A Loosely-coupled Semantic Model for Diverse and Comprehensive Cloud Service Search and Retrieval

Daren Fang, Xiaodong Liu, Imed Romdhani
School of Computing
Edinburgh Napier University, UK
emails: {d.fang, x.liu, i.romdhani}@napier.ac.uk

***Abstract*—As cloud services propagate along with the rapid development of cloud computing, issues raised in service selection and retrieval processes become increasingly critical. While many approaches are proposed on the specification and discovery of cloud services, existing service models and recommendation systems cannot achieve ultimate effectiveness while dealing with a variety of cloud services of different categories/levels/characteristics. To this extent, this paper proposes a Cloud Service Explorer (CSE) tool, which takes advantage of a Loosely-Coupled Cloud Service Ontology (LCCSO) model as the knowledge base to assist user-friendly service search and service information access. Demonstrated using a number of real world cloud services and their official service data as examples, it proves that the model and tool are capable of offering an efficient and effortless means of handling diverse service information towards ultimate service discovery and retrieval.**

Keywords-cloud computing; cloud service; semantic model; ontology; service recommendation system.

I. INTRODUCTION

Cloud Computing (CC) introduces a revolutionary information and communication technology (ICT) paradigm to the world, known as on-demand provisioning, pay-per-use self-service, ubiquitous network access and location-independent resource pooling [1]. It provisions reliable, scalable and elastic computational resources that effectively adapt to nearly all kinds of needs. Within barely a decade, CC has permeated into all major industry sectors. However, as the number of cloud services continues growing whilst the market becomes more complex, it would take considerable time and efforts for service users to find the targeted services, by researching a great many service descriptions, properties, Service Level Agreements (SLAs), ratings, reviews, trials, etc. Moreover, regarding services' functionality, usability, customizability, etc., existing cloud providers pose various interfaces, standards, service level data and explanations [2][3], which result into serious difficulties in service information retrieval, interpretation and analysis.

Consequently, the rapid development of CC has imposed urgent needs yet great challenges on the specification and retrieval of cloud services, whereas an effective means of cloud service search and discovery is under demand for a diversity of users. A successful model of cloud services should be able to cope with the following challenges: 1) Cloud services and resources involved in CC systems are across multiple abstraction levels, from Infrastructure-as-a-Service (IaaS), to Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). Such a matrix structure has incurred that cloud services contain dependencies over one another, and there are close or loose relations across the different levels of cloud services/resources. 2) Cloud services are rather complicated in both functional and non-functional aspects, whereas certain well-resourced services can be used flexibly to achieve diverse ranges of functions. These can be seen as the potential capabilities of the services. Accordingly, a superior service explorer system which works on top of the model should then be capable of providing ultimate service search and retrieval to meet complex requirements for all users regardless of their types or levels of expertise.

Many efforts have been made on the semantic specification of cloud services using OWL/OWL2 [4] ontology modeling approach, e.g., [5][6][7][8][9][10]. Yet, the majority of the existing models focus on specifying certain specific service categories, whereas very few of them utilize the various types of OWL 2 property assertions for ultimate service information specification. These drawbacks expose significant issues when users try to retrieve cloud services from the pool, since there is not a unified knowledge base that can hold comprehensive cloud service information for all service categories (all IaaS, PaaS and SaaS services of different infrastructure provision, platform provision and software functions).

In this paper, the authors propose a Loosely-Coupled Cloud Service Ontology (LCCSO) model, which takes advantage of flexible concept naming as well as loosely-coupled axiom assertions to ultimately comprise comprehensive specifications of diverse cloud services from distinct levels and categories. Moreover, LCCSO employs the full range of OWL 2 axiom assertions including annotation, class, Object Property (OP), Data Property (DP) assertions. This consequently enhances service specification by involving various forms of service information. Using the semantic model as the central cloud service knowledge base, a Cloud Service Explorer (CSE) prototype tool is also developed to best assist users in cloud service search and retrieval tasks. The contributions of the paper are: 1) a loosely-coupled cloud service semantic model that is able to

 ISBN: 978-1-61208-338-4

present cloud services and comprehensive service information regardless of their categories, levels, characteristics, functional or non-functional properties; 2) a cloud service search and retrieval tool that is capable of providing an effective means of service discovery and service information access.

