Non-Functional Certification of Modern Distributed Systems: A Research Manifesto

Claudio A. Ardagna
Department of Computer Science
Università degli Studi di Milano
Milan, Italy
claudio.ardagna@unimi.it

Nicola Bena
Department of Computer Science
Università degli Studi di Milano
Milan, Italy
nicola.bena@unimi.it

***Abstract*—The huge progress of ICT is radically changing distributed systems at their roots, modifying their operation and engineering practices and introducing new non-functional (e.g., security and safety) risks. These risks are amplified by the crucial role played by machine learning, on one side, and by the pervasive involvement of users in the system operation, on the other side. Certification techniques have been largely adopted to reduce the above risks, though the recent evolution of distributed systems towards cloud-edge, IoT, 5G, and machine learning severely hindered certification diffusion and quality. The need of new certification techniques that prove compliance of distributed systems against non-functional requirements arises and is often pushed by strict laws and regulations. In this paper, we envision a research manifesto for non-functional certification of modern distributed systems that paves the way for the wide adoption of certification in the real world, also in those domains where certification is not mandatory. Its ultimate goal is to lead to a trustworthy and adaptive ecosystem based on a cost-effective, non-functional certification, where modern system development, assessment, and management are not only ruled by functional requirements. The manifesto discusses the research challenges, a roadmap built on 6 research directions, and a concrete implementation timeline for the roadmap.**

***Index Terms*—Certification; assurance; security; service engineering; service life cycle**

I. Introduction

Modern distributed systems consist of hundreds or thousands of miniaturized services running opaque machine learning (ML) models, orchestrated and composed at run time with cloud-native technologies [1], and deployed seamlessly along the cloud-edge continuum [2]. They not only deliver smart functionalities, but become smart themselves [3], being able to swiftly react and adapt to environmental changes, while following continuous release cycles based on DevOps [4]. In this context, software-defined networking and 5G equipment provide high-throughput and ultra-low latency communications, where data are collected by IoT devices and exchanged at high rates, achieving greater quality of services.

The increasing interplay with the physical counterpart of distributed systems, as well as the increasing reliance on ML for both functional and non-functional aspects, is radically changing the service engineering landscape, in general, and the assurance and certification domains, in particular. Established assurance and certification practices, which have been successfully used to ensure that a system behaves correctly according to a set of non-functional requirements [5], become virtually useless. The ability to properly and precisely model, assess, and certify a target system according to testing and monitoring techniques is challenged by the dynamic and constantly-evolving topology of the system, whose behavior is not known in advance. On top of this, assurance and certification are facing the additional need to increase automation, to reduce the burden and, in turn, the costs on all involved stakeholders, from service providers to final users.

Our paper presents a research manifesto for non-functional certification of modern distributed systems, where all involved parties, from service providers to certification authority, are assumed to behave correctly (i.e., no malicious parties). It outlines the steps to undertake to fill in the above gaps towards a trustworthy ecosystem built on certified compliance to non-functional requirements. Pushed by the increasing interest in certification, for instance in the European Union where the European Agency for Cybersecurity has received the mandate of developing several European certification schemes (*EUCC* [6], *EUCS* [7], and *EU5G* [8]), our manifesto considers three main pillars. First, a set of modern and lightweight certification building blocks, whose definition can be performed with little to none human intervention. Second, an autonomic certification process based on unobtrusive system evaluation, capable to certify application-level services while considering the impact of the environment and the entire distributed systems. Third, the confluence of the two aforementioned pillars towards a unified certification-based service engineering process, with certification part of the system life cycle management. These pillars aim to unleash a cost-effective and high-quality ecosystem that fosters the usage of certification in the real-world.

The contribution of our manifesto is threefold. We first identify the challenges posed by modern distributed systems to the state of the art of certification (Section III). We then propose a research roadmap for distributed system certification discussing 6 main research directions (Section IV). We finally describe a possible timeline for the roadmap implementation (Section V) in the short ($\leq$2 years), medium (between 2 and 5 years), and long ($\geq$5 years) terms.

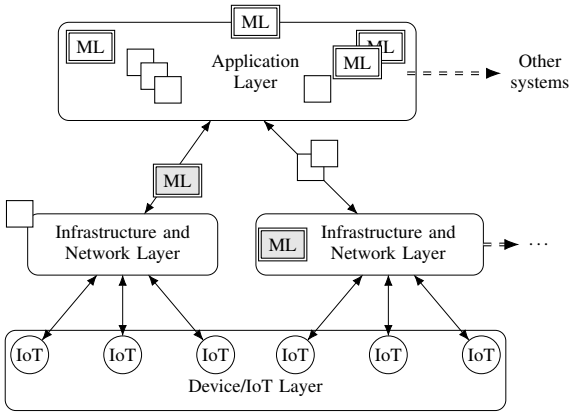

Fig. 1. Reference architecture of modern distributed systems. Single-line squares represent traditional services, double-line squares represent ML-based services, grey-filled squares represent ML-based services for system management.

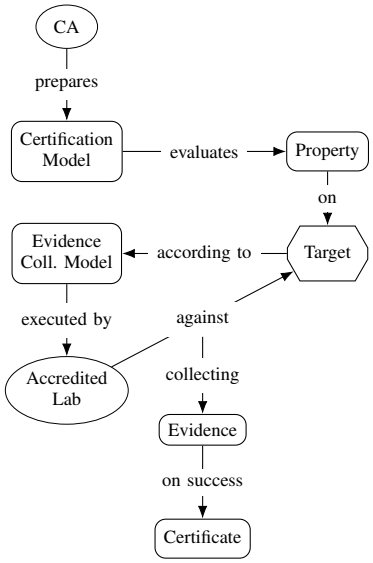

Fig. 2. Certification process

II. Background

We present the reference architecture for distributed systems used in this manifesto (Figure 1). We then discuss the current status of certification as the starting point to discuss the challenges and research roadmap according to the reference architecture.

