

Article

Assuring Brokerage Quality in the Cloud–Edge Continuum

Evangelos Barmpas ¹, Simeon Veloudis ^{1,*}, Yiannis Verginadis ^{2,3} and Iraklis Paraskakis ¹

¹ South-East European Research Centre, 546 22 Thessaloniki, Greece; evbarmpas@seerc.org (E.B.), iparaskakis@seerc.org (I.P.)

² Department of Business Administration, Athens University of Economics and Business, 104 34 Athens, Greece; jverg@aueb.gr or jverg@mail.ntua.gr

³ Institute of Communication and Computer Systems, 157 73 Zografou, Greece

* Correspondence: sveloudis@seerc.org

Abstract

The Cloud–Edge Continuum (CEC) has emerged as a paradigm for distributing computational resources across cloud, fog, and edge layers, enabling latency-sensitive applications to operate efficiently. However, ensuring the quality of service (QoS) brokerage in such environments remains a challenge. Existing frameworks primarily focus on resource management techniques such as allocation, scheduling, and offloading but fail to address the quality assurance of the brokerage process itself. This paper introduces SLA governance as a means of ensuring the quality of service brokerage by validating—through automated reasoning—Service Level Agreements (SLAs) against meta-quality constraints—high-level policies that define permissible QoS conditions. We propose an ontology-driven approach leveraging the ODRL ontology for SLA representation and capturing meta-quality constraints. Our method also enables introspective reasoning ensuring internal SLA consistency. Additionally, we integrate SLA governance with a real-time monitoring framework, the Event Management System (EMS), to continuously track workload performance and trigger SLA adaptation when necessary. This integration ensures that SLA-based brokerage decisions remain dynamic and context-aware.

Keywords: service brokerage; cloud–edge continuum; quality assurance; ontologies; reasoning

1. Introduction

The widespread adoption of Internet of Things (IoT) devices has led to an explosion in low-latency data generation and to the concomitant need for increased computation capacity at the network edge for processing this data. Traditional cloud-based architectures fail to satisfy this need due to their dependence on centralised data centres—typically located far away from data sources—for provisioning computation capacity [1].

Recently, the Cloud–Edge Continuum (CEC) has emerged as a paradigm capable of addressing this need by distributing infrastructural services—i.e., computational resources—across three distinct architectural layers [2–4]: the edge layer, which consists of resource-constrained heterogeneous devices located at the network edge (e.g., sensors); the cloud layer, characterised by resource-rich and predominantly homogeneous servers housed in large data centres; and the intermediary fog layer, which comprises heterogeneous servers (e.g., gateways, micro-data centres) positioned closer to data sources.

A key challenge in the CEC is service brokerage [5], which involves discovering and orchestrating infrastructural services across multiple heterogeneous administrative domains



Academic Editors: Jordi Mateo-Fornés and Choong Seon Hong

Received: 23 January 2026

Revised: 13 February 2026

Accepted: 16 February 2026

Published: 19 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

in the three CEC layers to optimally satisfy Service Level Objectives (SLOs)—i.e., preset quality of service (QoS) constraints articulated in Service Level Agreements (SLAs). Several frameworks have been proposed to address this challenge. These generally embrace efficient resource management as a fundamental strategy for achieving QoS. They focus on: resource allocation and auto-scaling, for dynamically provisioning sufficient computational resources for workloads [6–17]; resource scheduling, for ensuring coordinated and timely access to resources amidst fluctuating environmental conditions [13,18–20]; and computation offloading, for migrating workloads to resource-rich nodes to meet performance requirements [7,21–24].

However, these frameworks do not address the challenge of ensuring the quality of service brokerage, i.e., the extent to which service brokerage enables appropriate service consumption that is governed and enforced with respect to consumer requirements and contractual intent. Here, appropriateness does not refer to optimising operational metrics such as cost or resource utilisation per se, but to ensuring that brokerage decisions are based on SLAs that are valid, consistent, and compliant with higher-level governance constraints that encode consumer needs.

We argue that this is fundamentally a meta-quality assurance problem that amounts to determining whether the SLAs, on which service brokerage is based on, comply with meta-quality constraints. Meta-quality constraints define higher-level compliance conditions that govern individual SLOs or entire SLAs, ensuring that service brokerage decisions adhere to overarching quality principles. They may restrict the values of individual SLOs—e.g., “the response time SLO should invariably impose a threshold below 10 ms”. Or they may be more general, focusing on workload consumption as a whole—e.g., “if any SLO is violated more than n times in an hour, workload consumption must be downgraded to a lower QoS (and thus cheaper) level, and if no such level exists it must be terminated altogether”.

This paper proposes a solution to this problem that embraces SLA governance as a primary approach to ensuring the quality of service brokerage by reasoning about SLA compliance with meta-quality constraints, i.e., higher-level quality conditions, which may be simple—governing individual SLOs (such as the response time SLO in the example above), or more complex—governing the entire SLA and essentially defining the conditions under which a workload may or may not be consumed (cf. the more general meta-quality constraint in the example above). Beyond this core contribution, our approach also supports introspective reasoning, ensuring that SLAs are internally consistent, i.e., free from contradictory or overlapping terms.

Achieving effective SLA governance hinges upon a formalism that can accurately capture QoS requirements within the CEC, by accommodating several idiosyncratic characteristics of CEC environments. Firstly, it must support context-aware SLOs that dynamically assess quality across diverse CEC layers and across heterogeneous near-edge environments, with varying QoS characteristics and requirements. For instance, a workload may be associated with an SLO based on the composite metric “workload availability (Workload availability is interpreted in this example as the ability of a workload to deliver timely results.)”, represented as a vector of two simple (directly measurable) metrics—“uptime” and “response time”. However, when the workload is deployed in near-edge environments with high network variability, a third simple metric—“error rate”—must also be included in the composition to account for potential service disruptions. Secondly, it must support interoperable SLOs that are understood across heterogeneous CEC environments. Thirdly, it must enable the specification of diverse service levels to handle performance fluctuations due to high network and workload variability, as well as due to the limited processing power and energy constraints, in near-edge environments; it must also be capable of describing varied pricing models and compensation schemes that may be associated with

these service levels. Fourthly, it must be able to accurately describe monitoring schemes that can address diverse monitoring requirements in heterogeneous CEC environments and integrate seamlessly with monitoring frameworks that can implement these schemes.

This paper advocates semantic technologies—ontologies in particular—for the representation of SLAs in the CEC. Through their ability to support inferential reasoning, ontologies pave the way for formally assuring the quality of service brokerage. They offer an effective approach to addressing the idiosyncrasies of CEC environments and provide context awareness through integration of external domain models, as well as through the ability to precisely and unambiguously define and interpret custom composite metrics for assessing workload performance [25]. They ensure interoperability, they support diverse levels of service provisioning, and they enable the specification of associated settlement schemes that specify how SLO violations are addressed at each level. Appropriate compensation actions—ranging from transitioning to different service levels, to terminating SLAs (with penalties)—may then be automatically triggered at each level through semantic reasoning. For instance, going back to the workload availability example, semantic reasoning may be employed to determine, based on the current deployment location, if an increased error rate should lead to a penalty or trigger a transition to a different service level. This may entail semantic inferencing over an external geospatial ontology to determine if the current deployment location is near-edge. Such semantic inferencing eliminates the need for cumbersome syntactic checks to determine whether the current location qualifies as near-edge.

We opt for the ODRL ontology [26]: a sophisticated and comprehensive formalism that provides appropriate constructs for readily and accurately capturing the necessary knowledge for specifying and sustaining QoS in the CEC. ODRL is a W3C-recommended ontology for representing statements—policies—about the usage of services; automated reasoning is then employed to assess whether the knowledge conveyed by these policies is encapsulated in an SLA. Lastly, we integrate ODRL with a generic monitoring mechanism—the Event Management System (EMS)—able to track real-time workload performance in heterogeneous CEC environments.

From a methodological perspective, this work introduces a governance approach for Cloud–Edge brokerage that separates decision admissibility from decision execution. Brokerage quality is assured by validating that the SLA specifications themselves comply with explicitly modelled meta-quality constraints prior to deployment, while runtime mechanisms are limited to enforcing settlement actions derived from these validated specifications. This abstraction allows brokerage assurance to be treated as a formal reasoning problem, independent of specific resource management or optimisation strategies.

