Methodology for Detection of Cloud Interoperability Problems

Preliminary Communication

Darko Andročec
dandrocec@foi.hr

Neven Vrček
nvrcek@foi.hr

University of Zagreb,
Faculty of Organization and Informatics, Department of Information System Development
Pavlinska 2, 42000 Varaždin, Croatia

Abstract *– Interoperability problems between cloud providers are one of the most serious issues of this new computing paradigm. A methodology is needed to systematically and effectively find and solve interoperability problems. For these reasons, a new methodology with detailed steps to find and solve interoperability problems is proposed here. This new methodology is focused and implemented on the platform as a service model, but it can be used in any of the three main models of cloud computing. The methodology uses an iterative approach, because platform as a service offers and their application programming interfaces evolve and change very often. The focus of the methodology is to use remote cloud application programming interfaces to solve interoperability problems on technical, storage and services levels, respectively. Finally, we show the application of the methodology to achieve service-level interoperability among different providers of platform as a service.*

Keywords *– cloud interoperability, methodology, ontology, service composition*

1. INTRODUCTION

Cloud computing is nowadays becoming a popular paradigm for the provision of computing infrastructure that enables organizations to achieve financial savings. On the other hand, there are some known obstacles, among which vendor lock-in stands out. The aforementioned problem is characterized by time-consuming and costly migration of applications and data to alternative cloud solutions offered by different vendors, the inability or limited ability to use some computing resources, applications or data outside the selected cloud computing service and dependence on a specific programming language used by the selected cloud computing vendor.

Numerous heterogeneities among different vendors make cloud interoperability an interesting and complex research and practical problem. A methodology is needed to systematically and effectively find and solve interoperability problems. Currently, there is still no methodology that aims at identification and resolution of interoperability problems, either among APIs of commercial platforms as a service or among cloud offers in general. Most relevant similar interoperability methodologies are explained in the related work section of this paper. For these reasons, a new methodology with detailed steps to find and solve interoperability problems is proposed here. This new methodology is focused and implemented on platform as a service, but it can be used in any of the three main cloud computing models. The methodology uses an iterative approach, because platform as a service (PaaS) offers and their APIs evolve and change very often. User's interoperability requirements also change over time and new interoperability problems might arise. This paper focuses on the use of remote PaaS APIs to solve interoperability problems on technical, PaaS storage and services levels, respectively. Other levels of interoperability (for example, legal and organizational level) cannot be solved by using remote APIs, and they are not the subject of this paper and the proposed methodology.

This paper is organized as follows. First, in Section 2, the related work is listed. In Section 3, we proposed the cloud interoperability methodology. Section 4 shows the practical application of the proposed methodology. Our conclusions are provided in the final section.

2. RELATED WORK

Several European research projects used to deal with cloud computing interoperability. The main objective of the FP7-funded Cloud4SOA project [1] was to semantically interconnect heterogeneous platform as a service (PaaS) solutions sharing the same technology (programming languages and frameworks). The main research objectives were as follows: design of a semantic interoperability framework, introduction of a reference architecture to interconnect different clouds and development of the Cloud4SOA system. This project dealt with semantic interoperability at platform level.

The European project mOSAIC [2] aimed at developing the open-source cloud application programming interface. Petcu et al. [3] presented the mOSAIC architecture and its various usage scenarios. The aforementioned FP7

project covers three basic business scenarios for using multiple clouds: switch to the cloud (application developers or their clients should easily change the cloud provider); service brokerage (finding the best cloud services for a certain application); and development of cloud applications. The concepts of mOSAIC cloud ontology were identified by analyzing standards and the existing literature on cloud interoperability and integration.

