# Research Challenges in Legal-rule and QoS-aware Cloud Service Brokerage

Emiliano Casalicchio[a,1,*], Valeria Cardellini[b], Gianluca Interino[b], Monica Palmirani[c]

[a] *DIDD, Blekinge Institute of Technology, Sweden*
[b] *DICII, University of Rome Tor Vergata, Italy*
[c] *CIRFID, University of Bologna, Italy*

**Abstract**

The ICT industry and specifically critical sectors, such as healthcare, transportation, energy and government, require as mandatory the compliance of ICT systems and services with legislation and regulation, as well as with standards. In the era of cloud computing, this compliance management issue is exacerbated by the distributed nature of the system and by the limited control that customers have on the services. Today, the cloud industry is aware of this problem (as evidenced by the compliance program of many cloud service providers), and the research community is addressing the many facets of the legal-rule compliance checking and quality assurance problem.

Cloud service brokerage plays an important role in legislation compliance and QoS management of cloud services. In this paper we discuss our experience in designing a legal-rule and QoS-aware cloud service broker, and we explore relate research issues. Specifically we provide three main contributions to the literature: first, we describe the detailed design architecture of the legal-rule and QoS-aware broker. Second, we discuss our design choices which rely on the state of the art solutions available in literature. We cover four main research areas: cloud broker service deployment, seamless cloud service migration, cloud service monitoring, and legal rule compliance checking. Finally, from the literature review in these research areas, we identify and discuss research challenges.

*Keywords:* Cloud computing, Autonomic computing, Legislation compliance checking, Optimization, Quality of Service, Monitoring, Service Migration, Service portability

## 1. Introduction

In the ICT industry service providers, developers and integrators as well as customers should be aware that law and regulation introduce functional and non-functional constraints that must be included by-design inside the systems and maintained during their operation. In the era of cloud computing, and specifically in a public cloud scenario, this compliance management issue is exacerbated, because the customer essentially outsources data processing and storage to service providers that could be non-compliant with the customer legislation (e.g., as regards data protection).

An important role in law/regulation compliance management of cloud services can be played by a cloud broker [1, 2, 3] that works as an intermediary in the service procurement process and as a third party controller during the whole service life cycle. The broker should provide services to both customers and cloud service providers, for example: discovery of services compliant with law and Service Level Agreements (SLAs); run-time monitoring of service level metrics; monitoring of legislation changes; law and QoS compliance checking during the service on-boarding phase and, at run-time, during the service evolution phase; aggregation, composition, optimization, orchestration of cloud services.

In this paper we describe the detailed design of a legal-rule aware cloud service broker in the framework of the Cloud for Europe initiative [4], and we extensively discuss the emerged design and research challenges. Specifically, we focus on four of the most demanding functionalities of the broker, which we refer to also as *design challenges*, and that are:

- *F1 – Legal-rule compliance checking* is the broker capability of verifying (off-line and at run-time) that cloud service providers and cloud services are compliant with a legal framework, also as a mix of different supra-national or national legislations. An example of a tool to check cloud service law compliance is described in [5]. The legal-rule compliance must be monitored during the service on-boarding phase as well as at run-time, during the service evolution phase.
- *F2 – Legislation dynamic management* is the broker capability of dynamically tracking changes in legislation or changes in service features that may bring to a violation of the legal requirements.
- *F3 – QoS monitoring* is the broker capability of monitoring and analyzing QoS metrics. The monitoring is functional

*Corresponding author
*Email addresses:* emiliano.casalicchio@bth.se (Emiliano Casalicchio), cardellini@ing.uniroma2.it (Valeria Cardellini), gianluca.interino@gmail.com (Gianluca Interino), monica.palmirani@unibo.it (Monica Palmirani)

[1]Part of this research was carried out when Emiliano Casalicchio was at DICII, University of Rome Tor Vergata

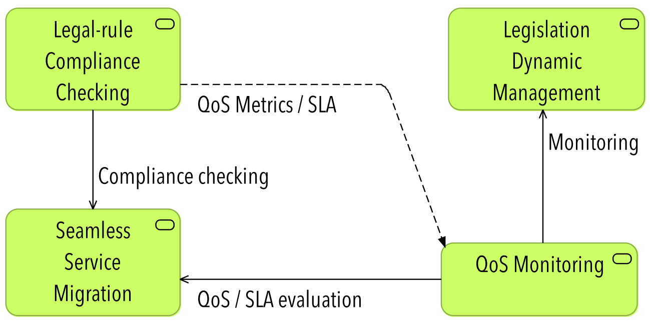

Legal-rule Compliance Checking
Legislation Dynamic Management
QoS Metrics / SLA
Monitoring
Compliance checking
Seamless Service Migration
QoS Monitoring
QoS / SLA evaluation

Figure 1: Dependencies among the broker's components implementing functionalities F1–F4.

both to check legal compliance and to verify that SLAs are guaranteed. The monitoring mechanism provided by a broker must be scalable to cope with a huge amount of data coming from many service instances.

- *F4 – Seamless service migration* refers to the ability to define and deploy a cloud service migration plan for the involved services, minimizing the service downtime. Seamless service migration relies on standard data formats and platform-independent computing environments.

An important pillar for both the F3 – QoS monitoring, and F4 – seamless service migration, is the deployment model of cloud services and the related technologies, as we will discuss later in this paper.

Functionalities F1 – F4 are strictly inter-dependent, as represented in Figure 1. Legal rules influence the service level objectives included in the SLA, the functional requirements (e.g., service migration towards another service provider to prevent vendor lock-in) and the structure of the processes behind the service implementation. Therefore, F1 influences the SLA metrics and the monitoring procedures (F3). On its turn, monitoring of QoS metrics (F3) is not only fundamental for guaranteeing SLAs, but it also takes part in the dynamic management of legislation compliance (F2). Finally, seamless service migration (F4) requires QoS monitoring (F3) and legal-rule compliance checking (F1) support from the broker.

Hence, the need arises for an integrated framework capable to offer these functionalities and to address the related research issues in an intertwined and integrated manner.

### *1.1. Research Contributions*

In literature many research works have focused on cloud service brokerage by addressing different issues, such as interoperability [6, 7], service discovery and matching [8], quality assurance and optimization [7, 9, 10, 11, 12], and legislation compliance [13]. However, those works address only one issue at a time, while only our previous works [3, 14] and this paper jointly address within a unified framework the design of a broker that integrates the functionalities F1 – F4.

This paper contributes to the literature as follows:

- We provide a more detailed design architecture of the broker initially proposed in [3]. The architecture proposed in our previous research has been largely extended and improved.

- We discuss our broker implementation choices selected from the state of the art solutions available in literature. We do not only address F1 – F4, but we also focus on the mechanisms and technologies for the service deployment, which are functional to F3 and F4.

- We review the literature and we identify and discuss research challenges in the following areas: cloud broker service deployment, seamless cloud service migration, cloud service monitoring, and legal rule compliance checking.

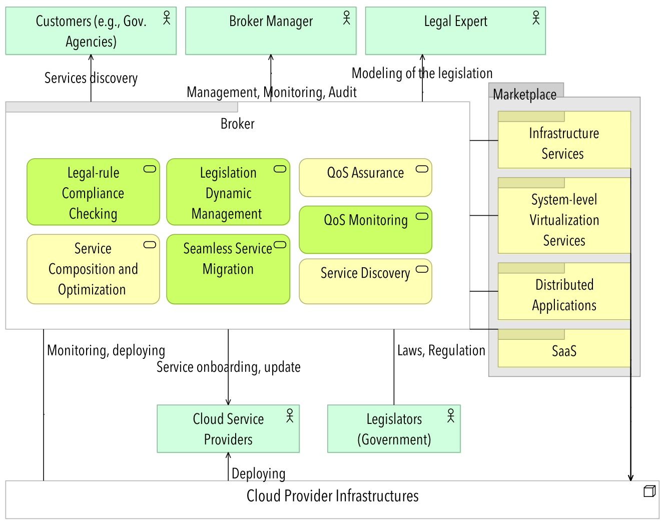
Our main findings are: we strongly recommend a broker which plays an active role in the service deployment. We suggest to use container technologies supported by TOSCA-based orchestration tools to cope with portability and provide seamless service migration. TOSCA, Docker, Cloudify and Kubernetes are examples of standards and technologies for service portability, but they are still not mature and the landscape is still fragmented. Monitoring of containers, assessment of scalability and elasticity, and evaluation of consistency in Cloud data storages are three challenges in QoS monitoring, which is functional to QoS assessment. Models and tools for legal compliance checking and management are available on the scene, but they are still fragmented and not accessible as a unique framework, and here the main challenge is to deal with different legal tradition sources, legal concepts in different languages, the interpretation level, and the interface module to allow human experts to take a decision.

### *1.2. Paper Organization*

To better contextualize the use of the proposed cloud service broker and to make easily understandable the functionalities we designed, we consider a real scenario described in Section 2. The purpose of the scenario is also to make practical and focused the discussion of the research challenges. In Section 3 we discuss related work on QoS and legal-rule aware cloud service brokerage and on related enabling technologies. In Section 4 we present the detailed design of the broker. Section 5 focuses on the proposed solution to address the design challenges. Finally, we discuss research open issues in Section 6 and we conclude with final remarks in Section 7.