The rest of this paper is structured as follows. Section 2 reviews formalisms for SLA specification based on appropriate criteria. Section 4 presents ODRL and justifies its suitability for CEC environments. Section 4 presents our approach for modelling meta-quality constraints in ODRL. Section 5 showcases the suitability of our approach for assuring service brokerage quality in the CEC. Section 6 presents the EMS. Finally, Section 7 presents conclusions and proposes future avenues of research.

2. Related Work

This section reviews the highly related work in SLA specification and semantic policy modelling.

2.1. SLA Specification

Several formalisms have been proposed for SLA specification [27–33]. We review them below with respect to their appropriateness for CEC environments. The review

is based on criteria drawn from the comprehensive set of criteria devised in [34] for comparing QoS description formalisms. We naturally focus on criteria that pertain to CEC's idiosyncratic characteristics.

2.1.1. Contextualisation

Contextualisation refers to the ability of a formalism to accurately encode the characteristics and requirements of a deployment environment. Table 1 presents several criteria that may be used to assess this ability.

Table 1. Contextualisation Criteria.

Criterion	Description
Customisability	Ability to base SLOs on custom metrics that address the idiosyncrasies of heterogeneous near-edge environments (cf. the workload availability example above) and close the semantic gap between lower-order device-level metrics and the higher-order concepts that users are typically interested in.
Qualifying condition	Ability to define assumptions on network and workload variability that must hold for an SLO to be considered assessable. For example, a response time SLO may be considered assessable only if the egress network bandwidth remains above a predefined threshold, ensuring that fluctuations in bandwidth do not distort response time measurements
Extensibility	Ability to incorporate external domain models. This ability is important in heterogeneous CEC environments as it enables expression of and reasoning about SLOs specific to certain CEC environments.

2.1.2. Interoperability

Interoperability refers to the ability of a formalism to establish a shared vocabulary that ensures the consistent representation, interpretation, and enforcement of SLOs across heterogeneous CEC environments. A main criterion to determine whether a formalism supports interoperability is its ability to semantically describe SLAs.

2.1.3. Quality Levels

Unpredictability near the edge calls for formalisms that are able to describe multiple quality levels of workload consumption in a single CEC environment. Table 2 presents two criteria that may be used to assess this ability. Notably, support for service levels (SLs) enables the definition of compensating actions that transition between different SLs, thus implementing different service degradation/upgrade settlement schemes.

Table 2. Quality levels.

Criterion	Description
Service levels	Ability to define SLO aggregations that capture different qualities of workload consumption [25]
Settlement schemes	Ability to specify different settlement schemes tailored to diverse service levels

2.1.4. Monitoring

This refers to the ability of a formalism to accurately describe monitoring schemes. Table 3 presents two criteria that may be used to assess this ability. Both are important in CEC environments: the former because different resources, hence metrics, may be provisioned by different entities who are responsible for their monitoring; the latter because different CEC environments have different monitoring requirements in terms of measurement frequencies.

Table 3. Monitoring.

Criterion	Description
Metric provider	Ability of a formalism to identify the entity responsible for monitoring the metrics involved in SLA terms [25]
Metric schedule	Ability of a formalism to accurately specify the production frequency of monitoring measurements [25]

Table 4 summarises the results of our review. Note that we consider semantic descriptions as a primary criterion for determining interoperability.

Table 4. Evaluation of SLA formalisms.

QoS-Related Aspect	Criterion	WSLA [27]	WS-A [28]	WSOL [29]	RBSLA [30]	LUA [32]	SLALOM [33]	Q-SLA [25]
Contextualisation	Customisability	yes	no	no	yes	no	yes	yes
	Qualifying condition	yes	no	yes	no	yes	no	yes
	Extensibility	no	no	no	yes	yes	no	yes
Interoperability	Semantic description	no	no	no	yes	yes	no	yes
	Service levels	no	no	no	no	no	no	yes
	Settlement schemes	yes	no	yes	no	yes	no	yes
Monitoring	Metric provider	no	no	no	yes	yes	no	no
	Metric schedule	yes	no	no	yes	yes	no	yes

2.2. Semantic Policy Specification

A number of approaches have been proposed for semantically representing policies [35,36]. These generally rely on OWL [37] for capturing the various knowledge artefacts that reside in policies. In [35], KaoS is presented as a generic framework offering: (i) a human interface layer for the expression of policies; (ii) a policy management layer that is capable of resolving conflicting policies; (iii) a monitoring and enforcement layer that encodes policies in a programmatic format suitable for enforcing them. KaoS lacks any mechanism for explicitly reasoning about inter-policy relations (contradiction, subsumption) and is thus inappropriate for modelling meta-quality constraints. In [36], Rei is proposed: a framework for specifying, analysing and reasoning about policies. A policy comprises a list of rules that take the form of OWL properties; it also comprises a context that defines the underlying policy domain. Rei resorts to the use of constructs adopted from rule-based programming languages for the definition of policy rules. This essentially prevents Rei from exploiting the full inferencing potential of OWL, as policy rules are expressed in a formalism external to OWL.

The Open Digital Rights Language (ODRL) [26] is a W3C-recommended ontology for expressing machine-interpretable policies that specify Permissions, Prohibitions, and Duties governing the use of resources. ODRL represents a policy as a collection of rules defined within a given domain. A rule is an abstract construct capturing the common structural characteristics shared by Permissions, Prohibitions, and Duties (see Figure 1).

correctness of SLAs themselves, prior to service deployment, by validating SLAs against meta-quality constraints and consistency requirements.

Ontology-based reasoning is therefore employed not to assess whether a service instance violates its SLA at runtime but to determine whether a candidate SLA constitutes a permissible and well-formed basis for brokerage decisions. While frameworks such as Linked USDL support semantic reasoning over service descriptions, they do not address this form of meta-level SLA governance. Other SLA formalisms either perform limited syntactic checks or rely on external components for violation detection and enforcement, which we intentionally treat as orthogonal concerns and delegate to other components.

3. Service Level Agreements

To the best of our knowledge, ODRL has not previously been employed as a formal language for modelling Service Level Agreements. At a conceptual level, this alignment is natural: an SLA's stipulations are normative commitments undertaken by a service provider to ensure that a specified service property satisfies given conditions. Such commitments correspond closely to ODRL Duties (`odr1:Duty`), which express obligations that must be fulfilled by a party under defined constraints. An SLA can therefore be interpreted as an `odr1:Policy` comprising multiple duty rules, each associated with specific actions, assets, parties, and constraints. We have extended this framework through the definition of the ODRL SLA profile. Below, we outline this profile and justify its suitability for the CEC environments.

3.1. ODRL SLA Profile

SLAs typically comprise three complementary categories of provisions. First, Service Level Objectives (SLOs) define the normative commitments of a service provider, specifying measurable guarantees such as availability, latency, or throughput [27]. Second, monitoring requirements detail how these guarantees are to be evaluated, including measurement methods, responsible parties, and reporting mechanisms [27,38]. Third, pricing and compensation obligations establish the economic conditions associated with service provision, including fees, penalties, or service credits contingent on SLO adherence or violations [38].

3.1.1. Service-Level Objectives

An SLO expresses a duty to ensure that a raw or composite service metric consistently takes values within a predefined range. Accordingly, an SLO is modelled as an ODRL Duty *D* with the following characteristics.

D is associated with exactly one specialisation of the action `sla:ensure`. This action is introduced by the proposed ODRL SLA profile (`sla:ensure` is not part of the normative ODRL vocabulary [39]). More specific actions may be defined as specialisations of `ensure` through the use of the `odr1:includedIn` object property (Using `sla:ensure` directly as the action of an SLO would be insufficiently specific and therefore semantically meaningless). As an illustrative example, consider the following SLO:

$$\text{response time} \leq 100 \text{ ms.}$$

This SLO is associated with the action `sla:ensure_response_time_H`, which is defined as a specialisation of `sla:ensure`.

The quantitative condition imposed by an SLO is modelled as a refinement of its associated action. In ODRL, refinements restrict the interpretation of actions by associating them with one or more constraints. In the above example, the `sla:ensure_response_time_H` action is refined by an `odr1:Constraint` that restricts the response time metric (i.e., the left operand of the constraint) to values below 100 ms.

D is additionally associated with a target asset representing the objective articulated by the SLO, such as `response_time_objective_H` in the example shown in Figure 2. Note that there may be several SLOs expressing different objectives on the same raw or composite metric. For example, `response_time_objective_H` imposes a stricter requirement on the response time metric than `response_time_objective_L`. As elaborated in Section 3.1.5, these objectives form parts of different service levels: a high service level and a low service level with the former imposing stricter requirements than the latter.