The FP7-funded Contrail project [4] aimed at designing an open source system for cloud federations. Contrail developed a software stack that enables a federation (combines services from different cloud providers), identity management (federated identity management to use all services from different cloud vendors), federated service level agreements (defined by a user and translated into requirements by the system), a cloud file system, and an interoperability layer that facilitates infrastructure management and application deployment. It describes four use cases that represent a diverse set of requirements: distributed provision of geo-referenced data which is an implementation of a 3D Virtual Tourist Guide (VTG service), a multimedia processing service marketplace that will exploit a Contrail federated cloud to develop a marketplace offering multimedia services to end-users, scientific data analysis that will archive climate model output data and neutron scattering, and an electronic drug discovery use case that plans to use modern bioinformatics tools/ applications on a federated cloud system.

The main aim of the Vision Cloud project [5] was to solve data management conflicts in cloud federations and multi-clouds. A federated cloud assumes a formal vendors' agreement, while the term multi-cloud [6] denotes the usage of multiple independent clouds. Five areas of innovation in the VISION Cloud platform [5] include the following: data objects are enriched with detailed metadata, data lock-in should be avoided, computations are put close to the data, efficient retrieval of objects is enabled, and strong QoS guarantees, security and compliance with international regulations are guaranteed. Real-world scenarios driving the Vision Cloud FP7 project are: SAP – Business intelligence on-demand (Vision Cloud is used for storage, data mobility and data federation), Telco use cases (telecommunication operators want to offer data-intensive applications with the high quality of service), media use cases (videos in clouds), and healthcare use cases (personalized healthcare applications based on patient health records).

The primary goal of the REMICS project [7] is to transform legacy systems into UML models and to manipulate these models to migrate applications to clouds. REMICS extracts the architecture of the legacy application, analyzes it and finds out how to modernize it. This information is converted into models that represent the start of the migration activity. Researchers working on the REMICS project defined a methodology [8] for the migration of legacy systems to clouds. Their methodology consists of the following activities: requirements and feasibility (to gather migration requirements), recovery (to get the application model of the legacy application), migration (to migrate to the cloud), validation (to define a testing strategy), supervision (to control the performance of the system), interoperability (to solve interoperability problems), and withdrawal (to stop the service).

There are also some interoperability methodologies in the existing literatures that are mostly concerned with enterprise interoperability. The ATHENA Interoperability Methodology (AIM) [9] is an extension of the Unified Software Development Process (UP) [10] which introduces a group of interoperability activities. AIM is used to identify interoperability issues and select the adequate ATHENA solutions. Chen and Daclin [11] proposed four main interoperability methodology phases, i.e., definition of interoperability objectives and needs, analysis of the existing system to identify interoperability barriers and measure the current interoperability level, select and combine solutions, and implementation and testing.

Sanati et al. [12] presented their E-service Integration Methodology (E-SIM) to solve complex interoperability problems and configure service workflow. The tasks in this methodology include specification of life-event service user requirements, specification of interoperability requirements at business process, data, and interface levels, detailed design of e-government services, design and implementation of Semantic Web specifications.

The European Interoperability Framework (EIF) [13] addresses interoperability of European public services at four identified interoperability levels, i.e., legal, organizational, semantic, and technical. The involved public organizations should make interoperability agreements for each level, such as agreements on the transposition of European directives to national legislation, SLAs, reference taxonomies, code lists, data dictionaries, interface specifications, data formats, etc. Interoperability agreements specify one or more interoperability solutions implemented by one or more interoperability solution instances. Due to environment changes, interoperability of European public services is a continuous task.

When dealing with the composition of web services, a dominant interoperability problem is how to map the inputs and outputs of involved services, and in most cases, data mediation is required to achieve interoperability among web services [14]. This problem has been addressed in many papers and books. Nagarajan et al. [14] proposed a data mediation architecture that uses WSDL-S for mapping from inputs and outputs to common ontology and vice versa. The web services should be semantically annotated by using WSDL-S, and the mapping engine was used to transform SOAP messages according to defined XSLT or XQuery mappings. WSDL-S later became the main input for W3C recommendation SAWSDL that provides a similar data mediation mechanism. The main contributors to the SAWSDL

standard were members of the METEOR-S research project and IBM [15]. Sheth et al. [15] claim that the key benefit of SAWSDL is systematic data mediation, where XSLT is used to map a service schema to ontology (lifting schema mapping) and vice versa (lowering schema mapping). Klímek and Necaský [16] introduced a model-driven method to automatically generate XSLT for lifting and lowering schema mappings and its prototype implementation.