A. Distributed Systems: Reference Architecture

The current ICT landscape has been deeply revolutionized in the last two decades, changing the way in which distributed systems are designed, developed, and operated. Systems moved from monolithic and static code bases to elastic and multi-layer distributed services, where each layer transparently provides functionalities to the upper layers. From the launch of services at the beginning of 2000s, distributed systems evolved to miniaturized micro- and nano-services, where horizontal aspects are delegated to orchestration and composition engines. In addition, IoT and ML models are increasingly permeating these systems, providing functionalities to end-users and supporting the service life cycle management.

We consider the simplified three-layer architecture in Figure 1 as our reference architecture for distributed systems, including: *i)* application layer (the target of this manifesto), *ii)* infrastructure and network layer, and *iii)* device/IoT layer. At application layer, individual (micro- and nano-)services are composed and orchestrated at run time along the cloud-edge continuum, delivering their functionalities to end users. Services are continuously and transparently re-instantiated when and where is needed, and increasingly deploy ML models. At infrastructure and network layer, orchestration technologies (e.g., Kubernetes) manage application-layer services together with virtualized and software-defined networking. They provide intelligent and transparent placement on top of a (virtualized) infrastructure and ultra-fast connectivity. ML is increasingly studied as a building block for this layer [3]. Finally, at device/IoT layer, data are collected from the field by low-end, resource-constrained devices, and efficiently sent to the upper layers for analysis. In the case of actuators, they may actively interact with the surrounding environment.

Although distributed systems built on this architecture have been already proposed and, in some cases, fully implemented and used in production [9], their certification is still in its infancy and more an art than a science. Additional effort must be put by the research community on certification to keep the pace of technological evolution, to the aim of providing certification techniques that address the peculiarities of modern distributed systems.

B. Certification in a Nutshell

Certification schemes have been successfully used to tackle the need for assurance of traditional software since the early 80s [10], [11] and, in the last 10 years, adapted to service-based and cloud-based environments [5]. A certification scheme details the process verifying that a target system behaves as expected and demonstrates one or more non-functional properties [5]. Figure 2 presents a generic certification process. The Certification Authority (CA) defines the *certification model* driving all activities needed to certify a *property* on a specific *target* of certification according to an *evidence collection model*. The accredited lab, delegated by the CA, executes the *evidence collection model* to collect the *evidence* that is used to possibly award a *certificate* to the target system [12]. A chain of trust is established involving the CA preparing the certification model, the accredited lab executing the certification model, and the awarded certificate [13]. Involved parties are usually assumed benign (no malicious actors), except few cases [14].

We present an overview the three main building blocks of a certification model and the techniques at the basis of their definition and usage.

Non-Functional Property can be traditionally expressed as *i) fixed property*, a property name (e.g., integrity) with no

TABLE I
AN EXAMPLE OF CERTIFICATION MODEL BUILDING BLOCKS

Property	Target	Evid. Coll. Model
p_{rel}=($\widehat{p}_{\text{rel}}$, {*replicas*=2, *replica zones*=2})	s_1={c_{db}, c_{api}, c_{cross}}	{`get-orchestrator`, `check-replicas`, `check-zones`}

additional details [15], [16], *ii) assurance case*, high-level objective defined according to *Goal Structuring Notation* [17], [18], or *iii) detailed property*, a fixed property (e.g., confidentiality) refined with attributes (e.g., encryption algorithm, key length) [12], [14], [19]. The latter is the preferred non-functional property type, and mostly used in the context of service-based and cloud-based certification. Its generic formulation allows to model a multitude of requirements, from the traditional CIA (*Confidentiality*, *Integrity*, and *Availability*) triad to reliability and performance, to name but a few.

Target of Certification models the target of a certification process. It corresponds to the set of components (e.g., endpoints, services, functions) that support a specific property, and the concrete implementation to which a certificate is bound. Different approaches exist for target modeling (e.g., [12], [18]), though it is often left unspecified (e.g., [20]).

Evidence Collection Model drives all activities aimed to collect the evidence proving a property for a specific target of certification. Evidence can be collected according to *i)* testing, *ii)* monitoring, *iii)* formal methods, and *iv)* other approaches. Testing evidence is in the form of test outputs retrieved from the target. It has been applied to service and cloud-based systems [12], [14], [20], and, recently, IoT-based systems [21]. Monitoring evidence is in the form of logs or other observability data retrieved during the target operations. It reduces invasiveness, at the expense of less accurate results; it has been mainly applied in cloud-based systems [22]–[25]. Evidence based on formal methods is in the form of formal proofs retrieved from a formal modeling of the target. It provides the strongest assurance, at the expense of an extensive modeling phase. For this reason, it is used in limited contexts, such as medical [26] and avionics [27], although it has been also applied to cloud-based systems [28]–[30]. Other approaches for evidence collection are also emerging and include hybrid approaches (e.g., combining testing and monitoring [31], fault injection [19]) to balance pros and cons of base techniques.