The rest of the paper is organized as follows: Section 2 discusses the related work. In Section 3, the design and implementation of LCCSO are demonstrated. Section 4 outlines the architecture of CSE prototype whilst the details of the system components are explained. The search and retrieval functions of CSE tool are further explored in Section 5, where examples of cloud service retrieval and tool screenshots are illustrated. Section 6 concludes the paper with summaries and future work.

II. RELATED WORK

Ontology-oriented or OWL-based approaches have been used widely in assisting service discovery, recommendation and composition. While its basic semantics provide comprehensive service annotations and descriptions, schema mapping and model reference, interface and operation parameter information, Description Logic (DL) reasoning is able to reveal additional inferred knowledge intelligently [2].

In the efforts of cloud computing/service semantic specification, existing ontology models expose various limitations considering the comprehensiveness and types of the knowledge revealed, whereas many of them are designed concentrating on certain restricted service categories, e.g., infrastructure services [5][9][10][11][12], platform services [7], software services [6][13]. More specifically, in Cloudle [9] and CSDS [12], both models developed cover only fundamental CC concepts, such as delivery models (e.g., IaaS, PaaS, SaaS), cloud hardware (e.g., CPU, memory, disk), software (e.g., OS, DBMS), programming languages (e.g., Python, C#, Java), etc. This implies that the specification would be limited to ordinary basic IaaS, PaaS and SaaS services. On the other hand, very few of the ontology models adopt comprehensive ontological assertion types to enhance the service specification. For instance, annotation properties are the main focus of the semantic platform for cloud service annotation and retrieval [6], whereas many others [9][11][12] are primarily concentrating on data-relevant service properties. Consequently, current existing cloud service models fail to deliver the best means of service specification.

Recently, various cloud service recommendation systems are developed, which rely on diverse service modeling and matchmaking techniques. The collaborative service recommender mechanism [14] is argued to specifically deal with consumer rated service quality matchmaking based on individual user's profile. While the prototype mainly works for non-functional (response time, availability, price, etc.) aspects-oriented service discovery, it suggests that such would not handle the diverse functional aspects effectively. In contrast, many others employ ontology models for cloud service discovery and recommendation; they produce enhanced results, yet still can be improved. CSDS [12] is primarily designed to deal with IaaS services; therefore the search/discovery is restricted to only that category. CloudRecommender [11] offers enhanced functions for various types of infrastructure service recommendations by allowing users to enter both functional and non-functional requirements. Nonetheless, it still cannot work perfectly for PaaS or SaaS service discovery. The cloud repository and discovery framework [7] advocates using a unified business and clouding service ontology to assist service discovery tasks. Although the proposed approach can support PaaS and IaaS services, the search functions are poorly provided due to the business function/process-oriented design where users have to specify certain business-relevant service requirements. Indeed, existing cloud service discovery/ recommendation systems cannot facilitate comprehensive service lookup across different service levels/categories whilst they fail to involve service characteristics aspects as search requirements.

In summary, current semantic models of cloud computing/services are not able to provide the best means of service specification due to the conventional category-restricted design and uneven use of ontology assertion types; existing cloud search/discovery approaches cannot effectively deal with the various cloud services of distinct categories, levels, characteristics or functional/non-functional properties.

III. LOOSELY-COUPLED ONTOLOGY DESIGN

In comparison with other work, LCCSO incorporates flexible concept naming and loosely-coupled axiom assertions to cover diverse service information across different service layers and categories. It utilizes a wide range of ontology assertion types to comprehensively reveal details regarding service functions, characteristics and features.