Li et al. [17] presented an approach to reconciliate semantic conflicts in the composition of web services. They used COIN ontology, SAWSDL and mapping algorithms to handle complex differences by using minimal numbers of predefined transformations. The method to automatically analyze data flows of the BPEL process and automatically determine possible semantic differences is also shown in the same paper. Stollberg et al. [18] proposed a mediation model for Semantic web services using WSMO mediators at data, functional, and process levels, respectively.

There is currently still no methodology that aims at identification and resolution of interoperability problems, either among APIs of commercial platforms as a service or among cloud offers in general. The only existing methodology that takes into consideration cloud interoperability problems is the methodology developed by the REMICS consortium [8], but its main purpose is to provide a model-driven approach to migrate the legacy application on software as a service.

Part of the methodology that addresses interoperability deals with finding possible interoperability problems for the future migrated system, and with building interoperability components in migrated software when needed. It does not consider interoperability problems between different cloud providers. In terms of the REMICS methodology, interoperability is modeled as one of five technical practices with five tasks, i.e., identification of interoperability problems/scenarios, definition of interoperability requirements, interoperability analysis, implementation of interoperability components and interoperability monitoring [8].

3. PROPOSED METHODOLOGY

Based on a literature review and the service-level cloud interoperability use case [19], the new methodology for the detection of interoperability problems among different providers of platform as a service was developed. This methodology uses semantic web annotations, semantic web services, ontology and the AI planning method to detect and solve common interoperability problems. Remote PaaS API operations are used to execute interoperability actions. The proposed methodology has five main steps: Requirement identification; Interoperability analysis; Solution design; Solution implementation; and Evaluation. The steps of the proposed methodology and their main activities are listed in Table 1. The methodology uses an iterative approach, because platform as a service (PaaS) offers and their APIs evolve and change very often, so we can repeat these steps over time.

The most important interoperability needs of users should be listed in the first step, i.e., interoperability actions such as migration of data from one PaaS offer to another cloud storage, working with external cloud data in PaaS applications, communication between two applications deployed on different PaaS offerings, composition of two or more API operations of different providers, etc. These actions can be derived from the available use cases presented in technical and research papers, deliverables of related projects, and proposals for cloud standards, where the authors have already done some research on user's interoperability requirements. Based on the identification of relevant interoperability actions, adequate use cases should be defined and described.

Interoperability analysis deals with identifying levels of interoperability problems and reasoning on possible interoperability problems between different commercial providers of platform as a service. This step starts with studying the existing literature with the aim of finding the most important known interoperability problems for a given context. The systematic mapping study or systematic review methods can be used to perform the said review. The final result of the review will be identification of levels of interoperability problems and specific problems on each level. Next, ontology of interoperability problems should be developed by using the chosen ontology development methodology such as Ontology Development 101 [20].

Technical and semantic interoperability issues among commercial providers of platform as a service can be derived from database interoperability problems, metadata interoperability problems, interoperability problems of web services and problems identified by different interoperability frameworks. In the platform as a service context, the following levels of interoperability problems were determined: legal, organizational, service level, application level, and storage level. At legal level, we differentiate the following main problems: different countries with different data privacy laws, data sovereignty interoperability problems, and cloud data ownership issues. However, incompatibilities at legal and organizational level cannot be solved by using cloud providers' remote APIs, and are not the focus of this paper. At service level, the main problems are operations and parameter level differences that arise because two semantically similar API operations or parameters are represented at different levels of abstraction, differences that arise because of using different descriptions of semantically similar API operations, some necessary API operations can be missing from cloud vendor's remote APIs, and there can be differences that arise because different descriptions for semantically similar parameters are used. At data (storage) level, the main interoperability problems are as follows: aggregation is used in one cloud storage

