symmetry

MDPI

Review

Addressing Semantics Standards for Cloud Portability and Interoperability in Multi Cloud Environment

Chithambaramani Ramalingam [1,*] and Prakash Mohan [2]

[1] Department of CSE, T.J.S Engineering College, Anna University, Thiruvallur 601206, Tamil Nadu, India
[2] Department of CSE, Karpagam College of Engineering, Coimbatore 641032, Tamil Nadu, India; prakashmohan@kce.ac.in
* Correspondence: chithambaramani.r@tjsec.in

Abstract: The increasing demand for cloud computing has shifted business toward a huge demand for cloud services, which offer platform, software, and infrastructure for the day-to-day use of cloud consumers. Numerous new cloud service providers have been introduced to the market with unique features that assist service developers collaborate and migrate services among multiple cloud service providers to address the varying requirements of cloud consumers. Many interfaces and proprietary application programming interfaces (API) are available for migration and collaboration services among cloud providers, but lack standardization efforts. The target of the research work was to summarize the issues involved in semantic cloud portability and interoperability in the multi-cloud environment and define the standardization effort imminently needed for migrating and collaborating services in the multi-cloud environment.

Keywords: platform as a service (PaaS); infrastructure as a service (IaaS); software as a service (SaaS); cloud interoperability and portability

Citation: Ramalingam, C.; Mohan, P. Addressing Semantics Standards for Cloud Portability and Interoperability in Multi Cloud Environment. *Symmetry* **2021**, *13*, 317. https://doi.org/10.3390/sym13020317

Academic Editors: Michel Planat and Tomohiro Inagaki

Received: 27 December 2020
Accepted: 8 February 2021
Published: 14 February 2021

1. Introduction

Cloud computing technology has been seriously accepted by organizations and numerous companies because of its widespread technology [1]. It generally aims to provide a computing paradigm. The objective of cloud computing is to distribute or extend access to a scalable infrastructure of software or virtualized hardware on the Internet [2–4].

Cloud computing is defined, according to the National Institute of Standards and Technology (NIST) [5], as a model for convenient, ubiquitous, on-demand network access to a shared pool of configurable computing resources such as servers, storage, application networks, and services that can be provisioned and released with minimal management effort or service provider interaction. The purpose of the cloud and the nature of the cloud location are defined by deployment models. Public cloud, private cloud, community cloud, and hybrid cloud are the four models of deployment. The platform as a service (PaaS), infrastructure as a service (IaaS), and software as a service (SaaS) are three kinds of cloud computing service models offered by the NIST cloud computing architecture [6] and can be generated. The major actors of NIST cloud reference architecture include the cloud consumer, provider, and broker. An organization or an individual can be a cloud consumer concerned with a business connection with a cloud provider and consumes their services. Monitoring and observing the usage such as the process, delivery of cloud services, managing, and handling the connection between cloud consumers and providers are the responsibility of the cloud broker [7]. An individual or organization offering services to clients is defined as the cloud provider.

A standout amongst the most critical boundaries of distributed computing reception is interoperability and conveyability at the platform as a service (PaaS) layer. Cloud interoperability is the measure to access the cloud resources of any cloud provider, and the cloud provider may interact with the resources of other cloud providers and insinuate a

circumstance in which various interfacing segments can trade specific data and execute, depending on the data on a lot of agreed useful semantics [8].

Portability is the capacity of exchanging workloads and information between cloud providers [9] when applications cannot be ported crosswise over heterogeneous PaaS stages, where the provider lock-in is probably going to happen. Vendor lock-in may happen at the application level, in which cases relocating an application require application adjustments at the information level, whereby the information cannot be exchanged on the grounds that each cloud supplier has its own information design.

NIST characterizes platform as a service to manage cloud infrastructure. After the provider initializes the hardware resources, they permit the client to create, run, and oversee web applications by making use of programming languages, libraries, administrations, and devices, ordinarily bolstered by the PaaS provider. Some of the PaaS providers are IBM Bluemix, Redhat OpenShift, Pivotal Cloud, Cloud Foundry, Microsoft Azure, and Google App Engine. To send and oversee requests on the PaaS cloud, each PaaS provider utilizes its own exclusive API. There is no standard API to have or oversee requests between various PaaS providers, which has prompted an absence of institutionalization endeavors.

As per Hoff [10], there are three different ways to investigate the interoperability and portability of the cloud provider by utilizing cloud broker, semantics, and proprietary API. In the accompanying segment, we present different methodologies to achieve the interoperability and portability by utilizing hardware resources, the need of semantics, and utilizing exclusive API is discussed in the next section. Finally, the study highlights the standardization efforts needed for generic API in a multi-cloud environment through service semantics, which highlights the future research ventures of this work.

2. Service Brokering Framework

The European Commission [11] has launched the Multidisciplinary drifting Observatory for the Study of Arctic Climate (MOSAIC) [12] project. The main purpose of this project is to build a connection between cloud services and their applications as well as link them with each other in an open source platform, with the belief that in future, this can drive out more competition between the cloud providers. This platform will authorize applications to obtain the service needs of the user's requirements, which will lead to the proposed requirements or modifications to the platform by a constant API. Then, the platform dispatches the information of the services that are compatible with the users' demands and requirements. Two critical sections inclusive of the resource broker and application executor are applied along with the multi-agent brokering mechanism [13]. The tasks in resource broking are booking and resource negotiation.

Cloud agency and client interface are the two sub-systems forming the resource broker. Justifying application specifications and creating service-level agreements (SLAs) are the two main tasks of the cloud agency. SLAs are created using tools such as a monitor, service registry, a service mediator, client semantic engine, a negotiator, quality of service (QoS) limits, and cloud ontology. An illustration of the service broker and interaction between the cloud provider and consumer are represented in Figure 1 with the SLA, service monitoring framework, QoS, semantic engine, and cloud ontology. The activity of the utilization agent [14] is to run applications in concurrence with an exceptional programming interface execution engine helps clients to gain API, which endows the clients to move on to the virtual group and physical assets [15]. Virtual resources are measured services that enable the provider to lease the resource need by the service client through an explicit connector whose action is to make a steady interface with the cloud resource, which are explicitly specified in the service level agreement (SLA) [16]. In the long run, the resource administrator's activity is to monitor and ensure the availability of cloud resources. It executes the elements of the SLA, monitor the cloud resource, and handles solicitations of the elasticity need of cloud resources.