A. Cloud services and service models

While other work classifies cloud services according to the delivery or deployment models (e.g., Amazon EC2 belongs to IaaS or Public cloud), LCCSO gathers all cloud services and their originated companies into one class named "Registered Cloud Entity". The class can be seen as a "service registry" within the ontology (see Fig. 1). The advantages of the design is seen as: 1) it specifies clear subsumption relationships between a cloud company/ provider (class) and the services (individuals) it owns, whereas it allows to assert relevant relationships among cloud companies and services; 2) it enables effective service retrieval, lookup and processing from one united class, instead of extracting services from multiple service model classes; 3) it achieves flexible service specification assertions for cloud services, e.g., Amazon S3 can be asserted as "belongs to" multiple service models such both IaaS and PaaS, or PaaS; 4) the comparison among cloud

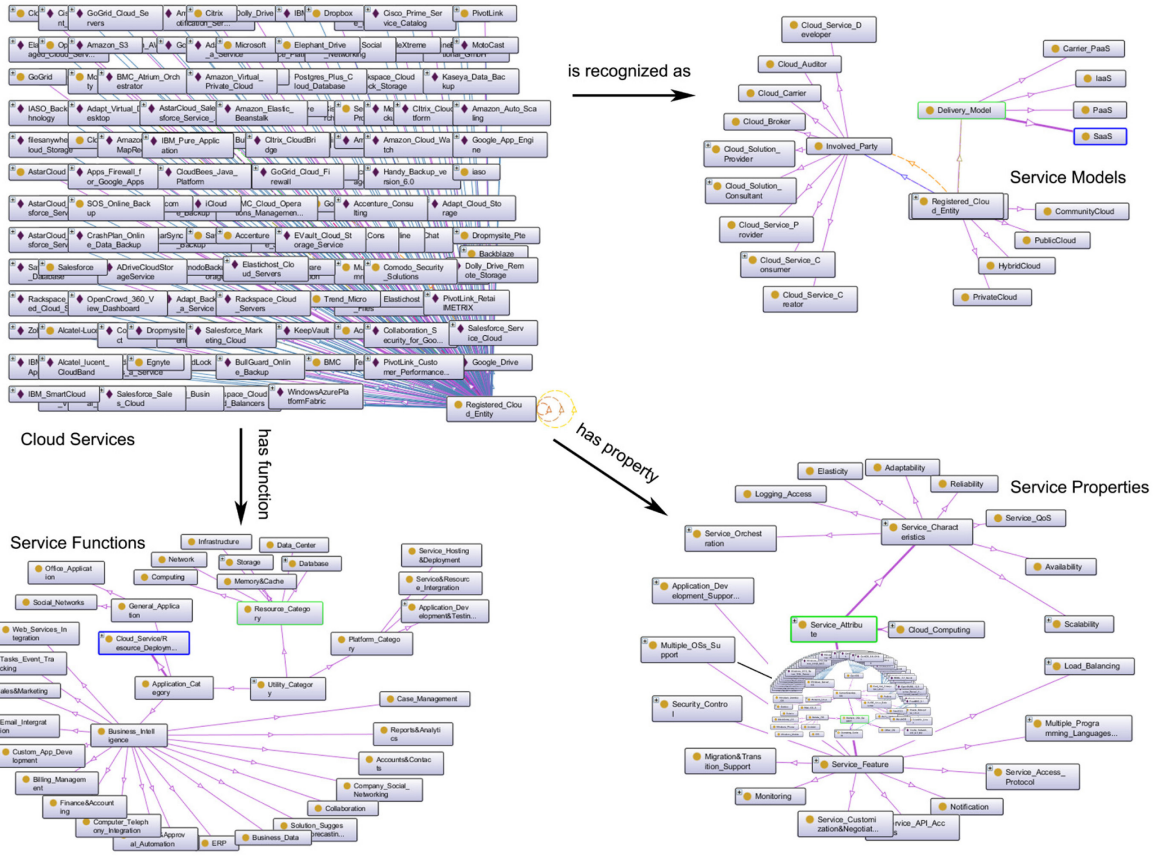

Figure 1. Loosely-Coupled Cloud Service Ontology.

companies and services can be effectively implemented, since they belong to a single class where their axiom assertions follow the same pattern.