to represent a set of entities in another cloud storage, the same entity can be modeled as an attribute in one cloud storage and a data container in another storage, two entities can be represented at different levels of generalization in various cloud storages, semantically unrelated entities might have the same name in different cloud storages (homonyms), semantically similar entities can be named differently in different PaaS storages (synonyms), some names are reserved and forbidden, and some types of names can be required (e.g., Salesforce requires that you name your custom object with postfix __c), different means to define relationships between two data containers (e.g., foreign key, no relationship between data containers, etc.), or maybe some cloud storage does not have any means to connect two data containers, two attributes might have different default values in different cloud storages, different cloud storages support different data types, and differences between cloud data models.

Solution design prepares the whole architecture. It includes activities such as the development of the ontology of resources, remote operations and data types [21], definition of the semantic web service, needed mappings and transformations, and defining the AI planning domain. Remote operations of commercial platform as a service, their data types and mappings are modeled by means of the ontology of resources, remote operations, and data type mappings. However, the landscape of cloud APIs is changing constantly, and the ontology should be upgraded over time. The refinement of the ontology is mandatory when users detect important changes in APIs of included providers and when they want to add a new cloud provider to its new remote functions, data types and new mappings. Next, the language for semantic web services is selected, and after that semantic web services are created by annotating operation, inputs and outputs, data types and needed mappings and transformations. Finally, an AI planner is chosen, and the planning domain is created by taking into account interoperability actions chosen in the previous steps.

Solution implementation deals with approach implementation and execution of the defined use cases. The initial state and goal for the AI planner are generated programmatically based on the chosen interoperability action, semantic annotations, the ontology, and defined mappings and transformations. The interoperability tool is developed or upgraded; the AI planner is executed to get a plan or list detected interoperability problems. If there is a suitable plan, appropriate service compositions are executed, taking into account possible mappings and transformations of inputs and outputs of different services representing remote APIs or composite service consisting of more remote APIs with additional logic. The evaluation step evaluates successful execution of use cases and correct identification of possible interoperability problems. If some problems are found, the AI domain and problem definitions, the interoperability tool and semantic annotations should be inspected and errors should be eliminated. Additionally, it is useful to evaluate developed ontologies by using known ontology evaluation techniques and methods.

Table 1. Steps and activities of the proposed methodology.

Step	Activities
1. Requirement identification	1.1 Choose a cloud model 1.2 Study the existing use cases 1.3 Identify relevant interoperability actions 1.4 Define use cases
2. Interoperability analysis	2.1 Review the existing literature on interoperability problems 2.2 Identify levels of interoperability problems 2.3 Identify specific interoperability issues at each level 2.4 Choose ontology development methodology 2.5 Create ontology of interoperability problems
3. Solution design	3.1 Create ontology of resources, remote operations, and data types 3.2 Choose language for semantic web services 3.3 Create mappings and transformations 3.4 Associate mappings and transformations to the appropriate elements of services 3.5 Choose the AI planner 3.6 Define the AI planning domain 3.7 Define algorithms for finding interoperability problems
4. Solution implementation	Use cases execution: 4.1 Implement needed web services to invoke remote APIs 4.2 Generate the AI planning problem based on semantic annotations, ontology and user choice 4.3 Develop or modify/upgrade an interoperability tool 4.4 Get a suitable plan from the AI planner or find interoperability problems 4.5 Execute service composition
5. Evaluation	5.1 Evaluation of ontologies 5.2 Validation of execution of use cases

