Semantic Sky: A Platform for Cloud Service Integration based on Semantic Web Technologies

Dimitar Trajanov
Faculty of Computer Science and Engineering
Skopje, Macedonia
dimitar.trajanov@finki.ukim.mk

Riste Stojanov
Faculty of Computer Science and Engineering
Skopje, Macedonia
riste.stojanov@finki.ukim.mk

Milos Jovanovik
Faculty of Computer Science and Engineering
Skopje, Macedonia
milos.jovanovik@finki.ukim.mk

Vladimir Zdraveski
Faculty of Computer Science and Engineering
Skopje, Macedonia
vladimir.zdraveski@finki.ukim.mk

Petar Ristoski
Faculty of Computer Science and Engineering
Skopje, Macedonia
petar.ristoski88@gmail.com

Marjan Georgiev
Faculty of Computer Science and Engineering
Skopje, Macedonia
marjan.georgiev@gmail.com

Sonja Filiposka
Faculty of Computer Science and Engineering
Skopje, Macedonia
sonja.filiposka@finki.ukim.mk

ABSTRACT

These days, the number of data sources an ordinary computer user works with every day is very large and continues to grow. With the increasing number of cloud services with specialized functionalities, the users are faced with the necessity to routinely perform manual actions to interchange data among different cloud and web services, in order to perform more complex and composite actions. These actions always require a certain amount of dedicated time from the user, who has to change the context in which he work, in order to take the actions and transfer data from one system to another. In this paper, we present a software platform, based on the concepts and technologies of the Semantic Web, which provides the users with a unified and simple composite approach to the different services they use, and crates a simple flow of information from one infrastructure to another. The system is able to automatically discover the context in which the user is working, and offer him the actions which can be used over the data within the context. In this way, the user can completely focus on his tasks in his work environment, and get relevant information and possible actions in that context. This system automates the execution of the users' tasks, which leads to improvements in their productivity, information exchange and efficiency. The system is called "Semantic Sky" and represents a platform where many cloud services are interconnected by the use of semantic web technologies.

I-Semantics 2012, 8th Int. Conf. on Semantic Systems, Sept. 5-7, 2012, Graz, Austria.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - *information filtering, retrieval models, search process*

General Terms

Algorithms, Performance, Design

Keywords

Semantic Web Technologies, Cloud Services, Named Entity Extraction, Automation of Processes.

1. INTRODUCTION

The information we work with every day is obtained from different sources – email, Facebook, Twitter, local documents, enterprise services, etc. Depending on that information, we usually take some actions (e.g. we share them, add them into a "to-do" list, etc.). In the last years, with the increasing number of cloud services [5] with specialized functionalities, we came across the need to routinely perform manual actions to interchange data among these cloud services, in order to perform more complex and composite actions. These actions always require a certain amount of dedicated time from the user, who has to change the context in which he works manually, in order to take the actions and transfer data from one system to another. Furthermore, we often need mechanisms for detection and description of the entities which are found within texts and information that we get from different services.

With the use of the technologies of the Semantic Web [18] in this approach, we can automate some of these tasks and processes, and eliminate the manual work. Even though the main purpose of the Semantic Web is to give meaning to the resources on the World Wide Web, semantic annotation of other resources can be used to automate some of the processes involved in the everyday service consuming flow.

In this paper we present a software platform, developed using the technologies of the Semantic Web, which provides the users with a unified and simple composite approach to the different services they use, and with a simple flow of information from one infrastructure to another. The system is able to automatically discover the context in which the user is working, and offer them the actions that can be used within the context. The users can completely focus on their tasks in their work environment, and get relevant information and possible actions in that context. This system automates the execution of the users' tasks, which leads to improvements in their productivity, information exchange and efficiency.

2. RELATED WORK

The potential and influence that the Cloud systems have had over the development and use of information technologies during the last years, attracts more researchers to work in this area. One of the challenging aspects in this area is the integration of semantic technologies that should enable drastic changes in the opportunities for application integration with the user environment and services [15][21][26][29]. Today, this research field is very active, with many projects aiming towards solving these existing problems. As an example in terms of development of desktop applications which use semantic technologies, the project Semantic Desktop [28] gives a proposal for layered architecture based on ontologies for different levels. By organizing the data and annotating the resources, which means creating associations between resources and files based on extraction of keywords and relations, and analyzing the history of use, it is possible for complex actions to be related to the context in which the user is working at the moment, and to be offered to him. The described project aims towards the integration of semantic technologies in desktop environments, but excludes the integration of many different Cloud infrastructures.

