Service and Workflow Engineering based on Semantic Web Technologies

Volkan Gezer and Simon Bergweiler

German Research Center for Artificial Intelligence (DFKI)
Innovative Factory Systems
Kaiserslautern, Germany
Email: firstname.lastname@dfki.de

***Abstract*—This paper presents the concept and implementation of a cloud-based infrastructure platform and tailored tools for graphical user interaction. The goal is the creation of a platform that allows users to generate workflows for their experiments in the field of product design and quality assurance without any knowledge of service engineering and the underlying Semantic Web technologies. An experiment is described as workflow and consists of orchestrated services from several vendors that encapsulate specific tasks. One advantage of this approach is the combination of a cloud-based platform with high-performance computing. Services that encapsulates complex calculation procedures can be outsourced to specific servers. This results in tremendous time savings and allows experts to carry out more experiments with products, which was omitted due to the complexity and the required computing power until now. The possibility to conduct these experiments improves the productive know-how of the companies and enhances the products they are selling.**

Keywords–Cloud infrastructure; Semantic Workflows; Semantic Web services; graphical workflow interface.

I. INTRODUCTION

Nowadays, cloud-based solutions are part of the daily life, and their usage is increasing day by day. These solutions increase the mobility of data by allowing access from multiple locations with minimum effort [1]. In this paper, we present the concept and implementation of a flexible cloud-based platform for the vendor-independent integration of Semantic Web services in the engineering domain. This platform is provided as Infrastructure as a Service (IaaS), and is able to combine and orchestrate Web services. Involving semantic technologies inside a cloud-based solution significantly improves usability by structuring the data in a standardized way that these can be understood by machines and humans. The structured data can be utilized to create interoperable and vendor-independent applications, and thereby avoid vendor lock-in problems [2]. The platform enables experts from various application domains to independently plan and execute their experiments with strong calculation procedures, e.g., to check the quality of products and their compliance with construction rules by comparison of 3D-models, or to identify weaknesses and subsequently improve the positive effects of their products. Each experiment is described as workflow that orchestrates services from different vendors. Each service encapsulates specific tasks, calculation procedures or complex sub-systems and require highly scalable computing clusters for their execution within an acceptable time-frame. Therefore, to provide an added value, the developed platform is combined with a cluster of high-performance computers spread across different virtualization solutions, which take over the calculation of complex tasks. This results in enormous time savings and allows experts to carry out more experiments with products, which was neglected due to the complexity of the task and the required computing resources until now. The developed tailored tools of this cloud-based platform, described below as core components, allow engineering companies or software providers to integrate their Software as a Service (SaaS) and orchestrate them in a specific workflow, seamlessly supported by graphical user interfaces and without requiring specific skills or knowledge of the underlying Semantic Web technologies. The developed solution uses standardized Web-based technologies and all workflows can be executed using a Web browser, requiring no additional software. Due to this distributed architecture, the platform offers optimal conditions for both short and long-running experiments.

Section II introduces used technologies and describes the topics under consideration. For a better understanding, Section III describes the requirements in the engineering domain and leads over to Section IV, methodology and concept of the developed approach. The next section describes the architecture and developed core components. The paper ends with a conclusion and an outlook on future work and extensions.

II. BACKGROUND

Web services are designed to support machine-to-machine interaction over a network and allow interoperable communication [3]. With the help of description languages, which will be discussed in the upcoming sections, Web services create communication between peer-platforms, prevent vendor dependency and increase reusability.

A. Web Services Description Language

The Web Service Description Language (WSDL) is a language- and platform independent XML-based interface definition language, designed with the aim to create a standardized mechanism for the description of Web services. It describes SOAP-based Web services in detail, their technical input and output parameters, ports, data types, and how services must be invoked. With this machine-readable description language, the automatic detection and execution of Web services is possible. A ready-revised language draft was submitted to the World Wide Web Consortium (W3C) [4], but only version 2.0 was standardized and proclaimed as W3C recommendation [5]. Unfortunately, WSDL is a lower level interface description language that addresses the technical mechanisms and aspects of Web services, and it does not reflect the functionality of a service. Furthermore, it is difficult to create and understand for humans. In this approach, WSDL is used for the technical description of Web services, their input and output parameters, and the SOAP messaging mechanism.