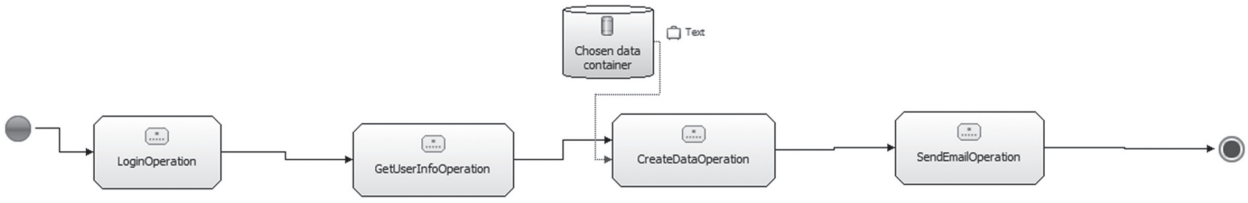

Fig. 1. API operations executed in use case 2

4. METHODOLOGY APPLICATION

All listed steps and activities were performed on the platform as a service model and two use cases: migration of data between different PaaS storages and adding a user to the application deployed on other PaaS offers. These two use cases were constructed to illustrate how PaaS storage interoperability and service-level interoperability can be solved by using this approach and remote APIs of PaaS providers.

In the first use case, data was migrated between different providers of platform as a service. Successful execution of more complex interoperability scenarios cannot be imagined without being able to move data from one PaaS vendor to another. The majority of vendors' API operations deals with data manipulation and management, so the first use case is also important to learn more about the mentioned APIs in practical problems. We have chosen the three prominent platform as a service providers (Microsoft, Google and Salesforce) and managed to migrate simple data (three tables) from one to another provider using the presented interoperability methodology at data level. We have determined the differences in data and application models between chosen commercial vendors of platform as a service. At Force.com platform, data objects are called custom objects (similarly to tables in databases). The Google App Engine Datastore holds data objects named entities; each entity has one or more properties of one of the supported data types and each entity is identified by its kind and key. Finally, Microsoft's Azure SQL Database is based on SQL Server technology and it provides a relational database for the Azure platform. Google App Engine, Microsoft Azure, and Salesforce do not allow applications deployed on their cloud to directly access external databases (other than their predefined ones that are part of their platform as a service offer or one of their other cloud storage options). The external data can only be accessed using REST or SOAP web services.

In the future, we plan to build an architecture for data migration among PaaS providers that uses data ontology (OWL is an intermediate data format) and data type mappings stored as individuals in PaaS ontology. Validation of the first use case and the data migration architecture will be done by migrating a more complex set of data (to be more specific, data of an open-source content management system) and manually checking all of the migrated data elements.

Methodology application at service level is shown in detail in paper [19]. In this paper, we tried to identify and address service-level interoperability issues when using APIs from different commercial providers of platform as a service. First, we have defined a use case to add current user information from one platform as a service offer to the application hosted on another offer. To address interoperability problems, ontology driven

Table 1. Steps and activities of the proposed methodology.

Use Case ID:	UC-2
Use Case Name:	Add the existing user to other PaaS providers
Actors:	PaaS application administrator, PaaS provider 1, PaaS provider 2
Description:	This use case shows how to add a user from one PaaS offer to an application hosted on another PaaS. PaaS administrator specifies the data container of target PaaS where user information will be stored. Adequate schema mapping files should also be created. Finally, an e-mail is sent to the PaaS application administrator a new user is added to.
Trigger:	This use case is initiated by the PaaS application administrator when he decides that he wants to add the existing user information (from other PaaS offer) to the PaaS application he manages.
Preconditions:	1. The user required to migrate must be logged-in on the source PaaS offer 2. The PaaS application administrator must be able to put data into the data container with user information on the target PaaS offer
Post Conditions:	1. The existing user from the source PaaS offer is added to the application hosted on the target PaaS offer, an e-mail is sent to the PaaS application administrator
Normal Flow:	1. The PaaS application administrator selects the source and connections of the target PaaS offer and specifies the name of the data container where user information is stored for target application 2. The PaaS application administrator initiates user migration 3. Input/output mappings are performed, appropriate web services are called, the user is added to the application stored on the target PaaS offer, and an e-mail on the new user is sent to the administrator
Exceptions:	1. If there is a problem with connection to the chosen source or target PaaS offers, the exception is raised and an error message is shown 2. If the system detects the interoperability problem during the planning phase or service execution, the action is stopped and detected interoperability problems are shown in user interface
Special Requirements:	This use case should validate API service level interoperability, using ontology based data mediation and lifting and lowering schema as defined in SAWSDL.
Assumptions:	The PaaS user understands English.

