_4$ is released. $\mathcal{C}_4=\langle\mathcal{M}_4, \{ev\}, \{(\textit{Input Partition Coverage}, 0.977), (\textit{Branch Coverage}, 0.876)\}\rangle$, where:

- $\mathcal{M}_4$ is the certification model in Example 3;
- $\{ev\}=\{ev\}_{D_{art}} \cup \{ev\}_{D_{dev}} \cup \{ev\}_{D_{eval}}$ is the set of evidence in Table 3. For instance, test case {Check-Code}, checking whether the used programming language is Rust, returned as output {Rust} and therefore {*Success*}. For brevity, Table 3 only reports the name of the test cases.
- $\{(\textit{Input Partition Coverage}, 0.977), (\textit{Branch Coverage}, 0.876)\}$ is a subset of the metrics in Table 4.

Table 3
Example of evidence collected by certifying s_4.

Test Case	*to*	*tr*
{Get-Orchestrator, Check-Replicas, Check-Zones}	{`HA Kubernetes`, 3, 3}	{*Success*, *Success*, *Success*}
{Dry-Run}	{`Ok`}	{*Success*}
{Check-Code}	{`Rust`}	{*Success*}
{Check-Review-Document}	{`Document exists`}	{*Success*}
{Check-Document}	{`Document exists`}	{*Success*}

Table 4
Metrics [35], [36].

Input Partition Coverage	The degree to which test cases cover the partitions of the service inputs.
Branch Coverage	The degree to which test cases cover the branches of the service.
Condition Coverage	The degree to which test cases cover the conditions of the service.
Path Coverage	The degree to which test cases cover the possible linearly independent paths of the service [37].

6 Service Selection

The certification model execution in Section 5 enables end users to select certified services according to their specific non-functional needs, increasing the trustworthiness of their systems. We assume certified services to be functionally equivalent (i.e., offer the same functionality) and match the functional users' requirements (step (4) in Figure 1). The service selection process builds on *dimension lattices* as the means to specify users' requirements on services and rank services according to their certificates. Each lattice is induced by a dimension of a non-functional property, and is defined as follows.

Definition 11 **($(\mathcal{D},\succeq_p)$).** Let D be a dimension and p a non-functional property. The dimension lattice is a pair $(\mathcal{D},\succeq_p)$, where

- $\mathcal{D}$=$V_{a_1}\times\ldots\times V_{a_n}$, with V_{a_i} being the domain of attribute a_i (Definition 1);
- $\succeq_p$ is a partial order relationship over $\mathcal{D}$ such that for each pair of elements (dimensions) $D_i,D_j\in\mathcal{D}$, $D_i\succeq_p D_j$ iff for all attributes $a_k\in D_i,D_j$ either $D_i.a_k.v>_{a_k}D_j.a_k.v$ or $D_i.a_k.v=D_j.a_k.v$.

In other words a dimension lattice contains all the possible dimensions organized according to a partial order (dominance) relationship. We note that the empty value $\{\}$ models either an attribute not having a value or an attribute whose value is unknown. Such a value is ranked last in the total order relationship in Definition 1, that is, $\forall v_j\in V_{a_k}$, $v_j >_{a_k} \{\}$. We also note that the total order relationships $>_{a_k}$ at the basis of the lattice are defined by experts (i.e., the CA, see Definition 1).

A *ranking function* R: $(\mathcal{D},\succeq_p)\rightarrow[0,1]$ assigns a value to each element D of the lattice according to the following condition: $\forall D_i,D_j\in(\mathcal{D},\succeq_p)$, $R(D_i)\geq R(D_j)$ iff $D_i\succeq_p D_j$, that is, the ranking function is compatible with the lattice ordering. We consider a standard ranking function defined as follows.

Definition 12 **(R).** The *ranking function* is defined as $R(D)$=$\frac{L(D)+1}{n}$, where D is a lattice element, $L(D)$ is the function returning the number of arcs of the minimum path from the least element to D in the corresponding Hasse diagram of the lattice, and n=$max(L(D))$+1.

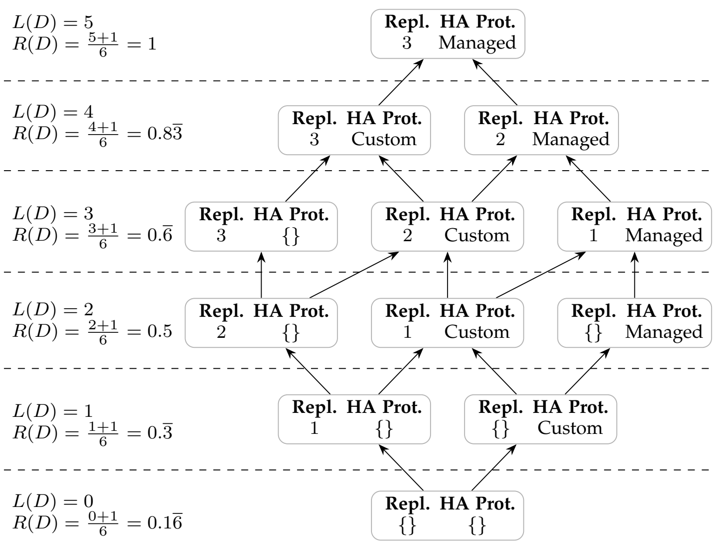
```
INPUT
{$s_1,\ldots,s_n$}: Certified services with certificates {$\mathcal{C}_1,\ldots,\mathcal{C}_n$}
{$(glb,lub)_i$}: User requirements
$W$: Preferences over dimensions
$W_m$: Preferences over metrics

OUTPUT
{$(s_j,pos_j)$}: Ranking list

SERVICE SELECTION

/* Service Filtering */
/* Filter according to requirements*/
services ← {$s_k|\forall i,\ lub_i\succeq_p \mathcal{C}_k.p.D_i\succeq_p glb_i$};

/* Service Ranking */
/* Apply VIKOR*/
{$(s_j,Q_j)$} ← VIKOR(services, $W$);
sort({$(s_j,Q_j)$});

/* Totally-Ordered Service Ranking*/
/* Compute the final ranking*/
forall $s_j\in\{(s_j,Q_j)\}$
    /* compute certificate strength using metrics*/
    $\mu_j$ ← compute_strength($\mathcal{C}_j,W_m$);
endfor;
forall $\{s_i,s_j\}\in\{(s_j,Q_j)\}\mid Q_i=Q_j$
    /* first, sort services with the same Q using strength;
    then, sort services with the same strenght randomly*/
    sort({$s_i,s_j$}, {$\mu_i,\mu_j$});
endfor;
{$(s_j,pos_j)$} ← build_ranking_list(services);
return {$(s_j,pos_j)$};
```

Figure 4. Service selection: pseudocode.