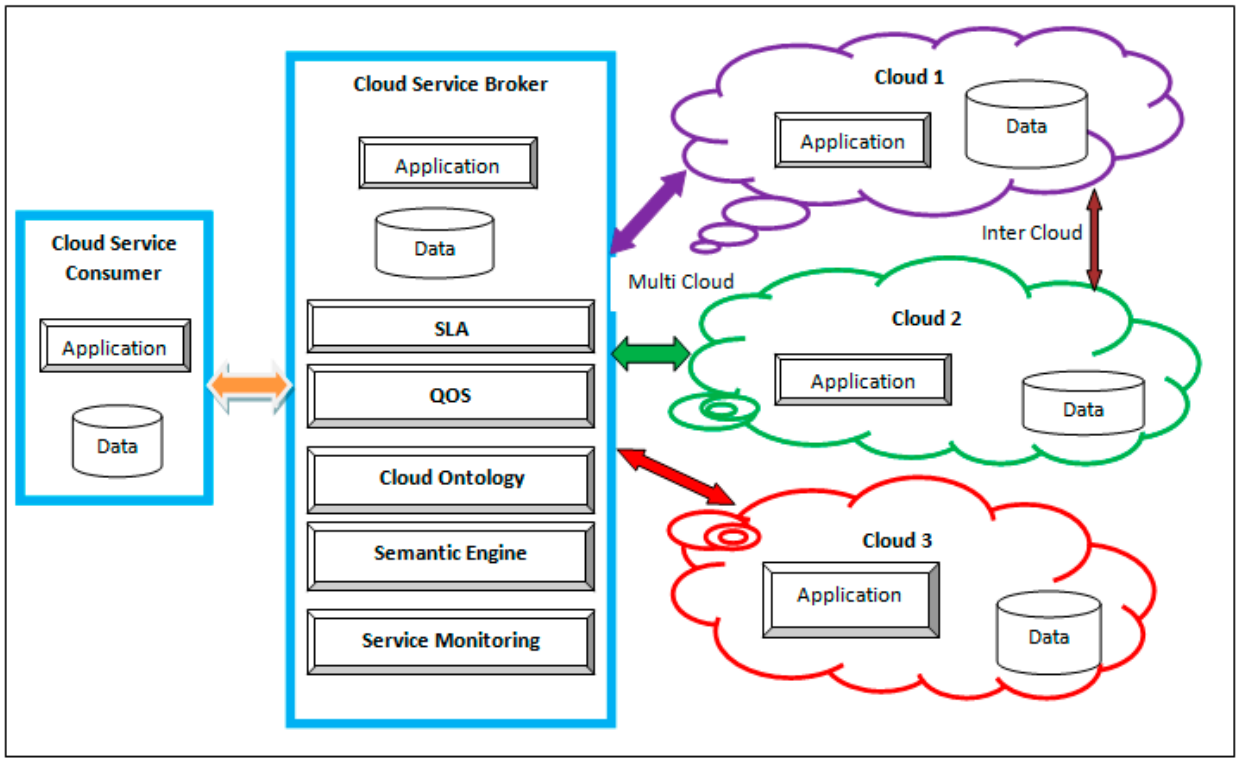

Figure 1. Service brokering framework.

Semantic business processes based on software as a service and cloud computing (SITIO) [17] centers around the making of a framework that will guarantee the entrance of intermediaries to the business through cloud computing services with a base expense. Analyzing this, a stage will be produced that can accomplish solid, secure, and gainful interoperability between unknown applications. Four essential components are found by SITIO: the User Interface (UI), process administrations, business administrations, and meta-data administrations. The UI permits distinctive clients to get to the SITIO stage and to utilize and deal with their records and applications. Programming designers and infrastructure providers check the interface to deploy their applications; finally, clients can find the cloud server and buy the necessary cloud resources to deploy the application. The procedure and business administration layers provide the capacity to run discretionary, web benefits in the distributed computing stage. At this end, appropriation, stack adjusting, and information perseverance administrations are added to a given application in a straightforward manner. In any case, to guarantee this usefulness, a few limitations on the administrations that keep running over the stage must be forced (e.g., determining the programming dialect where the application ought to be based on the dispersion of an application). The annotators of the administrations are the web benefit designers since they completely appreciate the techniques for the administrations created [18].

A semantic cloud broker framework for the automated composition of cloud services by representing the service description and discovery process can be undertaken through the ontology to enable proper service discovery at runtime [19]. The paper concentrated on the discovery of cloud service at runtime through a cloud broker, but provided only an abstract view of this approach. The concrete part such as service negotiation and service composition are planned as future work. The author addresses the IaaS layer of cloud stack and demonstrates this using the Amazon EC2 object and data properties. The extended work will concentrate on PaaS service description and discovery.

CloudSME Simulation Platform is an architecture proposed by Taylor, S.J. et al. [20] that uses a cloud broker platform to enable the capabilities of a multi cloud environment and combines with workflow development and the management of Web Service open-

source science gateway framework WS-PGRADE/gUSE. Different cloud provider resources are commercialized using the PaaS API of the cloud broker platform. Cloud brokers include IaaS providers such OpenStack, OpenNebula, Amazon, CloudSigma, and various other High Performance Computing HPC [21] resources. Cloud Small and Medium Enterprises SMEs address interoperability and portability at the infrastructure level by representing the physical layer of the cloud environment through resource managers, resource abstractions, and physical resources in Cloud-Ontology. The semantic engine of Cloud-Ontology helps in decision-making processes for the effective management of physical resources.

Bassiliades et al. [22] proposed an PaaSport Cloud broker or marketplace for representing semantic interoperable PaaSport ontology, which in turn represents the characteristics, attributes, and capabilities of the PaaS service, semantic information about the cloud platform in which the application is to be deployed, and SLAs between the cloud service provider and application developer (cloud consumer). Responsibilities of the cloud broker are to match the requirements of the cloud consumer with a provider through a semantic matchmaking engine and provide the best matched service to cloud consumers through the ranking algorithm. All semantic information are accessed through SPARQL queries from semantic repositories.

Quadir et al. [23] proposed an approach based on the consumer requirements where the cloud services can be chosen through a central registry, which acts as the database to store all the information of the existing cloud services. Semantic technology or semantic service representation is not used in service discovery, but Web Service Description Language (WSDL) and Resource Description Framework (RDF) are used for service description with input machine learning techniques applied for implementing the service discovery process by the service broker.

Petcu et al. [24] proposed a semantic framework for the brokering of service by matching the service requirement with the semantics for enabling the interoperability of the cloud service. Semantics can also be used to port the application by aligning and reconciling different APIs and resources, which enables strong support for code portability. mOSAIC [25] also addresses the above requirement from its mapping system, semantic engine, and dynamic discovery. The mOSAIC project [26] used the concept of cloud patterns and formed a research group in the Department of Industrial and Information Engineering at the Second University of Naples to enable the orchestration of cloud services that promote interoperability and portability of the cloud. The mOSAIC project derives a set of inference rules from the semantics of cloud services, resources used, cloud pattern, and orchestration to promote the discovery of services from the provider independently by enabling automated service discovery [27].

Di Martino et al. [28,29] stated that by using domain ontology description, standardization efforts can be derived from a non-uniform representation of services. This enables the semantic matching of the service, which will result in efficient service retrieval techniques by matching the semantic description offered by the PaaS providers. Using an agent to represent the semantic services by the service-provider and service-requestor can promote an automated retrieval of essential services. The survey likewise observed that the principled meaning of PaaS administrations is a crucial prerequisite for accomplishing the administration of interoperability among heterogeneous PaaSs to encourage benefit service discovery and composition. This can be accomplished through the specific service instances based on domain ontology formulated for PaaS services.

Alkalbani et al. [30] proposed two models for the cloud service discovery process which includes harvesting as a service (HaaS) and the service vault model. Centralized repository and web ontology language are used by service vault model for an efficient and effective cloud discovery process. Cloud service descriptions, which are available in different formats, are extracted through the Harvester module by extracting cloud service data from structured web portals through crawling the real time data of cloud service information.

Bouzerzour et al. [31] finalized a cloud broker, model based solution and semantic technologies to focus on achieving interoperability based on service client. Interoperability based on the service provider depends on standardized API, middleware, and protocols for solutions. Table 1 summarizes the efforts of service brokering, but with a lack of standardization effort for the seamless migration of data between cloud providers, this must be ensured with proper SLA management for interoperable services. The service provider has not fully accepted the standards to deploy applications to the cloud. The heterogeneity of cloud service description language is the major concern that requires standardization. To resolve provider lock-in, a generic model for interoperable cloud service and service description has to be implemented in the future.

Table 1. Summary of the service brokering framework.

Study	SLA	QOS	Service Monitoring	Cloud Ontology	Semantic Engine
European Commission [11]	✓	✓	×	×	×
SITIO [17]	✓	✓	✓	×	×
Parhi M. et al. [19].	×	×	×	✓	✓
Taylor, S.J. et al. [20]	✓	✓	×	✓	×
Bassiliades et al. [22]	✓	✓	✓	✓	✓
Quadir et al. [23]	✓	✓	✓	×	×
Petcu et al. [24]	✓	✓	×	✓	✓
mOSAIC project [26]	✓	✓	✓	✓	✓
Di Martino et al. [28,29]	×	✓	✓	✓	✓
Alkalbani et al. [30]	✓	✓	✓	✓	✓
Bouzerzour et al. [31]	✓	✓	×	✓	✓

3. Semantic Interoperable and Portable Needs of Cloud Services

Even though cloud computing has its advantages, it fails in a unique semantic approach. Fensel [32] says that the term ontology is closely associated with the semantic structure, meaning "theory of existence". The knowledge-sharing framework supporting the representation, sharing, and the subsequent reusability of field knowledge is the major contribution of this semantic structure. Information retrieval, knowledge management, semantic web, semantic search, information integration, and recommendation systems make use of ontologies [33].

