symmetry

MDPI

Article

A Generic Model for Identifying QoS Parameters Interrelations in Cloud Services Selection Ontology during Runtime

Babu Rajendiran * and Jayashree Kanniappan

Department of CSE, Rajalakshmi Engineering College, Anna University, Chennai 602105, India; jayashree.k@rajalakshmi.edu.in
* Correspondence: babu.r@rajalakshmi.edu.in

Abstract: Nowadays, many business organizations are operating on the cloud environment in order to diminish their operating costs and to select the best service from many cloud providers. The increasing number of Cloud Services available on the market encourages the cloud consumer to be conscious in selecting the most apt Cloud Service Provider that satisfies functionality, as well as QoS parameters. Many disciplines of computer-based applications use standardized ontology to represent information in their fields that indicate the necessity of an ontology-based representation. The proposed generic model can help service consumers to identify QoS parameters interrelations in the cloud services selection ontology during run-time, and for service providers to enhance their business by interpreting the various relations. The ontology has been developed using the intended attributes of QoS from various service providers. A generic model has been developed and it is tested with the developed ontology.

Keywords: cloud services; cloud service provider; quality of service; ontology

Citation: Rajendiran, B.; Kanniappan, J. A Generic Model for Identifying QoS Parameters Interrelations in Cloud Services Selection Ontology during Runtime. *Symmetry* **2021**, *13*, 563. https://doi.org/10.3390/sym13040563

Academic Editor: José Carlos R. Alcantud

Received: 17 February 2021
Accepted: 25 March 2021
Published: 29 March 2021

1. Introduction

Cloud Computing (CC) has become a large technology caterer for infrastructure, platforms, or software as a service. The flexible and scalable pay-per-use model, virtualization of resources, and significant cost reduction makes CC a widely accepted paradigm [1]. There is an enormous number of Cloud Service Providers (CSPs) providing a variety of Cloud Services (CSs), with varied Quality of Service (QoS) attributes, available on the market. The service portrayals look like functional descriptions and treats CS attributes that are independent of one another; this conveys that accurate decisions cannot be made with this minimum descriptive information. Most of the existing Cloud Service Selection (CSS) techniques fail to interrelate and identify interdependencies among various CS attributes, and miss showcasing their close correlation by identifying how one attribute impacts another attribute, as well as the level of impact. This makes the selection of optimal CSs offered by CSPs, that suit the requirements of cloud consumers, among plenty of alternatives available on the market, a challenging task. However, as more and more CSPs are available to users, maximizing profits has become a big challenge for CSPs [2].

To impart technical descriptions of CSs provided by CSPs, a formal method that considers different QoS attributes and their interrelations is required; ontology is used to deal with this problem. Ontology provides semantic information by exploring the meanings of different attributes and identifies relationships between those attributes.

In this paper, we have identified several attributes contributing to the Infrastructure as a Service (IaaS) selection with reference to the Service Measurement Index (SMI) consortium, and have developed a QoSOnto-CSS ontology to recognize relations among the attributes and proposed a generic CSS Onto archetype. The proposed model can be used in all domains where identifying interrelations among attributes is essential; that makes this archetype a generic one that suits all service users and service providers.

The remaining sections of this paper are systematized as follows. Section 2 provides a detailed study on existing methodologies—their merits and concerns to be addressed. Section 3 details the proposed archetype for CSS, whereas Section 4 illustrates the proposed ontology. Implementation and results are discussed in Section 5. The paper is concluded in Section 6 with the scope for possible future enhancement.

2. Related Work

Ontology represents the knowledge about specific domains in a hierarchical manner, in which each level demonstrates the characteristics of a particular concept. Ontology can be used to realize interoperability, to enable interaction among software systems, and to help in human-to-human communication [3,4]. Cloud computing possess two major rulers, namely provider and consumer, imbibed in it. Cloud computing technology literature focuses on resources and services descriptions of cloud, as well as the discovery and selection of cloud services, cloud security and interoperability. Ontologies can be developed for any of these application contexts [5]. Mihaela Oprea developed an ontology for the educational domain that supports the courses, Formal Languages and Compilers, to plot knowledge representation in a much better way. This OntoFormalLanguages-Compilers-1 ontology has been taught to undergraduate computer science students at the Petroleum-Gas University of Ploiesti [6].

Mohammed et al. [7] developed a service selection ontology for PaaS users by analyzing service description documents and Web Service Description Documents (WSDD) of service providers. The developed PaaS ontology does not address QoS attributes. Zhenglan et al. [8] constructed a QoS ontology for CSS based on some of the core attributes, and involved only a small number of QoS attributes. Weights are assigned to different QoS attributes, and services are ranked using an Analytical Hierarchy Process (AHP) approach. The ontology developed by Zhenglan has no provision to dynamically update new service information in the ontology. Samer et al. [9] presented a semantic reasoner and segregated it into domain ontology and relationship ontology. Semantic similarity, along with numerical similarity, dominant similarity, and recessive similarity, is calculated to find the number of matching services for a user request. The proposed semantic reasoner does not consider QoS attributes in its service selection process.

Richard et al. [10] generated semantic rules and a logical reasoning system to collect consistent and correct feedback from users about the usage of services provided by CSPs. The ontology introduced for SaaS by the author does not involve QoS attributes. Richard et al. [11] proposed a requirements ontology that, possibly, identified the necessary user requirements utilized in service searching process, and have provided freedom in developing its own user tasks. This ontological model of Richard also did not consider QoS attributes. Yasmine et al. [12] created a SaaS domain ontology that connects various concepts through an is-a relationship. Feature similarity and request–service hierarchical similarity is computed to provide an efficient service discovery capability to the user. The author has not involved QoS attributes in the decision-making of selecting services. Ontologies have been developed to address faults that occur during the run-time of Web Services (WS) [13].

Ontologies for various applications have been in use nowadays and, henceforth, there is a requirement for generic models to find associations among attributes in ontology. Hence, a generic ontological reference, suitable for all disciplines, is proposed. With this model, a service user can find related associations among attributes pertaining to their application, and it is also useful for the service providers to update their requirements according to the specified QoS.

3. Proposed Generic CSS onto Archetype

Ontology contains a set of concepts in the domain and the relationships between these concepts. It can be applied to information retrieval to deal with user queries [14]. Ontologies and taxonomies of research topics can support a variety of applications, such as

dataset integration, the exploration process in digital libraries, the production of scholarly analytics, and modelling research dynamics [15]. There are various ontologies which exist in the field of computer science, such as computer science ontology [16], gene ontology resources [17], crop ontology [18], ecology [19], eco-informatics [20], etc.

A generic ontological reference model suitable for all disciplines is a challenging issue. Hence, we propose a generic model that is used to retrieve the relationships between the various parameters in the CSS-QoS ontology as a testing environment. The proposed architecture is as shown in Figure 1, and it would be suitable for all kinds of ontologies.

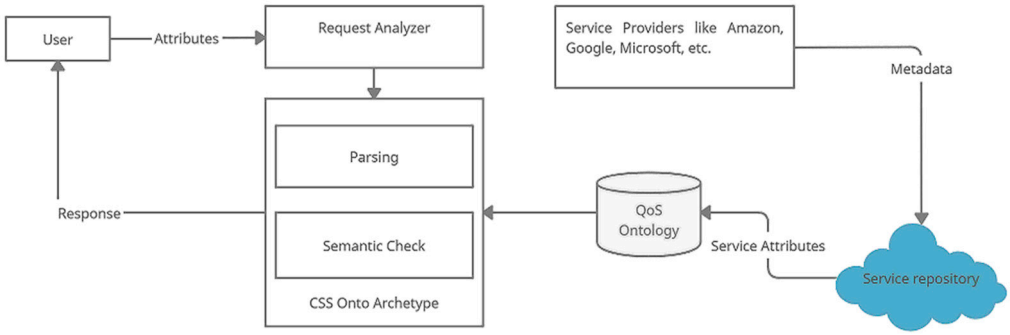

Figure 1. Generic CSS Onto Archetype.

The proposed architecture consists of various components, such as service user, service provider, service repository, CSS-QoS ontology, and the CSS Onto archetype. Service users are the ones who request the service. Service providers are the ones who provide services to the users and, the various CSPs are Amazon, Google, Microsoft, etc. CSPs provides numerous services to the users with several significant attributes, and with a fixed threshold for each of those attributes. The CSPs maintain and manage the metadata of all the services offered by them. The service repository acts as a repository of cloud services where service providers publish their cloud service. The attributes involved in cloud service selection are gathered from different CSPs and various research articles, and the ontology is developed by utilizing those attributes. The developed ontology is offered as a tool for CSPs, which need common vocabulary and semantics to communicate requirements and capabilities.