Example 6. Figure 5(a) shows an example of lattice $(\mathcal{D}_{art},\succeq_{rel})$ for dimension artifact D_{art} and property reliability. For brevity, we only consider attributes *Replicas* and *HA Prot.* (the former denoted as "Repl." in Figure 5(a)). We note that relationships $>_{\text{Repl.}}$ and $>_{\text{HA Prot.}}$ are defined by the CA. The least element of the lattice is the worst one, corresponding to $R(D)$=$0.1\overline{6}$. It refers to a service whose attributes are unknown/cannot be certified. The top element of the lattice is the optimum, corresponding to $R(D)$=1. It refers to a service with 3 replicas and a managed HA protocol. We note that elements located at the same lattice levels are equivalent and associated with the same value of ranking function, for instance, $R(D)$=0.5 for level 2, and $R(D)$=$0.8\overline{3}$ for level 4.

Service selection consists of three activities: *i) service filtering*, which collects compatible services matching users' requirements; *ii) service ranking*, which ranks compatible services according to the property they hold; *iii) totally-ordered service ranking*, which provides a totally-ordered ranking of services according to certificate metrics in Definition 10. Figure 4 shows the pseudocode of these steps, while Figures 5(a)–(d) shows an example based on services s_1–s_5 in Table 1.

Lattice $(\mathcal{D}_{art}, \succeq_{rel})$; $\succeq_{rel}$ defined by experts according to:
- $>_{\text{Repl.}} = [3 >_{\text{Repl.}} 2 >_{\text{Repl.}} 1 >_{\text{Repl.}} \{\}]$
- $>_{\text{HA Prot.}} = [\text{Managed} >_{\text{HA Prot.}} \text{Custom} >_{\text{HA Prot.}} \{\}]$

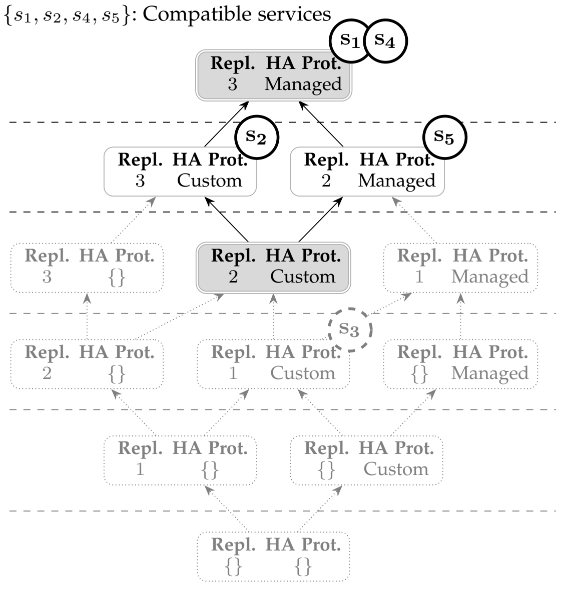


$L(D) = 5$, $R(D) = \frac{5+1}{6} = 1$
$L(D) = 4$, $R(D) = \frac{4+1}{6} = 0.8\overline{3}$
$L(D) = 3$, $R(D) = \frac{3+1}{6} = 0.\overline{6}$
$L(D) = 2$, $R(D) = \frac{2+1}{6} = 0.5$
$L(D) = 1$, $R(D) = \frac{1+1}{6} = 0.\overline{3}$
$L(D) = 0$, $R(D) = \frac{0+1}{6} = 0.1\overline{6}$
Repl. HA Prot. 3 Managed
Repl. HA Prot. 3 Custom
Repl. HA Prot. 2 Managed
Repl. HA Prot. 3 {}
Repl. HA Prot. 2 Custom
Repl. HA Prot. 1 Managed
Repl. HA Prot. 2 {}
Repl. HA Prot. 1 Custom
Repl. HA Prot. {} Managed
Repl. HA Prot. 1 {}
Repl. HA Prot. {} Custom
Repl. HA Prot. {} {}

(a) Dimension lattice.

INPUT
$\{s_1, s_2, s_3, s_4, s_5\}$: Set of services
$\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5\}$: Set of certificates
$((Repl{=}2,\ HA\ Prot.{=}\text{Custom}),$
$(Repl{=}3,\ HA\ Prot.{=}\text{Managed}))$: User requirements

OUTPUT
$\{s_1, s_2, s_4, s_5\}$: Compatible services



(b) Service filtering.

INPUT
$\{s_1, s_2, s_4, s_5\}$: Set of compatible services
$\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_4, \mathcal{C}_5\}$: Set of certificates
W: Preferences over dimensions

Serv.	$R(D_{art})$	$R(D_{dev})$	$R(D_{eval})$
s_1	1	0.4	0.6
s_2	$0.8\overline{3}$	$0.7\overline{3}$	0.8
s_4	1	1	1
s_5	$0.8\overline{3}$	$0.7\overline{3}$	0.8

$W[D_{art}]$	$W[D_{dev}]$	$W[D_{eval}]$
$0.\overline{3}$	$0.\overline{3}$	$0.\overline{3}$

OUTPUT
$\{(s_i, Q_i)\}$: Ranking list of service s_i and corresponding index Q_i

Serv.	Q	
$\mathbf{s_4}$	**0**	← compromise solution
s_2, s_5	0.986	
s_1	1	

(c) Service ranking.

INPUT
$\{(s_j, Q_j)\}$: Ranking list according to certificates $\{\mathcal{C}_j\}$
W_m: Preferences over metrics

Serv.	*IPC*	*BC*
s_1	0.331	0.259
s_3	0.613	0.373
s_4	0.977	0.876
s_5	0.856	0.574

W_m[*IPC*]	W_m[*BC*]
0.5	0.5

OUTPUT
$\{s_j\}$: Fully sorted ranking list

Serv.	Q
$\mathbf{s_4}$	**0**
s_2, s_5	0.986
s_1	1

$\bigwedge$

Cert.	μ
s_2	0.493
s_5	0.715

$\Longrightarrow$

Serv.	
$\mathbf{s_4}$	← best solution
s_5	
s_2	
s_1	

(d) Absolute ranking.

Figure 5. An example of a dimension lattice (a) and of the three activities of service selection (b)–(c)–(d) based on services in Table 1.

6.1 Service Filtering

Service filtering receives as input a set of certified services and user's requirements, and returns as output a subset of compatible services addressing the requirements. Given a non-functional property p, a *requirement* insisting on certified services holding p is a set of the form $\{(glb,\ lub)_1,\ \ldots,\ (glb,\ lub)_n\}$, where each $(glb,\ lub)_i$ defines the greatest lower bound (glb_i) and the least upper bound (lub_i) on the lattice induced by dimension D_i of p.