B. Technologies of the Semantic Web

The development of the current Web to the "Semantic Web" is pervasive. Efforts are aiming to add annotations to things and objects of daily life. Through the help of annotations, the vision of the Semantic Web allows better cooperation between

Copyright (c) IARIA, 2016. ISBN: 978-1-61208-505-0

people and computers; well-defined meanings are attached to information [6]. The Resource Description Framework (RDF) is one of the most important data formats that has been developed to implement this vision. The Semantic Web combines technologies that deal with the description of information and knowledge sources, such as ontologies, RDF triple stores, and Semantic Web services [7] [8]. Ontologies allow the definition of a vocabulary of a dedicated application domain and define for this purpose concepts and properties. These concepts can in turn be connected by relations, which promise a significant value, when conclusions are drawn about these structures. In that field, the W3C defines his recommendations as an open standard like RDF(S) [7][9] and the Web Ontology Language (OWL) [10].

In contrast to a complex and comprehensive infrastructure that tries to solve all problems of the interaction and communication of distributed applications, the Semantic Web Technology Stack, depicted in Fig. 1, is a family of modular standards mostly standardized by the W3C. Each of these standards aims at another part of problem or another sub-problem.

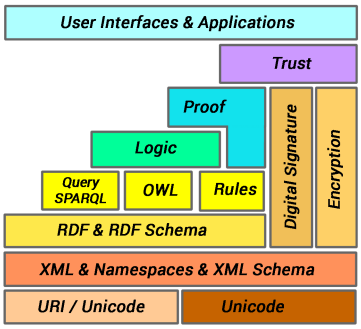

Figure 1. Semantic Web technology stack.

This stack of Semantic Web technologies describes the vision of the W3C to create a Web of linked data. The idea of open data stores on the Web, the ability to build vocabularies, and write rules for handling data based on these empowered technologies, such as RDF, OWL, and the SPARQL Protocol And RDF Query Language (SPARQL) [11].

C. Semantic Web Services

In the recent years, the tendency towards Semantic Web technologies increased the research in the domain, results in an elevated number of available ontologies as well as standards recommended by the W3C. To widen the scope of applicability, one of the submitted ontologies to W3C was the Web Ontology Language for Web Services (OWL-S), which allowed services on the Web to be found, executed, and monitored. The OWL-S ontology is designed on top of OWL with extensions to make service discovery, invocation, composition, and monitoring possible. The provided structure also allowed these operations to be performed autonomously, when desired [12]. Based on the previously described technologies, domain models must be created to form an important conceptual basis. Therefore, parts of the dedicated knowledge domain are categorized and structured in a machine readable form. OWL-S [12] extends this base to a set of constructs that relate to properties, specialties and dependencies of the Web service level and is also machine readable and processable.

A concrete service description in OWL-S is separated in several parts. Fig. 2 shows the main concepts and relations of a service model in OWL-S: service profile, service model, service grounding, and for our approach important, the processes. The *Service Profile* is used for service discovery and describes the functionality of the service and contains information about the service provider. Furthermore, this profile reflects the overall functionality of a service with its precondition, input and output types, features, and benefits.

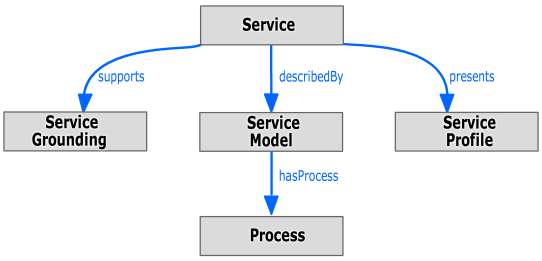

Figure 2. Main Web service concepts in OWL-S.