## 2. Reference Scenario

Let us consider, as driving example, the problem of procurement of governmental cloud services, that have functional and non-functional requirements imposed by European, national and local legislation and regulation. In this scenario (see Figure 2), a government agency willing to use a public cloud service needs to check the compliance of that service with the legislation framework. Furthermore, it needs to evaluate if the service can guarantee the QoS level needed by its customers.



Customers (e.g., Gov. Agencies)
Broker Manager
Legal Expert
Services discovery
Management, Monitoring, Audit
Modeling of the legislation
Marketplace
Broker
Legal-rule Compliance Checking
Legislation Dynamic Management
QoS Assurance
QoS Monitoring
Service Composition and Optimization
Seamless Service Migration
Service Discovery
Infrastructure Services
System-level Virtualization Services
Distributed Applications
SaaS
Monitoring, deploying
Service onboarding, update
Laws, Regulation
Cloud Service Providers
Legislators (Government)
Deploying
Cloud Provider Infrastructures

Figure 2: Reference scenario.

To carry out such controls, the government agency could benefit from an intermediary (a broker) that offers the access to a marketplace of services that are certified as being compliant with the current legislation. Furthermore, the broker allows the agency to discover services that fit the customers' functional and non functional needs. Additionally, the broker is in charge of monitoring and guaranteeing both legislation compliance and QoS assurance/SLA satisfaction at run-time, i.e., during the service usage. Finally, the broker must also support service portability [15] that is, to allow the customer to migrate an application from a cloud infrastructure and/or platform to another one with minimal or even without efforts, so to avoid vendor lock-in and retain the ability for future strategical decisions.

Figure 2 represents the business level view of the broker (we use the TOGAF representation [16]), that include the actors involved in the scenario, the business functionalities offered by the broker, and the product available by the marketplace. The broker is managed by the Broker Manager that is also in charge of supervising the cloud services enrollment in the marketplace, in monitoring SLAs and auditing the cloud service providers. The government agencies are the customers of the broker. They request for accreditation/registration and for service discovery, that include specification of the SLA and functional requirements. The marketplace emerges from the services exposed by cloud service providers. The latter generate requests for accreditation, requests for service on-boarding, and notify service updates. Cloud service providers are monitored by the broker to verify that the offered SLAs are guaranteed at runtime and to audit QoS and compliance with law and standards. Legal expert are in charge of modeling the legislation. Legislators and the Government are those who define and change laws and regulations. The broker functionalities are represented as business services (we show in light green the functionalities considered in the paper).

We also suppose that the marketplace offers the following types of services: *infrastructure services*, *system-level virtualization services*, *distributed applications* and *Software-as-a-Service*. Infrastructure services are used to setup virtual infrastructures. Those services are typically offered by Infrastructure as a Service (IaaS) providers. System-level virtualization services (i.e., containers) are needed to implement and deploy portable applications. Those services are typically offered by Container as a Service (CaaS) or IaaS providers. Containers have been selected as a universal cloud application and deployment technology [17] to support seamless service migration (as detailed in Section 5). Distributed applications run on top of virtualized infrastructures; typical examples are client/server applications realized by composing and orchestrating IaaS/CaaS services.

An example of such a broker is demanded by the Cloud for Europe initiative [4, 18, 19]. However, the principles behind the broker we propose are independent from the specific sector of application; furthermore, the broker can provide functionalities that are sector agnostic.

## 3. Related Work

In this section we review the state-of-the-art approaches in literature, broadly classified into *cloud service brokerage* related work and *enabling technologies* related work.

### *3.1. Cloud Service Brokerage*

According to the NIST definition [1], a cloud broker is "*an entity that manages the use, performance, and delivery of cloud services, and negotiates relationships between cloud providers and cloud consumers*". A similar definition is given by Gartner [20], while the concept of autonomicity is introduced in [2], that envisions a broker capable of performing automatic resource provisioning and management, as well as automatic deployment across multiple clouds.

The problem of cloud service brokerage has been addressed from different perspectives in the literature but, to the best of our knowledge, no research work addresses explicitly the problem of legal compliance checking, except for [13] where the authors propose a distributed cloud proxy for monitoring and controlling the cloud service consumption. The aim of the proxy is to enable compliance to all privacy, legal, and regulatory issues regarding the service consumption. However, the proxy is comparable to an application layer gateway for cloud computing service and cannot be exactly classified as a broker.

The need for brokering mechanisms and policies particularly arises in cloud federation architectures, such as Intercloud [2], which is the first approach going towards the direction of building a unified platform composed by federated providers that can exchange information through super-entities. Service brokering is expected also to facilitate cloud adoption by simplifying the matching of users' needs [8], to rise trust in cloud computing and to facilitate the procurement of cloud services in the public sector providing added-value services [7, 21]. In [22] the authors describe the concepts of cloud bursting and cloud brokerage and discuss the open management and security issues associated with the two models. They also present a possible architectural framework capable of powering the brokerage based cloud services; such framework is currently being developed in the scope of OPTIMIS [23], an EU FP7 project. Cloud Agency, which is proposed in [24] in the framework of the mOSAIC EU FP7 project [25], implements a multi-agent brokering mechanism that is vendor agnostic and allows for the deployment of mOSAIC applications on any cloud infrastructure. In [6] the same authors present the architecture of the Broker Agent and its implementation in Cloud Agency for the provisioning of brokering service at the cloud platform layer. A request splitting algorithm based on a mixed integer programming formulation is proposed in [10], where the cloud service broker distributes the user requests across multiple cloud providers.

Some works focus on the assignment of cloud providers' resources to cloud consumers so to guarantee the consumers' requirements. In [9] the authors propose a cloud brokerage approach that optimizes the placement of virtual infrastructures across multiple cloud providers (each one with a different infrastructure offer and pricing policy) and also abstracts the deployment and management of infrastructure components in these clouds. A cloud broker that can employ different assignment strategies for optimal deployment of virtual services across multiple clouds, based on different optimization criteria, user constraints, and different environmental conditions is presented in [26]. Some limitations of this work, where only one consumer at a time is considered and no preference can be assigned to non-functional requirements, are overcome in [27]. Their proposed brokering architectures allows cloud consumers to customize their requirements further to fine level and the assignment of consumer's requirements to provider's resources is dynamically managed through a multiple criteria decision making approach. However, all these approaches take a narrow perspective focused on optimizing the cloud resources allocation, without considering other issues.

STRATOS [11] is another cloud broker service that permits to deploy and manage cloud applications on multiple providers, based on requirements specified in higher level objectives. STRATOS solves a multi-objective optimization problem and addresses the runtime adaptation issue. In [28] the authors consider the service brokering at infrastructure (IaaS) level as a mean to realizing delegation in cloud federations that is, to allow IaaS providers to leverage the capabilities available in a federation. QBROKAGE [21] addresses the problem of scalability and vendor lock-in by exploiting only public information made available from service providers. The proposed solution allows the deployment of applications on virtual machines running on multiple clouds. To this end, the authors propose a genetic-based approach to choose a set of cloud providers that can host the application while guaranteeing the QoS negotiated for the application.

Smart Cloud Broker [12] is a suite of software tools that allows IaaS consumers to evaluate and compare the performance of services offered by different IaaS providers, and thus support the selection of the cloud configuration and provider with the specifications that best meet the user's requirements. Using Smart Cloud Broker, prospective cloud users can estimate the performance of the different cloud platforms by running live tests against representative benchmark applications under given load conditions.

In [29] the authors envisage a brokerage framework that offers mechanisms for recommending optimal services to consumers in a platform-agnostic manner. The proposed framework is based on a platform independent description of consumer-expressed preferences regarding the service delivery.

A broker-based approach similar to that presented in this paper is proposed in [7]. The broker receives from the customer a call for proposal indicating functional and not functional requirements (expressed through SLA) and returns as result the best proposal, i.e. the best offer from providers. The SLA takes into account the price, the time unit, a rating indicating the best accredited provider, and the minimum accepted availability. However, the authors do not explicitly consider legal rules in the adaptation process and do not address the runtime adaptation problem as we do in this work.

In [8] the authors address the problem of cloud service matching proposing an OWL-S based cloud services broker. The complex constraints considered regard service location, bandwidth, storage, cost, and usage. This solution can be also used to solve the interoperability problem due to the lack of a non standard way to expose the providers capabilities. Even if their approach uses semantic reasoning as our approach, the authors do not address the legal compliance problem and the run-time adaptation problem.

Our literature review on service brokerage confirms that none of the existing solutions addresses, through an integrated and unified architecture, the functionalities $F1$–$F4$.

### *3.2. Enabling Technologies*

In order to enable a legal-rule and QoS-aware cloud service broker, two challenging requirements arise as prevalent: *i*) a standard way to describe services and their orchestration, and *ii*) a portable application environment. These requirements are baseline for all the four functionalities we consider ($F1 - -F4$). We have identified in TOSCA (Topology and Orchestration Specification for Cloud Application) and application containers the state of the art solutions that can emerge as de-facto standards in the current cloud landscape to address the above design challenges. In what follow, we briefly describe these technologies and motivate their adoption.