Service filtering operates on each dimension independently, then combines the results as follows.

Definition 13 (Service Filtering). Let $\{s_1,\ \ldots,\ s_n\}$ be a set of services, $\{\mathcal{C}_1,\ \ldots,\ \mathcal{C}_n\}$ the set of corresponding certificates for property p, and $\{(glb,\ lub)_i\}$ user's requirements for each dimension D_i. The result of service filtering is the set $\{s_k | \forall i,\ lub_i \succeq_p \mathcal{C}_k.p.D_i \succeq_p glb_i\}$ of compatible services.

In other words, service filtering returns the set of compatible services satisfying users' requirements in all dimensions and lattices, according to partial order relationship $\succeq_p$ in Definition 11.

Example 7. Following Example 6, Figure 5(b) shows an example of service filtering. For brevity, we report only dimension D_{art}. User requirement $\{(Repl{=}2, HA\ Prot.{=}\text{Custom}), (Repl{=}3, HA\ Prot.{=}\text{Managed})\}$ indicates glb and lub in Definition 13, resp., depicted as grey-filled nodes in Figure 5(b). Services whose non-functional property is within glb and lub are kept for the following activities, namely s_1, s_2, s_4, s_5, while s_3, having a property dominated by glb, is filtered out.

6.2 Service Ranking

Service ranking ranks compatible services according to the non-functional property they hold. This activity is modeled as a Multi-Criteria Decision-Making (MCDM) problem, ranking n alternatives (the certified services) according to m (weighted) criteria (the dimensions forming the property in terms of ranking function in Definition 12). Each criterion is associated with a weight reflecting the importance thereof in the ranking. In our case, each dimension D_i is associated with a weight $W[i] \in [0, 1]$.

We use VIKOR [38] to find a compromise solution, that is, the closest to the ideal among conflicting criteria. It receives as input the set $\{s_1, \ldots, s_n\}$ of compatible services in Definition 13, the corresponding set of certificates $\{\mathcal{C}_1, \ldots, \mathcal{C}_n\}$ and a vector W of weights, and returns as output a ranking of the services, identifying the best services in terms of their non-functional property. The weights of the vector must sum to 1, formally $\sum_{i=0}^{|W|} W[i]{=}1$. VIKOR-based service ranking is a 5-step process working as follows.

Step 1. It takes as input the set $\{s_j\}$ of services, the corresponding set of certificates $\{\mathcal{C}_j\}$ and the weight vector W, and computes the positive-ideal (Equation 1) and the negative-ideal (Equation 2) alternative for all dimensions $D_1, \ldots, D_m$, as follows.

$$R_i^+ = \max_{j=1,\ldots,n} R(\mathcal{C}_j.\mathcal{M}.p.D_i) \tag{1}$$

$$R_i^- = \min_{j=1,\ldots,n} R(\mathcal{C}_j.\mathcal{M}.p.D_i) \tag{2}$$

where $j{=}1,\ldots,n$ is the number of certified services; $i{=}1,\ldots,m$ is the number of dimensions of the non-functional property; $R(\mathcal{C}_j.\mathcal{M}.p.D_i)$ is the value assigned by the ranking function in Definition 12 to the i-th dimension of the j-th service. The positive-ideal R_i^+ (negative-ideal R_i^-, resp.) represents the best (worst, resp.) certified service in terms of the non-functional property in the i-th dimension.

Step 2. It computes the group maximum utility U_j and the minimum individual regret Z_j of a certified service j, according to the following L_p-metric.

$$L_{p,j} = \left[\sum_{i=1}^{m} \left(W[i] \times \frac{R_i^+ - R(\mathcal{C}_j.\mathcal{M}.p.D_i)}{R_i^+ - R_i^-} \right)\right]^{1/p} \tag{3}$$

where $1{\leq}p{\leq}\infty$. $L_{p,j}$ is the (normalized) distance between service j and the positive ideal service. Based on such a metric, we compute group maximum utility U_j (Equation 4) and minimum individual regret Z_j (Equation 5) of a certified service j as follows.

$$L_{1,j} = U_j = \sum_{i=1}^{m} W[i] \times \frac{R_i^+ - R(\mathcal{C}_j.\mathcal{M}.p.D_i)}{R_i^+ - R_i^-} \tag{4}$$

$$L_{\infty,j} = Z_j = \max_{i=1,\ldots,m} \left[W[i] \times \frac{R_i^+ - R(\mathcal{C}_j.\mathcal{M}.p.D_i)}{R_i^+ - R_i^-} \right] \tag{5}$$

Step 3. It computes the sorting index Q_j for each service j using Equation 6.

$$Q_j = v \times \frac{U_j - U^+}{U^- - U^+} + (1 - v) \times \frac{Z_j - Z^+}{Z^- - Z^+} \tag{6}$$

where $U^+{=}\min_{j=1,\ldots,n} U_j$, $U^-{=}\max_{j=1,\ldots,n} U_j$, $Z^+{=}\min_{j=1,\ldots,n} Z_j$, $Z^-{=}\max_{j=1,\ldots,n} Z_j$. Parameter $v{\in}[0, 1]$ is a decision threshold; $v{>}0.5$ models *voting by majority*, $v{=}0.5$ models *consensus*, $v{<}0.5$ models *veto*.

Step 4. It ranks the services according to values U, Z, Q of corresponding certificates in ascending order, producing three ranking lists.

Step 5. It proposes the best compromise service having the lowest value according to index Q, denoted as Q', if the following conditions are met:

1) Q' provides an *acceptable advantage* over the second lowest value in Q, denoted as Q'', that is, $Q''{-}Q'{\geq}1/(n-1)$, with n being the number of alternatives (services).

2) Q' provides an *acceptable stability in the decision making*, that is, Q' is ranked the best also by U and/or Z.

If condition *1)* is not satisfied, the set of compromise solutions consists of Q', Q'', ..., $Q^{(k)}$, where $Q^{(k)}$ is given by the relation $Q^{(k)}{-}Q'{<}1/(n{-}1)$, for maximum k. In other words, it returns all the solutions laying in the *acceptable advantage interval*.

If only condition *2)* is not satisfied, the set of compromise solutions consists of Q' and Q'', since there is no *decision-making stability* and Q', Q'' represent the same compromise.

Example 8. Following Example 7, Figure 5(c) shows an example of service ranking. Values of ranking function R are taken according to the lattices of the three dimensions and the values in Table 1. Service s_4 is the only compromise solution, having the lowest value for index Q and both conditions satisfied. In particular, condition *1)* is satisfied because $Q_2 {-} Q_4 \geq 1/(n{-}1)$, as $0.986{\geq}0.\overline{3}$. We note that services s_2 and s_5 are equivalent, that is, $Q_2{=}Q_5$, and at the same ranking position.