Any Web services provided with a WSDL description and SOAP interface can be integrated into the infrastructure and converted into semantic descriptions as long as they satisfy the requirements. However, for a Web service that is converted into a Semantic Web service with the help of an upper ontology, the following types of the OWL-S Submission [12] are required:

- The *Service Profile* provides information to describe a service to a requester. The profile provides three types of information: the service creator, the service functionality, and the service characteristics [12].
- The *Service Model* is a mandatory type for the description how a Web service works. The model describes the inputs, outputs, preconditions and effects. It also specifies the *Process* concepts and their execution order. The process description consists of simple atomic processes or complex composite processes that are sometimes abstract and not executable. Each function provided by the service is considered as an *Atomic Process*, whereas combined multiple services are named as *Composite Service*.
- The *Service Grounding* stores the detailed technical communication information on protocols and formats. This concept provides the physical location to the technical description realized in WSDL. This WSDL-file is called when the service is executed, as well as during conversion process to retrieve the technical inputs and outputs of the service.

The listed types provide basis for OWL-S to create relations, which are utilized for improved interoperability. However, the relations generated using only OWL-S ontology form the minimal relationship for services, enough to be operated. If the usage scenario requires involvement of additional relations, these must be defined creating a *meta-ontology* and including it inside a semantic repository [13].

D. Workflows

Services provide functionality for special tasks, but complex tasks in the engineering domain usually consist of multiple steps. Therefore, one service is not sufficient and an orchestration of

Copyright (c) IARIA, 2016. ISBN: 978-1-61208-505-0

services is needed. The complete service chain is described in a *workflow*. Depending on the domain, a *workflow* can have multiple definitions, but in the context of this paper, a workflow will be considered as a set and ordered list of chained up Web services provided with a WSDL file and a SOAP interface to perform a specific task with or without user interaction.

Determined by the complexity, multiple workflows can be necessary to finalize the task. In this case, with the help of semantic descriptions, workflows can be grouped and chained up to create "sub-workflows." Workflows as well as sub-workflows are stored in semantic repositories. Similar to Semantic Web services, they are reusable and their descriptions are updated without causing any fragmentation.

E. Triple Stores

Computational tasks often require collection and storage of the results for further usage. Storage of information without a structured form increases the complexity and the time to access the data, and reducing the flexibility for further modifications and enhancements [14] causing to fragmentation problems. To address this issue, databases play the role as containers, which collect and organize the data for swift future access [15].

Semantic repositories are similar to the database management systems (DBMS) in terms of providing functionality for organization, storage, and querying the data, but differ from them in terms of the type of organization and data representation. Unlike DBMS, semantic repositories use schemata to structure the data thus allow defining the data stored to set relations between. Regarding to data representation, semantic repositories work with flexible and generic physical data models, which allow merging other ontologies "on the fly" and relate the data among merged schemata [16]. As OWL-S is based on OWL, which is built on top of RDF, see Section II-B, the data operations are performed using the same RDF structure. This structure provides descriptions to query the data, and allows optimal extension of relations allowing multiple use. The Sesame framework [17] is one RDF store solution, which can be used in this context. It creates, processes, edits, stores, and queries RDF data, therefore it is chosen to serve as a storage for the framework.

F. Related Work

The Business Process Execution Language (BPEL) is a language for describing and executing business processes in general. It provides an XML-based syntax and allows data manipulation for data processing and data flow. It also allows orchestration of services, after specifying the service set and the service execution order [18].

For the languages OWL-S and BPEL, there exist tools for the automated execution of Web services described in WSDL. They also permit implementations in any programming languages as long as they provide valid WSDL descriptions. Different from BPEL, OWL-S facilitates Semantic Web technologies, which make the structure meaningful for human and machines and allow automated design and orchestration of services, whereas BPEL does not [19].

The execution order of services is usually defined using a design tool (textual or graphical) which is then executed and monitored using an engine. For BPEL, Apache BPEL Designer and JBoss Tools BPEL Editor can be given as examples to design tools, whereas Oracle BPEL Process Manager, Apache ODE, IBM WebSphere Process Server, and Microsoft BizTalk Server can be listed as examples for execution and monitoring.