A cloud user makes a request by providing necessary attributes and the request is forwarded to the request analyzer module. The request analyzer component has been designed to delegate the validation of specified attributes. The verified attributes are given to the proposed generic CSS Onto Archetype for parsing, which results in the exploration of unknown relationships among attributes.

The steps are shown in Figure 2 and it is described below.

Step 1: Get the necessary input attribute from the user based on the relative importance of their business.

Step 2: Import the proposed ontology as an OWL file that consists of superclass–subclass relationships.

Step3: A Resource Description framework (RDF) graph is constructed from the python package mandated for RDF, called RDFLIB, and it stores the ontology diagram as a collection of RDF triples, namely subject, predicate, and object.

Step 4: Store the input attribute received from the cloud user in two fields, namely INPUT_LIST and SEMANTIC_MATCH_LIST.

Step 5: Parse the RDF Graph for identifying and formulating semantically related attributes based on the values in the INPUT_LIST.

Step 6: Find the parent of attributes in the INPUT_LIST using subclass of relation in the generated RDF Graph. If the parent is available procced for the next step else proceed for the step 9.

Step 7: Add the identified parent of INPUT_LIST to the SEMANTIC_MATCH_LIST.

Step 8: Include the identified parent to the INPUT_LIST and procced to Step 5.

Step 9: Inform the cloud user about the various service attributes they receive, in addition to the attributes stated in the service request.

Thus, the CSS Onto Archetype analyses the semantics of each attribute, and appropriate relations are identified. Cloud users are informed about the various service attributes they receive, in addition from the attributes stated in the service request.

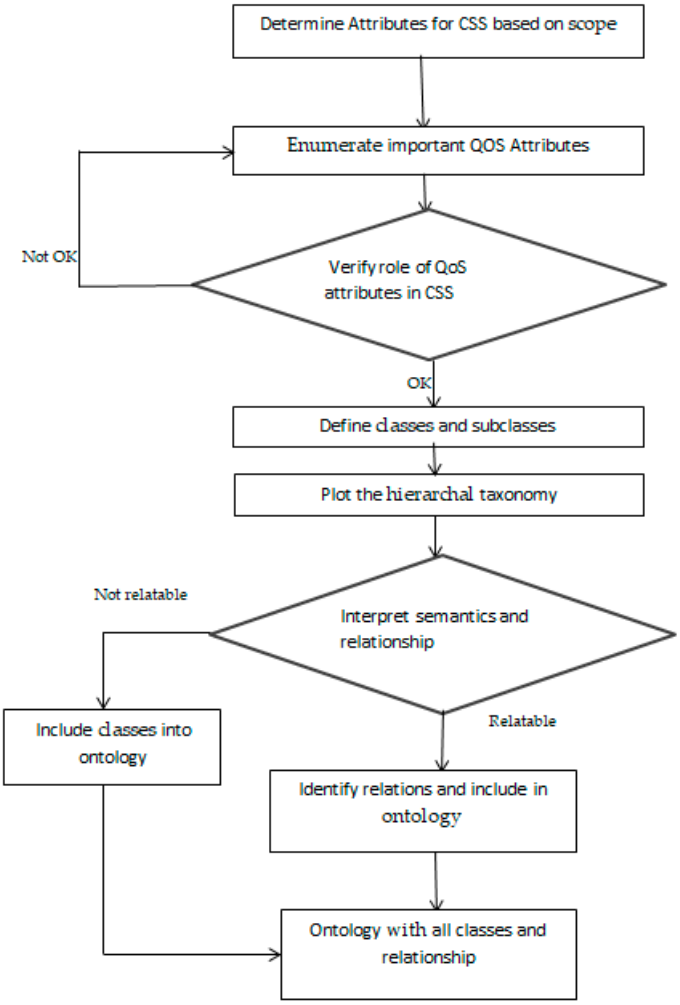

Figure 2. Development process of the QoSOnto-CSS ontology.

4. QoS Parameters Ontology for Cloud Service Selection

The ontology development process of QoSOnto-CSS is detailed in Figure 1. Determining the attributes for CSS and identifying the important QoS attributes contributing to CSS, as well as verifying the significance of QoS in service selection and, based on the inference, adding those QoS attributes as classes and its sub-attributes as subclasses will be added in ontology, is shown below. Then, we have plotted the attributes as hierarchial taxonomy and interpret the relationships between attributes, if any.

To solve service selection problems, we propose a QoSOnto-CSS based on attributes specified by Service Measurement Index (SMI) consortium as reference that can be used to provide most the relevant service to the user, as per their requirements. It also provides useful information to the user about the services and their relations with other QoS parameters. The ontology is created using the protege4.2 editor, which is freely available and platform-independent. Using Web Ontology Language (OWL), it is easy to publish and share ontologies on the World Wide Web. Based on the parameters, an ontology for service selection has been developed and it is shown in Figure 3.

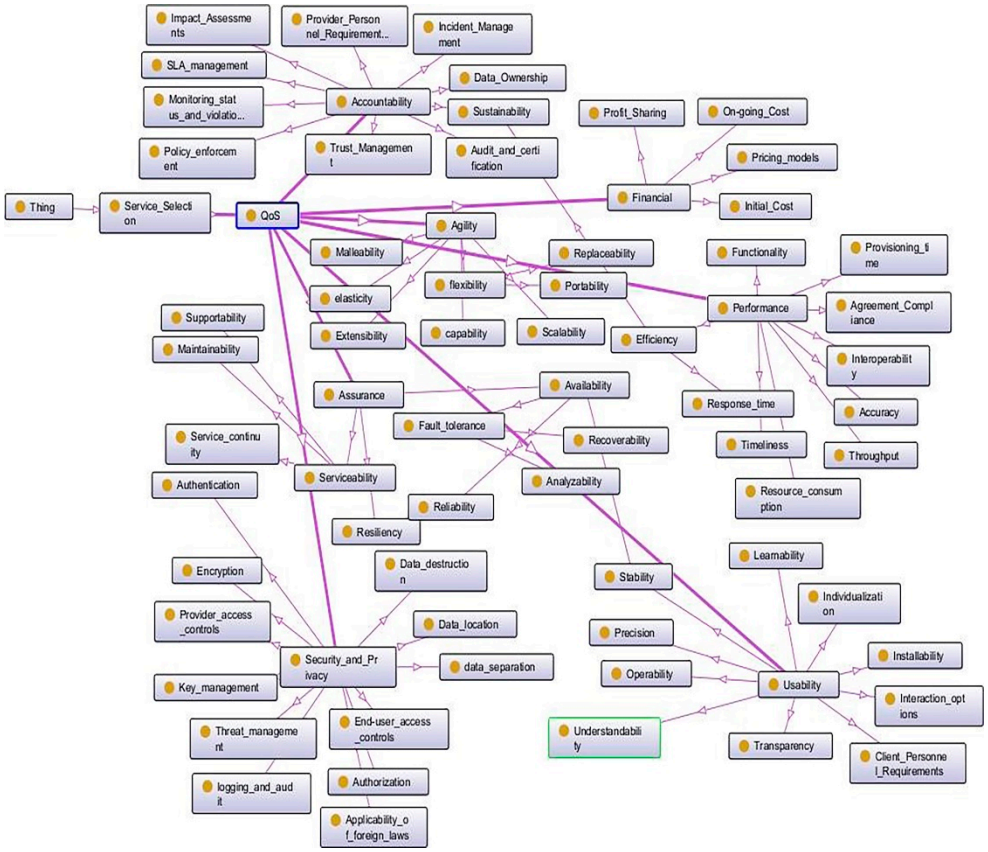

Figure 3. Proposed QoSOnto-CSS.

Functional attributes, such as Service Model, OS Series, OS Distribution, CPU Manufacturer etc., are provided by all CSPs by default. In order to make our approach different from other service selection approaches, we have considered QoS as a main class for our proposed ontology. The various subclasses that are included in QoS are illustrated in Figure 4.

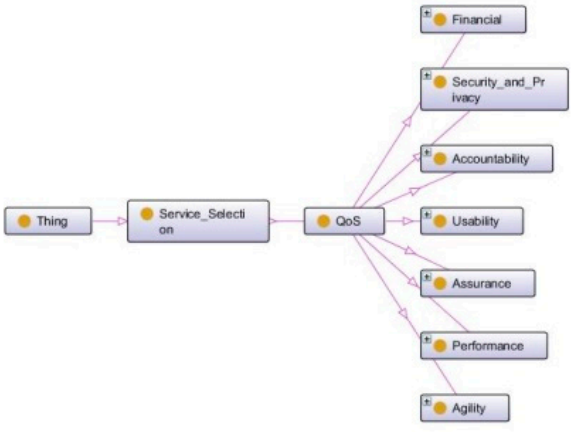

Figure 4. QoS subset in QoSOnto-CSS.