Example 1: Table I presents an example of certification model in traditional certification schemes. Let us consider a cloud-native Python microservice s_1 developed and operated using DevOps. Kubernetes takes care of automatic scaling and replication across multiple zones to guarantee *reliability*. Service s_1 can be modeled as s_1={c_{db}, c_{api}, c_{cross}}, with components referring to database interaction, HTTP endpoints, and cross-functionalities, respectively. Each component can be further refined in terms of low-level non-functional mechanisms [12]. Property reliability can then be modeled as a pair p_{rel}=($\widehat{p}_{\text{rel}}$, Attr), where $\widehat{p}_{\text{rel}}$ is the property name and $Attr$={*replicas*=2, *replica zones*=2} indicates the minimum number of acceptable replicas and replica zones, respectively. The corresponding evidence collection model can include a test case consisting of three test steps {`get-orchestrator`, `check-replicas`, `check-zones`} retrieving the orchestrator and evaluating that the number of replicas and replica zones is as expected (i.e., in p_{rel}). Each test step contains the corresponding inputs, expected outputs, and preconditions. The successful collection of this evidence triggers the release of a certificate for s_1, which contains a reference to the corresponding certification model and the collected evidence (i.e., test case's output).

Certification has been integrated within distributed system life cycle, to increase the quality of service and support stable behavior over time. It primarily focused on service selection, supporting the selection of the best service among a set of functionally-equivalent candidate services according to non-functional properties in certificates. It also focused on service composition, where services are dynamically composed at run time according to their non-functional properties. Several approaches based on SLAs have been also presented, such as [32]–[34]. Certification-based selection and composition support system adaptation, that is, the update of the running system to react to external changes. Preliminary approaches ensure that the system still holds non-functional properties during adaptation [17], [18], [25], [35], [36].

C. Towards Modern Distributed Systems Certification

Modern distributed systems in our reference scenario in Section II-A challenges the assumptions and, in turn, the soundness of existing certification schemes in Section II-B.

Modern systems are extremely elastic, adapting their topology to accomodate strict SLAs, contrasting with the ability to precisely define certification building blocks. System adaptation can even invalidate a certificate whose release requires a huge effort in terms of time and resources. Also, the introduction of ML at both application and infrastructure layers make things even worse. Certification building blocks assume static, predictable, and verifiable behavior (e.g., via testing or monitoring), contrasting with the unpredictable behavior of ML. Ultimately, there is a mismatch between the static and predictable nature of certification and the elastic and unpredictable nature of modern (ML-based) distributed systems.

The above scenario shows the way forward to define a certification scheme for modern distributed systems. It represents the starting point for the definition of our manifesto, which deeply analyzes existing certification challenges and corresponding research directions.

III. CHALLENGES IN THE CERTIFICATION OF MODERN DISTRIBUTED SYSTEMS

Modern distributed systems in Figure 1 challenge existing certification schemes, impairing their ability to retrieve high-quality, meaningful, and useful evidence in an automated way.

We identify 4 main domains of analysis; for each domain, we describe the research challenges affecting the certification of modern distributed systems and analyze their impact on certification at organizational and technical levels. First, we consider challenges in *certification design*, with particular reference to the impact that modern distributed systems have on the soundness of existing certification schemes (Section III-A). Then, we discuss challenges in *certification process*, with particular reference to certification process execution in large-scale cloud-edge environments (Section III-B). Next, we elaborate on the challenges introduced by the migration from deterministic to *ML-based systems* (Section III-C). Finally, we analyze challenges in *certification life cycle* management (Section III-D).

A. Challenges in Certification Design

Certification design defines all building blocks of a certification scheme with reference to the specific target environment. Certification schemes have traditionally been built without keeping the pace with IT evolution, trying to adapt existing techniques developed for service-based systems to current distributed systems in Figure 1. Expediency more than design [37] has been used in the development of new certification schemes, making certification unmanageable in systems that are miniaturized, distributed, and dynamic. New challenges emerge pushed by need to *i)* rethink the pillars of certification, that is, non-functional property, target of certification, and evidence collection model in Section II-B, and *ii)* consider certification since system design and development.

C1.1: Property definition. Detailed definition of attribute-based properties (see Section II-B) is at the basis of a sound certification scheme. The current definition implements a static approach, where properties are assumed to be instantiated a priori and maintained unmodified during the entire system and certification life cycles. The dynamic nature of modern distributed systems, however, requires a more flexible definition, where properties are defined according to multiple dimensions of interest (e.g., artifacts, development, evaluation) [12] and adapt to the surrounding environment and system evolution. For instance, some attributes may not be initially known (e.g., the autoscaling strategy in property p_{rel} in Example 1) while some other may change (e.g., the number of replicas in property p_{rel} in Example 1), still globally supporting the required property.

C1.2: Target modeling. As for C1.1, the ability of properly modeling the target of certification is impaired by the dynamic nature of modern distributed systems. Current modeling approaches (see Section II-B) assume targets of certification with known topologies and static boundaries. Today, (micro-)services and, more in general, computing nodes (e.g., IoT devices) continuously enter and exit the system, while being constrained by their lack of computational resources and need of efficiency. This results in a dynamic system topology, which changes according to the environment and the mobility of its smart devices. More flexible modeling approaches need to be defined to reliably measure the system boundaries, while not impacting on the availability of the system and its devices.

C1.3: Integration of development and certification processes. Certification is implemented as an independent, post-deployment activity disconnected from the service engineering process. Current development processes and existing certification techniques carry some critical mismatches, making certification a costly, time-consuming, error-prone, and suboptimal process, which does not support modern systems [38]. Inadequate planning and integration of certification within the development process in fact substantially limits its adoption in practical settings [38]. A tighter integration between the development and certification processes is in its infancy, though it has the potential to significantly increase the trust in certified systems, on one side [39], [40], and significantly decrease the certification costs, on the other side [38].