3.1. Need for Semantic Cloud Ontology

Web Ontology Dialectal, which is relayed in W3C standard Web Ontology Language (OWL) and depicts the organization of semantics, will be utilized to make the ontology for the proposed structure. OWL-S has the twofold limit of familiarizing ontology with depicting the ideas of service space and conventional ideas to portray the service themselves and the manner in which they identify with the area ontology by means of input, output, precondition, and effects. The expressive semantics of this ontology for services enables semantically rich descriptions facilitating automated machine reasoning over services and domain descriptions, enabling intelligent discovery, invocation, and composition of services. Ontology has three primary classes including a service class.

- Profile: This describes the role of the service via functional properties of hasOutput, hasInput, hasPrecondition, hasParameter, and hasResult with the assistance of non-functional attributes like service Name, service Category, text Description, etc., which helps in promoting the service and enables automatic discovery of services.
- Process: This can either be atomic or composite making to know how the service works at the background, which provides in-depth analysis of the services to the service agent for enabling the composition of services.
- Grounding: This provides the implementation details of services such as port numbers, communication protocols, message form, etc. through which services can be invoked from any other services. The abstract semantic description of services through service

profile and the process will be mapped into actual service by implementing them using service API by using any of the ontology tools such as protégé [34], WebVOwl, Graphical Online editor, etc. Services such as profile, process, and grounding jointly support an agent's service usage.

Semantic information of a cloud service should be inferred correctly to enable an application, so that it can be invoked from a cloud provider to another cloud provider, which promotes interoperability and portability by supporting inter-cloud operation [35]. If the application is under a particular cloud domain, it facilitates the re-engineering of an existing application using semantic information. The semantic information [36] of cloud service includes the physical resource used, library and background services, service-level agreement (SLAs) made between the service consumer and provider, and the non-functional parameter related cloud key performance indicators (KPIs), and the actual service description using services profile, process, and grounding. Many research works now use semantic technologies for representing the cloud domain, which enables generalization of the cloud concept using OWL (Web Ontology Language) by giving birth to cloud ontologies.

3.2. Interoperable and Portable Frameworks for Cloud Service Providers

There are various interoperable and portable frameworks for cloud service providers. N. Loutas et al. [37] proposed a platform as a service semantic interoperability framework (PSIF) that defined the semantics of PaaS application using three dimensions such as fundamental PaaS entities and types and levels of semantic conflicts. Fundamental PaaS entities include the system, offering, management interface, IaaS, and software components of the PaaS application. Type of semantics includes the functional, non-functional. and execution environment of the PaaS application. Level of semantic conflict includes the data and information models of a PaaS application. PSIF resolves the conflicts that occur while deploying the application to cloud by defining the three dimensions such as entities, semantics, and conflicts. The conflicts are resolved by adding a semantic interoperability layer with PaaS architecture, which would enable a common linking between heterogeneous clouds.

Arora Pankaj et al. [38] proposed an Inter Cloud Topology, which enables interoperability transaction between two cloud services by enabling a cloud instance to communicate with another cloud instance by exchanging the needed information as a precursor to the transaction. Semantics information of cloud services are stored in a resources description framework (RDF) and for semantic information exchange, Extensible Messaging and Presence Protocol XMPP is used as an inter-cloud protocol. Androcec et al. [39] proposed a unique PaaS API ontology. The PaaS API ontology was developed using Web Ontology Language (OWL), which includes several classes and associated operation related to resources of PaaS API, therefore enabling a proper understanding about cloud services provided by cloud providers. The initiative discussed above of PaaS APIs failed to explore differences in service descriptions, which are the level of PaaS semantic interoperability.

Cloud4SOA [40], as an EC-funded project, represents the interconnection or composition of heterogeneous PaaS services using semantics information for addressing issues of semantic interoperability in cloud infrastructures. Cloud4SOA combines three dimensions: cloud computing, semantics, and service-oriented architecture (SOA), and it is an open source project that has a time overhead compared to Cloud Foundry to enable semantic interoperability between different PaaS platforms. Zeginis et al. [41] proposed an implementation framework of Cloud4SOA by deploying, managing, monitoring, and migrating applications across private and public heterogeneous PaaS platforms. Cloud4SOA discovers the best PaaS services needed for application developers among equivalent PaaS services using a matchmaking algorithm by extracting the semantic description of application profiles and offering PaaS cloud services for computing the degree of similarities.

Baqa et al. [42] clearly explained the need for semantic interoperability and used an ontology recommendation framework for semantic interoperability standards such as ontology driven interoperability, setting up of an ontology practitioner, and establishing

ontology standards including domain and general ontology. Established ontology has been evaluated using syntactic, semantic based validation, and evolution based validation. After evaluation, the ontology was published for interoperable cloud services.

Kaur T et al. [43] developed two models such as the semantic based approach and TensorFlow for addressing application portability and interoperability. The semantic based approach uses WSDL for parameter level and operational level, and OWL representation for service level and infrastructure level to find a particular cloud service provider. TensorFlow is used for statistical computation to obtain the service request from the cloud service provider, which trains the TensorFlow and checks whether the job is portable, then ports the job if yes; else the service goes to the high end services that provide services to the end user.

Table 2 summarizes the various efforts carried out in an interoperable and portable framework. Many research activities have used semantic technology for representing cloud resources, but up to now, there is no common or uniform approach to representing semantic cloud services and resources, enabling users to make use of a common and familiar interface to interact with them.

Table 2. Summary of interoperable and portable frameworks.

Interoperable and Portable Frameworks	Techniques			Description
	OWL	WSDL	RDF	
Platform as a Service Semantic Interoperability Framework (PSIF)	✓	×	×	Resolves semantic conflicts in interoperability layer
Inter Cloud Topology	×	×	✓	Semantic information exchange in inter cloud domain
PaaS API Ontology	✓	×	×	Semantic information resources of the PaaS API
Cloud4SOA	✓	✓	×	Discovers best PaaS services for application developers
Ontology recommendation framework	✓	×	×	Evaluation and validation of semantic ontology
Tensor flow	✓	✓	×	Semantic approach for interoperable and portable cloud services

3.3. Migration and Composition of Cloud Service in a Multi-Cloud Environment

The main goal of cloud computing is to migrate a legacy application into any of the PaaS clouds. One of the major legacy applications is the banking sector. REuse and Migration of Legacy Applications to Interoperable Cloud Services (REMICS) [44] uses a reverse engineering concept such as extracting Unified Modeling Language(UML) models from a legacy application and converts the UML model to a SOA (service oriented architecture) model that wraps the legacy application to the cloud through the use of web services and adapts a model-driven approach [45]. A novel approach for the migration of legacy software on the Cloud (ARTIST) [46] performs the migration of SME business legacy applications to the cloud for the modernization of the business model by reusing some of the REMICS concepts. The semantic information of the two projects for the migration of legacy cloud applications involves identifying the behavioral specification for managing software evolution have been proposed by Langer et al. Common features of REMICS and PaaSport migration of cloud where REMICS uses a model-driven approach and PaaSport is semantic brokering architecture, while for service failure and recovery, REMICS uses replacement for the migration of service, but PaaSport uses a recommendation framework that provides the best matched PaaS offering from the marketplace. Semantic information of REMICS deals with behavioral aspects that are obtained from UML models whereas PaaSport uses the computing semantic information of applications such as the platform in which it is executed, the physical resource it has used, the key performance indicator of the service, and functional parameters including programming environment, database service,

and various platform services, library, and API. Using this information can provide the best matched PaaS offering from the marketplace.