#### *3.2.1. Containers*

The cloud industry has a growing interest in system level virtualization [30]. The idea of containers dates back to 1992 [31] and have matured over the years with the introduction of Linux namespace [32] and the LXC project [33], a solution designed to execute full operating system images in containers. Application containers [34] are an evolution of operating system virtualization. Rather than packaging the whole system, containers package application or even application components (the so called microservices) which introduce a new granularity level of virtualization and thus become appealing for PaaS providers [30]. The main idea behind containers is the possibility of defining a container specific environment where to install all the library dependencies, the binaries, and a basic configuration needed to run an application.

There are several management tools for Linux containers: LXC, systemd-nspawn, lmctfy, Warden, and Docker [30, 34]. Furthermore, rkt is the container management tool for CoreOS. The latter is a minimal operating system that supports popular container systems out of the box. The operating system is designed to be operated in clusters and can run directly on bare metal or on virtual machines. CoreOS supports hybrid architectures (e.g., virtual machines plus bare metal). This approach enables the CaaS solutions that are becoming widely available.

In our scenario (cf. Figure 2), there are many reasons to elect containers as the building block technology to support the Seamless Service Migration functionality. First, containers give the possibility to execute a containerized application on any platform that supports the container technology. Second, containers provide a higher level of abstraction for the process lifecycle management, with the ability not only to start/stop but also to upgrade and release a new version of a containerized service. The latter facilitates compliance checking against both QoS requirements and legislation. Third, containers provide also a solution to data portability. Docker provides the Docker Volumes, which are specialized containers intended for data management and facilitate data portability. Finally, containers allow to implement orchestration by means of specific platform tools. For example, Kubernetes [35, 17] is an open-source platform developed by Google for the automating deployment, scaling, and operations of application containers across clusters which provide the container-centric infrastructure. Kubernetes supports run time scaling, seamlessly roll out and resource usage optimization.

### *3.2.2. TOSCA*

TOSCA is an OASIS open standard [36] that defines an interoperable meta-model of services and applications, enabling portability and automated management across cloud providers, regardless of the underlying platform or infrastructure [37]. These characteristics also facilitate the portable, continuous delivery of applications across their entire lifecycle. In short, they empower a much higher level of agility and accuracy for business in the cloud.

The TOSCA meta-model defines both the structure of a service (e.g., an application or a cloud infrastructure service) as well as how to manage it (e.g., deploy, patch and shutdown). TOSCA's models are technology agnostic. Indeed, TOSCA's artifacts intended as topologies (service structures) and plans (process models), are portable and can thus be deployed on any cloud provider infrastructure.

TOSCA is a suitable model to describe services implemented as containerized applications and to orchestrate them. Furthermore, the TOSCA standard, thanks to its flexibility, allows to map the container lifecycle. As observed in [38], TOSCA runtime lifecycle events (create, pre-configure, configure, post-configure, start and stop) are superset of typical container runtime events (deploy, init, stop and dispose).

Concerning orchestration, TOSCA allows to define non-functional behavior or QoS by means of Policies. A Policy can describe different things like monitoring behavior, payment conditions, scalability, or continuous availability. The implementation of these policies are workflows and the TOSCA standard does not commit to any workflow language to describe the necessary steps to implement any orchestration process.

Cloudify [39] is an open source cloud orchestration software platform based on TOSCA. It automates the process of installation, deployment and also post-deployment such as monitoring, remediation, and auto-scaling of the application stack. Cloudify offers a plugin for Docker and Kubernetes.

In our reference scenario (cf. Figure 2) TOSCA is the pillar to enable service interoperability and portability that require a standard way to describe services and their orchestration. Interoperability and portability allow to implement Service Composition and Optimization, Service Discovery, and Seamless Service Migration.

## 4. Broker Architecture

The solution we propose can be classified, according to the NIST definition, as an intermediation – aggregation broker [40, 41, 42]. The detailed design architecture we present in this section implement the broker described in the reference scenario. It is an evolution of the high level architecture proposed in [3], and is organized around four main groups of services as depicted in Figure 3: *User Interface*; *Quality Assurance & Optimization*; *Migration*; *Legal Execution Framework*.

The system architecture is represented using the TOGAF framework [16] and the ArchiMate modeling language [43]. Put simply, TOGAF is a tool for assisting in the acceptance, production, use, and maintenance of architectures. ArchiMate provides uniform representations for diagrams that describe enterprise architectures.

In the architecture representation (see Figure 3) we use two layered views. The *Business Layer* describes how the business processes, services, functions and events are related to each other and with the associated individuals and business units. This layer is defined to be consisting of Information, Product, Process and Organization domains. In the *Application Layer*, the software applications that support the components in the business layer, along with the information processed by these applications, are described. This layer is defined to be consisting of application and data domains.

In the following sections we provide a description of the main services exposed by the broker and of how these services have been implemented. We always refer to Figure 3 and implicitly to our reference scenario (cf. Fig. 2 and Sec. 2). In the text we use the `verbatim` font to denote architecture elements, e.g., components, services, actors, data objects.

### *4.1. User Interface*

The User Interface group of services represents the access point for the Cloud Service Providers (SPs), the Customers or service consumers (SCs), and the Broker Manager. These actors interact with:

- The `Registration & Authentication` service that manages the accreditation with the broker and the authentication for accessing the broker services.

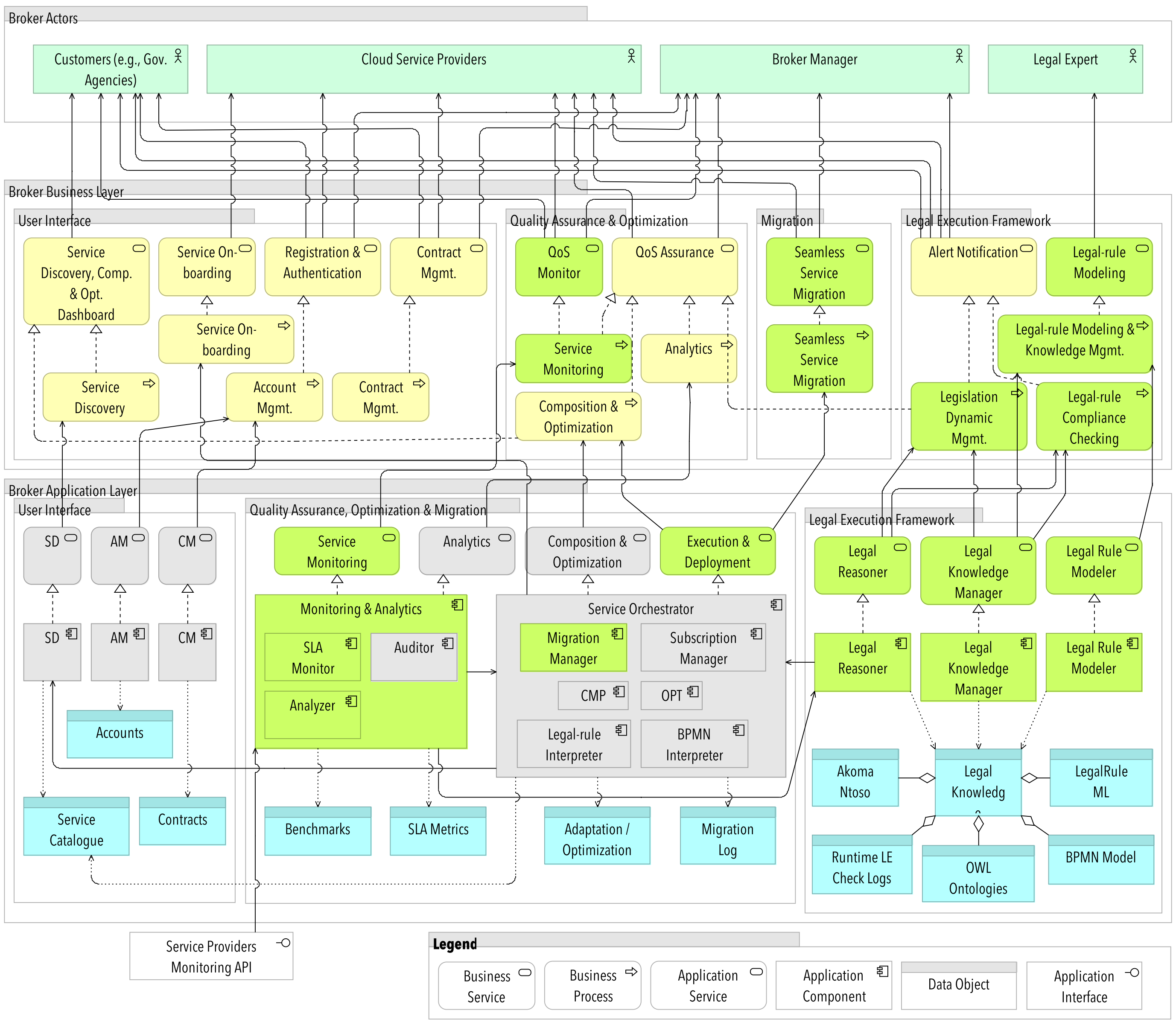


Figure 3: Detailed design architecture of the legal-rule and QoS-aware cloud service broker (we show in green the services, processes, and components that are discussed in details in the paper.

- The `Service On-boarding` service that manages the enrollment of new services in the broker, i.e., it allows the SPs to submit the service description, the offered SLA and it allows to start and manage the service on-boarding phase and the service evolution during the service life cycle. The `Service On-boarding` process is implemented by the `Service Orchestrator` macro-component at the application layer.