Using OWL-S increases the interoperability and enables automatic orchestration between the services, but it requires a deep knowledge in the domain. Hence, there are few editors available for OWL-S. However, all of them must be locally installed to be used. To create complex workflows, Protégé OWL-S Editor [20], which is a plug-in for Protégé, can be utilized. Nevertheless, the usage of this plug-in also requires advanced knowledge in the domain. To convert Web services into Semantic Web services, a design tool and an execution engine are necessary.

In another approach, created in the context of the THESEUS funding program, a framework for the discovery, integration, processing, and fusion of Semantic Web services is described [21]. According to a user request, the framework identifies and assembles matching services for problem solving and creates a plan for the composition and execution order. The focus is on the matching of heterogeneous services and the fusion of all gathered information in real time. The harmonizing and mapping of knowledge is carried out based on ontologies.

The advantage of our approach is the continuous integration of services, from the UI to an automatic executable experiment, described as a specific workflow. Within the developed infrastructure, specific services can be deployed and assigned to workflows graphically, without detailed knowledge of the underlying Semantic Web technologies.

III. Scenario

In the engineering domain, a conventional practice for quality assurance of the manufactured final product are comparison checks against the virtual designed product model. This accuracy check is performed by comparing two 3D models. First a scanning process creates and transfers accurate points, and in this way a virtual 3D model is created. The entire model consists of millions of 3D points, which must be matched and compared with the designed product model to find out the discrepancies by calculating the distances of points in both, the designed model and the virtual clone of the final product [22] [23].

The manufacturer of these big turbine blades uses different tools to perform this comparison task and these supplementary tools generate additional license and training costs. The handling of different software solutions requires many hours of work. By using the workflow and service infrastructure and the distributed High Performance Computing (HPC) solution described here, the comparison time is significantly reduced. These advantages allow the company to focus on quality measurement and also increase the capacity of the company for initiating new projects. Fig. 3 shows a complete Kaplan turbine (a), one blade that is to be evaluated (b) and the scanned and virtualized 3D model with color-coded comparison results (c) [23]. The virtual model is created by an open-source tool for rendering and visualization [24].

To perform such a comparison task, the complete workflow has to be formally described. The workflow description defines that two CAD files must be loaded and compared by the help of additional comparison services. These services make compatibility checks, they check whether components of the production environment created by different tools in different formats for Computer-Aided Design (CAD) and Computer-Aided Engineering (CAE) fit together, by comparing compatible

Copyright (c) IARIA, 2016. ISBN: 978-1-61208-505-0

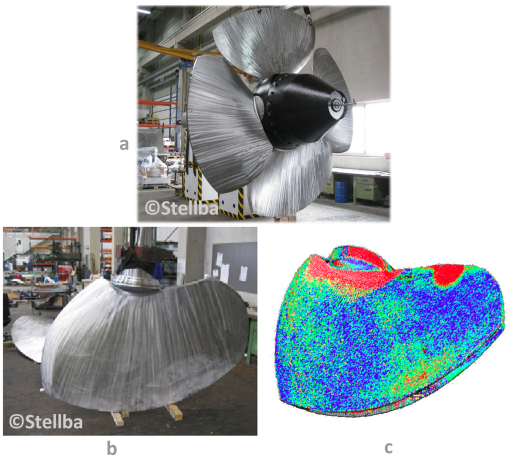

Figure 3. A complete manufactured Kaplan turbine (a), its single turbine blade (b) and the color-coded comparison of its scan and design (c).

heights and dimensions.

IV. CONCEPT

The cloud-based infrastructure allows to orchestrate services individually, formalized by a workflow. To design the workflow, a special tooling is needed, where each step in the workflow or each state change is assigned to a service. To solve this problem, all available technical Web service interfaces needs to be described using standards, such as WSDL. Based on this technical description with its functions, input, and output types, a semantic Web service model, described in OWL-S, is generated and integrated into the semantic repository. This transformation and conversion is automatically performed by the provided converter libraries, shown in Fig. 4. As depicted, the creator retrieves the service type and the URL of the WSDL file and converts the service into a specific instantiated Semantic Web service description that is stored in the semantic repository. Using the cloud-based approach, it is easily possible to execute the service execution task in an HPC cluster, which extremely saves computation time.