B. *Service functions*

In LCCSO, "Utility Category" class comprises all the service functions that cloud services are capable of providing. As illustrated in Fig. 1, they are divided into three categories: resource, platform and application. "Resource Category" involves the various computational resources that cloud services (typically IaaS) can provision, i.e., "Computing, Storage, Database, Network, Data Center", etc. "Platform Category" covers the usages of provisioned cloud platforms (most likely PaaS), seen as "Application Development and Testing, Service and Resource Integration, Service Hosting and Deployment", etc. "Application Category" gathers the various application-alike (SaaS) cloud service functions, including "General Applications, Business Intelligence, and Cloud Service or Resource Deployment and Management", etc. The use of these classes is to reveal the exact capability of cloud services, instead of simply justify the primary designed function(s). For instance, a SaaS service may only provide one specific or very limited function(s) (in Software Category). Some PaaS services, however, can have multiple functions (from Platform Category), e.g., for application testing and deployment, whereas any service functions provisioned on top of the platforms can also be attributed to these PaaS services. Likewise, typical IaaS services, like VM provision services, can be used for multiple resource functions, such as provisioning computing, database, network and storage flexibly. Plus, for certain well-resourced IaaS services which are capable of providing application development platforms (used as PaaS services); their additional usages would be even more diverse [15]. As such, these service function specifications provide a comprehensive view for cloud services of different models.

C. *Service properties*

Service properties are divided into "Service Characteristics" and "Service Features", in LCCSO. While the former involves the common and unique cloud service properties such as "Adaptability, Elasticity, Scalability, Availability, Reliability", etc., the latter consists of relevant "extra" cloud service properties such as "Service Access Protocol, Service API Access, Service Customization and Negotiation, Security Control, Multiple operating system (OS)/Programming Language/Platform Support, Monitor, Notification", etc. (demonstrated in Fig. 1). By associating these properties with relevant cloud services, the comparison among services can be implemented effectively.

 ISBN: 978-1-61208-338-4

D. *Ontology axiom assertions*

In contrast with other existing cloud (service) ontology models, which only utilize partial ontology assertion types, LCCSO adopts full range of OWL2 assertions for comprehensive service specifications. Firstly, all CC concepts (including cloud services) come with Annotation assertions which provide general knowledge of them. Secondly, Description assertions (or Class assertion), in the form of "individual of/subclass of assertions" and "individual-to-class" and "class-to-class" OP assertions, are employed to declare overall specifications of CC concepts (e.g., to specify overall cloud companies affiliations and relationships, delivery/deployment models, service characteristics and features information). Thirdly, additional CC concepts specifications (e.g., to clarify specific details of the individuals) are revealed by using Property assertions. These involve both DP assertions and "individual-to-individual" OP assertions.

This way, the axiom assertions in LCCSO become logical. To some extent, the Description assertions can be considered as further information of the Annotation assertion of a cloud service, whereas Property assertions can be regarded as extensions of the contents appeared in the Description assertions. Moreover, with flexible and loosely-coupled naming, domain, ranges, classification and OP deployment, a number of classes, individuals and properties are being used for multiple assertion needs where potential redundancy issue can be avoided.