B. Challenges in Certification Process

Certification process refers to the execution of a certification scheme in operation. Certification processes have traditionally been executed in lab environments, only recently moving to production environments (e.g., cloud). However, the increasing migration towards the edge and smart devices represents a barrier in the application of current certification techniques to modern systems. New challenges emerge pushed by the need to *i)* cope with the multi-layer and composition-based nature of modern systems, and *ii)* relax some trust assumptions that become invalid in open systems.

C2.1: Multi-layer service composition. Traditional certification processes collect evidence from the tip of the iceberg of modern distributed systems, that is, individual services (see Figure 1). This approach assumes collected evidence to be enough for a proper system assessment, while it is not. For instance, availability and reliability of services are guaranteed by means of an orchestration framework such as Kubernetes at the infrastructure layer (see Figure 1 and Example 1). A proper and comprehensive certification should depart from a single-layer evaluation and rather extend to any layers of the system [41]. In particular, it should cope with multi-layer composition, while limiting the impact on system functioning, in terms of resource consumption, and on privacy protection, in terms of data access and sharing. It should also consider how the composition of individually-certified building blocks can contribute to the certification of the composition.

C2.2: Evidence collection and lineage. Collected evidence is stored in the certificates and represents the concrete proofs at the basis of certificate awarding [12]. Collected evidence and, in turn, certificates, are challenged by *i)* low-quality evidence when IoT is involved, due to sensors' malfunctions, misconfigurations, or unavailability [41]; *ii)* privacy requirements, where evidence

collection may require access to system's internals and private data, and sensitive information might be stored in publicly-available certificates; and *iii)* reliance on blind trust (see C4.3). Ultimately, it requires a framework for *evidence lineage*.

C2.3: Dishonest behavior. Certification processes assume that all involved parties (CA, accredited lab, service/product provider) act correctly and honestly, that is, the target does not aim to wrongfully obtain a certificate for a non-functional property, and certification model preparation and execution are performed correctly [12]. This limits the ability to evaluate open systems, where honest behavior cannot be presumed.

C. Challenges in ML-Based Systems Certification

Machine learning plays an important role in the operations of modern distributed systems, and deserves a separate discussion. It is in fact difficult to assess the behavior of a ML-based system, which is inherently non-deterministic. A lot of effort on ML trustworthiness [42] is going on at regulatory, academic, and industrial levels. The European Artificial Intelligence Act [43] is under preparation and introduces the need of *conformance assessment*. Several approaches focused on explainability and robustness of ML [44], [45], which are however not the only properties that matter. Also, the proposed approaches are far from providing a general-purpose certification scheme for ML.

The need to cope with the intrinsic non-determinism and opaqueness of ML models and their operations makes certification of ML-based systems radically different from traditional services, and push towards a complete rethinking and redesign of the certification schemes in Section II-B.

C3.1: Property and target definition. Current definitions of non-functional property and target of certification are deterministic and therefore not applicable to ML. Even flexible attribute-based definitions (see C1.1, C1.2) cannot properly model a behavior that is, at least partially, unpredictable and not known a priori [44]. Novel non-deterministic building blocks are required. On top of this, the concrete definition of non-functional properties (e.g., fairness, transparency, robustness [42]) is still an open challenge itself [46].

C3.2: Certification process modeling. Traditional approaches for evidence collection connect to (public) APIs and execute pre-defined test cases and monitoring rules [5]. However, these approaches are not applicable to ML-based systems because the latter *i)* are opaque, making it difficult to find a common API-like connector, and *ii)* present an unknown behavior that can vary over time. Certification process modeling must address this uncertainty following the ML evolution.

C3.3: ML pipelines. Similarly to application-level services (see C2.1), deployed ML models are the tip of the iceberg of ML-based systems. The latter often build on ML pipelines, that is, a composition of components each one coping with a specific aspect of ML, such as data ingestion and preparation, processing and analytics, visualization and reporting [42]. Although the certification community is still trying to find concrete approaches to the evaluation of a standalone ML model, new certification solutions must be defined with the goal of verifying ML pipelines (i.e., ML model *operations*), including the surrounding services interacting with them.

D. Certification Life Cycle

Certification life cycle includes all activities concerning the usage and management of awarded certificates. Often certification life cycle is managed through an *all-or-nothing* approach, where the certificate is either supported or revoked in case a new set of evidence invalidates its properties. Existing life cycle management is usually a manual process fully relying on the manual definition of the certification model by the CA and its execution by the accredited lab, with some preliminary forms of automation (e.g., [13], [17], [18], [25]). The result is a costly and time-consuming process that requires re-certification from scratch at almost each system evolution. New challenges emerge pushed by the need to *i)* increase automation towards a continuous and autonomic life cycle, *ii)* integrate certification and service life cycles to preserve quality of service, and *iii)* define a sound chain of trust.

C4.1: Increase automation. The continuous evolution and adaptation of modern distributed systems implies that certification models and corresponding certificates can quickly become outdated and useless, requiring to re-execute the process from scratch with a waste of resources and time. There is an urgent need of *continuous and autonomic life cycle*, which is impaired by *i)* reliance on ineffective change detection, and *ii)* absent/inefficient automation.

Change detection refers to the approaches triggering re-certification at system changes. It is currently based on code updates [40] and fixed timers [31], triggering many unnecessary re-certifications and missing relevant ones caused by environmental changes. On one side, code updates in DevOps are frequent but mostly minor, with a negligible impact on certified non-functional properties; timer expiration is blind and does not necessarily imply a re-certification. On the other side, code updates and timers ignore changes due to external conditions, for instance, traffic peaks.