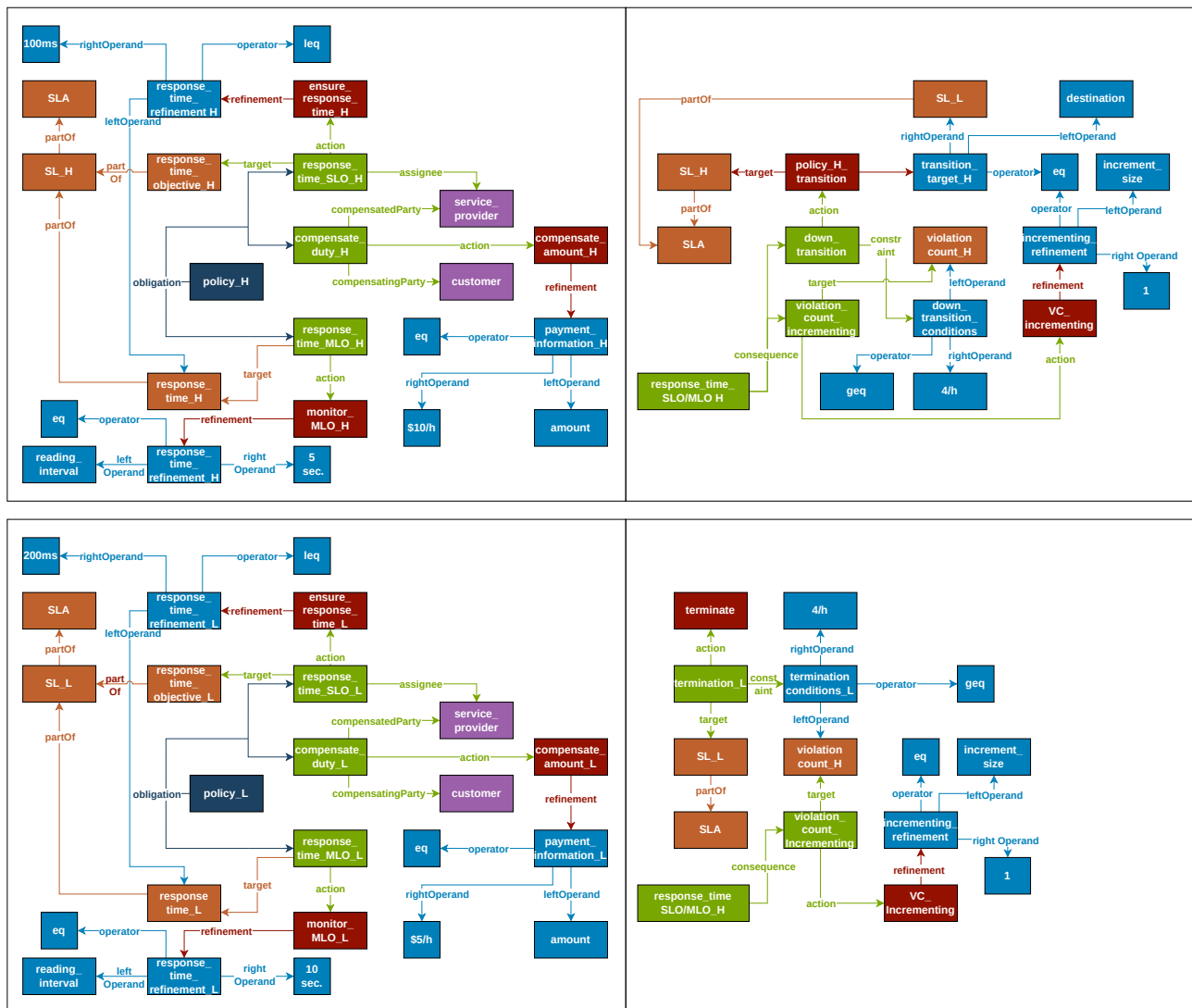


Figure 2. ODRL SLA example: Top and bottom show the high and low SLs, respectively. The color schema in this figure is the same as the following figures. Policies are dark blue, rules are green, actions are red, assets are orange, parties are purple, and constraints, left operands, operators, and right operands are blue. The arrows inherit the color of their origin.

Finally, D is linked via the `odr1:assigner` and `odr1:assignee` object properties to the service consumer and service provider, respectively, thereby explicitly capturing the parties involved in the service commitment.

In ODRL-SLA, duties associated with the `sla:ensure` action are interpreted as persistent obligations to maintain a service property within specified bounds, rather than as one-off action occurrences. Formally, let D be a duty with action a , refined by a set of constraints C over one or more service metrics. The duty D is considered satisfied at time t if and only if all constraints in C evaluate to true with respect to the observed metric values

at t . Conversely, D is considered violated whenever at least one constraint in C evaluates to false. This interpretation assumes continuous or periodic observation of service metrics and adopts an open-world monotonic semantics in which the absence of evidence of violation does not imply satisfaction. These semantic assumptions are introduced at the level of the ODRL-SLA profile and do not form part of the normative ODRL specification.

3.1.2. Monitoring-Level Objectives

MLOs specify requirements on how the raw metrics underlying SLOs are to be observed and evaluated, including aspects such as measurement methods, responsible parties, and reporting mechanisms. MLOs are modelled analogously to SLOs, with the following distinguishing characteristics:

- (i) The action associated with an MLO is semantically subsumed by the ODRL action `odr1:stream`. This action is refined with constraints that specify monitoring parameters, such as measurement or sampling intervals (as illustrated in Figure 2).
- (ii) Each MLO is associated with an `odr1:target` representing a raw metric that may be used either in the construction of composite metrics (cf. Figure 3) or as a left operand in the refinement constraints of SLO actions.
- (iii) Each MLO is linked via the `odr1:assignee` object property to the party responsible for performing the measurements (which need not coincide with the service provider).

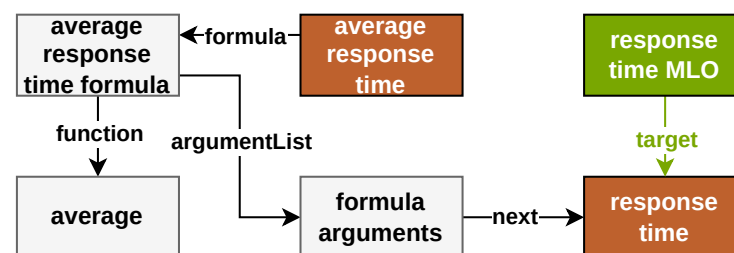


Figure 3. Definition of the “average response time” composite metric.

3.1.3. Pricing

An SLA typically includes one or more compensation duties that specify the pricing and remedial aspects of the agreement. These duties capture the economic obligations associated with service consumption and, where applicable, the consequences of SLO violations (see also Section 3.1.6). Compensation duties are characterised as follows:

- (i) The associated action is defined as a specialisation subsumed by `odr1:compensate`. This action is refined with constraints that specify payment-related information, such as the amount, unit of payment, and payment method.
- (ii) Each compensation duty is associated with both an `odr1:compensatedParty` and an `odr1:compensatingParty`, denoting the party receiving the compensation and the party responsible for providing it, respectively.

3.1.4. Qualifying Conditions

SLOs, MLOs, and pricing duties may additionally be associated with constraints that articulate the conditions under which an assignee must fulfil them. For instance, an SLO or MLO may apply only under high request volumes in order to control operational costs. Similarly, such constraints may be associated with compensation duties to specify payment schedules or tiered pricing schemes. For instance, an SLA may define two distinct compensation duties: one stating that the first 100 h of usage are charged at €0.1 per hour and another stating that thereafter the charge per hour is €0.15.

3.1.5. Service Levels

The ODRL SLA profile supports the definition of multiple Service Levels (SLs) as aggregations of SLA stipulations, encompassing SLOs, MLOs, and pricing-related obligations. Each SL represents a coherent set of normative commitments that defines a specific level of service quality within a CEC environment. Moreover, the profile enables the specification of transitions between SLs in response to SLA violations or changing operational conditions. These transitions are modelled as consequence duties that link the current SL to a target SL and are triggered when predefined conditions—such as cumulative SLO violations or monitored performance thresholds—are met, thereby supporting both service degradation and upgrade settlement mechanisms.