Take virtual server IaaS services as an example, in their service annotations there would be relevant OS provision descriptions as such are typical aspects of the services; therefore, these services could own the "Multiple OS Support" assertion (Description assertion). Likewise, a large number of SaaS services are offered with multiple OS platforms accessibility, which means they could also be asserted with the "Multiple OS Support" assertion. As a consequence, the "OS" class would then consist of all current mainstream OSs involved (for both IaaS and SaaS) here: "Mobile OS" class comprises "Android, IOS, Windows Mobile, Blackberry OS", etc.; "Server/ desktop OS" class comprises different OS platforms, e.g., "Debian", "Fedora", "Gentoo", "Solaris", "Ubuntu", "Windows", etc., where each one has own detailed OS versions (refer to Fig. 1). Then, in order to clearly present the differences between the two "Multiple OS Support", OP "supports VM OS of" is used to relate the IaaS services to their achievable OSs whilst OP "offers management application for OS of" is adopted to relate the accessible OSs with the SaaS services.

IV. CLOUD SERVICE EXLORER SYSTEM OVERVIEW

As depicted in Fig. 2, CSE prototype employs LCCSO as the formal and consistent knowledge source base to provide cloud service discovery and retrieval functions. The system consists of three main components, namely, Ontology Manager, Service Search Engine, and User Interface.

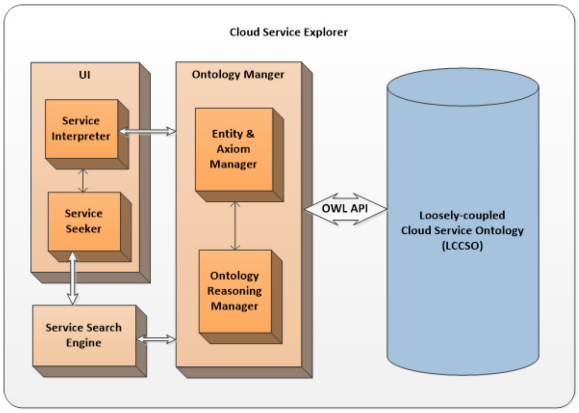

Figure 2. Cloud Service Explorer system architecture.

While the role of **LCCSO** is to provide comprehensive raw data of cloud services and other relevant CC concepts, CSE interprets it through OWL API [16] and translate the information into various general and detailed service descriptions.

Ontology Manager manages raw data extraction from LCCSO. It incorporates Entity and Axiom Manager and Ontology Reasoning Manager, which together manage the ontology parsing tasks. While **Entity and Axiom Manger** reads all types of asserted ontology concepts and axioms, including individual, class, OP and DP concepts, class, and OP and DP assertions, **Ontology Reasoning Manager** handles ontology consistency checks and inference controls by importing OWL2 ontology reasoner. Here, FaCT++ is chosen due to its faster reasoning process and better syntax and property characteristics support than other reasoners such as HermiT and Pellet [17]. Here, to take advantage of ontological modeling, any new knowledge discovered after reasoning process (inferred axioms) is also be used for service discovery and explore.

Service Search Engine accepts user's keywords or/and filters entries input to provide service/concept search functions. Through Ontology Manager, it extracts asserted service descriptions, attributes and other data details and analyzes such against users' input. As the information pairing process is complete, the component outputs a list of relevant cloud services User Interface.

User Interface comprises Service Interpreter and Service Seeker which interact with system users while they search/view cloud services. **Service Interpreter** translates the raw ontology axioms into naturally described contents so that they can be easily understood. **Service Seeker** interacts with Service Search Engine and manages the display of search keywords, filters and service list result.