On the other hand, the project mOSAIC (Open-Source API and Platform for Multiple Clouds) [20], intents to build an application platform that will integrate many Cloud services in order to answer complex user requirements. This project is under way and will provide a framework for development of applications that does not depend on the infrastructure they will be executed on. This framework will use a Cloud ontology and semantic representation of the resources [23]. An application will be able to define its own requirements and the decision about the place of execution will depend on them. The execution will be driven by certain "agent applications".

The project SITIO [12] is another research project that integrates many Cloud services, but unlike mOSAIC, the integration here is done only at the service level. The system implemented here enables different developers to create applications and integrate them in their own system. Several ontologies for semantic annotation of the available services, as well as certain tools for semi-automated annotation of their services, are available to the developers. Both mentioned projects are mainly aiming at the issues of interconnecting different Cloud infrastructures and defining interfaces for their mutual communication.

Anzo Data Collaboration Server lies at the heart of the Cambridge Semantics architecture [4]. The Open Anzo project includes an open source enterprise-featured RDF quad store, coupled with a unique semantic middleware platform. Together they provide the developers with a host of features necessary for the creation of sophisticated semantic technology standard (RDF, OWL, SPARQL) grounded applications. Features include support for multiple users, distributed clients and services, offline work, real-time notifications, named-graph modularization, versioning, access controls, and transactions with preconditions.

Babylon–Enterprise [6] is an enterprise information retrieval system, which represents a web-configured client-server system, based on a Windows client installed on the end-user's workstation and an enterprise application server. All of the employees in the enterprise can easily access all company information from their work environment.

OntoBroker is an ontology repository from ontoprise, which includes a high performance deductive reasoning engine [2][3]. Reasoning with rules is a major unique selling point for ontoprise. OntoBroker integrates a connector framework which makes it easy to connect a multitude of data sources like databases, web services, etc. Although it combines structured and unstructured data in one framework, OntoBroker is easy to extend and to integrate into existing IT landscapes and applications, as it offers a variety of open interfaces. OntoBroker is also closely connected to ontoprise's ontology modeling environment OntoStudio, which is the development environment for handling ontologies, mappings to information sources, rules, generating queries, creating business intelligence reports, etc. [2]. For many customers OntoBroker serves as a common semantic layer which is accessed by various applications and integrates different information sources.

None of the systems described above fully automates the process of user context extraction and invocation of relevant actions. These systems have some semantic features, but none of them integrates a context recognition capability, automatic action proposition, automatic action execution and auto-generated API interface, which makes them require a customization for every future extension, in terms of both the knowledgebase or and a client-side application.

3. SOLUTION DESCRIPTION

The system we developed is called "Semantic Sky", because it is an environment where many cloud services are interconnected by the use of semantic web technologies. A detailed description of the solution follows.

3.1. Architecture

The core of the system is developed with the Play MVC framework [27]. The system architecture, shown on Figure 1, consists of several components, listed below.

The user interaction with Semantic Sky happens through *Application plug-ins*. Any third party application, which implements the Semantic Sky API, can be integrated as a plug-in into the system. These plug-ins extract the user's working context and, when possible, infer additional metadata. This information is then sent to the Context Extraction Module. In the current implementation of the system, plug-ins for Gmail, Microsoft Office Word and Microsoft Office Outlook are used (see Section 3.3).

The *Knowledgebase* contains all of the data used by the user's applications. Our system extracts all of the information from it. New data can be easily added to the Knowledgebase, either by uploading a new OWL/RDF file, by adding a link to an OWL/RDF file or by adding an URL to a SPARQL endpoint. After the data is added to the Knowledgebase, it is indexed and ready to use.