SLAs have a tree-like structure. At the bottom, SLOs and MLOs get aggregated into SLs, which in turn, get aggregated into the SLA. Ontologically, this is achieved through `odr1:Asset` and `odr1:AssetCollection` instances related through the `odr1:partOf` object property. For example, in Figure 2, `response_time_objective_H` and `response_time_H` are `partOf` `SL_H`. In turn, `SL_H` `partOf` `SLA`. An equivalent structure can be seen on the low SL. Overall, the SLA's structure can be seen in Figure 4.

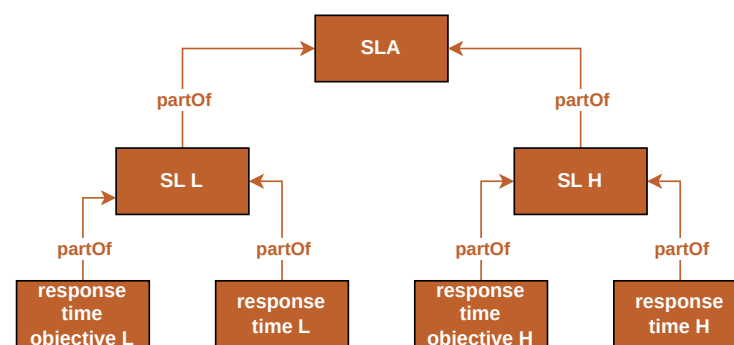


Figure 4. The structure of the SLA shown in Figure 2.

3.1.6. Settlement

Violations of SLOs and MLOs carry settlement consequences which can be categorised into three types:

1. Compensation penalties: providing discounts or other remedial payments when objectives are not met.
2. Transitions: moving from a higher service level (SL) to a lower one.
3. Terminations: ending the SLA.

Settlement consequences are modelled as duties linked to the relevant SLOs and MLOs via the `odr1:consequence` object property and are triggered whenever a duty is unfulfilled. Compensation penalties are analogous to compensation duties, as they require one party to compensate another. However, the roles differ: the service provider, being responsible for failing to meet the SLO, compensates the customer. Additionally, the refinement of the compensation action may indicate a percentage of the price rather than a fixed amount.

Cumulative SLO violations exceeding a predefined threshold can trigger transitions between service levels or termination of the SLA. These are modelled through transition and termination duties (see Figure 5). Each transition links the current SL to the target SL using the `odr1:target` property, while the `odr1:constraint` property defines its activation conditions, typically specifying the number of violations required to trigger the transition or termination.

Transitions and terminations rely on supporting violation-count duties, which maintain a running tally of violations. These duties have a target of `violation_count` and employ the `sla:increment` action (`odr1:includedIn odr1:modify`) with a refinement in-

dicating the increment size (see Figure 5). This mechanism allows different duties to carry different “weights”; for example, a duty incrementing the count by two is more impactful than one incrementing by a single violation. Conversely, a duty may be made soft by omitting its consequences.

Transitions serve to downgrade the service level when the provider cannot meet the SLOs or MLOs. Once lower levels are reached, it may be desirable to upgrade the SL if the provider can sustain a higher QoS. Accordingly, transitions are classified as down-transitions and up-transitions. Both types share the same structural modelling, but they differ in certain associations:

1. The `odr1:refinement` of an up-transition points to a higher SL rather than a lower one.
2. The `odr1:constraint` of an up-transition specifies positive conditions. Whereas down-transition constraints indicate violation thresholds, up-transitions may require a period with no violations or verify that the objectives of the higher SL can be satisfied. For example, an up-transition from SL_L to SL_H (Figure 2) could include a constraint requiring $responsetime \leq 100$ ms.

In Figure 2, the `down_transition` instance enforces a downgrade from the high to the low SL if more than four SLO violations occur within a 1 h window. The `termination_L` instance mandates SLA termination if four or more SLO violations occur in the same period. Notably, this condition applies only when the workload is already operating at the low SL, as indicated by the `termination_L` duty’s `odr1:target` association with the low SL.

Settlement actions can be triggered automatically through ontology-based reasoning. Depending on the scenario, this reasoning may be simple—e.g., evaluating the conditions for a transition—or more complex—e.g., performing semantic inference over an external geospatial ontology to determine whether the current deployment is near-edge and, hence, whether settlement actions should be enforced (cf. the workload availability example in Section 1).

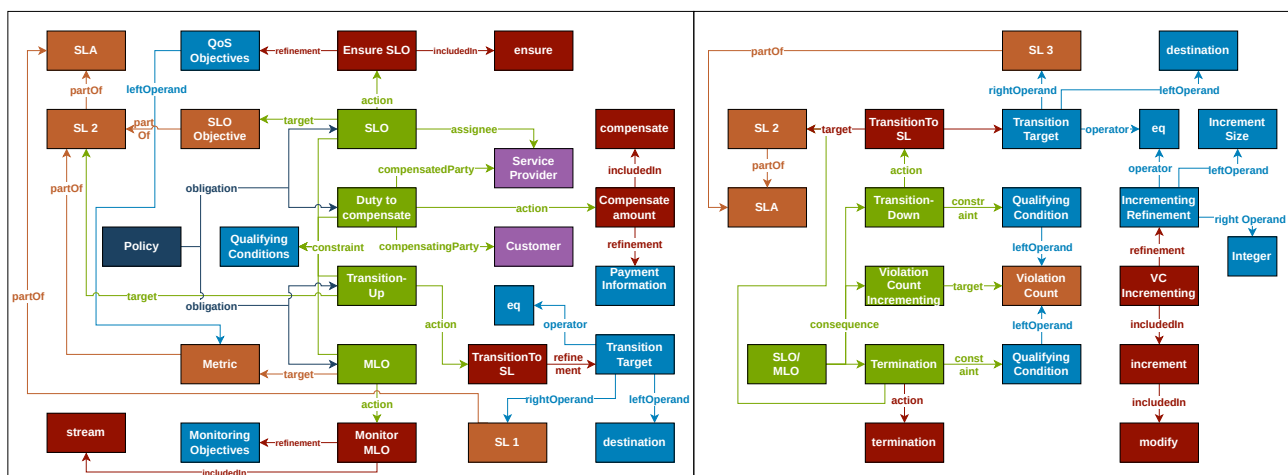


Figure 5. ODRL SLA profile: Left side shows the structure of SLA commitment duties. Right side shows duty consequences.

3.1.7. Metrics

The targets of MLOs represent raw metrics (e.g., “response time”), i.e., metrics that can be directly measured by hardware or software sensors. These raw metrics can be used in SLO action refinements to define the objectives of the SLO, as illustrated in Figure 2. However, raw metrics are often insufficient indicators of overall service quality, as they are

susceptible to noise and transient spikes. For instance, the `response_time_SLO_H` could be considered violated due to a single measurement reaching 101 ms.

As already discussed in Table 1, this limitation is addressed through composite metrics, which are derived by applying a formula to a collection of simple or other composite metrics. Composite metrics allow the definition of higher-order metrics that bridge the semantic gap between low-level device-centric measurements and the performance concepts relevant to end users. In other words, they transform raw data into actionable information.

To support composite metrics, we integrated elements of Q-SLA's metric facet [25]. Specifically, we declare that

$$\text{ComplexMetric} \sqsubseteq \text{LeftOperand},$$

allowing the ODRL model to seamlessly incorporate higher-order metrics. As illustrated in Figure 3, each complex metric is associated with a `qsla:Formula`, comprising a function (e.g., average) and a list of arguments to which the function applies. For example, the “average response time” metric is computed by applying the average function to the “response time” raw metric. More sophisticated constraints can be modelled by nesting multiple raw or composite metrics.

We opt for Q-SLA's metric model because it is more expressive than the models provided by other existing SLA formalisms that enable metric customisability (namely, WSLA, SLALOM, and RBSLA). More specifically, Q-SLA allows metrics to be defined as functions over other metrics, including recursive compositions. In contrast, WSLA treats metrics as measurement directives, SLALOM as UML attributes, and RBSLA as atomic evaluation variables and not as composable entities.

In the SLA example shown in Figure 2, two service levels (SLs) are defined: high (H) and low (L). Each SL comprises an SLO constraining response time, a corresponding MLO measuring the metric, and a compensation duty. Specifically, the high SL aggregates the high-tier versions of these duties (`response_time_SLO_H`, `compensate_duty_H`, and `response_time_MLO_H`), whereas the low SL aggregates their low-tier counterparts (`response_time_SLO_L`, `compensate_duty_L`, and `response_time_MLO_L`). The response time SLOs are linked to the corresponding response time metrics (`response_time_H` and `response_time_L`) via the object property `odrl:leftOperand`, which is constrained by the data property `odrl:rightOperand` (100 ms for the high SL and 200 ms for the low SL).