6.3 Totally-Ordered Service Ranking

The index Q, along with the results of conditions *1)* and *2)*, are not enough to compute a total ordering of certified services. The likelihood of having different services with the same ranking is in fact not negligible and, when condition *1)* does not hold, can result in all services tied for first. The reason lies in how a dimension is mapped into a number given as input to VIKOR. In our case, it is computed by ranking function R in Definition 12, and only depends on the position of the dimension in the lattice, calculated as the distance between the current dimension and the least dimension. Hence, even on lattices with a large number of elements, the number of possible outputs of R is typically small. To this aim, we follow the approach in [35] and further compare equivalent services using metrics in Definition 10. Metrics express the strength of the collected evidence supporting the certified non-functional property, and in turn the strength of the certificate.

The totally-ordered ranking is retrieved according to a function that takes as input *i)* the ranking $\{(s_j, Q_j)\}$ returned by VIKOR (Section 6.2), where s_j is a compatible service with its certificate $\mathcal{C}_j$ and Q_j the corresponding VIKOR index; *ii)* a vector of weights $W_m \in [0,1]$ expressing the importance of the metrics, such that $\sum_{i=0}^{|W_m|} W_m[i]=1$, with $|W_m|$ the number of metrics. We note that, for simplicity, a set of predefined vectors can be used. The function producing the totally-ordered ranking works as follows.

Step 1. It takes as input the ranking list and the weight vector W_m, and produces as output the strength μ_j of each certificate $\mathcal{C}_j$, according to Equation 7.

$$\mu_j = \sum_{i=0}^{|W_m|} W_m[i] \times \mathcal{C}_j.m_i.v \tag{7}$$

In other words, certificate strength μ_j is computed as a weighted sum of the metrics contained in the certificate.

Step 2. It takes as input the certificate strengths and the ranking list, and returns as output a total ordering, computed by refining the VIKOR ordering according to the certificate strength. Formally, a service s_i is ranked higher (is better) than a service s_j iff one of the following holds.

1) $Q_i < Q_j$

2) $Q_i = Q_j \wedge \mu_i > \mu_j$

Condition *1)* states that s_i is better than s_j if VIKOR index Q_i is lower. In this case, no additional sorting is needed. Condition *2)* states that, when the VIKOR index of the two services is equal, certificate strength is used to provide a total order. In other words, services are ranked according to their VIKOR index first and, in case they are equal, to their certificate strength. If there are services still ranked at the same position, random sort is used.

Example 9. Following Example 8, Figure 5(d) shows an example of totally-ordered ranking. For brevity, we consider only metrics *Input Partition Coverage* and *Branch Coverage* in Table 4, abbreviated as *IPC* and *BC*, resp. Totally-ordered ranking disambiguates between services having the same VIKOR index, namely s_2 and s_5, according to certificate strength μ. The resulting totally-ordered ranking is therefore s_4, s_5, s_2, s_1.

7 Experiments

We experimentally evaluated the performance and quality of the proposed approach in a simulated environment. Experiments have been run on a laptop equipped with an Intel® Core i7-5500U @ 2.4 GHz (2 cores, 4 threads), 16 GBs of RAM, operating system Ubuntu 20.04 x64, Java runtime OpenJDK 11.0.10, Python runtime 3.8.6. We compared our approach with state-of-the-art certification schemes (e.g., [6], [11]) that, according to assumptions A3) and A4) in Section 2, evaluate software artifacts only. However, since source code, as well as precise modeling, of existing solutions are generally not available, we compared our solution with an approximation of the state of the art, instantiating our scheme on dimension D_{art} only.

We evaluated our scheme in terms of *i)* performance, measuring the execution time of its phases (Section 7.1); and *ii)* quality, comparing the result of service selection against the state of the art and the global optimum (Section 7.2).

7.1 Performance

We evaluated the performance of our approach by running two experiments measuring the execution time of lattice building and service ranking.

The first experiment measured the time needed to construct the data structure holding the dimension lattice in Definition 11, using jhpl, an optimized Java library modeling lattices as sets of tries.[2] We generated lattices with a large number of elements varying the number of attributes in 1, 3, 6, 9 and the number of possible values of each attribute in 15, 25, 50, 75, 100. Figures 6(a)–(b) show that the time needed for lattice building is negligible, never exceeding 0.025 milliseconds.

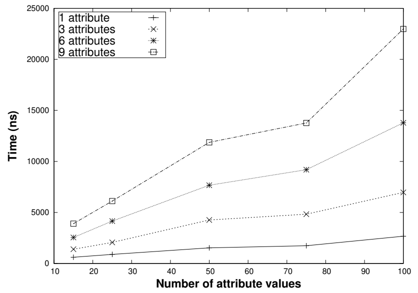
The second experiment measured the time needed to perform service ranking in Section 6.2. We implemented VIKOR on top of a Python library optimized for row- and column-wise operations,[3] and varied the number of services in 100, 300, 600, 900, 1200, 1500, 1800 and dimensions in 1, 3, 6, 9, 12. Figures 6(c)–(d) show that the performance depends only on the number of services involved, while the impact of the number of dimensions is negligible. The reason is that the time complexity is $\mathcal{O}(|\text{services}| \cdot |\text{dimensions}|) \approx \mathcal{O}(|\text{services}|)$, because the number of services is typically one or more orders of magnitude larger than the number of dimensions. Even in a worst case scenario, the execution time is very low, never exceeding 0.9 seconds. We note that our experiments did not measure the time for building a totally-ordered service ranking in Section 6.3, since its cost is the one of a sorting algorithm and therefore well-known, that is, $\mathcal{O}(|\text{services}| \cdot \log|\text{services}|)$.

7.2 Quality

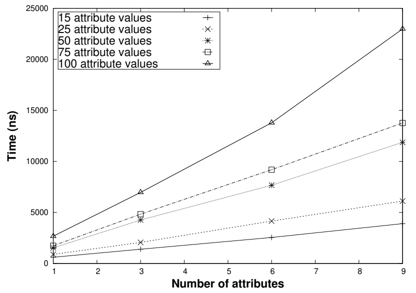
We evaluated the quality of our multi-dimensional service ranking with respect to two approaches: *i)* global optimum, and *ii)* state-of-the-art. The optimum approach is a manual approach retrieving the service with highest quality, while the state-of-the-art approach only considers dimension artifacts D_{art}. Quality evaluation analyzes *i)* the cumulative penalty introduced by each dimension with respect to the optimum approach (Section 7.2.1), *ii)* the similarities between the optimum ranking and the other approaches (Section 7.2.2).