The *Context Extractor Module*, shown on Figure 2, is the entry point to the core of the solution. It accepts the text-based data sent from the Application plug-ins. For each token (word) in the

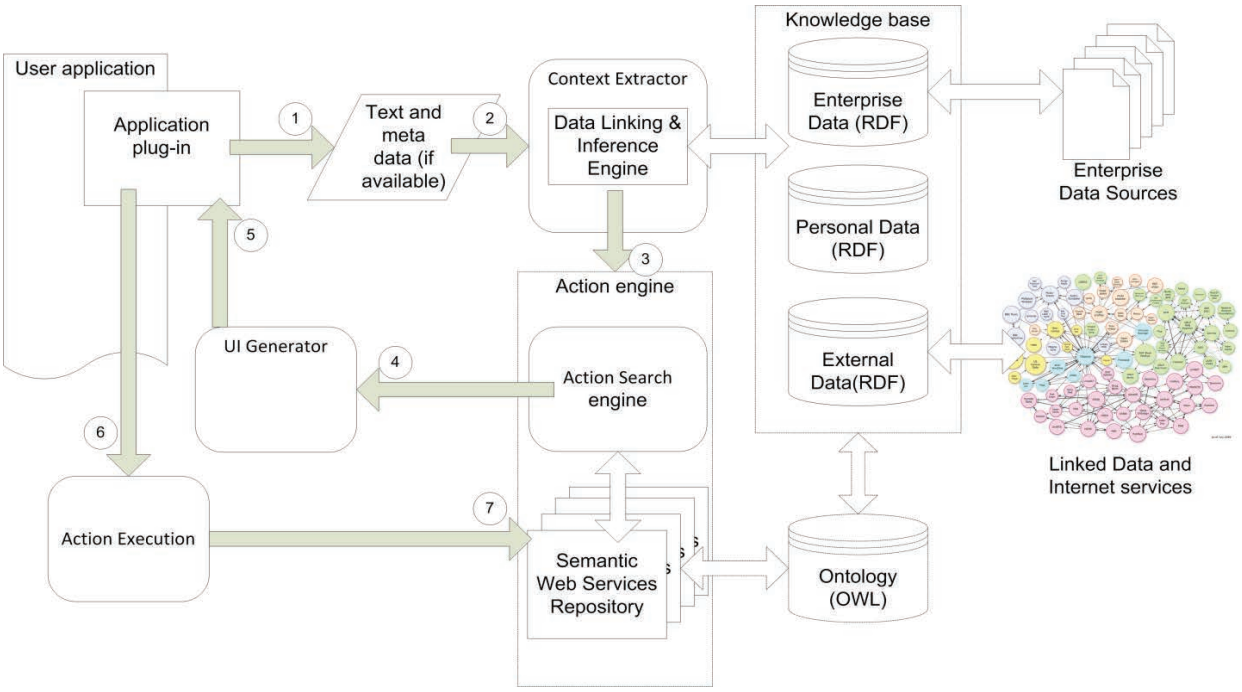

Figure 1: System Architecture.

text, the system extracts all resources related to it, using an index previously created in the Knowledgebase. All entities retrieved from the search are converted into semantic resources, and grouped by their type. This module uses a modification of the Apache Lucene Core engine [14], adapted to fit our solution. The modifications included using B+ hash trees to speed up the on-disk RDF lookups, as proposed in [24].

The *Semantic Web Services Repository* contains semantically annotated SOAP and RESTful web services, which will be used later to build actions for the user to execute. The system provides a simple form for annotation [8] and uploading of new web services. The SOAP web services are annotated using the Semantic Annotations for WSDL and XML Schema (SAWSDL) framework [10]. Although there are several

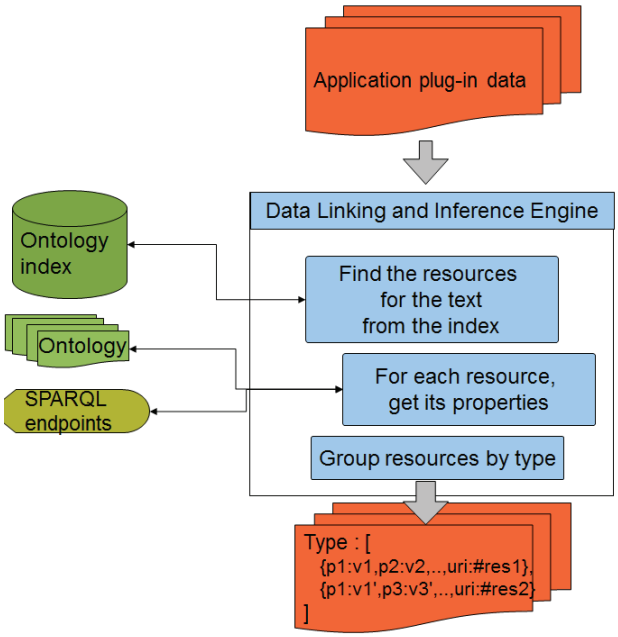

Figure 2: Context Extractor Module.