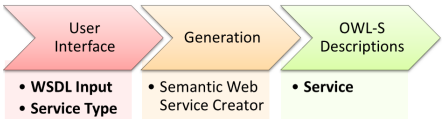

Figure 4. Generation of services in OWL-S.

However, in this approach three service categories must be differentiated:

- Synchronous Service
- Asynchronous Service
- Asynchronous Web Application

These service categories are disjoint, each Web service can only be assigned to one category of the service repository. Standard Web services belong to the *Synchronous Service* category. Whenever a request is made, they must respond within 60 seconds, which is defined as default timeout limit in SOAP. Every service, which does not require an interaction with the user is part of this category. An *Asynchronous Service* is a special category, which returns information whenever the calling component checks the status. Unlike the previous category, asynchronous services can display feedback messages and these services can last days or even weeks to complete. A response to the calling component reports the status by telling either the service is completed or still ongoing. Lastly, the *Asynchronous Web Application* category contains Web services, which are similar to asynchronous services, but without a status check. This type of service is used by interactive Web pages in the Web portal.

With this kind of service categorization, it is possible to support users and their specific needs to complete their tasks with synchronous or asynchronous processes executed in the background. A detailed description of the system design that uses annotations for workflow modeling is given in Section V. A service orchestration is performed by using a component for workflow editing to create a semantic workflow description. Fig. 5 summarizes the generation of workflows. First the workflows define the order in which the services have to be executed. In the process chain, the output of a service is passed to the input of the next service. A tool supporting graphical user input has the advantage that the user must not have a detailed knowledge of OWL-S to describe a workflow. The graphically sketched sequence of services is formalized in a workflow and stored in an XML-based meta-format that serves as input for the conversion into OWL-S.

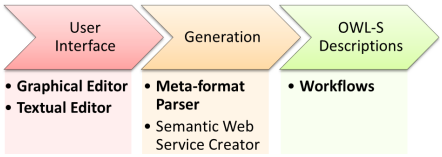

Figure 5. Generation of workflows in OWL-S.

The service domain is structured by an upper model for the generic description and vocabulary of services and workflows in OWL-S. It defines how services must be described and specified, using annotations and technical descriptions. The detailed knowledge of different application domains is represented by several domain ontologies that describe detailed application functionality.

V. INTERACTION OF CORE COMPONENTS

The creation of an interoperable and flexible platform provided as IaaS requires an embodiment of core components, which are compatible with each other. These core components are presented as Web services with a technical interface description in WSDL and form the infrastructure and host all the functionality:

- *Semantic Repository for services and workflows*: This component hosts the OWL-S descriptions of services and workflows.
- *Workflow Editor*: The graphical workflow editor assists in the creation of workflows and associates service functionality. The defined workflows are transferred into an XML-based meta-format, based on a predefined schema.
- *Semantic Web Service Creator*: This component creates the Semantic Web services out of WSDL. In another context, it creates semantic workflow descriptions in OWL-S based on the XML-meta-format, introduced by the Workflow Editor.

Copyright (c) IARIA, 2016. ISBN: 978-1-61208-505-0

- *Workflow Manager process control system*: This component manages, orchestrates, monitors workflows, and checks permissions of the users for the execution.

Vendor specific Web services provide and wrap functionality for specific software components of different complexity levels. Generic services are provided to load data structures with different formats, e.g., Computer-Aided Engineering (CAE) and Computer-Aided Design (CAD) data. Higher services encapsulate complex calculations and comparison operators or even provide the interface to third-party systems to perform complex calculations. The service providers are able to deploy the WSDL description of their Web services via the Workflow Editor (WFE) in the Web portal. The Semantic Web Service Creator uses the absolute URL to the WSDL description of the service to generate the semantic Web service descriptions in OWL-S, and it stores and registers them in the Semantic Repository including the inputs and outputs of the services. The basic requirement for all saved workflows and services are unique names. Each Semantic Repository is based on a central domain model, formalized as an OWL ontology, that describes the input and output types for the matchmaking process of the services.