Automation is fundamental to limit manual update of certification models and re-certification from scratch, especially in scenarios where incremental system changes driven by DevOps impact just a small fraction of the certification model. Automation is also important to limit the burden on the CA and the accredited lab, as well as to reduce the costs on the service provider.

C4.2: Certification-based system life cycle. The management of the system life cycle is driven by external conditions and aims to continuously maximize the system QoS. However, this maximization does not typically consider

and may even contrast with already certified non-functional properties, due to the lack of integration between certification and development processes (see C1.3). For instance, let us consider property reliability in Example 1. A decrease in the amount of requests typically triggers a decrease in the number of replicas (horizontal scalability), which may violate the requirement on the minimum number of replicas thus invalidating the property according to C4.1. This introduces the need of synchronizing the system and certification life cycles, such that certification is part of the process driving system evolution (e.g., certificate-based service selection [12]).

C4.3: Reduce reliance on blind trust. The practical usability of a certification process cannot depart from the definition of a trust model that permits CAs to delegate part of the process, while increasing the confidence on its final results [13]. Pushed by the need to address evidence lineage (C2.2), dishonest behavior (C2.3), and automation (C4.1), a modern chain of trust should not be "taken for granted" and bound by contractual obligations only. Also, the proliferation of service providers/device owners with unknown reputation, on one side, and the increasing reliance on ML, on the other side, undermine traditional approaches where the chain of trust is rooted at the certification model originally defined by the CA.

In summary, modern distributed systems challenge the traditional assumptions on which existing certification leans, demanding a rethink of certification from its building blocks to its operations and life cycle management. In the remaining of this paper, we analyze a possible roadmap of certification evolution and its implementation timeline.

IV. Certification Roadmap

This manifesto paves the way for the wide adoption of certification in the real world, also in those domains where certification is not mandatory. Its ultimate goal is to lead to a *trustworthy and adaptive ecosystem based on a cost-effective, non-functional certification*, where modern system development, assessment, and management are not only ruled by functional requirements. Our roadmap defines 6 research directions that aim to address the challenges in Section III. Given the complexity of the considered scenario, each research direction can map to multiple challenges in the 4 domains of observation (i.e., certification design, certification process, ML-based system certification, certification life cycle) at the same time. The complete mapping is summarized in Table II; column *Timeline* refers to the implementation timeline discussed in Section V, column *Related fields and techniques* shows relevant technologies that can be used in the roadmap implementation.

RD1: Non-functional property definition. The specification of a certification scheme that addresses modern distributed system's requirements starts from the definition of a dynamic non-functional property that builds on the state of the art [12].

Non-functional properties, which traditionally specify attributes describing system artifacts only (e.g., number of replicas), should consider additional attributes describing system development (e.g., development process) and system evaluation (e.g., verification approach) [12].

Non-functional properties should be evaluated over time according to system changes [47], departing from a static attribute-based definition, and support probabilistic and fuzzy evaluation, departing from crisp *yes-or-no* evaluation (C1.1). To this aim, properties should be enriched with their *strength* (often referred to as *label* [20], [21]), modeling the property magnitude, to support advanced certificate management within the system life cycle (C4.2). The property strength can proportionally decrease as a function of time (C2.1, C4.2) [20].

Finally, non-functional properties should natively support certification of composite services, providing clear rules for their composition. Property composition [47] has already been introduced for traditional services and static properties only.

RD2: Behavior-based certification. Traditional certification schemes are based on *i)* a manual definition of the target of certification, which precisely specifies the boundaries of the target system components (e.g., endpoints, services, functions), and *ii)* a static evaluation (i.e., monitoring and testing) of such components.

The intrinsic mobility of modern distributed systems result in scenarios where the unclear boundaries of the target system impair the ability of precisely assessing an endpoint, service, or function. The certification scheme should be extended towards a *behavioral* evaluation of the target system, which requires to: *i)* define the target of certification as a composition of individual system components, rather than the list of individual components themselves, *ii)* model the expected behavior of the system to be certified. The latter departs from a manual and precise definition of target components (C1.2, C2.1, C2.3, C4.1), and rather defines the system as whole through its behavior.

Behavioral evidence should complement testing and monitoring evidence to naturally capture the system behavior as a whole also considering the corresponding environment (C2.1, C4.2). Behavioral evidence should be matched with the behavioral modeling of the target system to evaluate the support of a given non-functional property [23]. It should be at the basis of a *continuous certification process*, where the target system behavior is continuously compared against the expected behavior of the target of certification to detect changes possibly affecting an awarded certificate (C4.1).

The importance of behavior-based certification has been already recognized in traditional web service-based environments [51] as well as modern distributed systems [41], and few preliminary solutions have been presented for the latter [23].

RD3: Trustworthy evidence management. An ecosystem based on certificates requires the definition of a certification process detailing how evidence is *managed*. The process should facilitate trustworthy circulation of evidence and certificates among the stakeholders, according to the principles