- The `Contract Mgmt.` service manages and stores contracts (Service Provider – Broker, Customer – Broker, and Customer – Service Provider);

- The `Service Discovery, Comp. & Opt. Dashboard` provides: a graphical interface allowing the SCs to specify their functional and non-functional requirements as facts for the instantiation of the Legal Knowledge; an automated service discovery tool to search in the cloud service metadata registries the services matching the customers' needs. This tool returns to the customers the (possibly aggregated) services matching their requirements, sorted by some utility criterion. The services are ranked on the basis of an acceptance score.

These end-user functionalities are not further discussed in the paper.

### 4.2. *Quality Assurance and Optimization*

The second group of services is the `Quality Assurance & Optimization` which is composed of:

- The `QoS Assurance` service: i) it provides the capabilities to verify off-line the service compliance with the constraints imposed by law and regulation. It takes as input all the information provided by the SP during the service registration process. ii) It uses the Legal Execution Framework to automatically check (at run-time) the compliance to legislation in term of non-functional requirements, business processes, standard adherence, and other constraints. iii) It uses the data produced by the monitoring and analytics processes to asses the QoS level of the services.

- The `QoS Monitor` service is used by the Broker Manager and the Customers to continuously monitor the QoS level of their services and to asses the fulfillment of contract obligations. The Cloud Service Provider accesses this service to stream QoS monitored data through the broker.

The `QoS Assurance` service offers a wide range of functionalities, hence it is realized by means of a set of business processes, that are: `Service Monitoring`, `Analytics`, and `Composition & Optimization`. Such processes are implemented by a complex set of application layer services and components.

`Service Monitoring` is implemented by the `Monitoring & Analytics` component and is responsible for the continuous runtime monitoring (`SLA Monitor` component) of the cloud services. Monitoring is related to the SLA metrics described in the `SLA Metrics` data object. Monitoring will be implemented mixing push and pull modes. This is required because some QoS metrics, such as scalability, elasticity and consistency (of storage systems), are evaluated through a periodic benchmarking activity. The goal of the `Analytics` component is to put in place data analysis techniques with goal of predicting and detecting SLA violations. The component inspects the monitored data, computes direct and indirect metrics, determines if the SLAs are violated, and/or forecasts the short term value for those metrics. Hence, the `Analytics` component can trigger some form of system adaptation to avoid SLA and legal-rule violations.

The `Service Orchestrator` component implements the `Composition & Optimization` service and the `Execution & Deployment` service. The `Service Orchestrator` is notified by the `Monitoring & Analytics` component and the `Legal Reasoner` component in case of SLAs/law/regulation violations. This component is in charge of planning some adaptation policy that could involve: service re-configuration, resource provisioning, traffic re-routing, service migration. The service adaptation policy has, of course, the goal of maintaining the compliance with law and regulation, but also to guarantee that all the non-functional constraints are satisfied and the broker and/or customer utility is maximized. Our solution uses a linear programming approach to optimize the service configuration [14].

The `Execution & Deployment` application level service is in charge of: managing the deployment of cloud services subscribed by customers; implementing the decisions determined by the adaptation policy in order to meet customers requirements; and managing the migration of cloud services.

The deployment can be managed in different ways depending on the type of service. For example, the deployment of an application can be entirely managed by the broker composing IaaS or CaaS resources. In case of SaaS, the deployment will be directly managed by the service provider.

## *4.3. Migration*

The `Seamless Service Migration` service provides tools in support of the service migration.

Service migration can be explicitly requested by the customer or can be suggested by the broker as adaptation action. In both cases migration is governed by the `Seamless Service Migration` process, implemented by the `Migration Manager` component (part of the `Service Orchestration component`). The `Migration Manager` is supported by the `Monitoring & Analytics` component and by the `Legal Reasoner` to assess the compliance of the migration with SLAs and legislation.

As discussed in the next section, seamless service migration is based on the assumption that applications and data use portable and interoperable formats and technologies, like containers and container data volumes.

## *4.4. Legal Execution Framework*

The forth group of services is the `Legal Execution Framework`, which is composed of:

- The `Alert Notification` service, which notifies the Broker Manager, the Cloud Service Provider and the Customers about violations of legal-rules or about changes in legislation that impact on functional and non-functional requirements of cloud services.

- The `Legal-rule Modeling` service enables the description of law and processes. It is realized by the `Legal-rule Modeling & Knowledge Mgmt.` process, implemented by the application services `Legal Knowledge Manager` and `Legal Rule Modeler`.

The `Alert Notification` is realized by the `Legislation Dynamic Mgmt.` process and the `Legal-rule Compliance Checking` processes, implemented by the `Legal Reasoner` and the `Legal Knowledge Manager` application layer services.

The `Legal Reasoner` is in charge of evaluating the legislation compliance by means of legal reasoning engines (e.g., SPINdle [44]) and business process modeling. The approach used is both for forward compliance checking [45] in order to forestall violations and for backward compliance checking [46, 47].

The `Legal Rule Modeler` service is essentially a set of tools (e.g., RAWE [48] and LIME editor for rules) to describe law and processes. Such tools are based on a set of XML standards (e.g., Akoma Ntoso [49], LegalRuleML [50] and LIME [51]).

The `Legal Knowledge Manager` application layer service is in charge of constantly monitoring and analyzing the law and regulation landscape, by means of natural language processing (NLP) tools and legal ontologies oriented to minimize the legal sources modeling and also to improve the semantic Web query on legal documents. This service will notify the `Legislation Dynamic Mgmt.` in case of any legislation change.

### *4.5. Data Model*

The broker components use and produce many data which are stored in specific repositories. In the bottom of Figure 3 we represent the main data objects. Data objects contain the information needed to implement each business process. In what follow we describe only the data object strictly related with functionalities F1 - F4, that are `Benchmark`, `SLA metrics`, `Migration log` and the `Legal Knowledge`.

The `Benchmark` data object is designed to represent all the information needed to run the benchmark for scalability and integrity evaluation (e.g. the dataset for workload generation and the benchmark configuration parameters) and the benchmark results.

The `SLA metrics` data object is designed to represent the list of metrics and the values for each metric and for each service.

The `Migration log` data object is designed to represent all the information monitored during a migration process, e.g. start timestamp for the migration, peers active in the migration, data successfully migrated, progress of the migration, errors, aborts.

The `Legal Knowledge` data object aggregates many data objects that allow to represent the legislation and to assess legislation compliance: Akoma Ntoso and LegalRule ML descriptions of law, OWL ontologies, BPMN model description of legal compliant business processes and the logs of run-time check of legal rule compliance and dynamic legislation changes.

## 5. Design Choices

In the section, we discuss the main choices that we have faced in designing the legal-rule and QoS-aware cloud service broker. As mentioned in Section 1, our analysis is focused only on functionalities F1–F4 plus the cloud service deployment model. Table 1 summarizes the solutions we adopt and the technologies used to tackle our four design challenges.

### *5.1. Cloud Service Deployment Model*

In the proposed architecture, the broker plays an active role in the deployment phase of the cloud services subscribed by customers, except for SaaS solutions. In our solution we assume has the broker has the full control on the service deployment process, since such approach offers many advantages. First, it makes possible the activation of QoS monitoring agents and, eventually, the initiation/implementation of reconfiguration actions to satisfy SLA and maintain legal-rule compliance. Second, keeping separated the infrastructure and the container services from the applications allows the broker to select the application that best fits the customer requirements (both functional and non functional ones) as well as the infrastructure on which such application will be deployed that allows to match non-functional requirements regarding the location of data and computation, regulation, and standard compliance. The selection, composition, and orchestration of services can be either human-assisted or automatic [14]. Third, it makes easy to notify the legislation changes and the related corrective actions to be taken to maintain compliance. Finally, the full control of the broker permits to suspend an application or an infrastructure service at any time, so the enforcement process of any policy violation may be immediately taken.

TOSCA is our state of the art choice to enable this deployment model and the related benefits, since it provides a common service description and orchestration format, independent from the specific technology and vendor-neutral.

Application service providers willing to leverage the broker functionalities have to comply with the broker deployment model and therefore have to offer containerized applications. Until a standard for containers and their orchestration will not clearly emerge from the ongoing work of standardization bodies and groups, our brokering solution will rely on Docker and Kubernetes, along with Cloudify.

### *5.2. Seamless Service Migration*

The vendor lock-in is an essential issue in cloud computing, mainly because there is no agreement on common standards among cloud providers and/or because proprietary solutions are often used. One of the goals of our cloud service broker is to address the vendor lock-in problem by providing service portability and guaranteeing seamless service migration.

Portability is related both to services and data [52, 15]. Service portability is intended as IaaS-to-IaaS (or CaaS-to-CaaS) portability and is mainly concerned with the transferability of the application from a computing environment to another. Data portability aims at allowing the customer to migrate its data independently from the application. While service portability implies data portability, the vice-versa does not hold. Portability not only is instrumental in avoiding vendor lock-in, but it also allows to guarantee the continuity of the service in case the cloud service provider goes bankrupt, and to guarantee optimal service adaptation by means of service composition and service migration.