4.1. Accountability

Accountability is considered to provide business reputation for CSPs due to a lack of transparency and less control over data on the market, as well as supplying good data stewardship for CSPs with compliance across geographic boundaries [21]. The accountability subset ontology is shown in Figure 5.

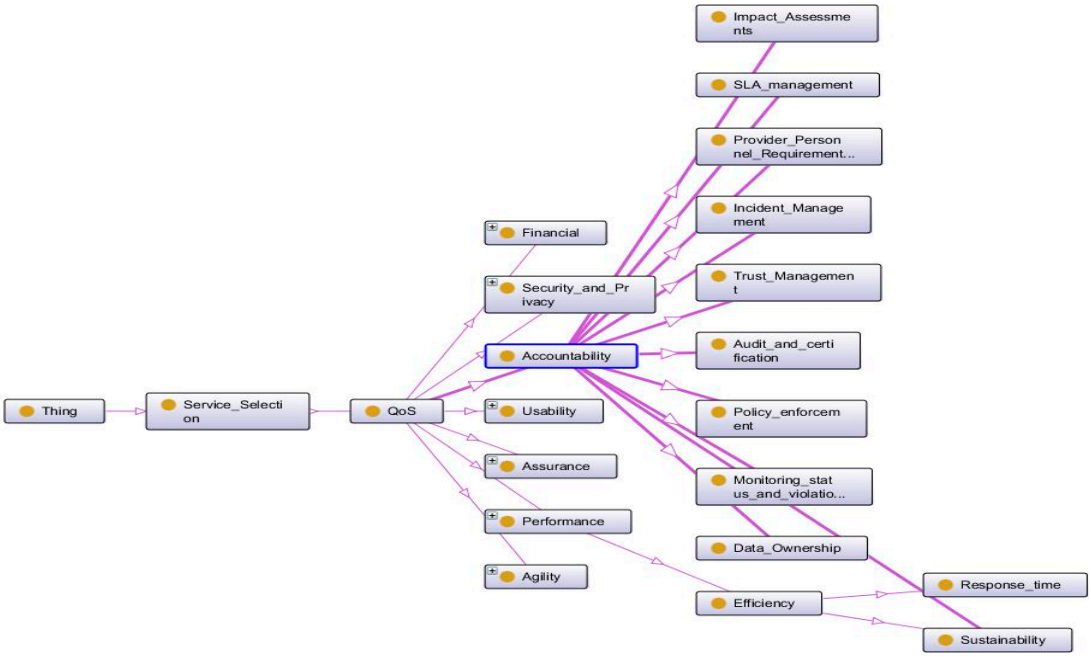

Figure 5. QoSOnto-CSS: Accountability subset.

Accountability includes subclasses, such as impact assessments, SLA management, provider personnel requirements, sustainability, incident management, trust management, audit and certification, policy enforcement, monitoring status and violations, and data ownership. An impact assessment is a means of measuring the possible consequences that an activity may have on the privacy of an individual. SLA management is considered to be important for allocating and managing resources, as well as negation, controlling the service, reporting, and monitoring service levels with high standards. Provider personnel requirements show the extent to which CSP personnel have the skills, education, certifications, and experience required to rightly distribute a service. Sustainability has an impact on the society, the environment and the economy of the CSP by offering resources dynamically and serving multiple business users by a common infrastructure. Sustainability is common to classes such as accountability and a subclass of performance known as efficiency. Incident management has been taken into consideration for identifying any unplanned interruptions to an IT service and for recording the response actions essential for mitigating the incident.

Trust management is another vital component that helps in relational exchanges between ecommerce trading partners in a cloud environment. Audit and certification verify that the requirements follow internal policies, laws and regulations, corporate contracts, or other factors. Policy enforcement is indispensable for providing robust and flexible security and nursing for cloud-based applications and data. Monitoring status and violations parameters are incorporated, as they are important for managing and maintaining software and hardware resources, and for providing uninterrupted information for those resources and consumers deployed applications on the cloud. Data ownership details the actual proprietor of data in the CC and the landscape of data stored, as well as where it was created.

4.2. Agility

Agility is primary for any organization in order to adapt quickly to a continuously growing business environment by integrating new capabilities to address users' growing needs, and to quickly create, test and, introduce software applications that initiates business growth [22]. Elasticity, flexibility, scalability, extensibility, malleability, and capability are the service parameters that comprise agility, and these are shown as subclasses of agility

in Figure 6. Elasticity is the capacity of a system module to quickly adapt to the real-world workload demand by automatically allocating and deallocating resources. Flexibility of a CSP is based on its capability to include or eliminate predefined features from a cloud computing service (CCS). Flexibility includes two subclasses, such as portability and replaceability. Portability indicates the way a service can be shifted from one CSP to another CSP with a minimum level of distraction. Replaceability dictates the possibility of changing from one CSP to another. Scalability is characterized by increasing or decreasing the number of CCS obtainable to meet the SLAs and objectives, as agreed with clients.

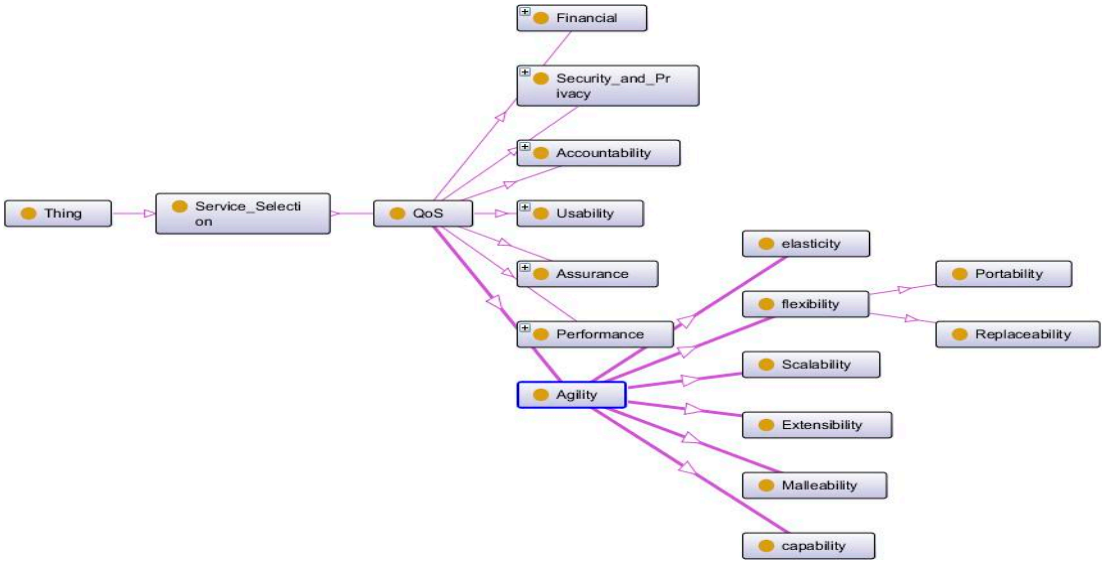

Figure 6. QoSOnto-CSS: Agility subset.

Extensibility concentrates on how to include novel, real-time background support through community build packs. Malleability drives the CSP to adjust to new inclusions by clients in their previously stated requirements. Capability is significant for determining the capacity of a CSP by identifying the level of satisfaction by comparing with the standards.

4.3. Assurance

Assurance is important to avoid misinterpretations in SLAs, security or privacy policies, and standard terms and conditions, which leads to increased adoption of CSs by consumers in their business environments [23]. Assurance incorporates subclasses, such as availability, resiliency, and serviceability. Stability, fault tolerance, and reliability are the subclasses considered for availability. Service continuity, supportability, and maintainability are the subclasses included in serviceability, and these are demonstrated in Figure 7.

Availability signifies the degree to which CCS works without any failure. Stability signifies the importance of predicting interactions among independently developed but interacting CSs. The parameter of stability is common to the class usability and subclass availability of assurance. Fault tolerance is the capability of a CS to work continuously without stalling due to any unknown or unpredictable conditions or situations. Reliability is the measure of consistency provided by the CSP while delivering a service. Resiliency, as a form of failover, rectifies it by distributing redundant implementations of IT resources across physical locations. Serviceability quantifies the efficiency of the CSP in accomplishing maintenance and revising problems with the CCS. Service continuity provides the capability to deliver protection for critical applications and data that help businesses to avoid, prepare for, and recover from, a disruption. Supportability is the level of comfort provided to the users in solving their queries after delivering a CS. Maintainability is essential to ensure that adequate performance is guaranteed, with minimum maintenance costs.