TABLE II
MAPPING BETWEEN RESEARCH DIRECTIONS, THEIR TIMELINE, AND CHALLENGES

Research direction	Challenge	Timeline	Preliminary works	Related fields and techniques
RD1: Non-functional property definition	C1.1: Property definition	M	[12], [47]	Statistics, soft computing
	C2.1: Multi-layer service composition	S, M	[23], [41]	
	C4.2: Certification-based system life cycle	M, L	[12], [17], [18], [25], [35], [36], [38]	
RD2: Behavior-based certification	C1.2: Target modeling	M	[23]	ML, software engineering
	C2.1: Multi-layer service composition	S, M	[23], [41]	
	C2.3: Dishonest behavior	M	[14]	
	C4.1: Increase automation	S, M	[13], [20], [28], [48]	
	C4.2: Certification-based system life cycle	M, L	[12], [17], [18], [25], [35], [36], [38]	
RD3: Trustworthy evidence management	C2.2: Evidence lineage	M	[49], [50]	Distributed ledgers, data governance
	C4.3: Reduce reliance on blind trust	M	–	
RD4: Certification for ML	C1.1: Property definition	M	[12], [47]	Statistics, XAI
	C1.2: Target modeling	M	[23]	
	C2.1: Multi-layer service composition	S, M	[23], [41]	
	C3.1: Property and target definition	M, L	[44]	
	C3.2: Certification process modeling	M, L	[46]	
	C3.3: ML pipelines	L	–	
RD5: ML-based automation	C1.1: Property definition	M	[12], [47]	ML
	C1.2: Target modeling	M	[23]	
	C2.3: Dishonest behavior	M	[14]	
	C4.1: Increase automation	S, M	[13], [20], [28], [48]	
RD6: *DevCertOps* and beyond	C1.3: Integration of development and certification processes	M, L	[38]	DevOps, MLOps, Software engineering
	C2.1: Multi-layer service composition	S, M	[23], [41]	
	C4.1: Increase automation	S, M	[13], [20], [28], [48]	
	C4.2: Certification-based system life cycle	M, L	[12], [17], [18], [25], [35], [36], [38]	

of openness, privacy-preserving sharing, and traceability.

Openness requires evidence to be human and machine-readable, like the non-functional properties it aims to support (see RD1), and accessible to anyone (C2.2). However, openness does not mean unregulated access. Evidence should be shared according to the target system's owner rules to avoid any leakages of information, which can favor *phase reconnaissance* in the cyber kill chain. For instance, approaches based on data anonymization could be exploited (C2.2).

To achieve trustworthy sharing, the whole evidence collection process should be traced and trusted by design. For instance, approaches based on dedicated data structures such as distributed ledgers could be exploited (C4.3) [49], [50].

RD4: Certification of machine learning. The requirements introduced by distributed systems built on ML models (see Section III-C) ask for a new definition of the certification model and its building blocks.

Following RD1 and RD2, the certification model should be extended to assess the behavior of a ML model at inference time, supporting its evolution over time and coping with environmental conditions, such as a change in the incoming data distribution (C1.1, C1.2, C2.1, C3.1) [44].

ML certification should *i)* statistically define ML non-functional properties, such as, fairness, robustness, confidentiality [46] (C3.1); *ii)* monitor model inference performance with dedicated statistical techniques [46] (C3.2); *iii)* analyze the internal structure of ML model, if feasible. This issue is directly connected to explainable artificial intelligence (XAI) and the need to "open the black box" [52]. For instance, XAI could be used to generate a surrogate white-box model of the ML model target of certification, which could be inspected and certified on its behalf (C3.2).

Similarly to RD1, ML certification should go beyond ML model assessment, and evaluate the overall ML-based system and corresponding ML pipelines as a whole (C1.3, C2.1, C3.3).

RD5: ML-based automation. The specification of a behavioral-based certification process and life cycle cannot leave aside automation. We argue that ML can play a pivotal role in this, boosting automation in the *i)* creation of certification model building blocks, and *ii)* continuous certification.

First, ML should be used to automatically define the behavior of the target system and evaluate it according to behavioral evidence (see RD2). This approach could relieve the CA from manual and time-consuming activities (C1.2, C4.1).

Also, ML should support continuous certification, by continuously retrieving system behavior, detecting if a change occurred, and preserving the validity of the certificate while ensuring stable quality of service (C1.1, C1.2, C2.3, C4.1).

The usage of ML carries the risk of retrieving unexpected results and may reduce their explainability, conflicting with RD1 and RD2 [41]. For this reason, the life cycle of the corresponding ML models should be carefully evaluated (e.g., using an explainable ML model), as already acknowledged in [41].

RD6: *DevCertOps* and beyond. The full potential of certification can only be unleashed when the certification life cycle meets the distributed system life cycle.

DevCertOps refers to the definition of a novel service engineering process unified with certification. It extends and builds on the same principles of *DevSecOps*: the integration of security within the whole development process. *DevCertOps* should aim to *shift certification to the left* by integrating certification into development (*Dev*) and operations (*Ops*). For what concerns *Dev*, it should *i)* introduce certification as a part of the system design [38], and *ii)* certify all development artifacts according to a multi-dimensional certification [12] (C1.3). The need to consider certification in advance is well-

recognized in literature as a mean to reduce costs [27] and improve quality and security [39], [40], although a limited number of seminal works have been proposed [38].

For what concerns *Ops*, it should go beyond the certification of artifacts along the system operations. External stimuli drive system evolution, and the corresponding non-functional properties inevitably change as well (see RD1, RD2, RD5). We argue that autonomic certification should not be limited to *follow* system evolution, but rather *be a part of the process driving evolution*, such that the system correctly reacts to stimuli *and* property changes still preserving high certification strength (C2.1, C4.1, C4.2). Certificate-based adaptation is mostly considered in peculiar cyber-physical systems [17], [18], [25], with few works considering it in traditional distributed systems [35], [36].

From the above roadmap, it clearly emerges the central role of ML, both as a target and facilitator of new certification schemes. On one side, ML certification requires a paradigm shift (C3.1, C3.2, C3.3, and RD4); on the other side, ML has the potential to revolutionize certification by increasing automation and exploiting system behavior (RD2, RD5).