3.2. Suitability for the Cloud-Edge Continuum

We demonstrate the suitability of ODRL SLA for CEC environments by assessing it against the criteria of Section 2.

The second criterion is achieved through the object property `odrl:constraint` (see Figure 5), which defines the circumstances under which SLOs are significant and may act as QoS constraints that dictate when they can be triggered. The third criterion is inherently satisfied due to ODRL's ontological nature that imposes a modular and hierarchical structure that can be easily extended through subclassing and/or ontology alignment without modifying existing concepts and properties. OWL's Open World Assumption also reinforces extensibility.

3.2.1. Interoperability

This is achieved through ODRL's ability to provide semantic descriptions, i.e., to formalise the knowledge behind SLAs by characterising the concepts involved and the properties thereof, rather than merely syntactically serialising SLAs.

3.2.2. Quality Levels

According to Section 2.1.3, the ability of a formalism to model different levels of service quality hinges on its ability to discern different SLs and on the ability to attach compensating actions to these levels. ODRL SLA meets both criteria as explained in Sections 3.1.5 and 3.1.6.

3.2.3. Monitoring

According to Section 2.1.4, the ability of a formalism to accurately describe monitoring schemes hinges on two criteria: the ability to identify the entity responsible for monitoring the metrics involved in SLA terms and the ability to accurately specify monitoring schedules. With respect to the first criterion, ODRL offers the `odrl:Party` and `odrl:PartyCollection` concepts, along with the `odrl:partOf`, and `odrl:function` object properties.

4. Meta-Quality Constraints

Meta-quality constraints can be understood as *policies* that govern the quality of service brokerage in the CEC. Unlike traditional SLA terms, which specify duties for individual workloads, meta-quality constraints operate at a higher level of abstraction, imposing governance duties that constrain how SLAs are formulated. These duties formulate and impose the conditions under which service brokerage decisions are made, ensuring that SLA constraints abide by overarching quality requirements. For example, a meta-quality constraint may impose a requirement whereby the response time SLO must always impose a threshold below 10 ms. It may also impose a more complex requirement whereby if any SLO is violated more than n times in an hour, workload consumption must be downgraded to a lower QoS level, and if no such level exists, the workload must be terminated.

As policies, meta-quality constraints can be formally expressed using policy languages such as ODRL, enabling automated reasoning for policy enforcement. This allows the dynamic assessment of SLA compliance through formal reasoning and supports decision-making mechanisms that adjust service provisioning based on real-time monitoring.

Meta-Quality Constraints in ODRL

A meta-quality constraint is modelled as an ODRL Duty say D . D takes the form of an OWL class that satisfies the following axioms:

1. It is linked via the object property `odrl:target` to exactly one `odrl:Asset`, which is invariably the SLA that the meta-quality constraint constrains;
2. It is associated via the object property `odrl:constraint` with exactly one `odrl:Constraint` or `odrl:LogicalConstraint`, which articulates the requirements imposed by the meta-quality constraint;
3. It is linked via the object property `odrl:function` to exactly one `odrl:Party`. This property is specialised into either the `odrl:assigner` sub-property, denoting the entity issuing the meta-quality constraint or the `odrl:assignee` sub-property, denoting the entity subject to the constraint, typically the workload consumer;
4. It is connected via the object property `odrl:action` to exactly one `odrl:permit` `odrl:Action`.

These axioms may be formally expressed as terminological (TBox) and assertional (ABox) axioms in the *SRIOQ* Description Logic [40] (DL (*SRIOQ* is the DL underlying

OWL 2; we resort to *SRIOIQ* due to the conciseness and rigorousness of its notation.)) as follows:

$$\begin{aligned}
 D \sqsubseteq & \quad (= 1 \text{ target.Asset}) \sqcap \\
 & \quad (= 1 \text{ constraint.}(\text{Constraint} \sqcup \\
 & \quad \quad \text{LogicalConstraint})) \sqcap \\
 & \quad (= 1 \text{ assigner.Party}) \sqcap (= 1 \text{ assignee.Party}) \sqcap \\
 & \quad (= 1 \text{ action.}\{\text{permit}\}).
 \end{aligned} \quad (1)$$

For any object property P and concept C , $(\leq 1P.C)$ represents the class that comprises all those individuals that have at most one association through P with an instance of C ; similarly, $(\geq 1P.C)$ represents the abstract class that comprises all those individuals that have at least one association through P with an instance of C . The symbols $\sqcup$ and $\sqcap$ represent, respectively, class union and intersection. $(= 1P.C)$ is an abbreviation for $(\leq 1P.C) \sqcap (\geq 1P.C)$.

Notably an SLA may be linked to several meta-quality constraints. Intuitively, an SLA is ‘permissible’ if the logical condition of each of its linked meta-quality constraints is true. All meta-quality constraints linked to a particular SLA should be encapsulated within a single overarching Set policy corresponding to the SLA.

Below, we provide two example meta-quality constraints. The first binds an individual SLO. It states that the response time SLO should invariably impose a threshold below 10 ms. Formally, in *SRIOIQ*,

$$\begin{aligned}
 D_1 \sqsubseteq & \quad \text{constraint.} \\
 & \quad (\text{leftOperand.}\{\text{ResponseTime}\} \sqcap \\
 & \quad \text{operator.}\{\text{lt, leq}\} \sqcap \text{rightOperand} \leq 10 \text{ ms}).
 \end{aligned} \quad (2)$$

The second binds the entire SLA. It states that if any SLO is violated more than 6 times in an hour, workload consumption must be downgraded to a lower QoS level, or (if no such level exists) it must be terminated altogether. Formally, in *SRIOIQ*,

$$\begin{aligned}
 D_2 \sqsubseteq & \quad \text{constraint.}(\text{evaluationPeriod} = 1 \text{ h} \sqcap \\
 & \quad (\text{violationThreshold} \leq 6)).
 \end{aligned} \quad (3)$$

D_2 is also depicted in Figure 6.

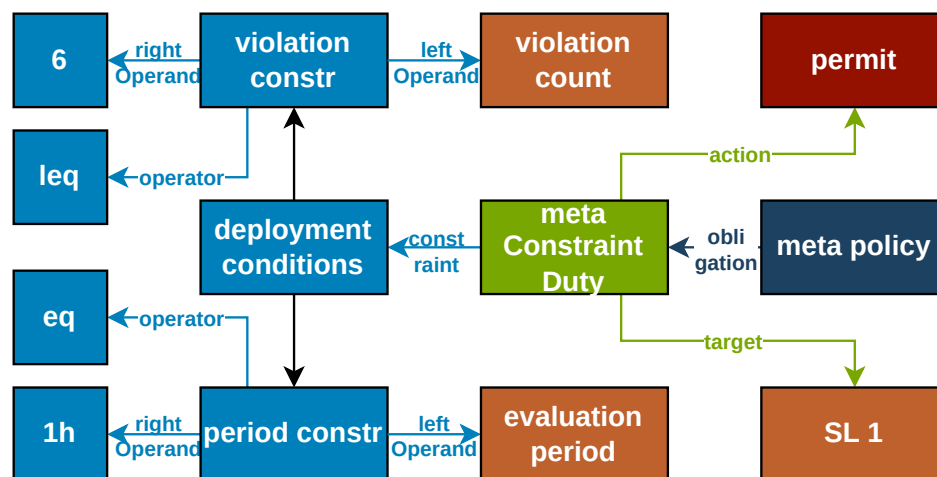


Figure 6. Meta-quality constraint example in ODRL.

5. Service Brokerage Quality Assurance

Brokerage quality assurance is implemented through the two-tier mechanism depicted in Figure 7. The Ontology Server is the mechanism's backbone. It uses the OWL API [41] for manipulating the ontology and performing reasoning. The SLA Generator receives a JSON representation of an SLA and calls the ontology server to parse this representation and capture the SLA using the SLA ODRL profile by creating the required instances. Similarly, the Quality Assurance mechanism receives a JSON representation of the meta-quality constraints that will be harnessing the SLA's terms and conditions and represents them ontologically as ODRL Duties. It then calls the Ontology Server to reason over the admissibility of the generated SLA by ensuring its compliance with meta-quality constraints and also by guaranteeing its internal consistency through introspective checks. It is to be emphasised that both compliance checking against meta-quality constraints and introspective checks are processes that take place offline, prior to application deployment, and run in the cloud as part of the control plane of the brokerage architecture.