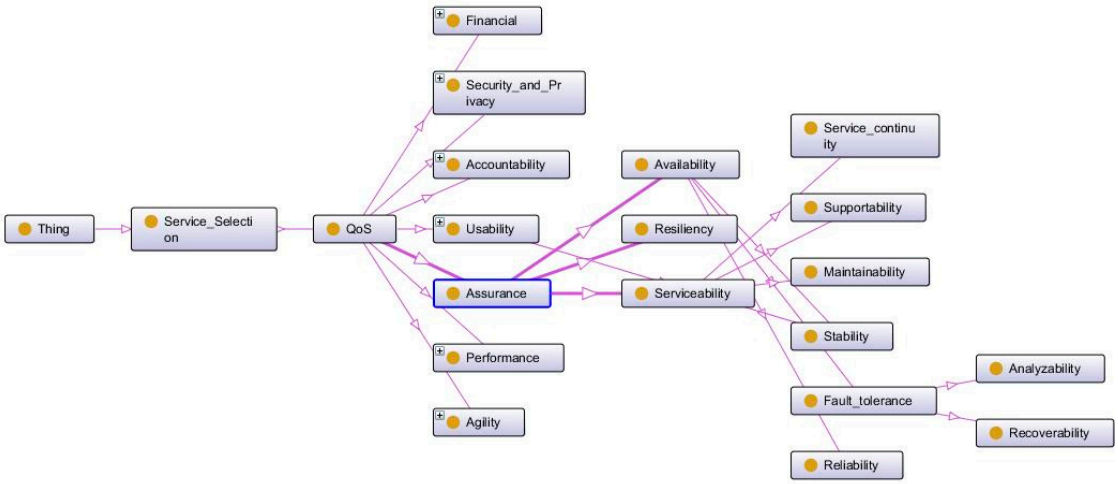

Figure 7. QoSOnto-CSS: Assurance subset.

4.4. Financial

Cost is one of the major factors considered by users when selecting a service, and for providers to deliver contemporary IT solutions [24]. The subclasses that characterize the financial class are on-going cost, pricing models, profit sharing, and initial cost. These are illustrated in Figure 8.

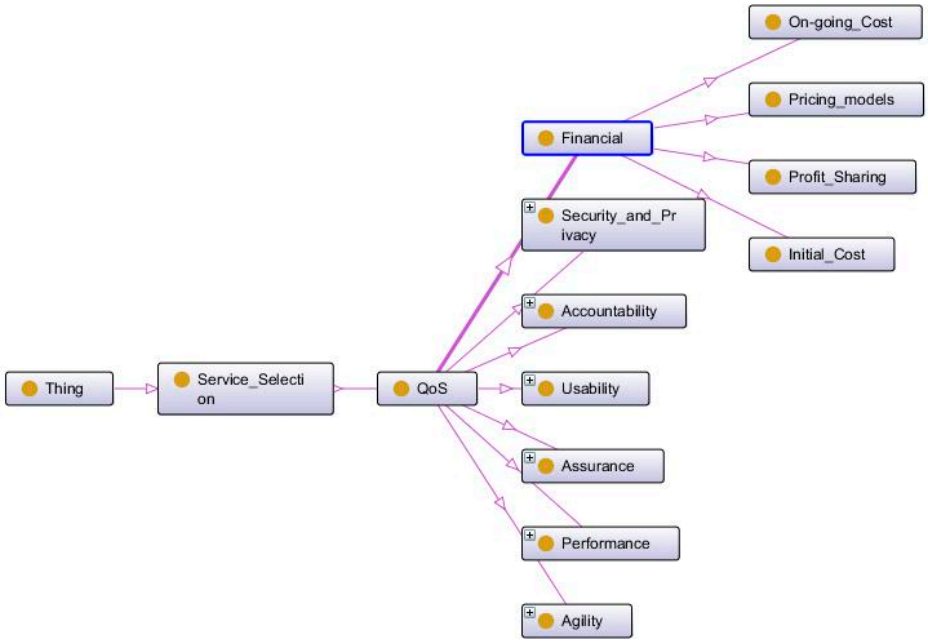

Figure 8. QoSOnto-CSS: Financial subset.

On-going cost is necessary to decide on the cost involved in managing and maintaining activities. The pricing model tends to satisfy both customers and providers by providing different packages to consumers based on their usage pattern. In the case where a CS involves multiple providers, profit sharing allows providers to split their profit based on their contribution towards the service. Initial cost involves the pricing for the data centre and the cost of installation charges, including cooling resources, real estates, electricity costs and network connection etc.

4.5. Performance

Performance indicates the ability of CS resources for carrying out jobs concurrently with extreme parallel processing, and thereby reducing the time involved, and promises more flexibility [25]. The various subclasses that have been included in performance are functionality, provisioning time, timeliness, agreement compliance, resource consumption, interoperability, accuracy, throughput, and efficiency. The subset ontology of performance is shown in Figure 9.

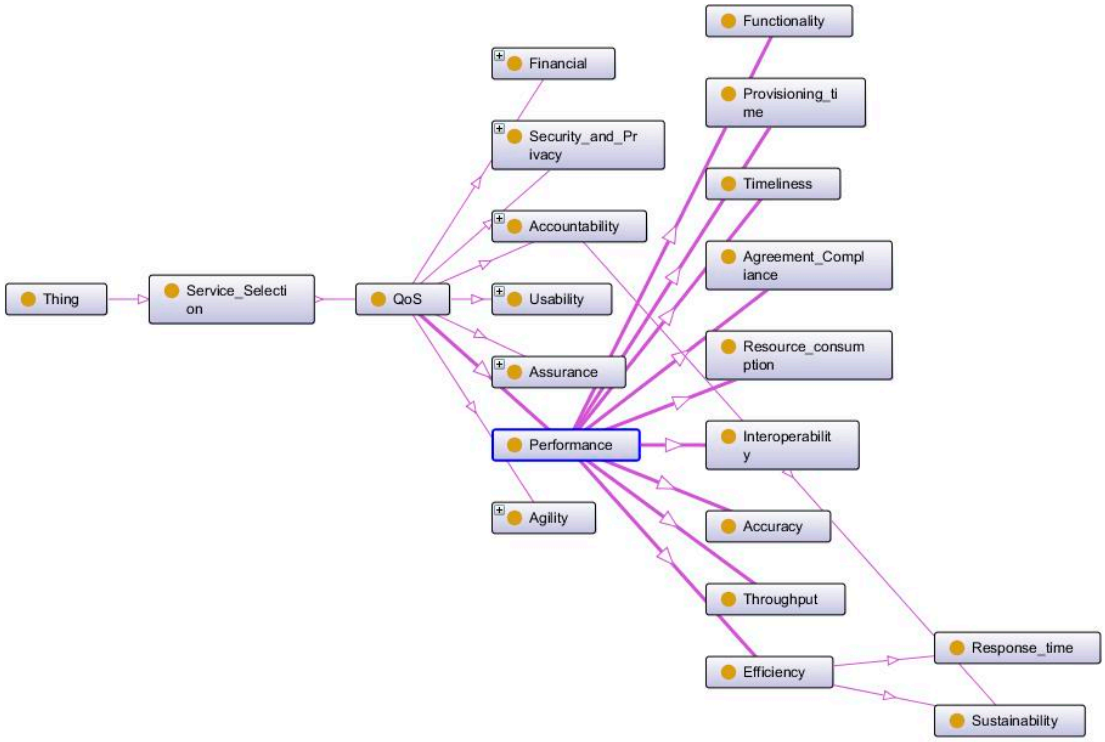

Figure 9. QoSOnto-CSS: Performance subset.

Functionality specifies whether the features provided by the CCS meet clients' needs. Provisioning time refers to the time taken to provide resources to the customer or requesting application in run time. Timeliness guarantees the on-time delivery of a service to its consumers to keep them satisfied. Agreement compliance is necessary to carefully evaluate the degree to which the CSP agrees to the jurisdiction laws and policy mentioned in the SLA. Resource consumption describes the quantity of resources utilized by the CS in delivering a request to the consumer. Interoperability determines the ability of the CCS to easily interact with other services. Accuracy indicates how far the CCS adheres to its requirements. Throughput specifies the number of services successfully supplied by a CSP in a given unit of time. Efficiency measures the amount of energy consumed by the resources involved in the CS delivery.