Fig. 6 gives an overview of the interaction of the core components of the developed platform. The main user interface of the developed platform is a Web portal, and it translates user actions into core component specific requests, e.g., workflow design, workflow and service execution, service monitoring, and result management. With the graphical interface of the Workflow Editor, the user gets access to the services stored in the Service Repository and services dedicated to experiments can be chained up to create dedicated workflows, such as for the comparison of 3D models. Each created workflow is stored in the Semantic Repository and can be found by querying with simple properties. To store the workflows, the graphical contents of the workflows are transferred into a XML-based meta-format. This format serves as input for the Semantic Web Service Creator that generates the workflow descriptions in OWL-S.

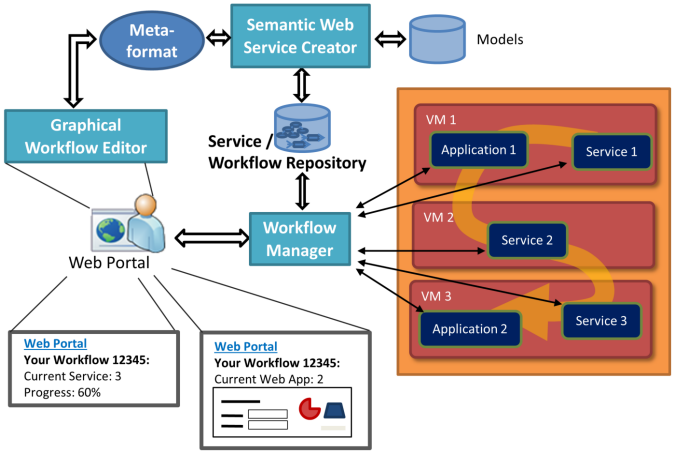

Figure 6. Overview of the interactions of the core components.

The Workflow Manager (WFM) component is used for the management and execution of individual predetermined workflows. Each workflow is a formalized orchestration of Semantic Web services and consists of at least one integrated service or many complex services, defined in sub-workflows. In the execution task, the component processes the individual workflows and accordingly queries the listed services in the defined order. If the service execution is completed and the answer of a service is received, the next step in the sequence is activated. The results of respective services are unified and added to a single representation structure, which is passed at the end of all processing steps to the UI of the Web portal.

For each workflow, the manager initiates processing procedures and tracks the progress individually. Before starting a workflow, it checks whether the user has permission to run it to prevent unauthorized execution. It also provides a monitoring functionality, which allows users to leave the workflow anytime and return at later stage to continue where they left off. This maximizes the benefits of such a cloud-based platform, supporting access anytime and from any desktop or mobile device with internet access. If the workflow does not need user input, the WFM is even able to complete it automatically and display its results to the user at a later time. As explained in the previous sections, services of different vendors can be used that are implemented by different programming techniques and run within the cloud on different application servers. But during the lifetime of a workflow, the user does not need to know, where the services are stored and how the data is forwarded to the next service. The manager component retrieves the service descriptions and performs the tasks without user notification and the complexity of all associated services within a workflow remains hidden from the user.

An example of a graphical workflow is shown in Fig. 7. The execution order is represented by dashed arrows inside WFE and the blue blocks are the individual workflow steps. The green marked block is an HPC sub-workflow, which executes the service in an HPC server environment. This sub-workflow consists of three tasks: (1) *pre-processing task* to generate the command to be executed by HPC process, (2) *HPC command task*, which receives the command by user interface and gives feedback to the user, and (3) *post-processor task*, which converts the output from the HPC process into application specific output.