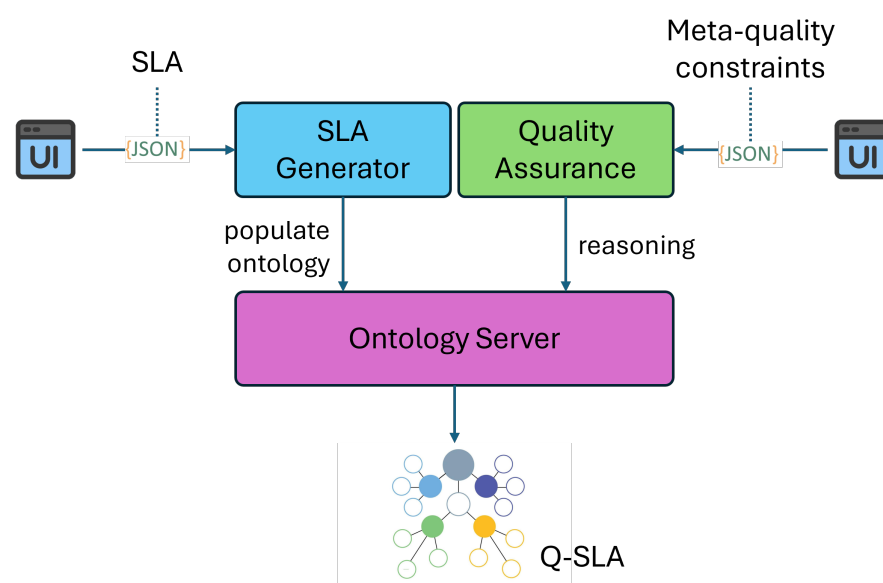


Figure 7. Brokerage Quality Assurance mechanism.

Notably, while introspective reasoning provides an additional layer of validation, our primary focus remains on enforcing the meta-quality constraints to guide policy-driven SLA compliance.

5.1. Meta-Quality Assurance

Meta-quality assurance involves inferential reasoning that aims at ensuring that SLAs abide by ODRL Duties that represent meta-quality constraints. These Duties may be either simple, binding only individual SLOs (see for example D_1 of Section 4), or more complex, governing the entire SLA and defining the terms under which workload consumption must be terminated (see for example D_2 of Section 4). This reasoning process entails the programmatic construction of two OWL classes—one representing the SLA and one representing a meta-quality constraint—and evaluating their relationship. If the SLA class is inferred to be a subclass of the meta-quality constraint class, then the SLA is compliant with the meta-quality constraint, hence admissible as far as this constraint is concerned. To be fully admissible, an SLA must of course satisfy all meta-quality constraints that refer to it.

The computational performance of this reasoning process is illustrated in Figure 8 (Benchmarking was conducted on a system with an AMD Ryzen 7 4800H (8-core) CPU,

16GB of RAM (3200 MT/s), and an NVIDIA GeForce RTX 3050 GPU, running Windows 11 Pro.), which presents the total reasoning time as a function of SLOs and meta-quality constraints. This time is further decomposed into the following: (i) the time required for constructing OWL class expressions and (ii) the time required for subsumption checking. As expected, the reasoning time increases with the SLA complexity, since a larger number of SLOs and constraints results in more intricate class definitions and a higher computational burden. Notably, however, the primary overhead stems from the class expression creation rather than from subsumption checking. This is due to the high volume of calls issued by the Quality Assurance mechanism to the Ontology Server. Each call incurs thread blocking delays while awaiting responses, and additional communication overheads (e.g., propagation delay) further increase latency. The time required for request processing varies significantly. The initial few requests tend to be slower, with one request dropping from 871 ms to 30 ms after approximately 15 similar queries. This behaviour likely results from reasoner optimisations and caching mechanisms.

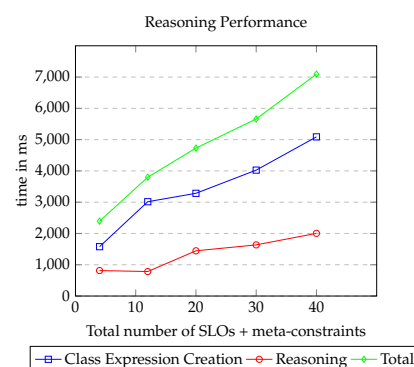


Figure 8. Reasoning performance for different numbers of SLOs and meta-constraints.

The objective of this evaluation is to assess the feasibility and scalability of ontology-based admissibility reasoning as a brokerage control-plane mechanism, rather than to simulate full Cloud–Edge deployment scenarios. Since SLA admissibility checking is performed offline during SLA generation and negotiation, its computational characteristics are independent of specific workload placement or orchestration strategies. Consequently, synthetic benchmarks that vary the number of SLOs and meta-quality constraints provide a controlled and representative assessment of reasoning scalability in realistic brokerage settings.

Benchmarking further revealed that the dominant factor influencing the reasoning time is the length of class expressions, dictated primarily by the number of conditions and SLOs. Other factors—such as the type of check (individual SLOs vs. workload consumption), logical complexity (nested structures), and constraint operators (conjunction vs. disjunction)—had minimal impact on performance. Interestingly, an SLA with two constraints and a meta-quality constraint with 10 conditions performed equivalently to the reverse case, suggesting that the total number of conditions is the key determinant of the reasoning time.

5.2. Introspective Checks for Internal SLA Consistency

Introspective checks entail reasoning to ensure that no contradicting or overlapping information exists within an SLA. Such reasoning may be conducted at the SLO, MLO, and consequence level.

At the SLO and MLO level, it determines whether there are two or more distinct SLOs in the same SL that constrain the same QoS (e.g., two “response time” SLOs or two “availability” SLOs). This may be readily checked by determining whether the SLOs’

`odr1:action odr1:refinement` uses the same `odr1:leftOperand`. An admissible SLA should be free of duplication.

It also determines whether there are two or more distinct MLOs in the same SL that measure the same metric. This can be checked by determining the existence of a metric that is the `odr1:target` of multiple MLOs. An admissible SLA should be free of duplication.

At the consequence level, it determines whether all SLs are reachable via transitions. This is achieved by checking whether every SL that is `odr1:partOf` the SLA is the `odr1:rightOperand` in a constraint where the `odr1:leftOperand` is `sla:destination`, and the operator is `odr1:eq`. An admissible SLA should only contain reachable SLs.

6. Monitoring

As outlined in Section 3.1.6, we propose ontology-based reasoning to trigger settlement actions at the SLO, SL, and SLA levels. However, the effectiveness of this approach depends on the ability to measure, propagate, and process raw metric values in near-edge environments, a task that presents significant challenges due to the limited processing power and energy constraints in these environments. The traditional monitoring approaches are often centralised and therefore inefficient, as they typically require substantial bandwidth to aggregate monitoring data. Such approaches are susceptible to a single point of failure, jeopardising the overall resilience of the monitoring infrastructure.

We address these challenges by proposing the integration of the brokerage quality assurance mechanism of Section 5 with an efficient and resilient monitoring mechanism: the Event Management System (EMS) [42]. The EMS leverages the distributed complex event processing paradigm (DCEP), by employing a network of monitoring agents, which is dynamically and automatically deployed across the CEC. These agents can process real-time monitoring streams, without centralising them, while adapting to any required application deployment topologies within the CEC. Towards minimising the monitoring overhead, EMS evaluates resource nodes' health (e.g., detecting failures) and identifies specific application reconfiguration opportunities. It achieves this by transmitting only the relevant raw and composite metrics, which are associated with the triggering of remediation/settlement actions at the SLO, SL, and SLA levels.

The EMS comprises an Event Processing Manager (EPM) and several Event Processing Agents (EPAs), which autonomously manage application component monitoring tasks. EPM and EPAs formulate an adequate event processing network, where the most resource-capable node automatically undertakes the role of the local monitoring data aggregator. Monitoring occurs at multiple levels, starting from individual instances/pods to geographic areas, availability zones, and cloud regions, ensuring scalability and efficiency. Metric values reaching the brokerage quality assurance mechanism trigger reasoning over the instantiated ODRL SLA profile. Additionally, EMS tracks any SLA settlements and SL transitions by monitoring all the predefined thresholds over specific time and event windows to ensure continuous SLA enforcement.