4.6. Security and Privacy

Security in CC is a group of policies and access controls to be adhered to in order to keep data applications and information safe [26]. Privacy is the capability of an object to control the information that it acknowledges about itself to the service provider. The subclasses contributing to the security and privacy class is shown in Figure 10.

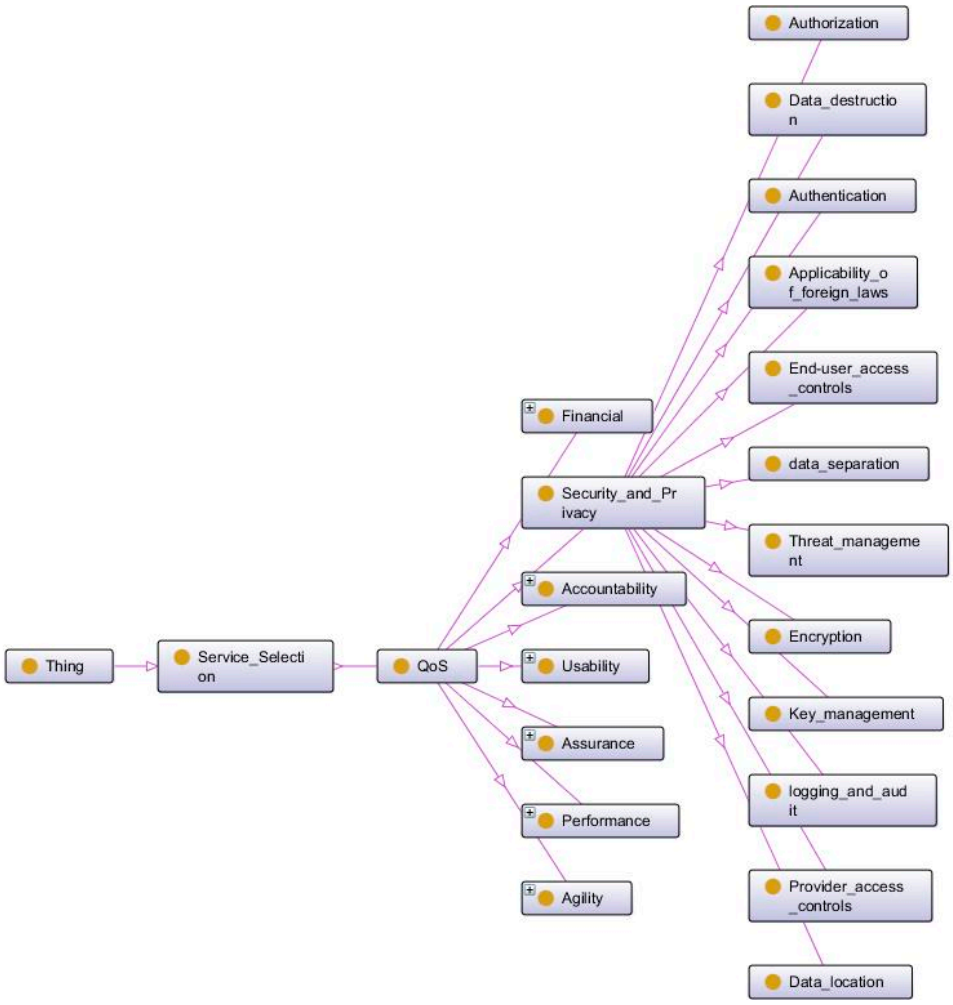

Figure 10. QoSOnto-CSS: Security and privacy subset.

Authorization checks the correctness of data created, stored, and used, which, in turn, offers confidence to the clients that they are using accurate and valid data. Data destruction prompts the CSP to direct clients to use and share data with certain limitations. Authentication is required to validate customers who are accessing the services provided by CSP. Foreign laws are applicable if the service provider accessed by the consumer is in a different country from the user. Access controls ensures the policies and processes in use by the CSP guarantee certain privileges based on personnel and make use of or modify data or work products.

Data separation or segregation is primary in a cloud environment, as data from several users are stored in a shared environment, and one user of a CS can interfere with data of another user. Threat management security solutions are necessary to protect service provider organizations from possible attacks. Encryption necessitates the secure transmission of information. Key management tells consumers about their permissible level to read information. Logging and audits are necessary to keep track the number of clients using their service, as well as the service they are accessing. Provider access controls set by CSPs limit the user in accessing their services without proper authentication. Data location allows clients to select the location of data centers based on geographical or political factors. Physical security of data centers is to protect the infrastructure by various means of safeguarding measures that are independently audited on a need basis.

4.7. Usability

The process of achieving required goals in an effective and efficient manner is coined as usability of a system [27]. To reveal the usability of the cloud and CSs that is available to the end user, the following factors are considered, namely learnability, individualization, client personnel requirements, availability, operability, precision, installability, understandability, transparency, and interaction options. This subset of usability ontology is shown in Figure 11.

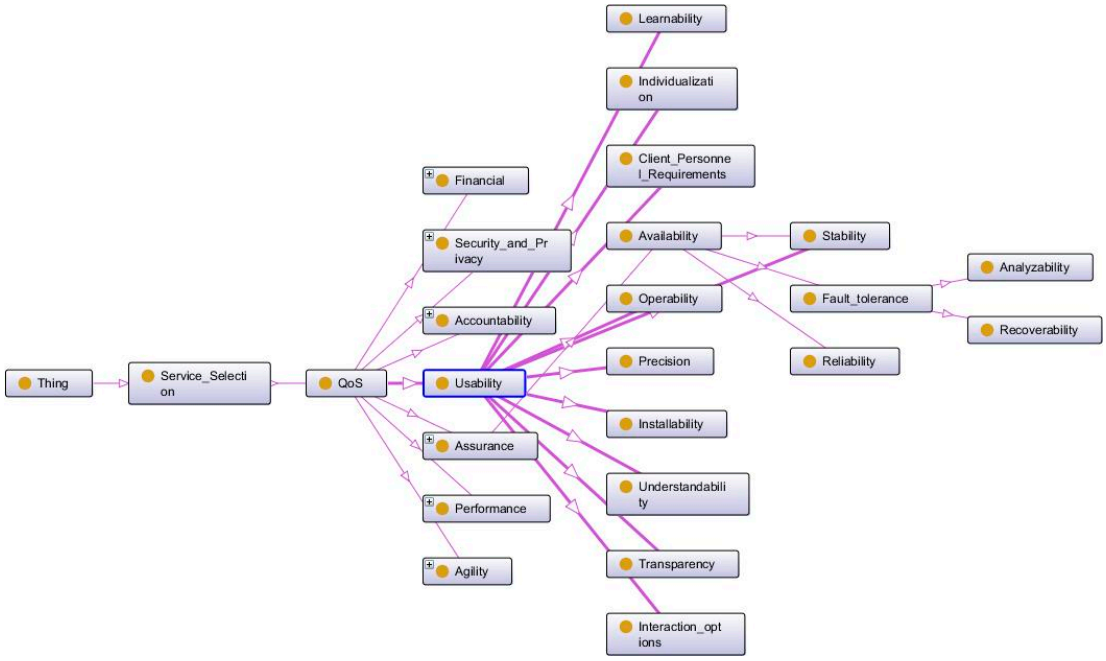

Figure 11. QoSOnto-CSS: Usability subset.

Learnability signifies the QoS and interfaces that allow consumers to quickly become familiar with the features and capabilities of that service. Individualization necessitates that service providers treat consumers as individuals and reward them with relevant offers by making them feel as if they are being remembered, being listened to and felt in control. The client personnel requirement is a crucial aspect to be considered when a service is being created and rendered to consumers for their usage. Availability indicates the amount of time a service is available for use without any failure. Operability dictates the extent to which the CCS is operable by users with disabilities. Precision initiates a detailed assessment of the client's needs, with a focus on the business needs, the business continuity plan, and the application utilization aspects. Installability characterizes the effort, time and privilege required to get the CCS ready for deployment in a client environment. Understandability defines the level of ease for the consumer to understand a CS. Transparency demonstrates the maximum level to which users can determine the changes in a feature or component of the CCS, and the level of impression it has on usability. Interaction options show increases in conversations, create brand advocates, onboard customers, boost lifetime values, and improve customer service. Stability is important to indicate to consumers that they can access a service at any time without any problems.

5. Results and Discussion

The proposed generic model is developed using python programming language that utilizes a third-party web framework library called "flask" and a special package called "rdflib" to work exclusively with RDF.

Simple protocol and RDF query Language (SPARQL) works as a data access query interaction protocol and language that matches graphical patterns against data sources.