data mediation is used and tested in this use case. Remote vendors' APIs are implemented as web services. Resulting web operations and their inputs/outputs are semantically annotated by using cross-PaaS concepts from the developed platform as a service OWL ontology. Next, SAWSDL and XSLT are used to define service type mappings. The actual composition of platform as a service APIs is implemented by means of the AI planner and the developed Java web application. Testing and validation are performed on a case where the current Salesforce user is added to the data container in the Vosao content management system deployed on Google App Engine [19]. The flow of cloud API operations with operation names as defined in the ontology is shown in Figure 1. The use case is described in Table 2. The service annotated with *GetUserInformation* has output *UserInfoType* and provides basic information on the user logged in a specific PaaS offer. Its output is used by other two operations (*CreateDataOperation* and *SendEmailOperation*) to create a data object in other PaaS offer, and send an e-mail to the application administrator that a new user has been added. We have executed a composite operation by using a prototype we developed, "an AI planner has successfully found the plan, CXF interceptor class and service data mapping and transformation were successfully finished, and web services defined in composition were successfully invoked. *UserEntity* was successfully created with the appropriate properties for username, name and email filled-in. Finally, the email message was sent to test e-mail representing mail of the application administrator" [19].

5. CONCLUSION

Interoperability problems between cloud providers are one of the most serious issues of this new computing paradigm. For this reason, a methodology is needed to systematically and effectively detect and solve interoperability problems. Currently, there is still no methodology that aims at identification and resolution of interoperability problems, either among APIs of commercial platforms as a service or among cloud offers in general. Our proposed methodology is focused and implemented on platform as a service, but it can be used in any of the three main models of cloud computing. The methodology uses an iterative approach because platform as a service (PaaS) offers and their APIs evolve and change very often. The user interoperability requirements also change over time and new interoperability problems could arise.

The plan for the future is to apply the proposed methodology to additional use cases regarding PaaS interoperability, by using other AI planners (for example, some contingent planner to address non-determinism of the domain), and try to apply it to other two cloud computing models (infrastructure as a service and software as a service). Hopefully, other researchers will find this methodology useful and apply it in their research.

6. ACKNOWLEDGEMENT

This work has been fully supported by the Croatian Science Foundation under the project IP-2014-09-3877.

7. REFERENCES:

[1] E. Kamateri, N. Loutas, D. Zeginis, J. Ahtes, F. D'Andria, S. Bocconi, P. Gouvas, G. Ledakis, F. Ravagli, O. Lobunets, K. A. Tarabanis, "Cloud4SOA: A Semantic-Interoperability PaaS Solution for Multi-cloud Platform Management and Portability", Service-Oriented and Cloud Computing, Vol. 8135, pp. 64–78, Springer Berlin Heidelberg, 2013.

[2] D. Petcu, C. Sandru, B. Di Martino, S. Venticinque, M. Rak, R. Aversa, M. Ficco, G. Cretella, D 1.1 - Architectural Design of the mOSAIC's API and Platform, http://www.mosaic-cloud.eu/index.php?option=com_chronocontact&Itemid=186 (accessed: January 16, 2016)

[3] D. Petcu, C. Craciun, M. Neagul, I. Lazcanotegui, M. Rak, "Building an interoperability API for Sky computing", Proceedings of the 2011 International Conference on High Performance Computing and Simulation, Istanbul, Turkey, 4-8 July 2011, pp. 405–411.

[4] J. Jensen, P. Dazzi, P. Mori, P. Kershaw, I. Johnson, M. Coppola, A. Lazouski, F. Martinelli, "The CONTRAIL approach to Cloud Federations", Proceedings of the International Symposium on Grids & Clouds 2012, Taipei, Taiwan, 26 February - 2 March 2012, pp. 1–14.