The EMS is based on the metric model offered by the CAMEL DSL. This model enables users to specify monitoring metrics, logical compositions across time and event windows and SLOs related to the execution of multi-cloud applications. The EMS is integrated into our approach by extending this model to cater for hyper-distributed applications comprising multiple workloads deployed across the CEC. This extension essentially serialises SLOs and the monitoring requirements they impose in CEC environments.

Notably, both the brokerage quality assurance mechanism and the EMS are practically developed as part of the EC-funded NebulOuS project (<https://nebulouscloud.eu/> (accessed on 15 February 2026)). NebulOuS aims to enable transient fog brokerage ecosystems, seamlessly integrating edge and fog nodes with multi-cloud resources to support

low-latency applications. In NebulOuS, the brokerage quality assurance mechanism is responsible for generating SLAs that govern workload consumption and for validating these SLAs against user-defined meta-quality constraints. Meanwhile, the EMS continuously monitors workload execution, detecting SLO violations and triggering settlement actions at the SLO, SL, and SLA levels when necessary. More specifically, the EMS has been deployed across heterogeneous multi-cloud and near-edge environments comprising multiple distributed monitoring agents. The experimental evaluations reported in [42] demonstrate that the distributed CEP architecture achieves near real-time propagation of monitoring events with sub-second processing latency under typical workload conditions, while significantly reducing bandwidth consumption through selective transmission of relevant metrics. In this work, the EMS is leveraged as a scalable monitoring substrate that supplies event streams to the brokerage quality assurance mechanism, rather than as the primary subject of performance evaluation.

7. Conclusions

This paper introduced SLA governance as a fundamental approach to ensuring the quality of service brokerage in the CEC. Existing resource management frameworks focus on provisioning, scheduling, and offloading to meet SLOs but lack mechanisms to ensure that SLAs comply with high-level meta-quality constraints. To address this gap, we proposed an ontology-driven approach that integrates SLA reasoning with real-time monitoring.

We extended the ODRL Information Model to formally represent SLAs and their constraints. We also leveraged ODRL to express meta-quality constraints as policies. Our approach enables both introspective reasoning—verifying SLA consistency—and, crucially, ‘extrospective’ reasoning, ensuring compliance with external constraints.

To facilitate continuous SLA assessment, SLA governance is integrated with a distributed monitoring framework, the Event Management System (EMS). The EMS employs Complex Event Processing (CEP) to track real-time workload performance across heterogeneous CEC environments. It efficiently aggregates, processes, and propagates monitoring metrics while minimising bandwidth and computational overhead. This integration ensures that SLA violations trigger automated transitions between service levels or enforce penalties as per the agreed compensation schemes.

This work has introduced a governance-based approach to assuring brokerage quality in the Cloud–Edge Continuum by validating the admissibility of SLA specifications against explicitly modelled meta-quality constraints. By separating offline SLA admissibility reasoning from runtime monitoring and settlement, the proposed method enables brokerage decisions to be governed in a principled and technology-agnostic manner, independently of specific resource management or optimisation strategies. Looking ahead, emerging Large AI Model services—such as collaborative and partitioned inference across cloud and edge resources—further amplify the importance of such governance mechanisms. Recent work on split and distributed inference of large models highlights the increasing prevalence of multi-stage execution paths, heterogeneous performance characteristics, and context-dependent trade-offs across edge environments [43,44]. In these settings, the service quality depends on composite and dynamically evaluated metrics rather than on isolated measurements. Within this context, the proposed approach can be applied to govern Large AI Model inference services by ensuring that their SLAs—covering dimensions such as latency, accuracy, availability, energy consumption, and cost—remain compliant with higher-level consumer and governance requirements, thereby complementing recent advances in Large AI Model inference optimisation for heterogeneous edge environments.

Author Contributions: Conceptualisation, E.B., S.V. and Y.V.; methodology, S.V.; software, E.B.; original draft preparation, E.B., S.V. and Y.V.; review and editing, S.V. and I.P.; project administration, I.P.; funding acquisition, S.V., Y.V. and I.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work has received funding from the EU's Horizon research and innovation programme under grant agreement No. 101070516 NebulOuS project.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012*; Association for Computing Machinery: New York, NY, USA, 2012; pp. 13–16. [\[CrossRef\]](#)
2. Habibi, P.; Farhoudi, M.; Kazemian, S.; Khorsandi, S.; Leon-Garcia, A. Fog Computing: A Comprehensive Architectural Survey. *IEEE Access* **2020**, *8*, 69105–69133. [\[CrossRef\]](#)
3. Mahmud, R.; Kotagiri, R.; Buyya, R. Fog Computing: A Taxonomy, Survey and Future Directions. In *Internet of Everything. Internet of Things*; Springer: Singapore, 2018; pp. 103–130. [\[CrossRef\]](#)
4. Mouradian, C.; Naboulsi, D.; Yangui, S.; Glitho, R.H.; Morrow, M.J.; Polakos, P.A. A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges. *IEEE Commun. Surv. Tutorials* **2018**, *20*, 416–464. [\[CrossRef\]](#)
5. Costa, B.; Bachiega, J.; De Carvalho, L.R.; Araujo, A.P.F. Orchestration in Fog Computing: A Comprehensive Survey. *ACM Comput. Surv.* **2023**, *55*, 1–34. [\[CrossRef\]](#)
6. Avasalcari, C.; Dustdar, S. Latency-aware decentralized resource management for IoT applications. In *Proceedings of the 8th International Conference on the Internet of Things, Santa Barbara, CA, USA, 15–18 October 2018*; Association for Computing Machinery: New York, NY, USA, 2018; pp. 1–4. [\[CrossRef\]](#)
7. Castellano, G.; Esposito, F.; Risso, F. A Service-Defined Approach for Orchestration of Heterogeneous Applications in Cloud/Edge Platforms. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 1404–1418. [\[CrossRef\]](#)
8. Daoud, W.B.; Obaidat, M.S.; Meddeb-Makhlouf, A.; Zarai, F.; Hsiao, K.F. TACRM: Trust access control and resource management mechanism in fog computing. *Hum.-Centric Comput. Inf. Sci.* **2019**, *9*, 28. [\[CrossRef\]](#)
9. Davoli, G.; Borsatti, D.; Tarchi, D.; Cerroni, W. FORCH: An Orchestrator for Fog Computing service deployment. In *Proceedings of the 2020 IFIP Networking Conference (Networking), Paris, France, 22–26 June 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 677–678.
10. de Brito, M.S.; Hoque, S.; Magedanz, T.; Steinke, R.; Willner, A.; Nehls, D.; Keils, O.; Schreiner, F. A service orchestration architecture for Fog-enabled infrastructures. In *Proceedings of the 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), Valencia, Spain, 8–11 May 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 127–132. [\[CrossRef\]](#)
11. Etemadi, M.; Ghobaei-Arani, M.; Shahidinejad, A. A cost-efficient auto-scaling mechanism for IoT applications in fog computing environment: A deep learning-based approach. *Clust. Comput.* **2021**, *24*, 3277–3292. [\[CrossRef\]](#)
12. Halén, J.; Hellkvist, S.; Baucke, S.; Wuhib, F.; Yazir, Y.O. Wind: Management and Orchestration in the Distributed Heterogeneous Cloud. In *Proceedings of the 2015 IEEE World Congress on Services, New York, NY, USA, 27 June–2 July 2015*; IEEE: Piscataway, NJ, USA, 2015; pp. 39–46, ISSN 2378-3818. [\[CrossRef\]](#)
13. Jiménez, L.L.; Schelén, O. HYDRA: Decentralized Location-Aware Orchestration of Containerized Applications. *IEEE Trans. Cloud Comput.* **2022**, *10*, 2664–2678. [\[CrossRef\]](#)
14. Östberg, P.O.; Byrne, J.; Casari, P.; Eardley, P.; Anta, A.F.; Forsman, J.; Kennedy, J.; Le Duc, T.; Mariño, M.N.; Loomba, R.; et al. Reliable capacity provisioning for distributed cloud/edge/fog computing applications. In *Proceedings of the 2017 European Conference on Networks and Communications (EuCNC), Oulu, Finland, 12–15 June 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6. [\[CrossRef\]](#)
15. Skarlat, O.; Schulte, S.; Borkowski, M.; Leitner, P. Resource Provisioning for IoT Services in the Fog. In *Proceedings of the 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China, 4–6 November 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 32–39. [\[CrossRef\]](#)
16. Nath, S.B.; Chattopadhyay, S.; Karmakar, R.; Addya, S.K.; Chakraborty, S.; Ghosh, S.K. PTC: Pick-Test-Choose to Place Containerized Micro-Services in IoT. In *Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6, ISSN 2576-6813. [\[CrossRef\]](#)
17. Wang, N.; Varghese, B.; Matthaïou, M.; Nikolopoulos, D.S. ENORM: A Framework For Edge NNode Resource Management. *IEEE Trans. Serv. Comput.* **2020**, *13*, 1086–1099. [\[CrossRef\]](#)