Like REMICS, MODAClouds [47] is also a modern driven engineering approach that works based on a decision support system. The decision of a support system provides multiple providers to satisfy the requirement of the PaaS offering for migration services and MODAClouds decides the best PaaS offering, which is similar to PaaSport. The objective MODAClouds is to deploy the application to the cloud using a multi-cloud provider through a semi-automated code by hiding the proprietary API. MODAClouds and PaaSport both use recommendation services for the best matched PaaS offering. As identified by Lund et al., MODAClouds uses a risk-based assessment for offering multi-cloud PaaS services, but PaaSport uses a multi-criteria approach as defined by Gupta et al. The stakeholders of MODAClouds are the cloud application developer and administrator while the stakeholders of PaaSport are the PaaS providers and software development engineers. MODACloud does not use any semantic information at all, but it makes use of multiple vendors for different services and PaaSport uses a single vendor approach. Earlier VISION and Contrails are similar projects to PaaSport for addressing the vendor lock-in problem, but provides only abstract cloud management and does not provide any semantic technology provided by PaaSport.

European software SME association has developed a PaaSport project [48,49]. The prime goal of the PaaSport project is to offer an open source unified semantic cloud brokering model through the marketplace for enabling interoperable PaaS offerings. Additionally, it has developed a unified PaaS API, which allows developers to deploy and migrate business applications seamlessly without any constraints of PaaS offerings. The dynamic environment and complex characteristics of the cloud PaaS marketplace have been addressed using a unified service agreement model through semantically interconnected marketplace infrastructures. The PaaS marketplace infrastructure was integrated into a unified semantic model [50] through a persistence layer. The objectives of PaaSport are the discovery of PaaS services, designing a user-centric front end, delivering a set of PaaS marketplace utilities, and providing the best-PaaS offering through the recommendation algorithm. The capabilities of PaaS offerings are represented using the characteristics and attributes in a unified semantic model through OWL ontology.

Di Martino, B et al. [51] proposed a semantic technique for cloud composition, which uses the base of the mOSAic Fp7 Project and application deployment for cloud platforms through the cloud patterns of Microsoft Azure, Open Stack, IBM Bluemix, Google App Engine, and Amazon Web services, which describes the environment and application design of cloud computing. Semantic techniques use a distributed batch K-means algorithm with the semantic approach of cloud patterns, cloud services, and virtual appliances, which only supports the semantic composition of the multi-cloud application using pattern workflow and does not support the portability of the application from one cloud to another cloud.

The PaaSport ontology extends the DOLCE+DnS Ultralite (DUL) ontology [52] to specify the characteristics and attributes of the PaaS offering through the extendable OWL class. The cloud marketplace infrastructure defines the requirements of business application using a SLA (service level agreement) between the cloud consumer and provider. The persistence layer of the cloud marketplace is implemented using a relational database that does not affect the extensibility of the OWL class. The relational database of the cloud marketplace has been defined to easily extend, even if the semantic model grows, there is no need to change the table. It will only add a newer table since it has been added as a subclass. Since the characteristics and parameters of PaaS offerings are agnostic, they are represented as the semantic class and object for providing the best matching cloud PaaS offering through semantic matchmaking [53] and ranking algorithm for application developers. The PaaS ontology semantically represents the characteristics and attributes of the capability of the PaaS offering while migrating an application from one cloud to another cloud. Other cloud capabilities are obtained from the cloud broker and service level agreements made between the marketplace provider and application owner. The marketplace provider obtains all

the semantic information from the repository and processes it using SPARQL queries by providing the best-matched cloud PaaS provider using a ranking based recommendation algorithm to the application developer.

Carrasco et al. [54] proposed two standards such as Cloud Application Management for Platform (CAMP) [55] and Topology and Orchestration Specification for Cloud Application (TOSCA) [56] for the management of PaaS and IaaS services. Due to diversity in services offered by the service provider, varying the QoS requirements that are represented using SLAs make interoperability and portability more complex, which result in provider lock-in. In order to address the issue, they proposed a unified API for PaaS and IaaS services for multi-cloud cross cloud application management. Services offered by the PaaS and IaaS cloud are unified under a common interface through TOSCA and CAMP. The topology of the cloud application has been described in a portable way using the TOSCA standard by the elicitation of agnostic cloud resources in an independent manner. The provider centric information are abstracted so only the target location is provided to the client based on the CAMP standard, so any need to port the application to another cloud requires only a change in target location.

Table 3 summarizes the various efforts carried out in the migration and composition of cloud services. The migration of legacy applications to the cloud needs a more generic approach and standard, which should be accepted by all cloud services providers and enables the interoperability and portability in a multi-cloud environment.

Table 3. Summary of migration and composition of cloud services.

Study	Types	Techniques Used
REMICS [44]	Migration services	Model-driven approach, UML, SOA
ARTIST [46]		Modernization of SME business model
MODAClouds [47]		Modern Driven Engineering approach and Decision Support System
PaaSport project [48,49].		Semantic Brokering Architecture and Recommendation Services
mOSAic Fp7 [51]	Composition of Services	Distributed Batch K-Means Algorithm, Cloud Patterns
DUL ontology [52]		DOLCE+DnS Ultralite Ontology, OWL, SPARQL
Carrasco et al. [54]	Standards	TOSCA and CAMP

4. Using Proprietary Application Programming Interface (API)

The PaaS application programming interface (APIs) acts as a mediator for deploying and migrating applications across multiple cloud providers such IBM Bluemix, Pivotal CF, Redhat Open Stacks, etc. Each provider maintains an API through which application services can be started, stopped, deployed, and deleted, so many kinds of API are called proprietary API. API, which can be used to perform a PaaS operation across multiple platforms such as one drive or multi-cloud, enables all cloud storage at a single point access and one kind of API is called the cross-platform API. The API acts as an agent for performing the PaaS operation across the platform, which enables the interoperability and porting of the application across the multi-cloud environment [57]. Interoperability of cloud services has been enhanced through cross-platform API by enabling the information exchange and communication between a shared interface. Some of the cross-platform interfaces are JCloud, Delta Cloud, OpenNebula, and libcloud.

The main idea of the Unified Cloud Interface $(UCI)_1$ project is to create a unified interface that will have access to all the proprietary cloud API. Cross-platform API interacts with all other cloud providers, which leads to a unified cloud interface. Similar PaaS services offering information conflicts can be avoided by using an ontology (OWL) [58]. UCI describes physical resources and PaaS offering details using semantic information to

avoid semantic conflicts between cloud-based API and existing standards and protocols. The illustration of the unified application programming interface and interaction between the cloud provider and consumer are represented in Figure 2 with the semantic engine and cloud ontology. However, each PaaS provider has given a proprietary API for the application developer to host the application to a particular cloud, which leads to a vendor lock-in problem. VMWare Cloud (VMW) has established an open source PaaS cloud provider named as Cloud Foundry, which provides generic API for the host application to the public and private cloud. Other than CF, OpenShift, GAE, Amazon Web Services Elastic BeansTalk, and Amazon EC2 help developers to deploy their applications onto a public cloud platform. All the above PaaS platforms have individual API for interfacing with the public cloud, since each code supports programming languages and uses different architectural styles, which increases the heterogeneity of the PaaS platform and makes the developer to learn different APIs.