schemas for semantic annotation of RESTful services [19], most notably hRESTS [16] and MicroWSMO [17], we use a custom schema. The existing schemas are more robust than what we need, so we decided to go with a custom, more light-weight approach, which fits better with our solution. Our schema uses the base-URL of the service, the method type and the names of the input parameters and the output parameter as properties which are semantically annotated. Each web service, regardless of the type, can be annotated with any ontology within the system.

The *Action Search Module*, shown on Figure 3, accepts the previously extracted resources from the Context Extractor Module, and aligns them as inputs. The system uses a custom algorithm to search through the SWS Repository for web services or compositions of web services which can be invoked with the previously identified inputs. The algorithm detects the semantic type of the desired output, and scans through the SWS Repository for services which provide such outputs. The detected services are tested for the semantic types of their inputs, in order to select only the services which can be invoked with the provided inputs. For every input parameter which is not provided, the algorithm does the same search for services which can provide it as an output. This is done repeatedly, until the algorithm comes to a state in which all of the required inputs for the selected services are provided either from the Context Extractor Module or as outputs from other services. In this state, the algorithm has identified one or more candidate web services or compositions of web services. We call them *actions*. We rank the actions using a suitability calculation, based on the number of services in the composition and the number of parameters which need to be exchanged at each level of the composition. As a result, we have actions ordered by relevance to the extracted resources. To improve the performances of the module, the actions are stored in cache, structured as an XML file, which speeds up the search process.

The *UI Generator Module*, shown in Figure 4, is responsible for generating the Graphical User Interface (GUI) of the system. In this module we implemented an algorithm for generic visualization of different types of resources and entities. The

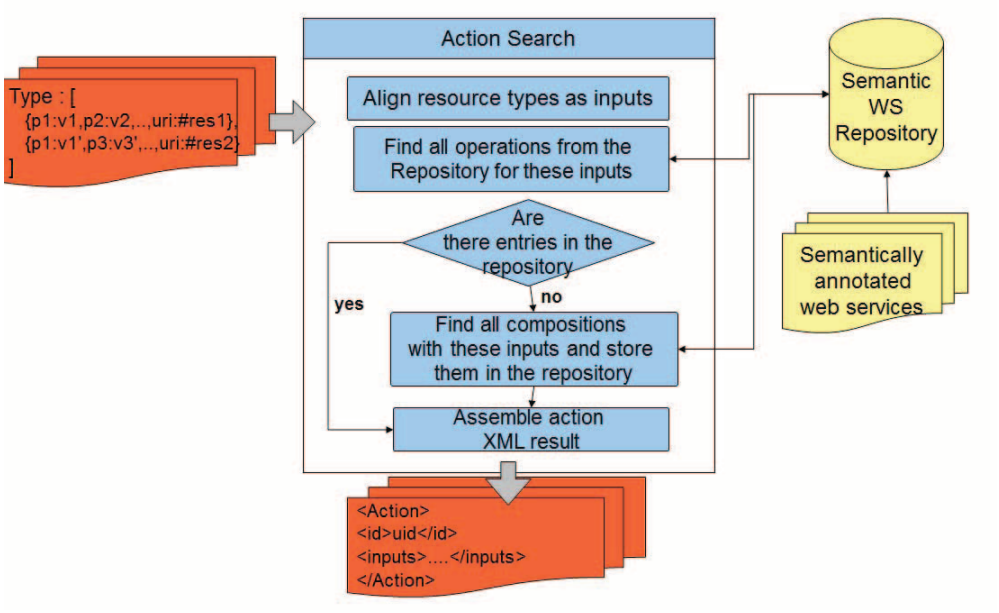

Figure 3: Action Search Module.

algorithm binds an XSLT transformer to each type of resources, which is used to render that type of resource. Using this approach, we enable previews of extracted resources, possible actions, results from the actions, and a preview of the configuration of the system.