Table 5 presents the experimental settings varying the number of dimensions in 3, 6, 9. For each setting, we defined 5 different configurations in the form of weights rating the importance of each single dimension. In particular, we considered different classes of weights: *i) increasing weights*, where each dimension i is more important than dimension

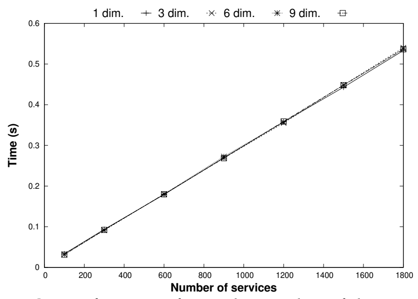
2. https://github.com/prasser/jhpl
3. https://pandas.pydata.org/



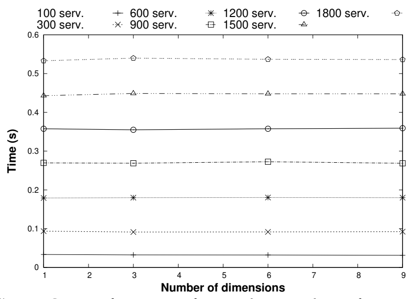
(a) Lattice building performance fixing the number of attributes.



(b) Lattice building performance fixing the number of attribute values.



(c) VIKOR performance fixing the number of dimensions.



(d) VIKOR performance fixing the number of services.

Figure 6. Performance of lattice building and VIKOR execution.

$i-1$ ($C3.1$, $C6.1$, $C9.1$ in Table 5); *ii) balanced weights*, where all dimensions have the same importance ($C3.2$, $C6.2$, $C9.2$ in Table 5); *iii) few-prevailing weights*, where three dimensions with similar weights have more importance ($C3.3$, $C6.3$, $C9.3$ in Table 5); *iv) decreasing weights*, where each dimension i is less important than dimension $i-1$ ($C3.4$, $C6.4$, $C9.4$ in Table 5); *v) unbalanced weights*, where one dimension has much higher importance ($C3.5$, $C6.5$, $C9.5$ in Table 5). We then randomly generated 38,400 certificates, covering the entire domain of our ranking function in Definition 12. For each configuration, we randomly split certificates in 30 data sets and *i)* manually calculated the optimum ranking, *ii)* executed our service ranking based on VIKOR, using the consensus decision threshold v=0.5 (see Step 3 in Section 6.2), and *iii)* executed the state-of-the-art ranking, where services have been ranked according to dimension D_{art}, that is, the first dimension in each setting. We finally averaged results retrieved according to each configuration. We note that, for simplicity, we assumed Service Filtering in Section 6.1 to return all certified services; this choice does not impact on the quality of our experiments as it applies to all approaches (global optimum, our approach, state of the art).

Table 5
Experimental settings

Abbr.	Dim.	W
$C3.1$	3	$[0.1, 0.3, 0.6]$
$C3.2$	3	$[0.\overline{3}, 0.\overline{3}, 0.\overline{3}]$
$C3.3$	3	$[0.35, 0.4, 0.25]$
$C3.4$	3	$[0.6, 0.3, 0.1]$
$C3.5$	3	$[0.8, 0.1, 0.1]$
$C6.1$	6	$[0.02, 0.065, 0.135, 0.20, 0.26, 0.32]$
$C6.2$	6	$[0.1\overline{6}, 01\overline{6}, 0.1\overline{6}, 0.1\overline{6}, 0.1\overline{6}, 0.1\overline{6}]$
$C6.3$	6	$[0.175, 0.20, 0.175, 0.15, 0.15, 0.15]$
$C6.4$	6	$[0.32, 0.26, 0.20, 0.135, 0.065, 0.02]$
$C6.5$	6	$[0.6, 0.08, 0.08, 0.08, 0.08, 0.08]$
$C9.1$	9	$[0.00260417, 0.00520833, 0.0078125, 0.015625, 0.03125, 0.0625, 0.125, 0.25, 0.5]$
$C9.2$	9	$[0.\overline{1}, 0.\overline{1}, 0.\overline{1}, 0.\overline{1}, 0.\overline{1}, 0.\overline{1}, 0.\overline{1}, 0.\overline{1}, 0.\overline{1}]$
$C9.3$	9	$[0.2, 0.3, 0.2, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05]$
$C9.4$	9	$[0.5, 0.25, 0.125, 0.0625, 0.03125, 0.015625, 0.0078125, 0.00520833, 0.00260417]$
$C9.5$	9	$[0.4, 0.075, 0.075, 0.075, 0.075, 0.075, 0.075, 0.075, 0.075]$

7.2.1 Captured Quality

A penalty metric $P(s)$ is first defined evaluating the degree to which a service s (and its certificate) diverges from the global optimum as follows.

$$P(s) = \sum_{i=0}^{n} W[i] \times \frac{max(D_i) - R(\mathcal{C}.\mathcal{M}.p.D_i)}{max(D_i)} \qquad (8)$$

where $max(D_i)$ is the highest value for the i-th dimension among all certified services and $R(\mathcal{C}.\mathcal{M}.p.D_i)$ is the value of the same dimension for the service under evaluation, both according to the ranking function in Definition 12. We note that penalty $P(s)$ is the sum of the normalized penalties contributed by each dimension.

The quality $QU(s)$ of a service s is then defined as follows.

$$QU(s) = 1 - \frac{P(s) - min(P)}{max(P) - min(P)} \qquad (9)$$

where $min(P)$ and $max(P)$ are the minimum and maximum penalties among all certified services, resp. $QU(s)$=0 is obtained when s has the lowest quality among all services, $QU(s)$=1 is obtained when s has the highest quality.

We retrieved the 10 best services in each approach as follows: *i) global optimum*: the 10 services with highest quality; *ii) our scheme*: the 10 best services according to service ranking; *iii) state of the art*: the 10 services with highest value of ranking function in dimension D_{art}. We then calculated

Table 6
Summary of the captured quality

Dim.	Our appr.	State of art
3	0.9178	0.755
6	0.9231	0.7166
9	0.924	0.6887
AVG	0.9217	0.7201

(a) Captured quality

Dim.	Our appr.	State of art
3	72	16
6	70	16.667
9	76.6667	5
AVG	72.889	12.556

(b) N. of occurrence when the global optimum is captured (%)

Dim.	Glob. opt.	Our appr.	State of art
3	0.5316	0.4825	0.5872
6	0.3253	0.2876	0.3091
9	0.3007	0.2681	0.3409
AVG	0.3859	0.3461	0.4124