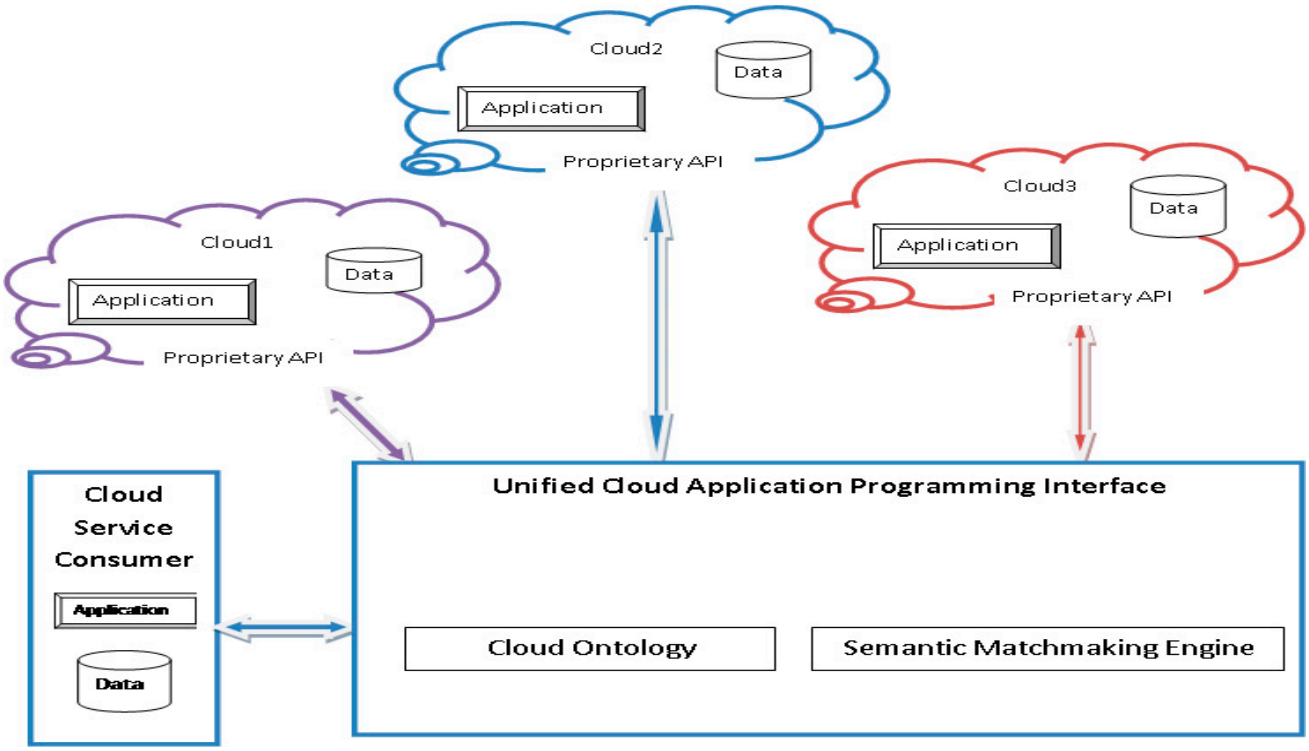

Figure 2. Unified cloud application programming interface.

Petcu et al. [59] proposed an open-source API work on multiple PaaS platforms, which helps developers to implement portable applications in such a way that the application is written once and deployed anywhere. Paraiso et al. [60,61] proposed the so cloud PaaS solution, which aims to satisfy application portability and ensure high availability across multiple clouds and is based on the service component architecture (SCA) standard. soCloud uses the APIs offered by the SCA standard to deal with application portability. However, it can deploy and execute only the service-oriented applications, which are a constraint when using the soCloud solution. PaaS Manager's logical architecture and integrating modules support the defined operational processes [62]. The PaaS Manager architecture has a modular design that allows the entire system to remain fully operational, even if some vendor or monitoring API is not operating correctly. Consequently, each API is implemented by distinct modules and managed by single entities. Finally, a REST interface exposes the specified operations to be invoked by any HTTP client application.

To migrate the PaaS application and also seamlessly interoperate between clouds, using a generic API called the Compatible One Application and Platform Service API

(COAPS) [63] helps to deploy and manage their application in two PaaS platforms, namely CF and OS [64]. The PaaS application description model (PADM) has been used to describe PaaS application resources such as runtime, framework, and services of software component along with specific PaaS properties, which are based on the Open Cloud Computing Interface (OCCI) standard. ServiceSs (Service Superscalar) [65] is an interoperable programming framework which helps the programmer to interoperate cloud application without using a specific API with the sequential programming model. The sequential programming model will implement sequential applications that can be executed in parallel on the cloud using a scaling factor. Without adopting the application code, the implemented applications can run on different PaaS platforms.

COAPS deployment API has been implemented [66] to include the GAE (Google App Engine), along with CF (Cloud Foundry) and OS (Open Shift) PaaS platforms that use the same application packaging, deploying the same application to different PaaS offerings. It provides a generic API for the packing and deployment of the same application to different cloud providers at the same time, which enables the porting of the legacy application or generic application to a multi-cloud environment. STAGER [67] is a framework for SemanTic-based GenERation of Generic-API Adapters. Upgrade of a PaaS provider needs to update the PaaS provider API, which in turn needs synchronization on a generic PaaS API that leads to the synchronization problem. The STAGER framework provides a semantic generic API for PaaS services for addressing synchronization issues. The author implemented the semantic generic API on Azure and GAE with blob storage and NOSQL datastore services. STAGER has also addressed data portability between GAE and Azure PaaS platforms.

OCCI is an open standard for the efficient management and access of cloud resources by the RESTful API and defines cloud resources through a generic extensible model. The model driven OCCIware approach [68] is composed of OCCIware Studio and OCCIware runtime. The OCCIware approach has to clearly specify the core model that can specify any cloud resources such as IaaS, PaaS, and SaaS and generates a cloud domain model for each domain and maps each domain to a designer. Each designer has mapped OCCIware runtime in five active implementations such as ACCORDS, erocci, OCCI4Java, pyOCNI, and pySSF. TOSCA Studio [69] uses API based implementation for managing cloud applications through TOSCA topology and OCCIware framework. The OCCIware approach and TOSCA standard were encoded as a topology for modeling and deploying cloud applications through the model-driven tool TOSCA Studio. It depends on Ecore and OCL for defining static semantics through the TOSCA Extension, which metamodels with the OCCI Metamodel. Using the Eclipse plug-in, TOSCO can be implemented with the help of TOSCA Designer for preparing, designing, and validating cloud applications. The deploying and management of applications has been taken care of by the OCCI Orchestrator.

From Table 4, it is clear that there is no proprietary API that satisfies all three types. There is no standard or single generic API adapted by all service providers to deploy applications to the cloud, which has led to fruitful research in the creation of semantic generic open source API and should be adopted by all kinds of service providers.

Table 4. Various types of proprietary application programming interface (API).

Proprietary API	Types of API			Description
	Open Source	Generic	Semantic	
UCI	×	×	✓	Semantic cloud-based API
Cloud Foundry	✓	✓	×	Deployment API
soCloud	×	✓	×	Uses SCA Standards
PaaS Manager	×	✓	×	Uses Modular Design
COAPS	✓	✓	×	Packaging and Deployment API
PADM	✓	×	×	Uses OCCI Standards
ServiceSs	×	✓	×	Sequential Programming Model
STAGER	×	✓	✓	Address data portability
OCCIware approach	✓	✓	×	Uses RESTful API
TOSCA Studio	✓	✓	×	Modeling and Deploying cloud application

5. Standard for Cloud Interoperability and Portability

Studies have collected existing standards and proposals to determine the specific cloud issues that they can solve or define as to how such standards can be used to build a cloud infrastructure. Currently, no standard has yet been accepted worldwide to definitively solve interoperability and portability issues. Instead, different efforts have been made toward the definition of such a standard, addressing the problem from different points of view. A cloud advisory group has been formed by the Cloud Standards Customer Council (CSCC) for the faster adoption of cloud computing and to address the key performance index through standards by addressing the wavering requirements such as the security, interoperability, and portability issues of the cloud. In virtual machines, Open Virtualization format (OVF) has been used as a packaging standard to address deployment and portability issues at the IaaS service level achieved by the Distributed Management Task Force. To perform CRUD operation on a storage cloud, a functional interface has been designed by the Storage Networking Industry Association (SNIA) through Cloud Data Management Interface (CDMI). Cloud auditing standards developed by the Distributed Management Task Force Access management functionality standard are enabled through LDAP, OAuth, OpenID Connect, and SAML. The cloud computing interoperability and portability ISO standard has been defined by ISO/IEC 9947. Packaging and runtime specification of the container image has been specified by the Open Container Initiative Image Specification and Open Container Initiative Runtime Specification. Deployment and management of docker or OCI container images have been defined using the Kubernetes container management platform.