The Web Ontology Language (OWL) file is extracted from the ontology of service selection using a Protégé editor that stores ontology relations in a Resource Description Format Schema (RDFS). The RDF data model-based query language, SPARQL 1.1, implementation has been utilized to query the semantics of each attribute and its associated relations. The ontology shown in Figure 2 is converted into an OWL file named as QoSonto.owl. This OWL file is given as input to our application that processes the hierarchies in an RDFS format and uses SPARQL to realize the relationship between attributes, if any.

The processing steps of the Generic CSS Onto Archetype are shown in Figure 12. Based on user's interest in attributes, possible associations are mined from the OWL file extracted from the ontology.

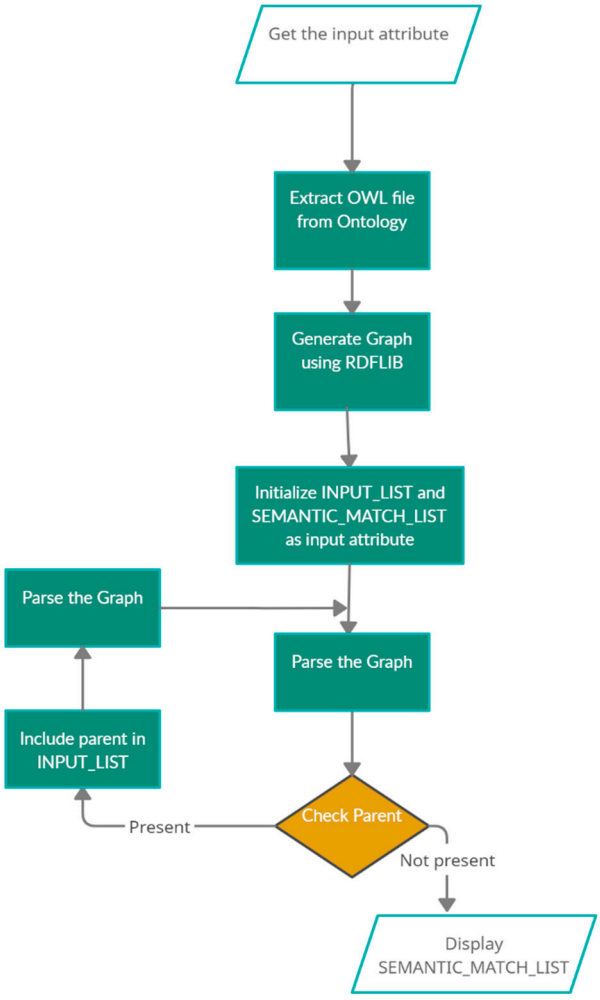

Figure 12. Workflow of generic CSS Onto archetype.

The sample input screenshot is as shown in Figure 13, and requests the user to provide the search criteria. For the given sample requirement as assurance from the user, the subset ontology and its tracing will be identified.

When a user wishes to search for a parameter, the search criteria are sent to the request analyser. The request analyser forwards the same to the parser present in QoS Onto Archetype, which performs necessary parsing to identify subclasses of the given search criteria. After identifying the subclasses, a semantic check is performed to find suitable relationships with other classes in the ontology. As an example, when a user requests a service with attributes in assurance, the QoS Onto model identifies the common attributes present under multiple classes.

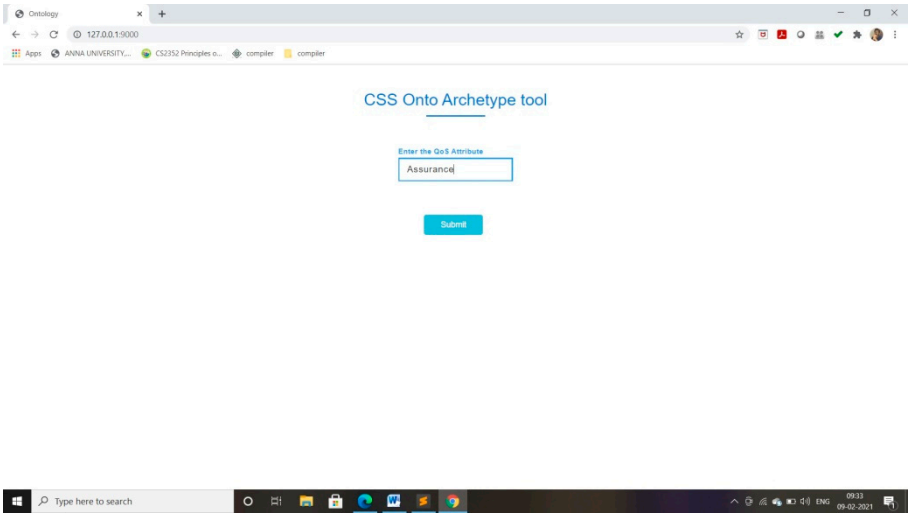

Figure 13. QoS Onto archetype: input requirement form.

Let us consider that one of the input attributes from the user is "assurance", and the user submits the request. The query is passed to the QoS Onto Archetype via SPARQL and a search for all the available subclasses from the RDFS is performed. The retrieved results are shown in Figure 14.

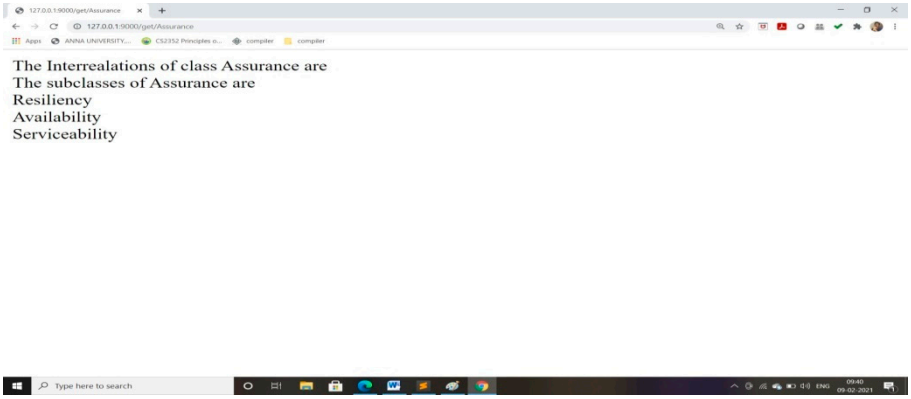

Figure 14. Sample output screenshot.

The service provider can develop an ontology for any application or any field of study using any editor, and can extract the OWL file. Our generic model accepts the OWL file of any type and identifies the relationship between attributes stated in their field of study. Thus, this generic model will be of greater advantage to anyone required to associate relations among attributes.

The proposed generic model has been tested, with fault ontology proposed by authors [13], and this generic model also suits that ontology. The RDF format of the fault ontology was extracted and given as input to the proposed generic CSS Onto archetype; it was observed that the model was able to trace the different types of errors and their correlations. Attribute selection plays a vital role in service selection and, in turn, in the development of an ontological model based on those attributes. Some authors considered fewer QoS attributes for the development of an ontology, as shown in Table 1. The same is illustrated in Figure 15.

Table 1. QoS attributes considered by other authors.

Response Time [8,28–30]
Reliability [8,12,28,30,31]
Security [8,10,12,28,31,32]
Cost [8–10,12,28,30,31,33,34]
Usability [8,28–30]
Location [9,28,33,34]
Availability [9,12,28,30,31]
Adaptability [10,12,28,30]
Scalability [12,28,29,31,32]
Efficiency [28,29]
Accountability [31,32]
Integrity [1,28,32,35]
Interoperability [29,30]
Elasticity [30,36,37]

The QoSOnto-CSS ontology was constructed using 73 parameters. These 73 parameters are classified under 7 broad classes, namely accountability, agility, assurance, financial, performance, privacy and security, and usability; other attributes are plotted as subclasses for these main classes are presented in Table 2. Response time is showcased for these seven main classes, but it also includes all the subclasses in it. This model is efficient when compared to other ontological models for service selection, as it considered a majority of QoS attributes.