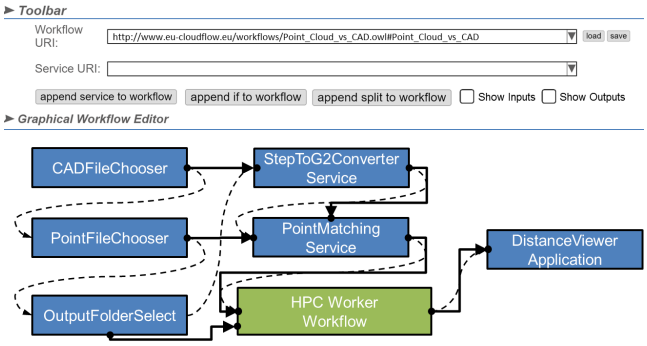

Figure 7. The graphical workflow editor UI shows the scenario workflow.

Based on their defined service types, synchronous or asynchronous, services are differentiated and executed by the execution engine of the WFM. The services for the pre- and post-processing tasks are implemented as *Synchronous Services*. Whereas, the HPC operating procedure is implemented as *Asynchronous Service*. If the duration of a Web service execution cannot be predicted, the service must be implemented as an *Asynchronous Service*, which provides its status via a method. This method is periodically requested by the WFM.

Copyright (c) IARIA, 2016. ISBN: 978-1-61208-505-0

As a result, this method is able to return HTML feedback, which is displayed on the Web portal. The usage of pre- and post-processing tasks varies from application to application.

If a Web service provides an UI to interact, the service must be implemented as an *Asynchronous Web Application*. Services that belong to this category are implemented similar to *Asynchronous Services*, but contrarily do not need to deliver their status. This service type explicitly tells the WFM that the task is completed. After receiving this notification, the WFM performs the next step and gives feedback on the Web portal.

VI. Conclusion and future work

This paper explained the concept and realization of a flexible cloud-based infrastructure platform, which involves Semantic Web technologies and tailored tools for the creation, execution, and management of workflows and conducted services by graphical user interface. The realized platform satisfies the requirements for the development and execution of experiments defined as workflows, without requiring knowledge on High Performance Computing or other underlying technologies of the Semantic Web. The platform offers an UI for the integration of Web services, described in WSDL. These services are automatically converted into Semantic Web services, without requiring specific knowledge of used complex Semantic Web technologies, such as OWL and OWL-S. With another graphical user interface, the Workflow Editor, the services can be orchestrated within the meaning of the experiment can be orchestrated and stored as workflow descriptions. For the execution of the experiment, the Workflow Manager uses these descriptions as a basis. One advantage of this approach is the combination of the created platform with high-performance servers. Complex tasks are outsourced to these servers and this results in enormous time savings and allows the experts to carry out more experiments with products, which was omitted due to the complexity and the required computing power until now. Of course, the possibility to conduct these experiments leads to an enormous increase in expert knowledge.

In future, the Workflow Editor will be able to give recommendations to the user, for an easier dynamic workflow design. A dynamic workflow formalizes an orchestration of services, supported by an automated matchmaking process that provides adequate services ordered by their confidence values, which is only possible using Semantic Web technologies. Furthermore, the Workflow Editor automatically inserts converter services into the workflow, just for adjustment of input and output types, e.g., convert units of measurement and file formats.

Acknowledgments

This research was funded in part by the 7th Framework Program of the European Union, project number 609100 (project CloudFlow). The responsibility for this publication lies with the authors.

References

[1] T. Barton, “Cloud Computing,” in E-Business mit Cloud Computing. Springer Fachmedien Wiesbaden, 2014, pp. 41–52.

[2] A. Ranabahu and A. Sheth, “Semantics Centric Solutions for Application and Data Portability in Cloud Computing,” in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, 2010, pp. 234–241.