V. Timeline

We organize our roadmap in Section IV and its implementation in the short ($\leq$2 years, denoted as S), medium (between 2 and 5 years, denoted as M), and long ($\geq$5 years, denoted as L) terms, as summarized in Table II.

Short-term timeline includes research directions that can be completed within the next 2 years, and aims to give a first boost towards next generation certification (RD1, RD2), while re-using as much as possible existing certification building blocks.

The first goal is the definition of a certification scheme supporting multi-layer service composition. Individual services deployed at different layers of the distributed system (see Figure 1) are composed according to their certificates, and a single and unified composite certificate is awarded for the whole distributed system.

The second goal is the definition of a novel approach that builds on ML for the automatic creation of the evidence collection model. The chain of trust should be revised accordingly.

Short term evolution will adapt existing certification techniques to modern distributed system peculiarities, while providing a rudimentary form of automation.

Medium-term timeline includes research directions that can be completed in 2 to 5 years, and aims to fully define a certification scheme for modern distributed systems (RD1, RD2, RD3, RD5), as well as the first building blocks for the certification of ML (RD4).

The first goal is the definition of a continuous and autonomic certification scheme, where system behavior and (explainable) ML models drive the certification process. The chain of trust should be revised accordingly.

The second goal is the refinement of the building blocks in Section II-B towards ML certification.

Medium term evolution will enable to meet critical challenges of modern distributed systems when entering the production, providing an early approach to ML certification.

Long-term timeline includes research directions that can be completed in a period longer than 5 years, and aims to fully define a certification scheme for ML (RD4) as well as achieve full integration within the distributed system life cycle (RD6).

The first goal is the definition of a complete, continuous and autonomic certification scheme for ML. It expands the scope of certification beyond the individual ML model (medium term) along two directions: *i)* the certification of a complete ML pipeline, *ii)* the certification of a complete ML-based distributed system.

The second goal is the integration of certification and development life cycles. *DevCertOps* should emerge as the paradigm where certification is an integral part of system design, development, operations, and management.

Long term evolution will permit to address all challenges in this roadmap.

VI. Conclusions

As modern distributed systems sitting at confluence of cloud, edge, IoT, and ML, are entering into commercial offerings, national and international authorities are finally acknowledging the need to introduce trust in the ICT sector.[1] Certification is a preferred means to achieve this goal. However, existing techniques fall short to meet the challenges posed by distributed systems, mainly due to lack of precise system boundaries, lack of continuous behavioral modeling, non-determinism, and opaqueness. Our manifesto discussed a roadmap for the certification of modern distributed systems presenting 6 research directions and a timeline of their implementation. We believe this manifesto can be a first step in enabling certification to fully meet modern distributed system expectations.

Acknowledgments

The work was partially supported by the projects *i)* MUSA - Multilayered Urban Sustainability Action - project, funded by the European Union - NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strengthening of research structures and creation of R&D "innovation ecosystems", set up of "territorial leaders in R&D" (CUP G43C22001370007, Code ECS00000037); *ii)* SERICS (PE00000014) under the NRRP MUR program funded by the EU - NextGenerationEU.

References

[1] M. Clark, "Moving BBC Online to the cloud," https://medium.com/bbc-product-technology/moving-bbc-online-to-the-cloud-afdfb7c072ff, 2020.

[2] D. Zeng, N. Ansari, M.-J. Montpetit, E. M. Schooler, and D. Tarchi, "Guest Editorial: In-Network Computing: Emerging Trends for the Edge-Cloud Continuum," *IEEE Network*, vol. 35, no. 5, 2021.

[1] https://certification.enisa.europa.eu/

[3] T. L. Duc, R. G. Leiva, P. Casari, and P.-O. Östberg, "Machine Learning Methods for Reliable Resource Provisioning in Edge-Cloud Computing: A Survey," *ACM CSUR*, vol. 52, no. 5, 2019.

[4] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," *IEEE Software*, vol. 33, no. 3, 2016.

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

[6] ENISA. (2021) Cybersecurity Certification: Candidate EUCC Scheme V1.1.1. [Online]. Available: https://www.enisa.europa.eu/publications/cybersecurity-certification-eucc-candidate-scheme-v1-1.1

[7] ——. (2020) EUCS - Cloud Services Scheme. [Online]. Available: https://www.enisa.europa.eu/publications/eucs-cloud-service-scheme

[8] ——. (2021) Securing EU's Vision on 5G: Cybersecurity Certification. [Online]. Available: https://www.enisa.europa.eu/news/enisa-news/securing_eu_vision_on_5g_cybersecurity_certification

[9] Singapore Public Utilities Board, "Managing the water distribution network with a Smart Water Grid," *Smart Water*, vol. 1, no. 1, 2016.

[10] DoD, "Department of Defense Trusted Computer System Evaluation Criteria," *USA Department of Defense*, December 1985.

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

[12] M. Anisetti, C. A. Ardagna, and N. Bena, "Multi-Dimensional Certification of Modern Distributed Systems," *IEEE TSC*, 2022.

[13] 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, 2020.

[14] 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.

[15] M. Aslam, B. Mohsin, A. Nasir, and S. Raza, "FoNAC - An automated Fog Node Audit and Certification scheme," *Computers & Security*, vol. 93, June 2020.

[16] A. Khurshid and S. Raza, "AutoCert: Automated TOCTOU-secure digital certification for IoT with combined authentication and assurance," *Computers & Security*, vol. 124, 2023.

[17] 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, 2018.

[18] 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, 2020.

[19] 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.

[20] A. Milánkovich, G. Eberhardt, and D. Lukács, "The AssureMOSS Security Certification Scheme," in *Proc. of ARES 2022*, Vienna, Austria, August 2022.