(c) Highest normalized contribution to penalty

Table 7
Kendall's τ (K) and Spearman's footrule (S) values

Config	K		S	
	Our Appr.	St. of Art	Our Appr.	St. of Art
$C3.1$	0.0782	0.4539	0.1129	0.6129
$C3.2$	0.116	0.3046	0.1665	0.4233
$C3.3$	0.1088	0.2977	0.1562	0.4143
$C3.4$	0.0785	0.1544	0.1135	0.2186
$C3.5$	0.0276	0.0562	0.041	0.08
$C3.AVG$	0.0818	0.2534	0.118	0.3498
$C6.1$	0.1212	0.489	0.1727	0.6531
$C6.2$	0.1539	0.368	0.2186	0.5053
$C6.3$	0.1444	0.3618	0.2056	0.4974
$C6.4$	0.1216	0.2693	0.1736	0.3758
$C6.5$	0.0471	0.0915	0.0678	0.1298
$C6.AVG$	0.1176	0.3159	0.1677	0.4323
$C9.1$	0.0857	0.4981	0.1234	0.6643
$C9.2$	0.1719	0.393	0.2435	0.5383
$C9.3$	0.1195	0.3489	0.1712	0.4807
$C9.4$	0.0869	0.1664	0.1247	0.2349
$C9.5$	0.0847	0.1543	0.121	0.218
$C9.AVG$	0.1097	0.3121	0.1568	0.4272
AVG	0.1030	0.2938	0.1475	0.4031

the average quality of the 10 selected services in each individual data set and configuration. Table 6(a) and Figures 7(a)–(c) summarize our results. Our approach captures 92% of the quality of the optimum approach on average, that is, $Avg(QU(s))$=92%, compared to 72% of the state of the art. As expected, the only settings where the quality of the state of the art is close to the quality of our approach are those where dimension D_{art} has (much) higher importance (i.e., $C3.5$, $C6.5$, $C9.5$). Furthermore, our approach provides a consistent quality of 92% on average in all configurations, while the quality of the state-of-the-art approach decreases from 76% (3 dimensions) to 72% (6 dimensions) and 69% (9 dimensions) on average. Table 6(b) and Figures 7(d)–(f) show the number of times, in percentage, when the first-ranked service of our and state-of-the-art approaches is the optimum one. Our approach retrieves the optimum in 73% of the cases, while the state-of-the-art approach in only 13% of the cases. Even when the optimum is not reached, our approach always provides a remarkable quality ≥85%, that, in most (98%) of the cases, is ≥90%.

A second important aspect is the contribution that each dimension gives to the penalty, that is, the distribution of the penalty among the dimensions. To this aim, we computed the highest normalized contribution to penalty of a service s according to the following Equation.

$$\frac{1}{P(s)} \times \max_i \left(W[i] \times \frac{max(D_i) - R(\mathcal{C}.\mathcal{M}.p.D_i)}{max(D_i)} \right) \quad (10)$$

Table 6(c) and Figures 7(g)–(i) show our results. In our approach, the highest contribution to penalty is 35% on average, compared to 39% of the global optimum and 41% of the state-of-the-art. This means that our ranking favors *balanced* services, reducing scenarios in which selected services show high variance in dimension penalty.

7.2.2 Ranking Evaluation

We finally compared the ranking produced by our approach with respect to state of the art and global optimum according to two metrics measuring distances between ranking lists: *i) Kendall's τ distance* [39] and *ii) Spearman's footrule distance* [40]. We used the experimental configurations adopted in Section 7.2.1 with totally-ordered service ranking, using metrics in Table 4 with random values and equally-distributed weights. This way, metrics values are given randomly both to high- and low-quality services, and certificate strength in Section 6.3 is entirely computed on random values.

The Kendall's τ distance counts the number of pairwise disagreements between two ranking lists. Let σ_q be the global optimum ranking list, σ_v the VIKOR-based ranking list, an σ_s the state-of-the-art ranking list, and $\sigma(i)$ the notation indicating the rank of element i in a given ranking list σ. Kendall's τ distance of rankings σ_v, σ_s with respect to σ_q is defined as

$$K(\sigma_y, \sigma_q) = K(\sigma_y) = \sum_{(i,j):i<j}^{n} inv(i,j) \quad (11)$$

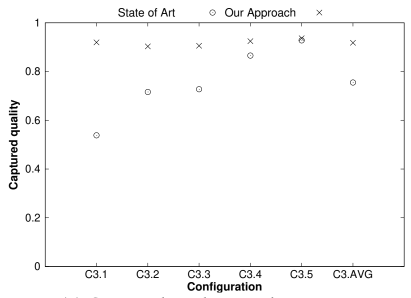
where $y \in \{v, s\}$ and $inv(i,j)$ returns 1 if $(\sigma_q(i) < \sigma_y(j) \land \sigma_q(i) > \sigma_y(j)) \lor (\sigma_q(i) > \sigma_y(j) \land \sigma_q(i) < \sigma_y(j))$, 0 otherwise. Such a distance can be normalized in the range $[0,1]$ by dividing by the highest possible value $\frac{n(n-1)}{n}$.

The Spearman's footrule distance measures the total displacement between σ_v, σ_s and σ_q, and is defined as

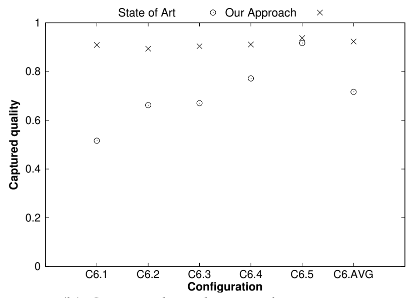
$$S(\sigma_y, \sigma_q) = S(\sigma_y) = \sum_{i}^{n} |\sigma_q(i) - \sigma_y(i)| \quad (12)$$

where $y \in \{v, s\}$. Such a distance can be normalized in the range $[0,1]$ by dividing by the highest possible value $\frac{n^2}{2}$.

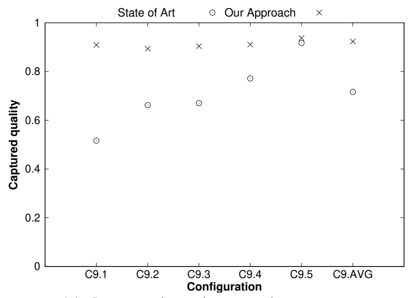
Table 7 summarizes our results. In all cases, our ranking σ_v outperformed the state of art. The number of pairwise disagreements $K(\sigma_v)$ and the total displacement $S(\sigma_v)$ are, on average, ≈3 times less than state-the-of-art $K(\sigma_s)$ and $S(\sigma_v)$. More in detail, σ_v requires to reorder only 10% of all pairs, compared to 29% of the state of art; it also shows a total displacement of 15%, compared to 40% of the state of art. Therefore, our approach improves the state of the art



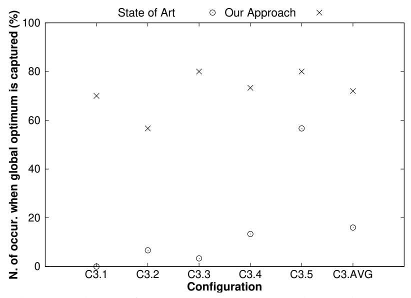
(a) Captured quality – 3 dimensions.