18. Buzachis, A.; Galletta, A.; Celesti, A.; Carnevale, L.; Villari, M. Towards Osmotic Computing: A Blue-Green Strategy for the Fast Re-Deployment of Microservices. In *Proceedings of the 2019 IEEE Symposium on Computers and Communications (ISCC), Barcelona, Spain, Barcelona, Spain, 29 June–3 July 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6. [\[CrossRef\]](#)
19. Díaz-de Arcaya, J.; Miñon, R.; Torre-Bastida, A.I. Towards an architecture for big data analytics leveraging edge/fog paradigms. In *Proceedings of the 13th European Conference on Software Architecture—Volume 2, Paris, France, 9–13 September 2019*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 173–176. [\[CrossRef\]](#)
20. Jiang, Y.; Huang, Z.; Tsang, D.H.K. Challenges and Solutions in Fog Computing Orchestration. *IEEE Netw.* **2018**, *32*, 122–129. [\[CrossRef\]](#)
21. Cozzolino, V.; Ott, J.; Ding, A.Y.; Mortier, R. ECCO: Edge-Cloud Chaining and Orchestration Framework for Road Context Assessment. In *Proceedings of the 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI), Sydney, Australia, Sydney, NSW, Australia, 21–24 April 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 223–230. [\[CrossRef\]](#)
22. Jalali, F.; Lynar, T.; Smith, O.J.; Kolluri, R.R.; Hardgrove, C.V.; Waywood, N.; Suits, F. Dynamic Edge Fabric Environment: Seamless and Automatic Switching among Resources at the Edge of IoT Network and Cloud. In *Proceedings of the 2019 IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 8–13 July 2019*; IEEE: Piscataway, NJ, USA, 2019; pp. 77–86. [\[CrossRef\]](#)
23. Mechalik, C.; Taktak, H.; Moussa, F. A Fuzzy Decision Tree Based Tasks Orchestration Algorithm for Edge Computing Environments. In *Advanced Information Networking and Applications*; Barolli, L., Amato, F., Moscato, F., Enokido, T., Takizawa, M., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; Volume 1151, pp. 193–203, Series Title: Advances in Intelligent Systems and Computing. [\[CrossRef\]](#)
24. Papathanail, G.; Fotoglou, I.; Demertzis, C.; Pentelas, A.; Sgouromitis, K.; Papadimitriou, P.; Spatharakis, D.; Dimolitsas, I.; Dechouniotis, D.; Papavassiliou, S. COSMOS: An Orchestration Framework for Smart Computation Offloading in Edge Clouds. In *Proceedings of the NOMS 2020–2020 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 20–24 April 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6. [\[CrossRef\]](#)
25. Kritikos, K.; Plexousakis, D.; Plebani, P. Semantic SLAs for Services with Q-SLA. *Procedia Comput. Sci.* **2016**, *97*, 24–33. [\[CrossRef\]](#)
26. World Wide Web Consortium (W3C). ODRL Information Model 2.2; W3C Recommendation, Retrieved 10 March 2024; World Wide Web Consortium (W3C): Wakefield, MA, USA, 2024.
27. Keller, A.; Ludwig, H. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *J. Netw. Syst. Manag.* **2003**, *11*, 57–81. [\[CrossRef\]](#)
28. Wäldrich, O.; Battre, D.; Brazier, F.; Clark, K.; Oey, M.; Papaspyrou, A.; Wieder, P.; Ziegler, W. WS-Agreement Negotiation Version 1.0. Available online: <https://publica.fraunhofer.de/handle/publica/295514> (accessed on 15 February 2026).
29. Zhang, L.J. (Ed.) *Proceedings of the International Conference on Web Services, ICWS '03: Las Vegas, Nevada, USA, 23–26 June 2003*; CSREA Press: Las Vegas, NV, USA, 2003.
30. Tomic, V.; Pagurek, B.; Patel, K. WSOL—A language for the formal specification of classes of service for web services. In *Proceedings of the International Conference on Web Services, –ICWS'03, Las Vegas, NV, USA, 23–26 June 2003*; Zhang, L., Ed., CSREA Press: Las Vegas, NV, USA, 2003; pp. 375–381.
31. Paschke, A. RBSLA A declarative Rule-based Service Level Agreement Language based on RuleML. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), Vienna, Austria, 28–30 November 2005*; IEEE: Piscataway, NJ, USA, 2005; Volume 2, pp. 308–314. [\[CrossRef\]](#)
32. García, J.M.; Pedrinaci, C.; Resinas, M.; Cardoso, J.; Fernandez, P.; Ruiz-Cortés, A. Linked USDL Agreement: Effectively Sharing Semantic Service Level Agreements on the Web. In *2015 IEEE International Conference on Web Services*; IEEE: Piscataway, NJ, USA, 2015; pp. 137–144. [\[CrossRef\]](#)
33. Correia, A.; Abreu, F.B.e.; Amaral, V. SLALOM: A Language for SLA specification and monitoring. *arXiv* **2011**, arXiv:1109.6740. [\[CrossRef\]](#)
34. Kritikos, K.; Pernici, B.; Plebani, P.; Cappiello, C.; Comuzzi, M.; Benrernou, S.; Brandic, I.; Kertész, A.; Parkin, M.; Carro, M. A survey on service quality description. *ACM Comput. Surv.* **2013**, *46*, 1–58. [\[CrossRef\]](#)
35. Uszok, A.; Bradshaw, J.M.; Jeffers, R.; Johnson, M.; Tate, A.; Dalton, J.; Aitken, S. KAoS Policy Management for Semantic Web Services. *IEEE Intell. Syst.* **2004**, *19*, 32–41. [\[CrossRef\]](#)
36. Kagal, L.; Finin, T.; Joshi, A. A Policy Language for a Pervasive Computing Environment. In *Proceedings of the 4th International Workshop on Policies for Distributed Systems and Networks (POLICY 2003), Lake Como, Italy, 4–6 June 2003*; IEEE: Piscataway, NJ, USA, 2003; pp. 63–74.
37. World Wide Web Consortium (W3C). OWL 2 Web Ontology Language: Overview; W3C Recommendation, Retrieved March 2024. World Wide Web Consortium (W3C): Wakefield, MA, USA, 2004.
38. Paschke, A.; Bichler, M. Knowledge representation concepts for automated SLA management. *Decis. Support Syst.* **2008**, *46*, 187–205. [\[CrossRef\]](#)

39. Iannella, R.; Michael, S.; Stuart, M.; Victor, R.D. ODRL Vocabulary & Expression 2.2. Available online: <https://www.w3.org/TR/odrl-vocab/> (accessed on 15 February 2026).
40. Horrocks, I.; Kutz, O.; Sattler, U. The Even More Irresistible SROIQ. In *Proceedings of the 10th International Conference on Knowledge Representation and Reasoning (KR 2006), Lake District, UK, 2–5 June 2006*; AAAI Press: Cambridge, MA, USA, 2006; pp. 57–67.
41. Horridge, M.; Bechhofer, S. The OWL API: A Java API for OWL Ontologies. *Semant. Web* **2011**, *2*, 11–21. [[CrossRef](#)]
42. Stefanidis, V.; Verginadis, Y.; Patiniotakis, I.; Mentzas, G. Distributed Complex Event Processing in Multiclouds. In *Proceedings of the 7th European Conference on Service-Oriented and Cloud Computing (ESOCC 2018), Como, Italy, 12–14 September 2018*.
43. Mudvari, A.; Jiang, Y.; Tassiulas, L. SplitLLM: Collaborative Inference of LLMs for Model Placement and Throughput Optimization. *arXiv* **2024**, arXiv:2410.10759. [[CrossRef](#)]
44. Lyu, Z.; Xiao, M.; Xu, J.; Skoglund, M.; Renzo, M.D. The Larger the Merrier? Efficient Large AI Model Inference in Wireless Edge Networks. *IEEE J. Sel. Areas Commun.* **2025**, *Early Access*. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.