[5] S. V. Gogouvitis, G. Kousiouris, G. Vafiadis, E. K. Kolodner, D. Kyriazis, "OPTIMIS and VISION Cloud: How to Manage Data in Clouds", Euro-Par 2011: Parallel Processing Workshops, Vol. 7155, pp. 35–44, Springer Berlin Heidelberg, 2012.

[6] N. Grozev, R. Buyya, Inter-Cloud architectures and application brokering: taxonomy and survey, Software: Practice and Experience, Vol. 44, No. 3, 2012, pp. 369–390.

[7] P. Mohagheghi, T. Sæther, "Software Engineering Challenges for Migration to the Service Cloud Paradigm: Ongoing Work in the REMICS Project", Proceedings of the IEEE 7th World Congress on Services, Washington, DC, USA, 4-9 July 2011, pp. 507–514.

[8] G. Benguria, P. Mohagheghi, Y. Gomez, C. Hein, B. Morin, Deliverable D.2.2 - REMICS Methodol-

ogy, http://www.remics.eu/system/files/REMICS_D2.2_V1.0.pdf (accessed: January 16, 2016)

[9] ATHENA Consortium, Interoperability methodology, http://athena.modelbased.net/methodology/index.pdf (accessed: January 16, 2016)

[10] I. Jacobson, G. Booch, J. Rumbaugh, "The Unified Software Development Process", Addison-Wesley, 1999.

[11] D. Chen, N. Daclin, "Barriers Driven Methodology for Enterprise Interoperability", Establishing the Foundation of Collaborative Networks, pp. 453–460, Springer US, 2007.

[12] F. Sanati, J. Lu, X. Zeng, "A methodological Framework for E-government Service Delivery Integration", eGovernment Interoperability Campus, Paris, 9 October 2007, pp. 1–9.

[13] European Commission, European Interoperability Framework (EIF) for European public services, http://ec.europa.eu/isa/documents/isa_annex_ii_eif_en.pdf (accessed: January 16, 2016)

[14] M. Nagarajan, K. Verma, A. P. Sheth, J. A. Miller, "Ontology Driven Data Mediation in Web Services", International Journal of Web Services Research., Vol. 4, No. 4, pp. 104–126, 2007.

[15] A. P. Sheth, K. Gomadam, A. Ranabahu, "Semantics Enhanced Services: METEOR-S, SAWSDL and SA-REST", IEEE Data Engineering Bulletin, Vol. 31, No. 3, pp. 8–12, 2008.

[16] J. Klímek, M. Necaský, "Generating Lowering and Lifting Schema Mappings for Semantic Web Services", IEEE Workshops of International Conference on Advanced Information Networking and Applications, Biopolis, Singapore , 22-25 March 2011, pp. 29–34.

[17] W. Li, J. Tordsson, E. Elmroth, "Modeling for Dynamic Cloud Scheduling Via Migration of Virtual Machines", Proceedings of the IEEE 3rd International Conference on Cloud Computing Technology and Science, Athens, Greece, 29 November - 1 December 2011, pp. 163–171.

[18] M. Stollberg, E. Cimpian, A. Mocan, D. Fensel, "A Semantic Web Mediation Architecture", Canadian Semantic Web, Vol. 2, pp. 3–22, Springer US, 2006.

[19] D. Andročec, N. Vrček, P. Kungas, "Service-Level Interoperability Issues of Platform as a Service", Proceedings of the IEEE 11th World Congress on Services, New York City, New York, USA, 27 June - 2 July 2015, pp. 349–356.

[20] N. F. Noy, D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology", Stanford Knowledge Systems Laboratory, Stanford University, USA, Technical Report KSL-01-05, 2001.

[21] D. Andročec, N. Vrček, "Platform as a Service API Ontology", Proceedings of the 12th European Conference on eGovernment, Barcelona, Spain, 14-15 June 2012, pp. 47–54.