V. CLOUD SERVICE SEARCH AND RETRIEVAL

CSE prototype is implemented in Java. Currently, with specifications of approximately two hundred cloud services and companies/providers available in LCCSO, it can provide flexible service search, logical service property

 ISBN: 978-1-61208-338-4

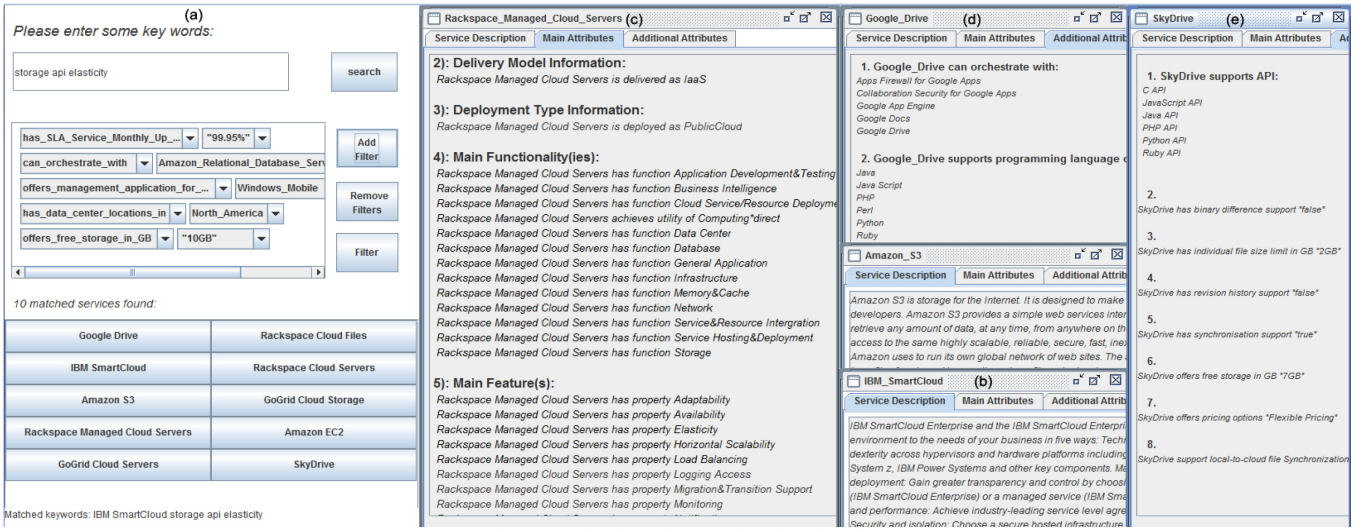

Figure 3. Cloud service search and retrieval.

view and effortless service comparison functions regardless of distinct service models, functions, or user expertise levels.

A. Cloud service search

In CSE, the two search options offered keywords and filters. Basically, system users can enter keywords first and then add filters to view the matched service result, or vice versa. The search requirements can be of any service functions, categories, levels, characteristics, features, functional or non-functional properties, etc., which achieves a flexible and user-friendly cloud explorer experience regardless of the expertise level of users.

The search and filter tasks are implemented by calling Ontology Manager to walk through the ontology to collect entire service properties and data of all cloud services for keyword/filter matches. Here, any services involved in those asserted/inferred axioms where certain information fits the keywords/filters are extracted for shown in the search result.

The purpose of the service filter mechanism is to enhance service discovery experience, especially for those users of limited CC knowledge. It allows them to view and select from a comprehensive list of service properties and property values (so that applicable cloud services can be extracted). The service data options are automatically retrieved from LCCSO, shown in the form of drop down lists.

As depicted in Fig. 3(a), by searching with example keywords "storage, api and elasticity" plus restrictions of a series of filters, a number of applicable cloud services are returned. Users can select them to view and compare the comprehensive service information stored in LCCSO.

B. Cloud service retrieval

As a system user attempts to retrieve cloud services, Service Interpreter reads the raw collected cloud service specification data from Ontology Manager and categorizes the interpreted service information into "Service Description", "Main Attributes" and "Additional Attributes" three categories. As seen in Fig. 3, they are displayed in three switchable tabs.

"Service Description" tab outlines the general descriptions of cloud services by extracting the annotations of them. To ensure the availability and reliability of the contents, such information is collected from the services' official resources. As depicted in Fig. 3(b), the descriptions of Amazon S3 [18] and IBM SmartCloud [19] enable effortless information access which can effectively help users understand the basics of the services.