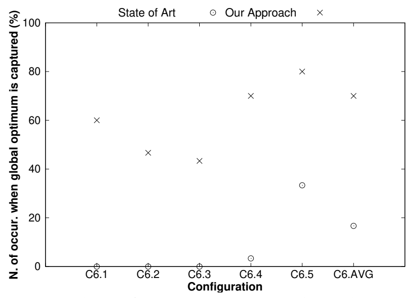
(b) Captured quality – 6 dimensions.



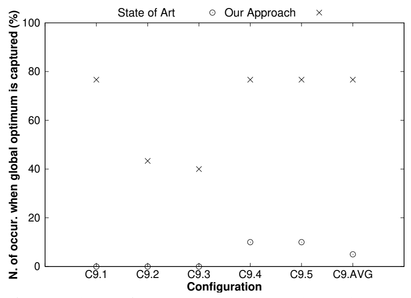
(c) Captured quality – 9 dimensions.



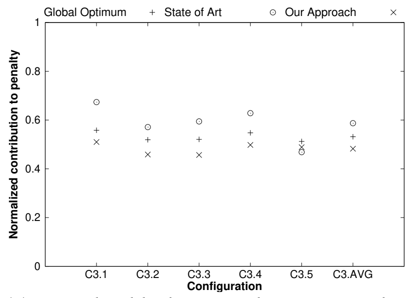
(d) Number of occurrence (%) when the optimum according to the proposed approaches is the global optimum – 3 dimensions.



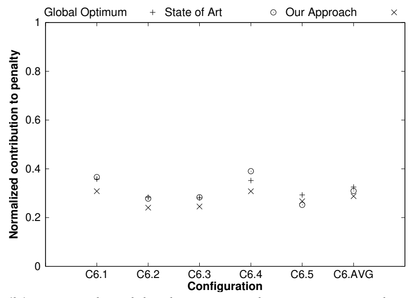
(e) Number of occurrence (%) when the optimum according to the proposed approaches is the global optimum – 6 dimensions.



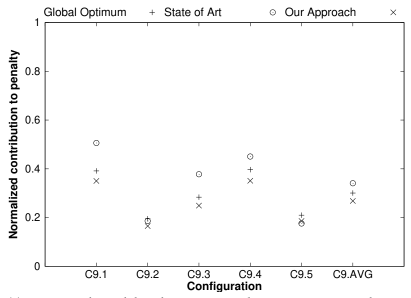
(f) Number of occurrence (%) when the optimum according to the proposed approaches is the global optimum – 9 dimensions.



(g) Normalized highest contribution to penalty – 3 dimensions.



(h) Normalized highest contribution to penalty – 6 dimensions.



(i) Normalized highest contribution to penalty – 9 dimensions.

Figure 7. Quality Evaluation.

producing a ranking that, in terms of the aforementioned distances, is similar to a global optimum-based approach.

To conclude, our experiments show that the proposed ranking has a good quality. It outperformed the state of the art, capturing 92% of the quality of the global optimum in all settings (Table 6(a)), while better approaching the global optimum-based ranking, almost six times more than the state of the art (Table 6(b)). Furthermore, our approach has the lowest value of the highest normalized contribution to penalty (Table 6(c)). This means that it favors *balanced* services, guaranteeing good quality and low variation among all dimensions.

8 Conclusions

An important goal of the evolution of ICT is to combine the opportunities provided by modern distributed systems composed of several services in terms of efficiency, flexibility and added-value applications, with an adequate level of trustworthiness over system behavior. The approach in this paper provides a concrete solution towards this goal, defining a novel certification scheme that goes beyond the simple evaluation of a service behavior, and considers additional factors describing, for instance, how the service has been implemented and verified, to increase the quality and performance of a distributed system. The proposed scheme enables a multi-dimensional certification approach based on a novel and fine-grained definition of non-functional properties, where the certification execution is split in different and logically-separated domains called dimensions. This modeling leads to more accurate certificates and, consequently, more accurate decision-making when services/software are dynamically selected at run time on the basis of their certificates. Our approach provides benefits for all the involved parties: service providers obtain certificates better reflecting all the best practices they followed; end users take decisions on more detailed and well-structured certificates; the CA provides a more useful certification scheme.

Acknowledgments

Research supported, in parts, by EC H2020 Project CONCORDIA GA 830927 and Università degli Studi di Milano under the program “Piano sostegno alla ricerca”.

References

[1] MIT Tech. Review. (2017) After Deployment Storms, Skies Turn Sunny for Multi-Cloud Environments.

[2] C. A. Ardagna, R. Asal, E. Damiani, and Q. H. Vu, "From Security to Assurance in the Cloud: A Survey," *ACM CSUR*, vol. 48, no. 1, August 2015.

[3] P. Stephanow, G. Srivastava, and J. Schütte, "Test-Based Cloud Service Certification of Opportunistic Providers," in *Proc. of IEEE CLOUD 2016*, San Francisco, CA, USA, June–July 2016.

[4] I. Kunz and P. Stephanow, "A Process Model to Support Continuous Certification of Cloud Services," in *Proc. of IEEE AINA 2017*, Taipei, Taiwan, March 2017.

[5] A. J. Muñoz-Gallego and J. Lopez, "A Security Pattern for Cloud service certification," in *Proc. of SugarLoaf PLoP 2018*, Valparaiso, Chile, November 2018.

[6] M. Anisetti, C. A. Ardagna, E. Damiani, and F. Gaudenzi, "A Semi-Automatic and Trustworthy Scheme for Continuous Cloud Service Certification," *IEEE TSC*, vol. 13, no. 1, January–February 2020.

[7] S. P. Kaluvuri, M. Bezzi, and Y. Roudier, "Bringing Common Criteria Certification to Web Services," in *Proc. of IEEE SERVICES 2013*, Santa Clara, CA, USA, June–July 2013.

[8] V. Lotz, "Cybersecurity Certification for Agile and Dynamic Software Systems – a Process-Based Approach," in *Proc. of EuroS&PW*, Genoa, Italy, September 2020.

[9] C. A. Ardagna, R. Asal, E. Damiani, T. Dimitrakos, N. E. Ioini, and C. Pahl, "Certification-Based Cloud Adaptation," *IEEE TSC*, vol. 14, no. 1, January–February 2021.

[10] D. S. Herrmann, *Using the Common Criteria for IT security evaluation*. CRC Press, 2002.

[11] P. Stephanow and N. Fallenbeck, "Towards Continuous Certification of Infrastructure-as-a-Service Using Low-Level Metrics," in *Proc. of IEEE UIC-ATC-ScalCom*, Beijing, China, August 2015.