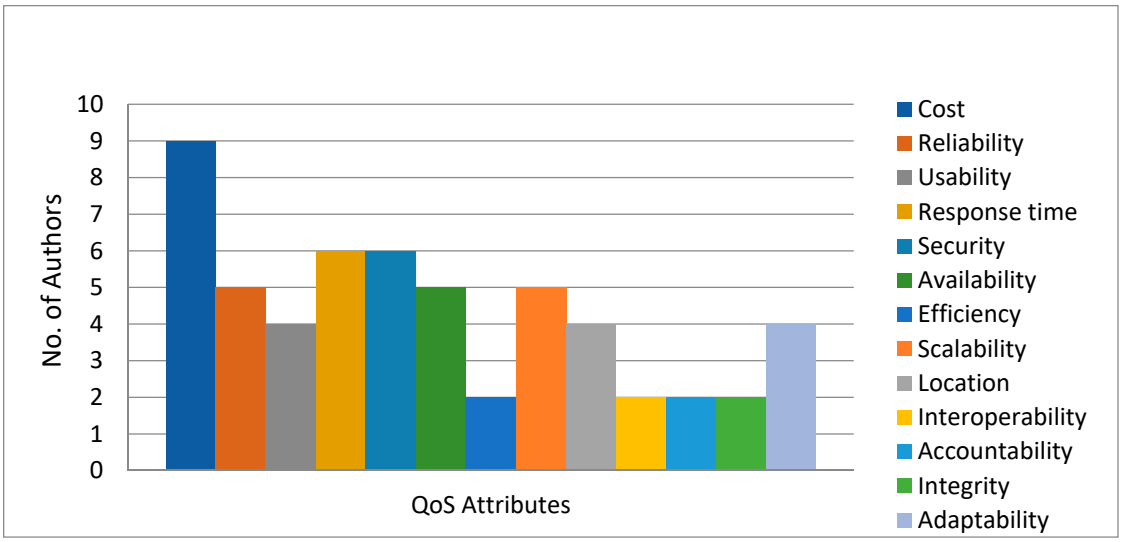

Figure 15. Attributes considered by different authors.

The response time of attributes that have a semantic association is comparatively lower than those attributes that do not have associations. This observation is plotted in Figure 16.

It is observed that the response time of attributes which have interrelations is comparatively higher than those attributes which do not have interrelations. The observations from our proposed ontology, was, it has seven major QoS attributes as classes and its corresponding response time, is shown in Figure 17; it is inferred that an attribute with semantic associations takes more time to execute when compared to attributes without semantic associations.

Table 2. Major QoS attributes considered to develop the ontology.

QoS Attributes	1st Level Attributes	2nd Level Attributes	QoS Attributes	1st Level Attributes	2nd Level Attributes
Accountability	SLA management		Assurance	Availability	Stability
	Incident Management				Fault Tolerance
	Trust Management				Reliability
	Policy enforcement			Resiliency	
	Impact Assessments			Serviceability	Service continuity
	Monitoring status and violations				Supportability
	Audit and certification				Maintainability
	Data Ownership		Financial	Ongoing Cost	
	Provider Personnel Requirements			Pricing models	
	Sustainability			Profit Sharing	
Agility	Elasticity			Initial Cost	
	Flexibility	Portability	Performance	Functionality	
		Replaceability		Provisioning time	
	Scalability			Timeliness	
	Extensibility			Agreement compliance	
	Malleability			Resource Consumption	
	Capability			Interoperability	
Security	Authorization			Accuracy	
	Data destruction			Throughput	
	Authentication			Efficiency	Response time
	Applicability of foreign laws				Sustainability
	End user access controls		Usability	Learnability	
	Data separation			Individualization	
	Threat Management			Client personnel requirements	
	Encryption			Availability	Stability
	Key management				Fault Tolerance
	Logging and audit			Reliability	
	Provider access controls			Operability	
	Data location			Precision	

Though the response time of attributes with interrelationships is comparatively higher than those without relationships, it is preferrable, as the semantic relationship among attributes has been found. The proposed ontology can be enhanced by considering data and object properties, axioms, and individuals for different attributes of services. Such additional inclusions to the ontology help to further refine the relationship among the provided attributes and may result in better output.

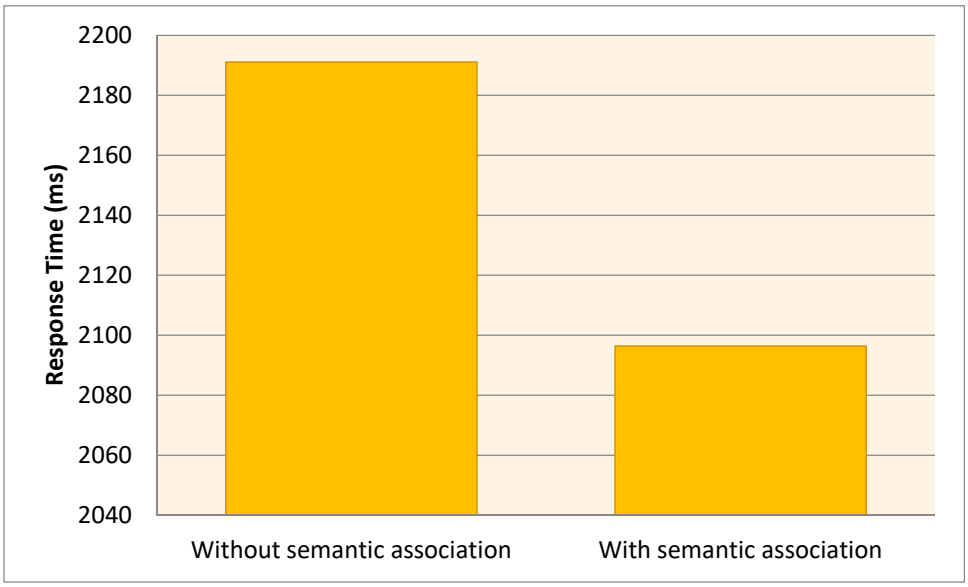

Figure 16. Response time of attributes with and without semantic association.

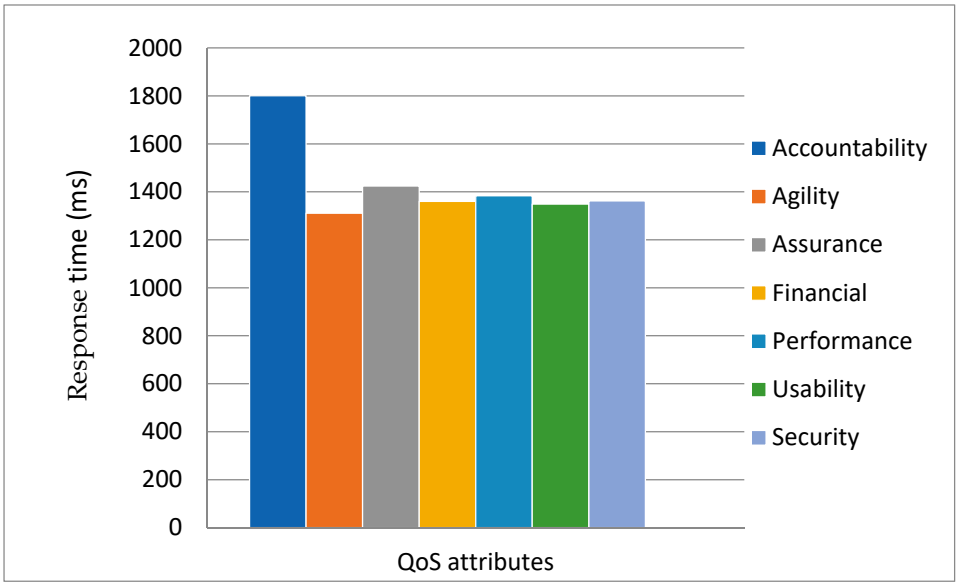

Figure 17. Response time of different QoS attributes.

6. Conclusions

Cloud Service Selection needs to be regularized as it guarantees services requested by users. To select the service based on mandatory functional and optional QoS attributes, an ontology is developed to provide semantic reasoning among attributes of different kinds. The proposed QoSOnto-CSS ontology developed has been written in a standard ontology language called OWL, which shows its generic capability. Thus, the proposed generic QoS Onto Archetype can be used to find semantic relationships between attributes present in any ontology. It also provides an abstract perspective with which it is possible to infer knowledge, follow common objectives, and interoperate for service providers and service users. The proposed model has been tested using an application with sample inputs and necessary interrelations derived as output shown. For the future work, we plan to extend the generic model to have more possible associations among the various attributes.

Author Contributions: B.R. and J.K. contributed equally to this article. Both authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declares no conflict of interest.