The *Module for Action Execution* is responsible for the execution of the semantically annotated actions. This module receives the RDF resources and the semantic description of the action as an input. The arguments for the action are provided by the user. The user selects each input from a group of possible resources. These resources are the ones returned from the Context Extractor Module, grouped by the service input type. If there is only one resource from the given type, it is automatically selected. One action can consist of a single RESTful web service, a single function from a SOAP web service, or a composition of functions from one or more SOAP web services. In the former two cases, the invocation is a single step process. However, for the latter case, we implemented an algorithm which invokes the functions in a given order and handles the parameter passing between them. After the execution of the action, the generated output is displayed to the user. If the action does not have a visible output, the system displays a status message. The implementation of this module uses the Apache Axis2 engine [1].

3.2. Advantages

The main advantage of the system is its capability for integration with already existing, proprietary and widely used systems, such as Gmail, Microsoft Office, etc. This provides a low learning curve for the end users: our system integrates within their work environment, so they will not have to adapt into a new, unfamiliar environment. This capability is possible due to the modular architecture of the solution. The integration is achieved in few simple configuration steps. First, the user data should be connected to the system. The data formats can be relational databases, semantic databases or OWL/RDF files. This connection process is different from a standard data import process. Instead, our system indexes the data and stores the index within the knowledgebase. The next step is a semantic annotation of the SOAP and RESTful web services, and their registration in the SWS repository. After this, the system is ready for use in the new environment.

Another advantage of the system is its scalability. The architecture allows easy expansion of the number of users, since

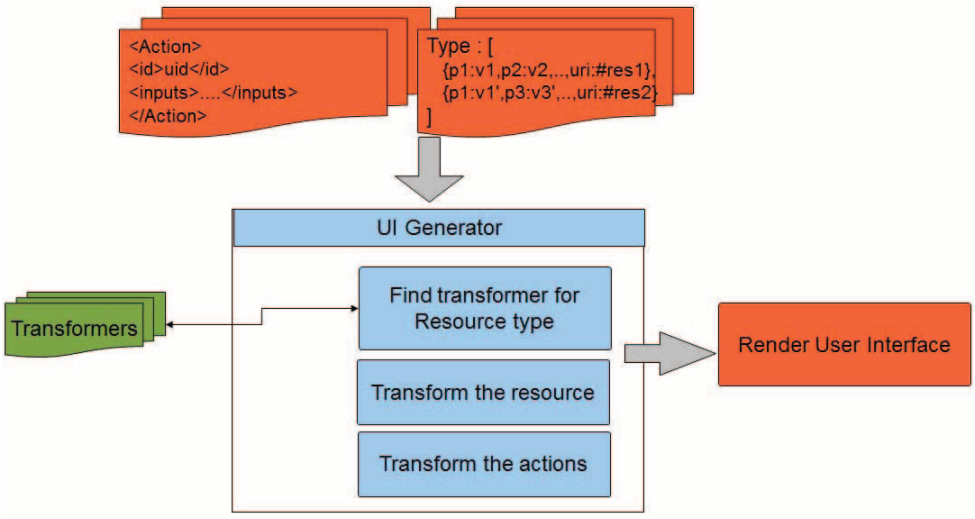

Figure 4: UI Generator Module.

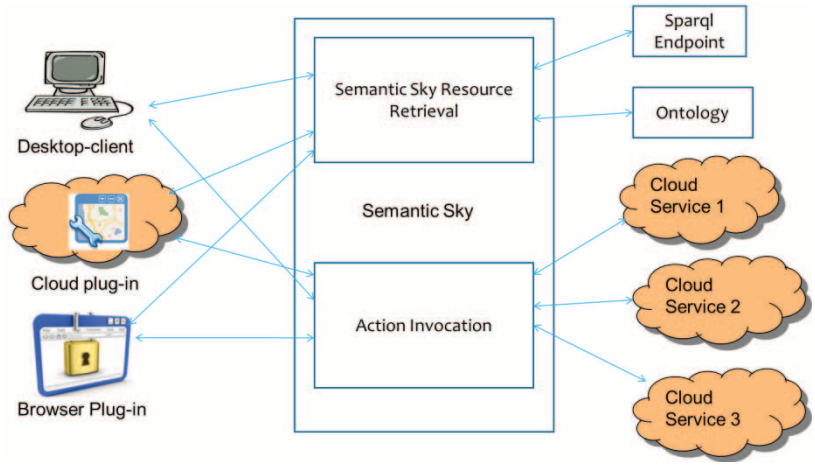

Figure 5: System integration with user application plug-ins.

the communication with the user environment is service-based.