[21] S. N. Matheu-García, J. L. Hernández-Ramos, A. F. Skarmeta, and G. Baldini, "Risk-based automated assessment and testing for the cybersecurity certification and labelling of IoT devices," *Computer Standards & Interfaces*, vol. 62, February 2019.

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

[23] M. Elsayed and M. Zulkernine, "Towards Security Monitoring for Cloud Analytic Applications," in *Proc. of IEEE BigDataSecurity/HPSC/IDS 2018*, Omaha, NE, USA, May 2018.

[24] S. Lins, S. Schneider, and A. Sunyaev, "Trust is Good, Control is Better: Creating Secure Clouds by Continuous Auditing," *IEEE TCC*, vol. 6, no. 3, 2018.

[25] R. Faqeh, C. Fetzer, H. Hermanns, J. Hoffmann, M. Klauck, M. A. Köhl, M. Steinmetz, and C. Weidenbach, "Towards Dynamic Dependable Systems Through Evidence-Based Continuous Certification," in *Proc. of ISoLA 2020*, Rhodes, Greece, October 2020.

[26] D. Méry and N. K. Singh, "Trustable Formal Specification for Software Certification," in *Proc. of ISoLA 2010*, Heraklion, Greece, October 2010.

[27] G. Gigante and D. Pascarella, "Formal Methods in Avionic Software Certification: The DO-178C Perspective," in *Proc. of ISoLA 2012*, Heraklion, Greece, October 2012.

[28] 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, 2020.

[29] 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, 2020.

[30] J. Bornholt, R. Joshi, V. Astrauskas, B. Cully, B. Kragl, S. Markle, K. Sauri, D. Schleit, G. Slatton, S. Tasiran, J. Van Geffen, and A. Warfield, "Using lightweight formal methods to validate a key-value storage node in Amazon S3," https://www.amazon.science/publications/using-lightweight-formal-methods-to-validate-a-key-value-storage-node-in-amazon-s3, 2021.

[31] M. Egea, K. Mahbub, G. Spanoudakis, and M. R. Vieira, "A Certification Framework for Cloud Security Properties: The Monitoring Path," in *Proc. of A4Cloud 2014*, Malaga, Spain, June 2014.

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

[33] 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.

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

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

[36] C. A. Ardagna, N. Bena, C. Hebert, M. Krotsiani, C. Kloukinas, and G. Spanoudakis, "Big Data Assurance: An Approach Based on Service-Level Agreements," *Big Data*, 2023.

[37] B. Foote and J. Yoder, "Big ball of mud," in *Pattern Languages of Program Design 4*. Addison-Wesley, 2000.

[38] C. A. Ardagna, N. Bena, and R. M. de Pozuelo, "Bridging the Gap Between Certification and Software Development," in *Proc. of ARES 2022*, Vienna, Austria, Aug. 2022.

[39] J. Voas and P. A. Laplante, "IoT's Certification Quagmire," *Computer*, vol. 51, no. 4, 2018.

[40] C. Baron and V. Louis, "Towards a continuous certification of safety-critical avionics software," *Computers in Industry*, vol. 125, 2021.

[41] N. Bena, R. Bondaruc, and A. Polimeno, "Security Assurance in Modern IoT Systems," in *Proc. of IEEE VTC 2022-Spring*, Helsinki, Finland, Jun. 2022.

[42] K. R. Varshney, *Trustworthy Machine Learning*. Chappaqua, NY, USA: Independently Published, 2022.

[43] European Commission, "Proposal for a Regulaton of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative Acts," 2021.

[44] G. Vidot, C. Gabreau, I. Ober, and I. Ober, "Certification of embedded systems based on Machine Learning: A survey," *arXiv preprint arXiv:2106.07221*, 2021.

[45] M. Anisetti, C. A. Ardagna, A. Balestrucci, N. Bena, E. Damiani, and C. Y. Yeun, "On the Robustness of Ensemble-Based Machine Learning Against Data Poisoning," *arXiv preprint arXiv:2209.14013*, 2022.

[46] M. Anisetti, C. A. Ardagna, E. Damiani, and P. G. Panero, "A Methodology for Non-Functional Property Evaluation of Machine Learning Models," in *Proc. of MEDES 2020*, Abu Dhabi, UAE, November 2020.

[47] M. Anisetti, C. A. Ardagna, E. Damiani, and G. Polegri, "Test-Based Security Certification of Composite Services," *ACM TWEB*, vol. 13, no. 1, 2019.

[48] A. Núñez, P. C. Cañizares, M. Núñez, and R. M. Hierons, "TEA-Cloud: A Formal Framework for Testing Cloud Computing Systems," *IEEE TRel*, vol. 70, no. 1, 2021.

[49] C. A. Ardagna, R. Asal, E. Damiani, N. El Ioini, M. Elahi, and C. Pahl, "From Trustworthy Data to Trustworthy IoT: A Data Collection Methodology Based on Blockchain," *ACM TCPS*, vol. 5, no. 1, 2021.

[50] Z. Wang, J. Lin, Q. Cai, Q. Wang, D. Zha, and J. Jing, "Blockchain-Based Certificate Transparency and Revocation Transparency," *IEEE TDSC*, vol. 19, no. 1, 2022.

[51] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti, "Run-Time Monitoring of Instances and Classes of Web Service Compositions," in *Proc. of IEEE ICWS 2006*, Chicago, IL, USA, September 2006.

[52] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A Survey of Methods for Explaining Black Box Models," *ACM CSUR*, vol. 51, no. 5, 2018.