Containers offer a partial solution for both service and data portability and this is the technology we selected as enabler for seamless service migration. Containers can be deployed in two modes: *container-on-VM* that is, containers run on top of any virtual machine (VM) provided by any cloud service provider; or *container-on-baremetal* that is, containers run directly on the server OS. The first approach allows to use consolidated solutions for resource allocation and scaling, but vanishes the lightweight nature of containers because of the double virtualization layer. The second solution is supported by new OSs, such as CoreOS. Our design choice does not impose a specific container deployment approach, hence is up to the cloud service provider to choose either for a container-on-vm or a container-on-bare-metal solution.

Another facet of the migration problem is the orchestration of containers. The orchestration logic has to be moved on the

Table 1: Design challenges, design solutions and supporting technologies.

| Design challenge | Solution | Supporting technologies |
|---|---|---|
| Cloud service deployment model | Full control of the broker on IaaS/CaaS service deployment | TOSCA, Docker, Kubernetes, Cloudify |
| Seamless service migration | Container as for data and application portability; TOSCA to define a portable model of orchestration logic | |
| Monitoring | A mix of push and pull monitoring models; service providers should expose a Monitoring-as-a-Service (MaaS) solution | Zabbix, Nagios, Prometheus, YCSB, TPC-VMS, TPCx-HS, TPCx-BB, BUNGEE |
| Legal-rule compliance checking; legislation dynamic management | Modeling of law phase to populate the Legal Knowledge base; Run-time checking phase based on the Legislation Dynamic Management process | Eunomos, LIME, RAWE, SPINdle, Regorus |

IaaS/CaaS service provider. A solution to this problem can be the adoption of TOSCA, that provides a way to define a portable model of orchestration logic. Cloudify and Kubernetes are the candidate technological solutions to the problem.

As regards data portability, Docker, which is the major implementation of container technology, provides the Docker Volumes, which are specialized containers intended for data management and keep all the nice features of portability of simple containers.

## 5.3. QoS Monitoring

Monitoring is one of the main challenges in cloud (e.g., [53, 54]) and multi-cloud environments [55], that is the context in which our cloud service broker operates. In [55] the authors clearly state that a multi-cloud monitoring service should provide push-based and/or pull-based options.

In order to monitor QoS properties and assure the SLA fulfillment, we select a combination of the pull-based and the push-based model. The *pull-based* approach foresees that the broker actively queries the deployed applications in order to obtain information about the QoS level and health status of the application. The pull-based monitoring can be implemented in two different modes, that we call active and passive. The *passive* mode foresees that the broker queries the service provider monitoring system to get the required monitored metrics and events. In this case, the main issue is the broker difficulty in customizing the metrics. In the *active* mode, the service provider furnishes the broker with a testing workload and proper benchmark tools and the broker will periodically run tests to evaluate the application performance and ensure SLA compliance. This approach may be used to evaluate key and challenging properties of the cloud environment, like elasticity [56] and consistency of data storage [57], whose related metrics can be negotiated as service level objectives (SLOs) in the SLA agreed with the consumers. However, its main drawbacks are the additional traffic and load it may impose on the service provider infrastructure and platform; furthermore, the active mode is not truly at runtime, so the detection of problematic event patterns and the subsequent triggering of corrective adaptation actions may be delayed.

The *push-based* model foresees that the application and infrastructure providers monitor the QoS metrics specified in each SLA for each customer and actively push notifications regarding the monitored metrics and events to the broker's monitoring server through real-time oriented communication. Essentially, the service provider should expose a Monitoring-as-a-Service (MaaS) solution.

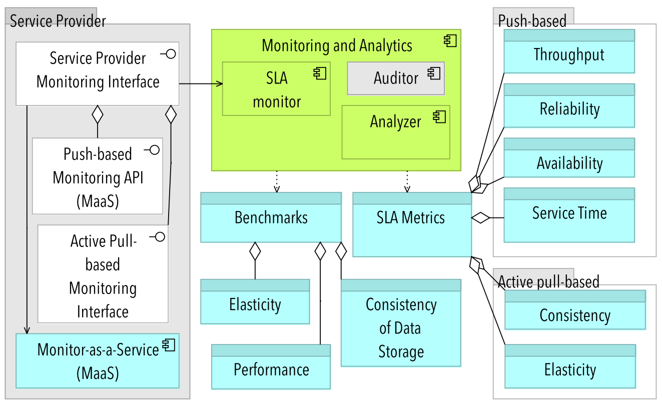


Figure 4: Detailed design of the Monitoring & Analytics component. The Data Objects represent the metrics collected using the push-based approach (e.g., service time, throughput, availability) and active pull-based approach (e.g., elasticity, consistency).

In our broker we opt for a mix of push and pull models (see Figure 4). The services are mainly monitored using a push-based approach; this choice requires the service providers to offer a MaaS service available at least for the broker. Active pull-based measurements can be periodically taken by the broker to perform an audit of the quality of the measures provided by the service provider MaaS. The audit is needed to establish trustworthiness among the consumers and the providers (and the broker itself).

The specific frameworks used for implementing the push-based monitoring are selected by the cloud service providers. Examples of frameworks are: Zabbix [58], Nagios [59], and Prometheus [60]. Furthermore, major cloud providers have their own customized monitoring systems. As regards the tools and benchmarks to support pull-based monitoring it is difficult to provide a comprehensive list because their choice depends on the type of offered cloud service. For example, for NoSQL data storage systems YCSB [61] is a popular benchmark tool that allows also to evaluate consistency. Performances of databases running in virtualized environments can be also measured using the TPC-VMS suite [62], while TPCx-HS [63] and TPCx-BB [64] can be used to benchmark Hadoop run-time, Hadoop Filesystem API compatible systems, and MapReduce layers. As regards elasticity, BUNGEE [65] is a tool for benchmarking the elasticity of IaaS cloud services.

Details about the `Monitoring & Analytics` component and the related data objects are represented in Figure 4. The component accesses the service provider API that are split in: push-based monitoring API (the MaaS), to get QoS metrics measurements, and active pull-based monitoring interfaces, to measure metrics such as elasticity and consistency of data stores, or simply measure performances for audit purposes. Monitored data are stored into corresponding data objects, one for each metric (a small example is provided) for online and off-line processing by the `Analyzer` and the `Auditor`.

## *5.4. Legal-rule Compliance Checking and Legislation Dynamic Management*

The legal compliance checking life cycle is based on two main phases (see Figure 5), the *Modeling phase* and the *Run-time phase*, that are two parallel and interacting processes. The first can be triggered by the legal experts or by the run-time phase process, while the latter uses the output of the modeling phase that is, the legal rules.

In the *Modeling phase*, as soon as a new command is introduced by law (e.g., new informed consent procedure, new archiving method), the models of the business processes that are regulating all the cloud services should be refined according to this modification. The legal knowledge resources where to discover obligations, permissions, rights, prohibitions, penalties, and reparations are mainly the law, the contracts, and the case-law documents.

In the *Run-time phase* the cloud service broker (and cloud service providers joining the marketplace) must invoke the Legislation Dynamic Management process that uses the Legal Knowledge Base for checking if the specific service requested is lawful and policy compliant (cf. Figure 5). This mechanism could be used when a service customer requires a service (forward) or after the service delivery in order to check a possible not eligible behavior (backward). This double approach permits either to avoid the violation of the legal provisions before the service delivery, from the beginning of the system operation, or to repair a possible violation after the delivery of the service.

Backward and forward legal compliance checking are integrated with business process modeling and are modeled adopting a legal-by-design approach. *Backward legal compliance checking* detects the violation after that the process is activated. It is an ex-post analysis of the log file for detecting if something was not properly managed. The cloud service providers should pass to the business process modeling engine all the events as log file. The *Forward legal compliance checking* approach allows to verify the compliance checking both in the design/modeling phase and in the run-time phase. This permits in the design/modeling phase to define correct business processes

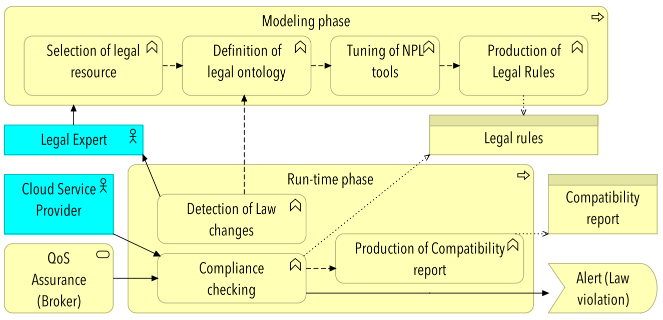


Figure 5: Interaction between the legal-rule compliance checking and legislation dynamic management processes.

according to the law regulations and in the run-time phase to prevent the violation of the law.