The modularity of the architecture allows adding and removing functionalities from the system, in order to conform to the requirements of the user.

The service-based communication with the user environment also goes into favor of the overall extensibility of the system. It can be extended by adding new services, as well as new ontologies and knowledge. For every new type of data entered into the system, the use of XSLT makes it is easy to configure a new UI representation.

Additionally, the system can be used from various types of application plug-ins, as shown on Figure 5:

- *Cloud e-mail plug-ins* – the system can be integrated with the e-mail client used within an enterprise, for example. After the integration, the users will get useful information extracted from each e-mail message they receive, along with a list of possible action which can be executed over the identified resources. As we will show later, we developed a Gmail contextual gadget as a plug-in for the system.
- *Desktop clients* – the system can be integrated with various desktop applications or it can work on top of all desktop applications. We developed both Microsoft Office Word and Outlook add-ins as plug-ins for the system, which are described in the next section.
- *Browser plug-ins* – the system can be integrated with the web browser used within an enterprise, for example. The users will get information relevant to their work and a list of possible actions which can be taken over the data extracted from the web pages they visit.
- *Miscellaneous systems* – Semantic Sky is designed as a universal system and because of its modular architecture it can be extended and integrated with most of the existing systems.

3.3. Implementations

In order to test its functionalities and create a working example, we implemented the Semantic Sky system in our environment at the Faculty of Computer Science and Engineering in Skopje.

First, we loaded data into the Knowledgebase. We used the Friend of a Friend (FOAF) ontology [11] to define the basic relationships among people at the Faculty. In order to use the relational database from our student e-services system, we designed a custom ontology which defines all of the concepts for a faculty environment, matching all the relations from the existing relational database [22]. The ontology was designed using the Protégé ontology editor [25]. We mapped the relational database with the ontology and deployed a SPARQL endpoint, using D2RQ server [7] as described in [22]. The SPARQL endpoint was then connected to the system and the data it points to was added into the Knowledgebase.

The second step was to register the existing web services into the SWS Repository. We semantically annotated the services with the previously created ontologies, using the tool for annotation provided by the system [8]. Then all of the services were registered into the SWS Repository.

As an application plug-in, we developed a Gmail contextual gadget [13]. The plug-in was tested by the faculty staff, in order to automate the most common tasks within their e-mail inbox. The plug-in extracts the entities and relations from each received e-mail message and suggests a list of possible actions which can be taken over the extracted context. Thus, the users

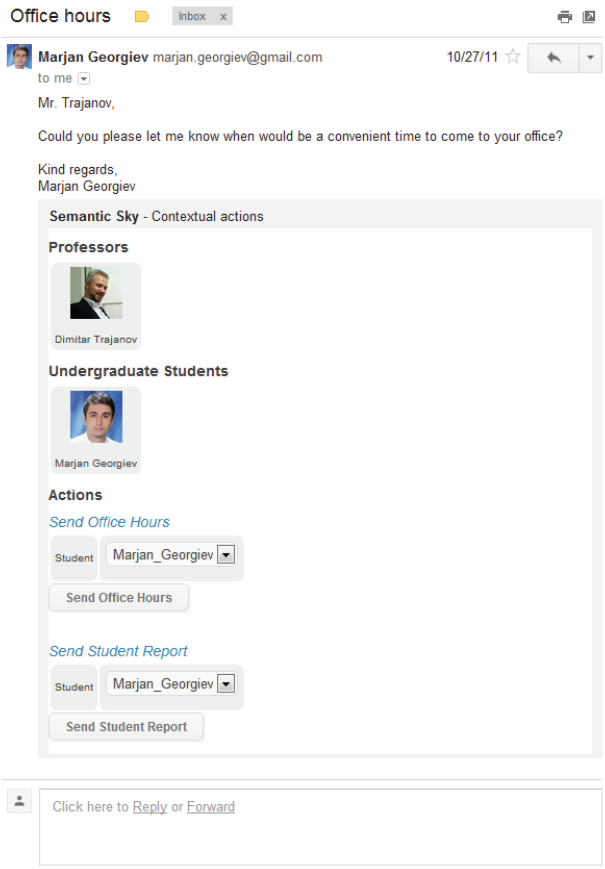

Figure 6: Gmail contextual gadget UI.