"Main Attributes" tab comprises a service's overall service properties, e.g., the delivery and deployment model, the primary designed service functions and other additional usages, plus the service characteristics and features. These are in fact the translations of the service's class assertions in the ontology. Using Rackspace Managed Cloud Servers [20] as an example (see Fig. 3(c)), the tab comprehensively illustrates a series of information: 1) "Main Functionalities": due to the fact that the service is seen an IaaS service which provisions virtual servers for general computing needs, it can be used for various other functionalities across multiple function categories, e.g., using such for storage and database provision as well as for application development and deployment. The arrangement helps users understand the ultimate capability of a cloud service on top of its main designed function. 2) "Main Features": the collected service characteristics and features are displayed here, regardless of whether functional or non-functional. These help users view the full picture of a cloud service clearly.

"Additional Attributes" tab displays the various additional service data by examine both cloud service's individual-to-individual OP and DP assertions in LCCSO. Due to the individual assertions for cloud services, they can have a variety of relationships among each other plus other CC concepts individuals. For instance, a service "can orchestrate with" another and "supports OS/programming language/API of" certain OSs/programming languages/APIs. As demonstrated in Fig. 3(d), Google Drive [21] can

 ISBN: 978-1-61208-338-4

orchestrate with a number of services as listed, whereas the supported programming languages are also given. On the other hand, cloud services are asserted with a series of DP axioms. They involve clarifying a range of detailed service data: VM services have a series of VM-specific data, i.e., virtual CPU frequency, hard drive capability, memory size, network through output, etc.; storage services own a number of storage-specific data, i.e., free storage, pricing, certain particular feature supports, etc. An example of SkyDrive's [22] additional attributes is illustrated in Fig. 3(e).

VI. CONCLUSION AND FUTURE WORK

The growing number and complexity of cloud services bring us numerous advantages whilst the issues exposed while searching and retrieving the services are becoming increasingly critical. Despite of the many efforts made on the semantic specification of cloud computing/services and service discovery/recommendation approaches, current service models and recommendation systems cannot work effectively on various service and usage scenarios to provide comprehensive service search and retrieval functions. This paper presented the design and implementation of the LCCSO model and the CSE tool, which together can facilitate effective service discovery and service information access regardless of service's functions, levels, categories, or characteristics. The novel LCCSO model employs flexible concepts naming and the full range of OWL2 axiom assertion types which ultimately comprise diverse types of service specification data and information, whereas the CSE prototype tool adopts a user-friendly design that enables effortless service search and retrieval for all CC user regardless of different level of expertise.

In future work, we intend to enhance the model by focusing on the granular details of unique cloud service properties (e.g., agility, elasticity). Moreover, we are experimenting on additional tool functions, including service rating, review, benchmarking and unified service access portals.

REFERENCES

[1] R. Buyya, C. Vecchiola, and S. Thamarai, "Master cloud computing: Foundations and applications programming", Elsevier, Waltham, USA, pp. 3-27, 2013.

[2] J. M. S. Orozco, "Applied ontology engineering in cloud services, networks, and management systems", Springer Science+Business Media, pp. 23-52, 2012.

[3] J. Shen, G. Beydoun, G. Low, and L. Wang, "Aligning ontology-based development with service oriented systems", Future Generation Computer Systems, vol. 32, 2014, pp. 263-273.

[4] M. Horridge, "A practical guide to building OWL ontology's using protégé 4 and CO-ODE tools", The University of Manchester, edition 1.3, available at: http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf, 2011 [retrieved: May, 2014].

[5] R. Aversa, et al., "An ontology for the cloud in mOSAIC", Cloud Computing Book 2010: Cloud Computing: Methodology, System, and Applications, CRC Press, pp. 467-486, 2011.