In both backward and forward approaches the steps of the life cycle are the following (cf. Figure 5):

1. A legal expert selects resources (all the legislation, regulation, policy, contract, case-law) pertinent with the application domain (e.g., all the privacy regulation) using a particular tool called Eunomos [66].
2. A legal knowledge engineer provides the first core of legal domain ontology of the domain (e.g., privacy, digital identification, cloud computing, etc.).
3. The computational linguists and the engineers tune the NLP tools on the base of the language of the legal documents.
4. Using the web specialized editors LIME and RAWE, the legal expert models the text transforming it into legal rules by using deontic and defeasible logic. In this phase the editors are supported by NLP tools and by legal ontology.
5. The legal engine SPINdle produces the compliance checking with the existing business processes that represent the cloud computing services. In this phase, requests are received from the broker to check the compliance of aggregated and mediated services, both in the on-boarding and in the evolution phase.
6. The legal compliance checking module Regorus [67] provides a report about the business processes not in line with the law. The report produces a list for priority, of justifications and evidence, of the legal original text related to. The report includes also some possible solutions for correcting the business process that breaks the law (e.g., introduce a new consent module, a new information web page). Right now the report is validated by a legal experts in the LIME editor in order to propagate the correct rule. The main challenge in this research is to automate this phase producing a report that will feed directly the broker modules in charge of adapting the services.
7. Periodically, the Eunomos crawler detects the law/regulation changes. New law or fragment of law/regulation/legal material is detected and the NLP tools extract from the new law the modifications that are applied to the existing legal documents database for

producing the updated version. A new updated legal ontology is produced using learning ontology techniques and the rules affected by the modifications are easily retrieved by the legal rule base and the legal knowledge expert tunes them using RAWE editor.

Steps 1-4 characterize the preparatory modeling phase, while steps 5-7 describe the run-time query phase and are cyclically repeated.

## 6. Research Challenges

Notwithstanding the solutions that we have selected to tackle the design challenges, there are some research challenges and open issues that call for an effort from the scientific community. Table 2 summarizes, for each research area of our interest, the research challenges and the state of the art solutions.

### *6.1. Deployment and Migration of Services*

The service deployment and seamless service migration are two strictly related research and technological challenges. Indeed, the choice of a specific deployment model impacts on the selection of proper models and technologies to enable portability (and vice-versa).

Portability, intended as both service portability and data portability, is a hot research topic in cloud computing [52, 15]. Different approaches have been proposed so far: middleware based solutions, e.g., [68] and the mOSAIC platform [25]; software engineering approaches, e.g., [69]; semantic approaches, e.g. [70]; and containers [71, 72], that are the focus of our paper.

The cloud industry recognizes containers as a new and promising solution for service portability, but the landscape is still fragmented despite the increasing adoption and support across major cloud providers. The Docker project has served to make the Docker image format a de facto standard for many purposes. However, this is only a first step toward a more standardization framework. In 2015, the Open Container Initiative [73] was founded with the aim of defining a more formal, open, industry specification and standardization of the container's runtime and format. Such specification should be not bound to higher level constructs, such as a particular client or orchestration stack, not tightly associated with any particular vendor or project, and portable across a wide variety of operating systems, hardware, CPU architectures, public clouds. It would enable portability across compliant runtimes, and provide a robust stand-alone runtime that can directly consume the specification and run a container. Unfortunately, the standardization process is still at its early stage.

Another issue concerning the wide adoption of containers as a portability standard is the fragmented landscape of orchestration frameworks, as observed in [74]. Cloudify and Kubernetes are the main TOSCA compliant implementations that allow the orchestration of Docker containers. However, how to execute and orchestrate containers in a distributed environment without leveraging on hypervisors is still an open issue. Core OS is a first step toward this direction, but it is a young solution and again it is affected by the standardization problem.

### *6.2. Monitoring of Cloud Services*

QoS monitoring in horizontal (i.e., multi-cloud) and vertical (i.e., cross-layer) dimensions of cloud architectures present many challenges [55]. The monitoring system should be capable to accept data from multiple heterogeneous sources and data should be represented in standard formats. The monitoring system should be scalable to cope with a large number of services to be monitored in near-real time and the services could be deployed on geographically distributed systems. The monitoring infrastructure itself should rely on a cloud infrastructure so to elastically scale. Moreover, data stream processing and complex event processing techniques can be leveraged to extract in real-time aggregated metrics from the large volume of detailed monitoring data and perform their analysis.

While the monitoring at the infrastructure layer is a mature field, the monitoring at the application layer requires further investigation. A first issue is determined by the need to observe applications independently from the infrastructure in use and aggregate QoS data from distributed application components that can be spread among multiple clouds. Another issue is related to the correlation and integration of cross-layer monitored data, that come from the application layer and the underlying platform and infrastructure layers, also considering that monitorable information at the lower layers could be limited, especially in public clouds. Some research efforts have recently explored this direction. CLAMBS is an application monitoring and benchmarking as-a-service framework proposed in [75]; it enables QoS monitoring and benchmarking of cloud application components hosted on multiple clouds and across multiple cloud layers. The Ceiloesper framework presented in [76] has the goal to combine together cross-layer cloud monitoring and data stream analysis techniques. The Smart CloudMonitor in [77] is a performance monitoring tool that provides cross-layer performance monitoring capabilities by capturing both the application performance and the infrastructure performance.

In our reference scenario, where cloud service providers expose Monitoring-as-a-Service solutions, there is also the challenge to allow the definition of customizable metrics according to the broker and consumers' needs. Finally, trust and security issues related to monitoring are still two open issues.

A further challenge which is independent of the inter-cloud scenario is the monitoring of containers, that includes monitoring of the containerized environment (i.e., of the application) and monitoring of the container engine/platform. Monitoring techniques and tools used for the operating system and application levels do not allow to catch a wide range of QoS metrics and health state metrics for containers. Docker offers the `docker stat` command that returns CPU and memory utilization for each running container. More detailed CPU, memory and network statistics can be accessed through the `/containers/(id)/stats` API. In [78] the authors modify Docker and Docker Swarm in order to monitor the I/O capacity and utilization of the containers with the goal of controlling the QoS level of a Docker cluster.

Finally, the evaluation of key cloud QoS properties such as elasticity is an open challenge for hypervisor-based cloud

Table 2: Research challenges and state of the art solutions.

| Research area | Research challenges | State of the art solutions |
|---|---|---|
| Cloud broker service deployment, seamless cloud service migration | Service portability | [52, 15, 68, 25, 69, 70, 71, 72] |
| | Standard container format and data volume | [73, 34] |
| | Container orchestration | [74, 39, 35, 17] |
| Cloud service monitoring | Application level monitoring | [75] |
| | Cross-layer monitoring | [76, 77] |
| | Container monitoring | [78] |
| | Elasticity monitoring | [79, 80, 56, 79, 81] |
| | Data consistency | [57] |
| Legal-rule compliance checking | Scalable and run-time legal reasoning | [82] |
| | Multilingual access and management of legal concepts management | [83, 84, 85] |

platforms and a completely unexplored field for operating system/application level virtualization.

Elasticity, which denotes the capability of autonomously adapting the system capacity to workload changes as exact, fast and cheap as possible, is a complex metric because it embeds multiple components, that is scalability, accuracy, time, and cost [79]. In addition, a plethora of factors may affect the elasticity provided by a system and there is not a simple and easy way to quantify this key performance indicator, which should be agnostic with respect to the measured system (e.g., no assumptions about the infrastructure, the technologies and strategies used for providing elasticity). Notwithstanding the recent research works in the literature devoted to the definition of elasticity metrics, e.g., [80, 56, 79, 81], it appears to us that there is not yet a mature representative metric that can be easily and unanimously adopted in the real reference scenario where our broker should operate. In addition, we observe that, to the best of our knowledge, no public cloud provider specifies some elasticity metric as SLO in its offered SLA.

Besides elasticity, another example of advanced QoS property which is of interest when using storage cloud services is data consistency [57]. In the cloud scenario, where storage systems tend to adopt the eventual consistency model, which relaxes consistency guarantees in favor of availability and latency tradeoffs as required by the CAP theorem, the runtime monitoring of the consistency level (and of the degree of inconsistency) cannot be neglected. Indeed, consistency guarantees offered by the storage system impacts on the design and performance of applications that rely on them.

### 6.3. Legal Rule Compliance Checking and Management

One of the most relevant challenges that arise in the definition of the Legal Execution Framework of the proposed broker is how to manage legal concepts in different languages [83, 84, 85], the interpretation level, and the interface modules for permitting human experts to take the final decisions. Furthermore, the large number of legal rules stresses the Legal Reasoner module and requires to define smart solutions for providing reasonable answers to real-time queries coming both from the service providers and consumers. To this end, parallel reasoning approaches can be considered [82].

## 7. Conclusions

In this paper we have proposed the detailed design of a legal-rule and QoS-aware cloud service broker and explored some research challenges that arise from its design. The main open issues from our analysis can be summarized as follows.

Technologies and de-facto standards for service portability are exiting from their infancy stage, but are not yet mature and widely adopted, especially in public clouds. Specifically, the standardization of container technologies and the orchestration of containers in a large-scale distributed environment represent the main open issues.

QoS assessment relies on QoS monitoring and analysis. We have explored the monitoring issues and some main challenges have emerged: monitoring of containers and assessment of key cloud QoS properties, such as elasticity and consistency provided by data storage systems. QoS monitoring techniques for containers are in their infancy and operating system level monitoring tools cannot be successfully applied. Furthermore, the trend towards the design of applications that use microservices running as containerized processes can exacerbate the complexity of monitoring, because the number of application components to be monitored will largely increase and these components could be deployed across geographically distributed data centers. Monitoring and assessment of key cloud properties such as elasticity and consistency still represent an open issue: notwithstanding the increasing amount of recent research efforts on the definition of related metrics and benchmarks, there is not yet a standard and largely adopted solution.