References

1. Majid, A.; Ali, E.; Fahimeh, R.; Farookh, H.K. Efficiency measurement of cloud service providers using network data envelopment analysis. *IEEE Trans. Cloud Comput.* **2019**, *1*. [CrossRef]
2. Al-Sayed, M.M.; Hassan, H.A.; Omara, F.A. CloudFNF: An ontology structure for functional and non-functional features of cloud services. *J. Parallel Distrib. Comput.* **2020**, *141*, 143–173. [CrossRef]
3. Uschold, M.; Jasper, R. A Framework for Understanding and Classifying Ontology Applications. In Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden, 2 August 1999.
4. Maedche, A.; Staab, S. Measuring Similarity between Ontologies. In *Knowledge Engineering and Knowledge Management*; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2473, pp. 15–21.
5. Darko, A.; Neven, V.; Jurica, S. Cloud Computing Ontologies: A Systematic Review. In Proceedings of the Third International Conference on Models and Ontology-Based Design of Protocols, Architectures and Services MOPAS 2012, Chamonix/Mont Blanc, France, 29 April—4 May 2012; pp. 9–14.
6. Oprea, M. An Educational Ontology for Formal Languages and Compilers. In Proceedings of the 15th International Conference on Virtual Learning, University of Bucharest, Romania, Europe, 31 October 2020; pp. 54–60.
7. Galety, M.G.; SaravanaBalaji, B.; SaleemBasha, M.S. OSSR-P: Ontological Service Searching and Ranking System for PaaS Services. *Int. J. Adv. Trends Comput. Sci. Eng.* **2019**, *8*, 271–276. [CrossRef]
8. Xie, Z.; Yin, H. Selection of optimal cloud services based on quality of service ontology. *Ing. Syst. Inf.* **2018**, *23*, 127–141. [CrossRef]
9. Hasan, S.; Kumari, V.V. Generic-distributed framework for cloud services marketplace based on unified ontology. *J. Adv. Res.* **2017**, *8*, 569–576. [CrossRef] [PubMed]
10. Otuka, R.I.; Tawil, A.-R.; Al-Nemrat, A. Cloudysme: An Ontological Framework for Aiding SMEs Adoption of SaaS in a Cloud Environment. *J. Comput. Commun.* **2017**, *5*, 86–112. [CrossRef]
11. Greenwell, R.; Liu, X.; Chalmers, K.; Pahl, C. A Task Orientated Requirements Ontology for Cloud Computing Services. In Proceedings of the 6th International Conference on Cloud Computing and Services Science, Rome, Italy, 23–25 April 2016; SCITEPRESS—Science and Technology Publications: Rome, Italy, 2016; pp. 121–128.
12. Yasmine, M.A.; Ibrahim, F.M.; Nagwa, L.B.; Tolba, M.F. Ontology-Based SAAS Catalogue for Cloud Services Publication and Discovery. *Asian J. Inf. Technol.* **2016**, *15*, 4900–4915.
13. Jayashree, K.; Anand, S.; Chithambaramani, R. A Fault Ontology for Managing Run-Time Faults in Web Services. *Asian J. Inf. Technol.* **2013**, *12*, 60–69.
14. Reshma, V.K.; Saravana, B. Cloud Service Publication and Discovery Using Ontology. *Int. J. Sci. Eng. Res.* **2012**, *3*, 1–4.
15. Osborne, F.; Salatino, A.; Birukou, A.; Motta, E. Automatic Classification of Springer Nature Proceedings with Smart Topic Miner. In Proceedings of the International Semantic Web Conference, Kobe, Japan, 17–21 October 2016; pp. 383–399.
16. Salatino, A.A.; Thanapalasingam, T.; Mannocci, A.; Osborne, F.; Motta, E. The Computer Science Ontology: A Large-Scale Taxonomy of Research Areas. In *International Semantic Web Conference*; Metzler, J.B., Ed.; Springer: Cham, Switzerland, 2018; pp. 187–205.
17. Consortium, T.G. The Gene Ontology Resource: 20 Years and Still Going Strong. *Nucleic Acids Res.* **2019**, *47*, D330–D338.
18. Matteis, L.; Chibon, P.Y.; Espinosa, H.; Skofic, M.; Finkers, H.J.; Bruskiewich, R.; Hyman, J.M.; Arnoud, E. Crop Ontology: Vocabulary For Crop-related Concepts. In Proceedings of the First International Workshop on Semantics for Biodiversity, Montpellier, France, 27 May 2013.
19. Madin, J.; Bowers, S.; Schildhauer, M.; Krivov, S.; Pennington, D.; Villa, F. An ontology for describing and synthesizing ecological observation data. *Ecol. Inform.* **2007**, *2*, 279–296. [CrossRef]
20. Williams, R.; Martinez, N.; Golbeck, J. Ontologies for Ecoinformatics. *J. Web Semant. First Look* **2006**, *4*, 237–242. [CrossRef]
21. Zou, J. Accountability in Cloud Services. Ph.D. Thesis, Department of Computing, MACQUARIE University, Sydney, Australia, 1 November 2016.
22. Fang, D.; Liu, X.; Romdhani, I.; Jamshidi, P.; Pahl, C. An agility-oriented and fuzziness-embedded semantic model for collaborative cloud service search, retrieval and recommendation. *Future Gener. Comput. Syst.* **2016**, *56*, 11–26. [CrossRef]
23. Dash, S.B.; Saini, H.; Panda, T.C.; Mishra, A. Service Level Agreement Assurance in Cloud Computing: A Trust Issue. *Int. J. Comput. Sci. Inf. Technol.* **2014**, *5*, 2899–2906.

24. Weintraub, E.; Cohen, Y. Cost Optimization of Cloud Computing Services in a Networked Environment. *Int. J. Adv. Comput. Sci. Appl.* **2015**, *6*, 148–157. [CrossRef]
25. Jatoth, C.; Gangadharan, G.R.; Fiore, U.; Buyya, R. SELCLOUD: A hybrid multi-criteria decision-making model for selection of cloud services. *Soft Comput.* **2019**, *23*, 4701–4715. [CrossRef]
26. Sun, Y.; Zhang, J.; Xiong, Y.; Zhu, G. Data Security and Privacy in Cloud Computing. *Int. J. Distrib. Sens. Netw.* **2014**, *10*, 1–9. [CrossRef]
27. Aniobi, D.E.; Alu, E.S. Usability Testing and Evaluation of a Cloud Computing based Mobile Learning App: Students Perspective. *Int. J. Comput. Sci. Issues* **2019**, *13*, 67–75.
28. Rekik, M.; Boukadi, K.; Ben-Abdallah, H. Cloud Description Ontology for Service Discovery and Selection. In Proceedings of the 10th International Conference on Software Engineering and Applications, Firenze, Italy, 18–19 May 2015; SCITEPRESS—Science and Technology Publications: Colamr, France, 2015; pp. 26–36.
29. Bassiliades, N.; Symeonidis, M.; Gouvas, P.; Kontopoulos, E.; Meditskos, G.; Vlahavas, I. PaaSport semantic model: An ontology for a platform-as-a-service semantically interoperable marketplace. *Data Knowl. Eng.* **2018**, *113*, 81–115. [CrossRef]
30. Garg, S.K.; Versteeg, S.; Buyya, R. SMICloud: A Framework for Comparing and Ranking Cloud Services. In Proceedings of the Fourth IEEE International Conference on Utility and Cloud Computing, Melbourne, VIC, Australia, 5–8 December 2011; pp. 210–218.
31. Moscato, F.; Aversa, R.; Di Martino, B.; Fortiş, T.; Munteanu, V. An analysis of mOSAIC ontology for Cloud resources annotation. In Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS), Szczecin, Poland, 18–21 September 2011; pp. 973–980.
32. Jayapraksh, S.; Aramudhan, M. Classical Probability Ranking Principle based Provider Selection in Federated Cloud. *Indian J. Sci. Technol.* **2016**, *9*, 1–5. [CrossRef]
33. Nepal, S.; Zhang, M.; Ranjan, R.; Haller, A.; Georgakopoulos, D. An Ontology-based System for Cloud Infrastructure Services' Discovery. In Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, Pittsburgh, PA, USA, 14–17 October 2012; pp. 524–530.
34. Ali, A.; Shamsuddin, S.M.; Eassa, F.E. Ontology-based Cloud Services Representation. *Res. J. Appl. Sci. Eng. Technol.* **2014**, *8*, 83–94. [CrossRef]
35. Goeke, L.; Mohammadi, N.G.; Heisel, M. Context Analysis of Cloud Computing Systems Using a Pattern-Based Approach. *Future Internet* **2018**, *10*, 72. [CrossRef]
36. Nagarajan, R.; AVC College of Engineering; Thirunavukarasu, R.; Shanmugam, S. VIT University A Cloud Broker Framework for Infrastructure Service Discovery Using Semantic Network. *Int. J. Intell. Eng. Syst.* **2018**, *11*, 11–19. [CrossRef]
37. Niha, K.; Aisha Banu, W.; Anette, R. A cloud service providers ranking system using ontology. *Int. J. Sci. Eng. Res.* **2015**, *6*, 41–45.