[6] M. A. Rodríguez-García, R. Valencia-García, F. García-Sánchez, and J. J. Samper-Zapater, "Creating a semantically-enhanced cloud services environment through ontology evolution", Future Generation Computer Systems, vol. 32, 2014, pp. 295-306.

[7] A. Tahamtan, S. A. Beheshti, A. Anjomshoaa, and A. M. Tjoa, "A Cloud Repository and Discovery Framework Based on a Unified Business and Cloud Service Ontology" IEEE World Congress on Services (SERVICES), 2012, pp. 203-210.

[8] T. B. Pedersen, D. Pedersen, and K. Riis, "On-demand multidimensional data integration: toward a semantic foundation for cloud intelligence", The J. Supercomputing, vol. 65, no. 1, 2013, pp. 217-257.

[9] J. Kang and K. Sim, "Cloudle: A Multi-criteria Cloud Service Search Engine," IEEE Asia-Pacific Services Computing Conference (APSCC), 2010, pp. 339-346.

[10] G. Manno, W. W. Smari, and L. palazzi, "FCFA: A semantic-based federated cloud framework architecture", International Conf. High Performance Computing and Simulation (HPCS), 2012, pp. 42-52.

[11] M. Zhang, et al., "An ontology-based system for Cloud infrastructure services' discovery", 8th International Conf. Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012, pp. 524-530.

[12] T. Han and K. M. Sim, "An Ontology-enhanced Cloud Service Discovery System", International Multi-Conf. Engineers and Computer Scientists (IMECS), available at http://www.iaeng.org/publication/IMECS2010/IMECS2010_pp644-649.pdf, 2010 [retrieved: May, 2014].

[13] M. A. Rodríguez-García, R. Valencia-García, F. García-Sánchez, and J. J. Samper-Zapater, "Ontology-based annotation and retrieval of services in the cloud", Knowledge-Based Systems, vol. 56, 2014, pp. 15-25.

[14] K. Tserpes, F. Aisopos, D. Kyriazis, and T. Varvarigou, "A Recommender Mechanism for Service Selection in Service-oriented Environment", Future Generation Computer Systems, vol. 28, no. 8, 2012, pp. 1285-1294.

[15] D. C. Marinescu, "Cloud computing: Theory and practice", Elsevier, Waltham, USA, pp. 339-356, 2013.

[16] M. Horridge and S. Bechhofer, "The OWL API: A Java API for OWL Ontologies Semantic Web Journal 2(1)", Special Issue on Semantic Web Tools and Systems, 2011, pp. 11-21.

[17] D. Tsarkov and I. Horrocks, "FaCT++ description logic reasoner: system description", 3ird international joint conf. Automaitc Reasoning, 2006, pp. 292-297.

[18] Amazon Web Services, Inc., "Amazon S3 Developer Guide", available at: http://awsdocs.s3.amazonaws.com/S3/latest/s3-dg.pdf, 2013 [retrieved: May, 2014].

[19] IBM Corporation, "IBM SmartCloud Entry user guide, version 3.2", available at: https://www.ibm.com/developerworks/community/wikis/form/anonymous/api/wiki/e6a8193b-bd0a-4c69-b641-328d20a4f69c/page/4cd16bdc-e3cb-4d71-948e-46f0e6548561/attachment/816e4610-008d-487a-9764-a60399ab6e1b/media/SCE_User_Guide_32.pdf, 2013 [retrieved: May, 2014].

[20] Rackspace, Inc., "Next Generation Cloud Servers Developer Guide", available at: http://docs.rackspace.com/servers/api/v2/cs- devguide/cs-devguide-latest.pdf, 2013 [retrieved: May, 2014].

[21] Google, Inc., "About Google Drive", available at: http://www.google.com/drive/about.html, 2013.

[22] Microsoft, "Live SDK developer guide", available at: http://msdn.microsoft.com/en-us/library/live/hh243641.aspx, 2013 [retrieved: May, 2014].

 ISBN: 978-1-61208-338-4