The runtime adaptation continues to be a challenging issue [86]. Although the large amount of research work conducted in the last fifteen years from the field of autonomic computing in the areas of service-oriented and cloud systems and services and the promising results therein obtained, the large-scale heterogeneous and multi-layered cloud environment in which the Cloud service broker will operate introduces deeply challenging problems for the design of effective runtime adaptation techniques.

Models and tools for legal compliance checking and management are available but they are still fragmented and not accessible as a unique framework, and here the main challenge is to deal with different legal tradition sources, legal concepts in different languages, the interpretation level and the interface module for permitting the final decision by human experts.

## Acknowledgments

The work of E. Casalicchio is part of the research project "Scalable resource-efficient systems for big data analytics" funded by the Knowledge Foundation (grant: 20140032) in Sweden. The work of V. Cardellini, G. Interino and M. Palmirani is partially supported by the project: Realization of a Research and Development Project (PCP) on "Cloud for Europe" - Tender Num. 5843932 - CUP C58I13000210006. The authors thank also F. Fiorentino from NTT Data Italia s.r.l. for the fruitful technical discussions.

## References

[1] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, D. Leaf, NIST cloud computing reference architecture, NIST special publication 500 (2011) 292.

[2] N. Grozev, R. Buyya, Inter-cloud architectures and application brokering: taxonomy and survey, Softw. Pract. Exp. 44 (3) (2014) 369–390.

[3] E. Casalicchio, M. Palmirani, A cloud service broker with legal-rule compliance checking and quality assurance capabilities, Procedia Computer Science 68 (2015) 136–150, 1st Int'l Conf. on Cloud Forward: From Distributed to Complete Computing.

[4] Cloud for Europe (C4E), `http://www.cloudforeurope.eu` (2015).

[5] B. D. Martino, G. Cretella, A. Esposito, Towards a legislation-aware cloud computing framework, Procedia Computer Science 68 (2015) 127–135, 1st Int'l Conf. on Cloud Forward: From Distributed to Complete Computing.

[6] A. Amato, S. Venticinque, Multi-objective decision support for brokering of cloud SLA, in: Proc. of 27th Int'l Conf. on Advanced Information Networking and Applications Workshops, WAINA '13, 2013, pp. 1241–1246.

[7] A. Amato, B. Di Martino, S. Venticinque, Cloud brokering as a service, in: Proc. of 8th Int'l Conf. on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC '13, IEEE, 2013, pp. 9–16.

[8] L. D. Ngan, R. Kanagasabai, OWL-S based semantic cloud service broker, in: Proc. of IEEE 19th Int'l Conf. on Web Services, ICWS '12, 2012, pp. 560–567.

[9] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, I. M. Llorente, Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers, Future Gener. Comput. Syst. 28 (2) (2012) 358–367.

[10] I. Houidi, M. Mechtri, W. Louati, D. Zeghlache, Cloud service delivery across multiple cloud platforms, in: Proc. of 2011 IEEE Int'l Conf. on Services Computing, SCC '11, 2011, pp. 741–742.

[11] P. Pawluk, B. Simmons, M. Smit, M. Litoiu, S. Mankovski, Introducing STRATOS: A cloud broker service, in: Proc. of IEEE 5th Int'l Conf. on Cloud Computing, CLOUD '12, 2012, pp. 891–898.

[12] M. B. Chhetri, S. Chichin, Q. B. Vo, R. Kowalczyk, Smart cloud broker: Finding your home in the clouds, in: Proc. of IEEE/ACM 28th Int'l Conf. on Automated Software Engineering, ASE '13, 2013, pp. 698–701.

[13] D. Thatmann, M. Slawik, S. Zickau, A. Küpper, Towards a federated cloud ecosystem: Enabling managed cloud service consumption, in: Economics of Grids, Clouds, Systems, and Services, Vol. 7714 of LNCS, Springer, 2012, pp. 223–233.

[14] E. Casalicchio, An autonomic legal-rule aware cloud service broker, in: Proc. of 2015 Int'l Conf. on Cloud and Autonomic Computing, ICCAC '15, 2015, pp. 216–219.

[15] D. Petcu, A. Vasilakos, Portability in clouds: approaches and research opportunities, Scalable Computing: Practice and Experience 15 (3) (2014) 251–270.

[16] The Open Group, TOGAF, `https://www.opengroup.org/togaf`.

[17] D. Bernstein, Containers and cloud: From LXC to Docker to Kubernetes, IEEE Cloud Computing 1 (3) (2014) 81–84.

[18] AgID, Annex IV (B) Technical Specification: federated certified service brokerage of EU public administration cloud, `http://www.agid.gov.it/cloudforeurope` (2015).

[19] A. Longo, M. Zappatore, M. A. Bochicchio, B. Livieri, N. Guarino, D. Napoleone, Cloud for Europe: The experience of a tenderer, in: Proc. of 30th Int'l Conf. on Advanced Information Networking and Applications Workshops, WAINA '16, 2016, pp. 153–158.

[20] Gartner, Cloud services brokerage (CSB), `http://www.gartner.com/it-glossary/cloud-services-brokerage-csb` (2015).

[21] G. F. Anastasi, E. Carlini, M. Coppola, P. Dazzi, QBROKAGE: A genetic approach for QoS cloud brokering, in: Proc. of IEEE 7th Int'l Conf. on Cloud Computing, CLOUD '14, 2014, pp. 304–311.

[22] S. Nair, et al., Towards secure cloud bursting, brokerage and aggregation, in: Proc. of IEEE 8th European Conf. on Web Services, ECOWS '10, 2010, pp. 189–196.

[23] OPTIMIS, OPTIMIS: Optimized Infrastructure Services, EU FP7 project, `http://www.optimis-project.eu` (2013).

[24] A. Amato, B. Di Martino, S. Venticinque, Evaluation and brokering of service level agreements for negotiation of cloud infrastructures, in: Proc. of 2012 Int'l Conf. on Internet Technology and Secured Transactions, 2012, pp. 144–149.

[25] mOSAIC, mOSAIC: Open source API and platform for multiple Clouds, EU FP7 project, `http://www.mosaic-cloud.eu` (2013).

[26] J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, I. M. Llorente, Scheduling strategies for optimal service deployment across multiple clouds, Future Gener. Comput. Syst. 29 (6) (2013) 1431–1441.

[27] D. Rane, A. Srivastava, Cloud brokering architecture for dynamic placement of virtual machines, in: Proc. of IEEE 8th Int'l Conf. on Cloud Computing, CLOUD '15, 2015, pp. 661–668.

[28] D. Villegas, et al., Cloud federation in a layered service model, J. Comput. Syst. Sci. 78 (5) (2012) 1330–1344.

[29] S. Veloudis, A. Friesen, I. Paraskakis, Y. Verginadis, I. Patiniotakis, Underpinning a cloud brokerage service framework for quality assurance and optimization, in: Proc. of IEEE 6th Int'l Conf. on Cloud Computing Technology and Science, CloudCom '14, 2014, pp. 660–663.

[30] R. Dua, A. R. Raja, D. Kakadia, Virtualization vs containerization to support PaaS, in: Proc. of 2014 IEEE Int'l Conf. on Cloud Engineering, IC2E '14, 2014, pp. 610–614.

[31] R. Pike, D. Presotto, K. Thompson, H. Trickey, P. Winterbottom, The use of name spaces in plan 9, SIGOPS Oper. Syst. Rev. 27 (2) (1993) 72–76.

[32] E. W. Biederman, Multiple instances of the global Linux namespaces, in: 2006 Ottawa Linux Symposium, 2006.

[33] Linux Containers, Linux Containers - LXC, `https://linuxcontainers.org/lxc/introduction` (2016).

[34] D. Merkel, Docker: Lightweight Linux containers for consistent development and deployment, Linux J. 2014 (239).

[35] Kubernetes, `http://kubernetes.io` (2016).

[36] OASIS, Topology and orchestration specification for cloud applications, Tech. Rep. Version 1.0, OASIS Standard (2013).

[37] T. Binz, U. Breitenbücher, O. Kopp, F. Leymann, TOSCA: Portable automated deployment and management of cloud applications, in: Advanced Web Services, Springer, 2014, pp. 527–549.

[38] H. Surti, D. Palma, K. Thangavelu, P. Lipton, S. Moser, Understanding TOSCA and containers, Tech. Rep. Version 1.0, OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC (2014).

[39] Cloudify, Cloudify: pure-play cloud orchestration & automation based on TOSCA, `http://getcloudify.org` (2016).

[40] A. J. H. Simons, et al., Advanced service brokerage capabilities as the catalyst for future cloud service ecosystems, in: Proc. of 2nd Int'l Workshop on CrossCloud Systems, CCB '14, ACM, 2014.

[41] F. Fowley, C. Pahl, L. Zhang, A comparison framework and review of service brokerage solutions for cloud architectures, in: Service-Oriented Computing – ICSOC 2013 Workshops, Vol. 8377 of LNCS, Springer, 2014, pp. 137–149.

[42] Broker@Cloud, Enabling continuous quality assurance and optimization in future enterprise cloud service brokers (Broker@Cloud), FP7-ICT EU project, `http://www.broker-cloud.eu` (2012).