[3] R. Cyganiak, D. Wood, and M. Lanthaler, “Web Services Architecture,” W3C Working Group Note, 2004, [retrieved: July 2016]. [Online]. Available: https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[4] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, “Web Services Description Language (WSDL) 1.1,” W3C, W3C Note, March 2001, [retrieved: July 2016]. [Online]. Available: http://www.w3.org/TR/wsdl

[5] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana, “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language,” W3C Recommendation, 2007, [retrieved: July 2016]. [Online]. Available: https://www.w3.org/TR/wsdl20/

[6] T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web,” Scientific American, 2001, [retrieved: July 2016]. [Online]. Available: http://www.jeckle.de/files/tblSW.pdf

[7] G. Klyne and J. J. Carroll, “Resource Description Framework (RDF): Concepts and Abstract Syntax,” W3C Recommendation, 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/

[8] P. Hitzler, M. Krötzsch, and S. Rudolph, Foundations of Semantic Web Technologies. Chapman & Hall/CRC, 2009.

[9] R. Cyganiak, D. Wood, and M. Lanthaler, “RDF 1.1 Concepts and Abstract Syntax,” W3C Recommendation, 2004, [retrieved: July 2016]. [Online]. Available: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

[10] P. F. Patel-Schneider, P. Hayes, and I. Horrocks, “OWL Web Ontology Language Semantics and Abstract Syntax,” Feb. 2004, [retrieved: July 2016]. [Online]. Available: http://www.w3.org/TR/2004/REC-owl-semantics-20040210/

[11] I. Horrocks, B. Parsia, P. Patel-Schneider, and J. Hendler, “Semantic Web Architecture: Stack or Two Towers?” in Principles and Practice of Semantic Web Reasoning, Third International Workshop, PPSWR 2005, Dagstuhl Castle, Germany, F. Fages and S. Soliman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 37–41.

[12] D. Martin et al., “OWL-S: Semantic Markup for Web Services,” 2004, [retrieved: July 2016]. [Online]. Available: http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/

[13] S. Bergweiler, “A Flexible Framework for Adaptive Knowledge Retrieval and Fusion for Kiosk Systems and Mobile Clients,” in Eighth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2014), International Academy, Research, and Industry Association (IARIA). IARIA, 8 2014, pp. 164–171.

[14] C. Casanave, “Designing a Semantic Repository - Integrating architectures for reuse and integration,” 2007, [retrieved: July 2016]. [Online]. Available: https://www.w3.org/2007/06/eGov-dc/papers/SemanticRepository.pdf

[15] “Webster Database Definition,” [retrieved: July 2016]. [Online]. Available: http://www.merriam-webster.com/dictionary/database

[16] Ontotext, “GraphDB - Semantic Repository,” [retrieved: July 2016]. [Online]. Available: http://ontotext.com/products/graphdb/semantic-repository/

[17] Sesame Framework Contributors, “Sesame Java Framework,” [retrieved: July 2016]. [Online]. Available: http://rdf4j.org/about.docbook?view

[18] “Web Services Business Process Execution Language Version 2.0,” OASIS Web Services Business Process Execution Language (WSBPEL) Technical Commitee, 2007, [retrieved: July 2016]. [Online]. Available: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[19] S. Bansal, A. Bansal, G. Gupta, and M. B. Blake, “Generalized semantic Web service composition,” Service Oriented Computing and Applications, vol. 10, no. 2, 2016, pp. 111–133.

[20] D. Elenius et al., “The OWL-S editor - a development tool for semantic web services,” in ESWC, 2005, pp. 78–92.

[21] S. Bergweiler, “Interactive service composition and query,” in Towards the Internet of Services: The Theseus Program. Springer Berlin Heidelberg, 2014, pp. 169–184.

[22] C. Stahl, E. Bellos, C. Altenhofen, and J. Hjelmervik, “Flexible Integration of Cloud-based Engineering Services using Semantic Technologies,” in Industrial Technology (ICIT), 2015 IEEE International Conference on, 2015, pp. 1520–1525.

[23] Stellba, “Comparing CAD Models with 3D Scanned Manufactured Parts on the Cloud,” [retrieved: July 2016]. [Online]. Available: http://eu-cloudflow.eu/experiments/first-wave/experiment_6.html

[24] C. Dyken et al., “A framework for OpenGL client-server rendering,” in Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference, 2012, pp. 729–734.

Copyright (c) IARIA, 2016. ISBN: 978-1-61208-505-0