[12] S. Lins, S. Schneider, J. Szefer, S. Ibraheem, and A. Ali, "Designing Monitoring Systems for Continuous Certification of Cloud Services: Deriving Meta-requirements and Design Guidelines," *CAIS*, vol. 44, 2019.

[13] M. Alhamad, T. Dillon, and E. Chang, "SLA-Based Trust Model for Cloud Computing," in *Proc. of NBIS 2010*, Takayama, Japan, September 2010.

[14] Y. Du, X. Wang, L. Ai, and X. Li, "Dynamic Selection of Services under Temporal Constraints in Cloud Computing," in *Proc. of ICEBE 2012*, Hangzhou, China, September 2012.

[15] R. Shaikh and M. Sasikumar, "Trust Model for Measuring Security Strength of Cloud Computing Service," *Procedia Computer Science*, vol. 45, 2015.

[16] M. Eisa, M. Younas, K. Basu, and H. Zhu, "Trends and Directions in Cloud Service Selection," in *Proc. of IEEE SOSE 2016*, Oxford, UK, March, April 2016.

[17] F. Jrad, J. Tao, A. Streit, R. Knapper, and C. Flath, "A utility-based approach for customised cloud service selection," *International Journal of Computational Science and Engineering*, vol. 10, no. 1-2, 2015.

[18] S. Ding, Z. Wang, D. Wu, and D. L. Olson, "Utilizing customer satisfaction in ranking prediction for personalized cloud service selection," *Decision Support Systems*, vol. 93, January 2017.

[19] T. Halabi and M. Bellaiche, "A broker-based framework for standardization and management of Cloud Security-SLAs," *COSE*, vol. 75, 2018.

[20] C. Redl, I. Breskovic, I. Brandic, and S. Dustdar, "Automatic SLA Matching and Provider Selection in Grid and Cloud Computing Markets," in *Proc. of ACM/IEEE Grid 2012*, Beijing, China, September 2012.

[21] A. Taha, R. Trapero, J. Luna, and N. Suri, "A Framework for Ranking Cloud Security Services," in *Proc. of IEEE SCC 2017*, Honolulu, HI, USA, September 2017.

[22] A. Taha, S. Manzoor, and N. Suri, "SLA-Based Service Selection for Multi-Cloud Environments," in *Proc. of IEEE EDGE 2017*, Honolulu, HI, USA, September 2017.

[23] K. J. Modi and S. Garg, "A QoS-based approach for cloud-service matchmaking, selection and composition using the Semantic Web," *Journal of Systems and Information Technology*, vol. 21, no. 1, Jan 2019.

[24] A. B. de Oliveira Dantas, F. H. de Carvalho Junior, and L. S. Barbosa, "A component-based framework for certification of components in a cloud of HPC services," *Science of Computer Programming*, vol. 191, June 2020.

[25] D. Cotroneo, L. De Simone, and R. Natella, "Dependability Certification Guidelines for NFVIs through Fault Injection," in *Proc. of IEEE ISSREW 2018*, Memphis, TN, USA, October 2018.

[26] F. Nawab, "WedgeChain: A Trusted Edge-Cloud Store With Asynchronous (Lazy) Trust," in *Proc. of IEEE ICDE 2021*, Chania, Greece, April 2021.

[27] A. J. H. Simons and R. Lefticaru, "A verified and optimized Stream X-Machine testing method, with application to cloud service certification," *Software Testing, Verification and Reliability*, vol. 30, no. 3, May 2020.

[28] R. Calinescu, D. Weyns, S. Gerasimou, M. U. Iftikhar, I. Habli, and T. Kelly, "Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases," *IEEE TSE*, vol. 44, no. 11, November 2018.

[29] S. Jahan, I. Riley, C. Walter, R. F. Gamble, M. Pasco, P. K. McKinley, and B. H. Cheng, "MAPE-K/MAPE-SAC: An interaction framework for adaptive systems with security assurance cases," *FGCS*, vol. 109, August 2020.

[30] J. Prüfer, "Trusting privacy in the cloud," *Information Economics and Policy*, vol. 45, December 2018.

[31] S. Pape, F. Paci, J. Jürjens, and F. Massacci, "Selecting a Secure Cloud Provider – An Empirical Study and Multi Criteria Approach," *Information*, vol. 11, no. 5, 2020.

[32] J. Lansing, A. Benlian, and A. Sunyaev, ""Unblackboxing" Decision Makers' Interpretations of IS Certifications in the Context of Cloud Service Certifications," *JAIS*, vol. 19, 2018.

[33] G. McGraw, "Software security," *IEEE Security & Privacy*, vol. 2, no. 2, March–April 2004.

[34] H. Teigeler, S. Lins, and A. Sunyaev, "Drivers vs. Inhibitors - What Clinches Continuous Service Certification Adoption by Cloud Service Providers?" in *Proc. of HICSS 2018*, Waikoloa, HI, USA, January 2018.

[35] M. Anisetti, C. A. Ardagna, E. Damiani, and F. Saonara, "A Test-based Security Certification Scheme for Web Services," *ACM TWEB*, vol. 7, no. 2, May 2013.

[36] G. J. Myers, T. Badgett, T. M. Thomas, and C. Sandler, *The art of software testing*. Wiley Online Library, 2004, vol. 2.

[37] T. J. McCabe, "A Complexity Measure," *IEEE TSE*, vol. SE-2, no. 4, December 1976.

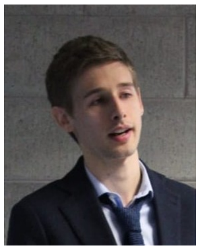
[38] S. Opricovic and G.-H. Tzeng, "Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS," *EJOR*, vol. 156, no. 2, 2004.

[39] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, 1938.

[40] C. Spearman, "The proof and measurement of association between two things," *International Journal of Epidemiology*, vol. 39, 1904.

Marco Anisetti is an Associate Professor at the Università degli Studi di Milano. His research interests are in the area of computational intelligence, and its application to the design and evaluation of complex systems. He has been investigating innovative solutions in the area of cloud security assurance evaluation. In this area he defined a new scheme for continuous and incremental cloud security certification, based on distributed assurance evaluation architecture.

Claudio A. Ardagna is Full Professor at the Università degli Studi di Milano, the Director of the CINI National Lab on Big Data, and co-founder of Moon Cloud srl. His research interests are in the area of cloud-edge security and assurance, and data science. He has published more than 140 contributions in international journals, conference/workshop proceedings, and chapters in international books. He has been visiting researcher at BUTP, Khalifa University, GMU.



Nicola Bena is a Ph.D. student at the Università degli Studi di Milano. His research interests are in the area of security of modern distributed systems with particular reference to certification, assurance, and risk management techniques.