[43] The Open Group, The ArchiMate® Enterprise Architecture Modeling Language, `http://www.opengroup.org/subjectareas/`

enterprise/archimate-overview.

[44] H.-P. Lam, G. Governatori, The making of SPINdle, in: Rule Interchange and Applications, Springer, 2009, pp. 315–322.

[45] G. Governatori, The Regorous approach to process compliance, in: EDOC Workshop, 2015, pp. 33–40.

[46] M. El Kharbili, Business process regulatory compliance management solution frameworks: A comparative evaluation, in: Proc. of 8th Asia-Pacific Conference on Conceptual Modelling, Vol. 130 of CRPIT, ACS, 2012, pp. 23–32.

[47] S. C. Tosatto, P. Kelsen, Q. Ma, M. E. Kharbili, G. Governatori, L. W. N. van der Torre, Algorithms for tractable compliance problems, Frontiers of Computer Science 9 (1) (2015) 55–74.

[48] M. Palmirani, L. Cervone, O. Bujor, M. Chiappetta, RAWE: An editor for rule markup of legal texts, in: Joint Proc. of 7th Int'l Rule Challenge, Vol. 1004 of CEUR Workshop Proceedings, 2013.

[49] UNDESA, Akoma Ntoso, XML for parliamentary, legislative and judiciary documents, http://www.akomantoso.org (2015).

[50] OASIS, LegalRuleML, enabling legal arguments to be created, evaluated, and compared using rule representation tools, https://www.oasis-open.org/committees/legalruleml (2015).

[51] CIRSFID and University of Bologna, Language independent markup editor (LIME), http://lime.cirsfid.unibo.it (2015).

[52] B. D. Martino, G. Cretella, A. Esposito, Advances in applications portability and services interoperability among multiple clouds, IEEE Cloud Computing 2 (2) (2015) 22–28.

[53] G. Katsaros, G. Kousiouris, S. V. Gogouvitis, D. Kyriazis, A. Menychtas, T. Varvarigou, A self-adaptive hierarchical monitoring mechanism for clouds, J. Syst. Softw. 85 (5) (2012) 1029–1041.

[54] J. Montes, A. Sánchez, B. Memishi, M. S. Pérez, G. Antoniu, GMonE: A complete approach to cloud monitoring, Future Gener. Comput. Syst. 29 (8) (2013) 2026–2040.

[55] M. Smit, B. Simmons, M. Litoiu, Distributed, application-level monitoring for heterogeneous clouds using stream processing, Future Gener. Comput. Syst. 29 (8) (2013) 2103–2114.

[56] M. Becker, S. Lehrig, S. Becker, Systematically deriving quality metrics for cloud computing systems, in: Proc. of 6th ACM/SPEC Int'l Conf. on Performance Engineering, ICPE '15, 2015, pp. 169–174.

[57] D. Bermbach, S. Tai, Benchmarking eventual consistency: Lessons learned from long-term experimental studies, in: Proc. of 2014 IEEE Int'l Conf. on Cloud Engineering, IC2E '14, 2014, pp. 47–56.

[58] Zabbix, http://www.zabbix.com (2016).

[59] Nagios, https://www.nagios.org (2016).

[60] Prometheus, https://prometheus.io (2016).

[61] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, R. Sears, Benchmarking cloud serving systems with YCSB, in: Proc. of 1st ACM Symp. on Cloud Computing, SoCC '10, 2010, pp. 143–154.

[62] W. D. Smith, S. Sebastian, Virtualization performance insights from TPC-VMS, Tech. rep., Intel Corporation (2013).

[63] TPC, TPC Express Benchmark HS (TPCx-HS), http://www.tpc.org/tpcx-hs/ (2016).

[64] TPC, TPC Express Benchmark BB (TPCx-BB), http://www.tpc.org/tpcx-bb/ (2016).

[65] N. R. Herbst, S. Kounev, A. Weber, H. Groenda, BUNGEE: An elasticity benchmark for self-adaptive IaaS cloud environments, in: Proc. of 10th Int'l Symp. on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '15, IEEE, 2015, pp. 46–56.

[66] G. Boella, R. Muthuri, L. Humphreys, Managing legal resources in open government and e-democracy: Eunomos - an AI and law response, in: Proc. of Int'l Conf. for E-Democracy and Open Government, CeDEM '14, Verlagshaus Monsenstein und Vannerdat OHG, 2014.

[67] G. Governatori, S. Shek, Regorous: A business process compliance checker, in: Proc. of 14th Int'l Conf. on Artificial Intelligence and Law, ICAIL '13, ACM, 2013, pp. 245–246.

[68] E. M. Maximilien, A. Ranabahu, R. Engehausen, L. C. Anderson, Toward cloud-agnostic middlewares, in: Proc. of 24th ACM SIGPLAN Conf. Companion on Object Oriented Programming Systems Languages and Applications, OOPSLA '09, 2009, pp. 619–626.

[69] J. Miranda, J. Guillén, J. M. Murillo, C. Canal, Assisting cloud service migration using software adaptation techniques, in: Proc. of IEEE 6th Int'l Conf. on Cloud Computing, CLOUD '13, 2013, pp. 573–580.

[70] E. Kamateri, et al., Cloud4SOA: A semantic-interoperability PaaS solution for multi-cloud platform management and portability, in: Proc. of 2nd European Conf. on Service-Oriented and Cloud Computing, Vol. 8135 of LNCS, Springer, 2013, pp. 64–78.

[71] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, L. Peterson, Container-based operating system virtualization: A scalable, high-performance alternative to hypervisors, SIGOPS Oper. Syst. Rev. 41 (3).

[72] W. Felter, A. Ferreira, R. Rajamony, J. Rubio, An updated performance comparison of virtual machines and Linux containers, in: Proc. of 2015 IEEE Int'l Symp. on Performance Analysis of Systems and Software, ISPASS '15, 2015, pp. 171–172.

[73] Open Container Initiative, https://www.opencontainers.org (2016).

[74] A. Tosatto, P. Ruiu, A. Attanasio, Container-based orchestration in cloud: State of the art and challenges, in: Proc. of 9th Int'l Conf. on Complex, Intelligent, and Software Intensive Systems, CISIS '15, 2015, pp. 70–75.

[75] K. Alhamazani, R. Ranjan, P. P. Jayaraman, K. Mitra, F. Rabhi, D. Georgakopoulos, L. Wang, Cross-layer multi-cloud real-time application QoS monitoring and benchmarking as-a-service framework, IEEE Trans. Cloud Comput.To appear.

[76] D. Bruneo, F. Longo, C. C. Marquezan, A framework for the 3-D cloud monitoring based on data stream generation and analysis, in: Proc. of 2015 IFIP/IEEE Int'l Symp. on Integrated Network Management, IM '15, 2015, pp. 62–70.

[77] M. B. Chhetri, S. Chichin, Q. B. Vo, R. Kowalczyk, Smart CloudMonitor - providing visibility into performance of black-box clouds, in: Proc. of IEEE 7th Int'l Conf. on Cloud Computing, CLOUD '14, 2014, pp. 777–784.

[78] S. McDaniel, S. Herbein, M. Taufer, A two-tiered approach to I/O quality of service in Docker containers, in: Proc. of 2015 IEEE Int'l Conf. on Cluster Computing, CLUSTER '15, 2015, pp. 490–491.

[79] M. Beltran, Defining an elasticity metric for cloud computing environments, in: Proc. of 9th EAI Int'l Conf. on Performance Evaluation Methodologies and Tools, VALUETOOLS '15, 2016, pp. 172–179.

[80] N. R. Herbst, S. Kounev, R. Reussner, Elasticity in cloud computing: What it is, and what it is not, in: Proc. of 10th Int'l Conf. on Autonomic Computing, ICAC '13, USENIX, 2013, pp. 23–27.

[81] E. F. Coutinho, P. A. Rego, D. G. Gomes, J. N. de Souza, Physics and microeconomics-based metrics for evaluating cloud computing elasticity, J. Netw. Comput. Appl. 63 (2016) 159–172.

[82] F. Cerutti, I. Tachmazidis, M. Vallati, S. Batsakis, M. Giacomin, G. Antoniou, Exploiting parallelism for hard problems in abstract argumentation, in: Proc. of 29th Int'l Conf. on Artificial Intelligence, AAAI '15, 2015, pp. 1475–1481.

[83] E. Francesconi, G. Peruginelli, E. Steigenga, D. Tiscornia, A semantic approach to support cross border e-justice, in: The Semantic Web: ESWC 2014 Satellite Events, Vol. 8798 of LNCS, Springer, 2014, pp. 204–208.

[84] D. Bourcier, P. De Filippi, Cloud computing: New research perspectives for computers and law, in: AI Approaches to the Complexity of Legal Systems - Models and Ethical Challenges for Legal Systems, Legal Language and Legal Ontologies, Argumentation and Software Agents, Vol. 7639 of LNAI, Springer, 2012, pp. 73–92.

[85] E. Francesconi, G. Peruginelli, Transnational access to legal information document identification standards for case law, AIDAinformazioni 3–4 (33).

[86] R. Ranjan, B. Benatallah, S. Dustdar, M. P. Papazoglou, Cloud resource orchestration programming: Overview, issues, and directions, IEEE Internet Comput. 19 (5) (2015) 46–56.