For interoperability of the PaaS cloud, Topology and Orchestration Specification for Cloud Applications (TOSCA) and Cloud Application Management for Platforms (CAMP) are two main OASIS3 standardization efforts in progress. TOSCA (2013) tries to enhance the portability framework of cloud application and services. CAMP (Carlson et al., 2012) aims at defining the interoperability standard of managing applications in PaaS environments. A topology description of cloud-based services has been provided using a standard language for defining the operational concepts of services such as start, stop, deploy, and scaling of cloud services and the relationship of services and components of PaaS applications represented using a topology by TOSCA. The topology of TOSCA includes a meta-model for the construction and management of services. PaaSport, offering an application semantic model, is used as a TOSCA notion of software dependencies between application requirements and service capabilities. The CAMP approach uses the semantics of PaaSport through CAMP-OWL, which includes the meta-model, artifacts for controlling PaaS operation, monitoring, administration, and integration of the application available in the cloud. The aims of the PaaS cloud is to define the meta-model, the artifacts, and APIs that need to be offered for building, running, monitoring, and patching applications in the cloud [70].

Amazon Web Services (AWS) is the world leader of IaaS services for providing data storage and data warehousing. It acts as a the de facto standard for the public cloud with

all other cloud APIs in the IaaS offering. AWS has been widely accepted and has proven its standard by scaling up the resource at peak time, which is nothing but resource elasticity of the cloud. In such cases, a huge number of resources should scale up and after the sale end, the resource should be scaled down. This has been tested under AWS services, which have proven its capabilities. The counterpart of AWS is provided through an open source version of IaaS offerings such as Open Stack and Open Shift.

Since Open Stack is open source, many organizations use it to set up a cloud environment within an organization under the free license of the Apache Software Foundation because of which it has received much support from large IT companies including Oracle, IBM, Red Hat, and Rackspace. For the above reason, it has been defined as the de facto standard for IaaS offerings. In the PaaS world, some of the major players including Microsoft Azure, VMware, Redhat Open shift, IBM Bluemix, Google App Engine, and Pivotal Web Services are now widely adopted. De facto standards for PaaS offerings at the API level that have received noteworthy attention are Redhat OpenShift and VMWare Cloud Foundry.

Adding semantics to PaaS cloud services can lead to the standardization of interoperable and portable cloud services. Interoperability can be achieved by representing cloud services through the semantics of the service profile, service grounding, and service model of cloud ontology. Semantic information can be used to invoke other cloud services easily by comparing the semantics of the cloud provider and cloud consumers. The cloud broker provides the best-matched service of service providers to service the client. Portability at the PaaS layer is an important constraint where some of the proprietary PaaS providers will never share the working model, which has to be represented in semantics for port applications from one cloud provider to another cloud provider. The comparison of various research works with the semantics of interoperability and portability of cloud services is summarized in Table 5. It is clear from Table 5 that adding semantics to the migration effort can lead to fruitful research on cloud portability, along with existing semantics on cloud interoperability, and realize interoperable and portable cloud services in the multi-cloud environment. It also clear from Table 5 that the migration and semantics of interoperability and portability are not achieved together, which pave the way for innovation. Adding semantics in the migration effort will lead to fruitful research work. In the future, we have planned to add semantics for the COAPS API, the creation of domain ontology for each proprietary API, migrate cloud application between different cloud provider to avoid the vendor lock-in problem, and interoperate seamlessly in a multi-cloud environment.

Table 5. Standardization efforts.

Standardization Efforts	Migration Effort	Semantics in Interoperability	Semantics in Portability	Description
mOSAIC	×	✓	×	Provides IaaS and PaaS API for Service Discovery Framework
SITIO	×	✓	×	Business Interface for Cloud Application
PSIF	✓	✓	×	Linking framework for heterogeneous cloud
Cloud4SOA	×	✓	✓	Abstraction among all PaaS Services
PaaS Manager	✓	×	×	Manual Migration of Cloud application in multi cloud environment
REMICS	×	✓	×	Migration of Legacy Application to SOA Web Service Application to Cloud Application
ARTIST	✓	✓	×	Migration of SME Legacy Application to Cloud Application
MODAClouds	✓	×	×	Migration of Legacy Application through proprietary API
PaaSport	✓	×	✓	Semantic matching making services for PaaS Services
SCA	✓	×	×	Porting of SOA application to soCloud.
TOSCA	×	✓	✓	Interoperability Standard for PaaS Services
Open Swift	✓	×	×	PaaS Provider allow to create, deploy application to cloud
Cloud Foundry	✓	×	×	PaaS Provider allow to create, deploy application to cloud
COAPS API	✓	✓	×	Generic API for packing and deploying of Application to Multi Cloud Environment
Cloud SME	×	✓	✓	Simulation framework for hosting application to multi cloud environment
STAGER	✓	×	✓	Semantic Data Portability between Microsoft Azure and GAE

6. Conclusions

An extensive survey on the portability and interoperability of services at the service broker level, semantic level, and API level have been presented for migrating and collaborating services, which will enable greater research opportunities for technical innovation efforts among multi-cloud environments, in particular at the SaaS and PaaS levels. The adoption of cloud computing will be increased through standardization. The standardization effort has led to common data model supports and provides a mechanism for improving the portability and interoperability aspect of cloud computing. Users will have a wide range of choices in cloud, and organizations can use their own existing data center resources seamlessly. Standardization also allows the users to choose and use services provided by numerous different cloud vendors based on various criteria. It also benefits the vendor by providing additional higher-level services apart from normal cloud services that are needed by the user. Thus, standardization will open the way toward realizing the true potential and benefits of cloud computing.

Future work will concentrate on the standardization of unified API through service semantics, which will drive the pace of innovation by building cloud computing on open source standards. Open source and open standards make it easy for designers and developers to share knowledge quickly, and this knowledge is the key to lowering operating costs when designing, programming, and implementing a cloud application system in a multi-cloud environment, thereby enabling interoperability and portability of cloud services at a lower cost.

Author Contributions: C.R., and P.M. contributed equally to this article. All 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.

Acknowledgments: The authors would like to acknowledge K. Jayashree, CSE Department, Rajalakshmi Engineering College, Chennai for providing technical support.

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

References

1. Lewis, G. *Basics about Cloud Computing*; Software Engineering Institute, Carniege Mellon University: Pittsburgh, PA, USA, 2010.
2. Strowd, H.D.; Lewis, G.A. *T-Check in System-of-Systems Technologies: Cloud Computing*; Software Engineering Institute, Carniege Mellon University: Pittsburgh, PA, USA, 2010.
3. Wang, L.; Von Laszewski, G.; Younge, A.; He, X.; Kunze, M.; Tao, J.; Fu, C. Cloud computing: A perspective study. *New Gener. Comput.* **2010**, *28*, 137–146. [CrossRef]
4. Gangemi, A.; Mika, P. Understanding the semantic web through descriptions and situations. In Proceedings of the OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", Sicily, Italy, 3–7 November 2003; Springer: Berlin/Heidelberg, Germany, 2003; pp. 689–706.
5. Mell, P.; Grance, T. *The NIST Definition of Cloud Computing (Draft) Recommendations of the National Institute of Standards and Technology*; NIST Special Publication: Gaithersburg, MD, USA, 2011; Volume 145, pp. 1–2.
6. Liu, F.; Tong, J.; Mao, J.; Bohn, R.; Messina, J.; Badger, L.; Leaf, D. NIST cloud computing reference architecture. *NIST Spec. Publ.* **2011**, *500*, 1–28.
7. Katsaros, G.; Gogouvitis, S.; Mavrogeorgi, N.; Voulodimos, A.; Kiriazis, D.; Varvarigou, T.; Talyansky, R. A holistic view of information management in Cloud environments. In Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA, 24–29 June 2012; pp. 989–990.
8. Brownsword, L.L.; Carney, D.J.; Fisher, D.; Lewis, G.; Meyers, C. *Current Perspectives on Interoperability (No. CMU/SEI-2004-TR-009)*; Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst: Pittsburgh, PA, USA, 2004.
9. Stravoskoufos, K.; Preventis, A.; Sotiriadis, S.; Petrakis, E.G. A Survey on Approaches for Interoperability and Portability of Cloud Computing Services. In Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014), Barcelona, Spain, 3–5 April 2014; pp. 112–117.

10. Hoff, C. Inter-Cloud Rock, Paper, Scissors: Service Brokers, Semantic Web or APIs? 2009. Available online: http://www.rationalsurvivability.com/blog/?p=1189 (accessed on 12 February 2021).
11. European Commission. Software & Services FP7 Project Portfolio—Internet of Services, Software and Visualisation Call 5. 2010. Available online: http://www.sequoiaproject.eu/index.php/documents/doc_download/17-software-a-services-fp7-projectportfolio (accessed on 28 January 2018).
12. Di Martino, B.; Petcu, D.; Cossu, R.; Goncalves, P.; Máhr, T.; Loichate, M. Building a mosaic of clouds. In Proceedings of the European Conference on Parallel Processing, Bordeaux, France, 29 August–2 September 2011; Springer: Berlin/Heidelberg, Germany, 2010; pp. 571–578.
13. Carlini, E.; Coppola, M.; Dazzi, P.; Ricci, L.; Righetti, G. Cloud federations in contrail. In Proceedings of the European Conference on Parallel Processing, Bordeaux, France, 29 August–2 September 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 159–168.
14. Sim, K.M. Agent-based cloud computing. *IEEE Trans. Serv. Comput.* **2012**, *5*, 564–577.
15. Gupta, S.; Muntes-Mulero, V.; Matthews, P.; Dominiak, J.; Omerovic, A.; Aranda, J.; Seycek, S. Risk-driven framework for decision support in cloud service selection. In Proceedings of the 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Shenzhen, China, 4–7 May 2015; pp. 545–554.
16. Buyya, R.; Garg, S.K.; Calheiros, R.N. SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In Proceedings of the 2011 international conference on cloud and service computing, Hong Kong, China, 12–14 December 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1–10.
17. Garcia-Sanchez, F.; Fernandez-Breis, E.; Valencia-Garcia, R.; Jimenez, E.; Gomez, J.M.; Torres-Niño, J.; Martinez-Maqueda, D. Sitio: Semantic Business Processes Based on Software-as-a-Service and Cloud Computing. 2009. Available online: https://www.semanticscholar.org/paper/SITIO%3A-semantic-business-processes-based-on-and-Garc%C3%ADa-S%C3%A1nchez-Fernandez-Breis/233e526b7788d7cda02ff6b0879b7d7d4691b73d (accessed on 12 February 2021).
18. Cretella, G.; Di Martino, B. A semantic engine for porting applications to the cloud and among clouds. *Softw. Pract. Exp.* **2015**, *45*, 1619–1637. [CrossRef]
19. Parhi, M.; Pattanayak, B.K.; Patra, M.R. A multi-agent-based framework for cloud service description and discovery using ontology. In *Intelligent Computing, Communication and Devices*; Springer: New Delhi, India, 2015; pp. 337–348.
20. Taylor, S.J.; Kiss, T.; Anagnostou, A.; Terstyanszky, G.; Kacsuk, P.; Costes, J.; Fantini, N. The CloudSME simulation platform and its applications: A generic multi-cloud platform for developing and executing commercial cloud-based simulations. *Future Gener. Comput. Syst.* **2018**, *88*, 524–539. [CrossRef]
21. Castañé, G.G.; Xiong, H.; Dong, D.; Morrison, J.P. An ontology for heterogeneous resources management interoperability and HPC in the cloud. *Future Gener. Comput. Syst.* **2018**, *88*, 373–384. [CrossRef]
22. 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]
23. Quadir, A.; Varadarajan, V.; Mandal, K. Efficient Algorithm for Identification and Cache Based Discovery of Cloud Services. *Mob. Netw. Appl.* **2019**, *24*, 1181–1197.
24. Petcu, D.; Di Martino, B.; Venticinque, S.; Rak, M.; Máhr, T.; Lopez, G.E.; Stankovski, V. Experiences in building a mOSAIC of clouds. *J. Cloud Comput. Adv. Syst. Appl.* **2013**, *2*, 12. [CrossRef]
25. Moscato, F.; Aversa, R.; Di Martino, B.; Fortiş, T.F.; Munteanu, V. An analysis of mosaic ontology for cloud resources annotation. In Proceedings of the 2011 federated conference on computer science and information systems (FedCSIS), Szczecin, Poland, 18–21 September 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 973–980.
26. mOSAIC FP7 Project. Available online: http://www.mosaic-cloud.eu (accessed on 28 February 2019).
27. Di Martino, B.; Cretella, G.; Esposito, A. Semantic and agnostic representation of cloud patterns for cloud interoperability and portability. In Proceedings of the 2013 IEEE 5th International Conference on Cloud Computing Technology and Science, Bristol, UK, 2–5 December 2013; IEEE: Piscataway, NJ, USA, 2013; Volume 2, pp. 182–187.
28. Di Martino, B.; Cretella, G.; Esposito, A.; Carta, G. Semantic representation of cloud services: A case study for openstack. In Proceedings of the International Conference on Internet and Distributed Computing Systems, Calabria, Italy, 22–24 September 2014; Springer: Cham, Switzerland, 2014; pp. 39–50.
29. Di Martino, B.; Cretella, G.; Esposito, A.; Sperandeo, R.G. Semantic representation of cloud services: A case study for Microsoft Windows Azure. In Proceedings of the 2014 International Conference on Intelligent Networking and Collaborative Systems, Salerno, Italy, 10–12 September 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 647–652.
30. Alkalbani, A.M.; Hussain, W.; Kim, J.Y. A Centralised Cloud Services Repository (CCSR) Framework for Optimal Cloud Service Advertisement Discovery from Heterogenous Web Portals. *IEEE Access* **2019**, *7*, 128213–128223. [CrossRef]
31. Bouzerzour, N.E.H.; Ghazouani, S.; Slimani, Y. A survey on the service interoperability in cloud computing: Client-centric and provider-centric perspectives. *Softw. Pract. Exp.* **2020**, *50*, 1025–1060. [CrossRef]
32. Fensel, D. Ontology-based knowledge management. *Computer* **2002**, *35*, 56–59. [CrossRef]
33. Trajanov, D.; Stojanov, R.; Jovanovik, M.; Zdraveski, V.; Ristoski, P.; Georgiev, M.; Filiposka, S. Semantic sky: A platform for cloud service integration based on semantic web technologies. In Proceedings of the 8th International Conference on Semantic Systems, Graz, Austria, 5–7 September 2012; ACM: New York, NY, USA, 2012; pp. 109–116.
34. Protégé. Available online: http://protege.stanford.edu/ (accessed on 28 February 2020).

35. Di Martino, B. Applications portability and services interoperability among multiple clouds. *IEEE Cloud Comput.* **2014**, *1*, 74–77. [CrossRef]
36. Garcia-Sanchez, F.; Fernandez-Breis, E.; Valencia-Garcia, R.; Jimenez, E.; Gomez, J.; Torres-Niño, J.; Martinez-Maqueda, D. Adding semantics to software-as-a-service and cloud computing. *WSEAS Trans. Comput.* **2010**, *9*, 154–163.
37. Loutas, N.; Kamateri, E.; Tarabanis, K. A semantic interoperability framework for cloud platform as a service. In Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, Athens, Greece, 29 November–1 December 2011; pp. 280–287.
38. Arora, P.; Wadhawan, R.C.; Ahuja, E.S.P. Cloud Computing Security Issues in Infrastructure as a Service. 2012. Available online: https://www.semanticscholar.org/paper/Cloud-Computing-Security-Issues-in-Infrastructure-a-Arora-Wadhawan/049038fcf0c820764b3d011293f8be46e46b37ec (accessed on 12 February 2021).
39. Androcec, D.; Vrcek, N.; Küngas, P. Service-level interoperability issues of platform as a service. In Proceedings of the 2015 IEEE World Congress on Services, New York, NY, USA, 27 June–2 July 2015; pp. 349–356.
40. Kamateri, E.; Loutas, N.; Zeginis, D.; Ahtes, J.; D'Andria, F.; Bocconi, S.; Tarabanis, K.A. Cloud4soa: A semantic-interoperability paas solution for multi-cloud platform management and portability. In *European Conference on Service-Oriented and Cloud Computing, Málaga, Spain, 11–13 September 2013*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 64–78.
41. Zeginis, D.; D'andria, F.; Bocconi, S.; Cruz, J.G.; Martin, O.C.; Gouvas, P.; Tarabanis, K.A. A user-centric multi-PaaS application management solution for hybrid multi-Cloud scenarios. *Scalable Comput. Pract. Exp.* **2013**, *14*, 17–32. [CrossRef]
42. Baqa, H.; Bauer, M.; Bilbao, S.; Corchero, A.; Daniele, L.; Esnaola, I.; Guillemin, P. Towards Semantic Interoperability Standards Based on Ontologies. 2019. Available online: https://hal-emse.ccsd.cnrs.fr/emse-02352822/document (accessed on 12 February 2021).
43. Kaur, T.; Kaur, K. TensorFlow-Based Semantic Techniques for Multi-cloud Application Portability and Interoperability. In *Inventive Communication and Computational Technologies*; Springer: Singapore, 2020; pp. 13–21.
44. REMICS FP7 Project. 2013. Available online: http://www.remics.eu (accessed on 10 December 2018).
45. Lund, M.S.; Solhaug, B.; Stølen, K. *Model-Driven Risk Analysis: The CORAS Approach*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2010.
46. ARTIST FP7 Project. 2016. Available online: http://www.artist-project.eu (accessed on 28 February 2019).
47. Di Nitto, E.; Casale, G.; Petcu, D. On MODAClouds toolkit support for DevOps. In Proceedings of the Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC2015, in: Revised Selected Papers, Taormina, Italy, 15–17 September 2015; Springer: Berlin/Heidelberg, Germany, 2015; Volume 567, p. 430.
48. PaaSport Consortium. Available online: http://linc.ucy.ac.cy/index.php?id=95 (accessed on 9 March 2017). Deliverable 1.1,1.2,1.3,4.1.
49. PaaSPort FP7 Project. 2018. Available online: http://paasport-project.eu (accessed on 8 December 2018).
50. Langer, P.; Mayerhofer, T.; Kappel, G. Semantic model differencing utilizing behavioral semantics specifications. In Proceedings of the International Conference on Model Driven Engineering Languages and Systems, Valencia, Spain, 28 September–3 October 2014; Springer: Cham, Switzerland, 2014; pp. 116–132.
51. Di Martino, B.; Esposito, A. Semantic Techniques for Multi-Cloud Applications Portability and Interoperability. *Procedia Comput. Sci.* **2016**, *97*, 104–113. [CrossRef]
52. Bassiliades, N.; Symeonidis, M.; Meditskos, G.; Kontopoulos, E.; Gouvas, P.; Vlahavas, I. A semantic recommendation algorithm for the PaaSport platform-as-a-service marketplace. *Expert Syst. Appl.* **2017**, *67*, 203–227. [CrossRef]
53. Cretella, G.; Di Martino, B. Semantic and matchmaking technologies for discovering, mapping and aligning cloud providers's services. In Proceedings of the International Conference on Information Integration and Web-Based Applications & Services, Vienna, Austria, 2–4 December 2013; ACM: New York, NY, USA, 2013; p. 380.
54. Carrasco, J.; Durán, F.; Pimentel, E. Trans-cloud: CAMP/TOSCA-based bidimensional cross-cloud. *Comput. Stand. Interfaces* **2018**, *58*, 167–179. [CrossRef]
55. Carlson, M.; Chapman, M.; Heneveld, A.; Hinkelman, S.; Johnston-Watt, D.; Karmarkar, A. Cloud Application Management for Platforms (CAMP). Available online: http://docs.oasis-open.org/camp/campspec/v1.1/camp-spec-v1.1.pdf (accessed on 28 February 2019).
56. TOSCA. Topology and Orchestration Specification for Cloud Applications. 2013. Available online: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCAv1.0-os.pdf (accessed on 28 February 2019).
57. Sellami, M.; Yangui, S.; Mohamed, M.; Tata, S. PaaS-independent provisioning and management of applications in the cloud. In Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, 28 June 28–3 July 2013; pp. 693–700.
58. Van Harmelen, F.; McGuinness, D.L. OWL Web Ontology Language Overview. 2004. Available online: https://www.w3.org/TR/owl-features/ (accessed on 12 February 2021).
59. Petcu, D.; Frincu, M.; Craciun, C.; Panica, S.; Neagul, M.; Macariu, G. Towards open-source cloudware. In Proceedings of the 2011 Fourth IEEE International Conference on Utility and Cloud Computing, Victoria, Australia, 5–8 December 2011; pp. 330–331.
60. Paraiso, F.; Merle, P.; Seinturier, L. soCloud: A service-oriented component-based PaaS for managing portability, provisioning, elasticity, and high availability across multiple clouds. *Computing* **2016**, *98*, 539–565. [CrossRef]

61. Paraiso, F.; Haderer, N.; Merle, P.; Rouvoy, R.; Seinturier, L. A federated multi-cloud PaaS infrastructure. In Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, Chicago, IL, USA, 5–8 November 2012; pp. 392–399.
62. Cunha, D.; Neves, P.; Sousa, P. PaaS manager: A platform-as-a-service aggregation framework. *Comput. Sci. Inf. Syst.* **2014**, *11*, 1209–1228. [CrossRef]
63. Sellami, M.; Sami, Y.; Mohamed, M.; Tata, S. *The Compatible One Application and Platform Service (COAPS) API Specification*; Telecom SudParis, Computer Science Department: Évry-Courcouronnes, France, 2013.
64. Sudparis, T. The Compatible One Application and Platform Service (COAPS) API Specification. 2013. Available online: http://www.compatibleone.org/bin/download/Download/Software/COAPS-Spec.v1.5.3.pdf (accessed on 10 December 2018).
65. Lordan, F.; Tejedor, E.; Ejarque, J.; Rafanell, R.; Alvarez, J.; Marozzo, F.; Badia, R.M. Servicess: An interoperable programming framework for the cloud. *J. Grid Comput.* **2014**, *12*, 67–91. [CrossRef]
66. Hossny, E.; Khattab, S.; Omara, F.A.; Hassan, H.A. Implementing generic PaaS deployment API: Repackaging and deploying applications on heterogeneous PaaS platforms. *Int. J. Big Data Intell.* **2016**, *3*, 257–269. [CrossRef]
67. Khattab, S.; Omara, F.A.; Hassan, H. STAGER: Semantic-based Framework for Generating Adapters of Service-based Generic-API for Portable Cloud Applications. *IEEE Trans. Serv. Comput.* **2018**, 1–14. [CrossRef]
68. Zalila, F.; Challita, S.; Merle, P. Model-driven cloud resource management with OCCIware. *Future Gener. Comput. Syst.* **2019**, *99*, 260–277. [CrossRef]
69. Challita, S.; Korte, F.; Erbel, J.; Zalila, F.; Grabowski, J.; Merle, P. Model-Based Cloud Resource Provisioning with TOSCA and OCCI. *arXiv* **2020**, arXiv:2001.07900.
70. Fehling, C.; Leymann, F.; Retter, R.; Schupeck, W.; Arbitter, P. *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2014.