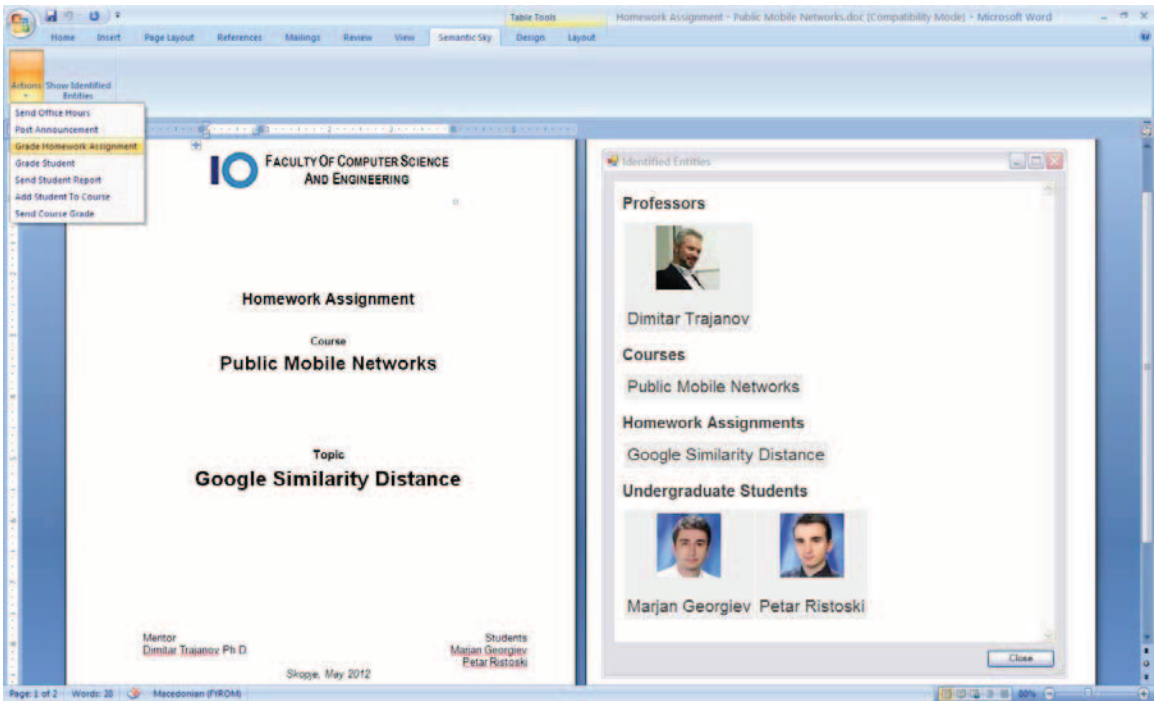

Figure 7: Microsoft Office Word add-in.

are able to quickly understand who the sender of the message is, what their mutual relation is, and what the task is. Then the user can automatically execute one or several actions in order to accomplish the demands from the e-mail.

The Gmail gadget interface is shown on Figure 6. The upper part lists the entities extracted from the currently opened e-mail message, grouped by their type. The lower part lists the possible actions which can be executed over the extracted entities. These actions can be executed here, directly from the user environment, saving the users a valuable time. Grading a student homework assignment, for example, normally requires a professor to log-in into a different cloud based e-services system of the Faculty, browse for the course, browse for the student and then enter the grade. Here, with the use of Semantic Sky, these steps are reduces to just one – a single click on a button within the e-mail message.

Another application plug-in we developed was a Microsoft Office Word add-in, shown on Figure 8, which analyzes the contents of the opened document and extract the entities and relations relevant for the user and his context. The add-in automatically lists the possible actions for the extracted resources. These actions can be executed directly from the add-in, saving valuable time for the user. The main purpose of this add-in is to assist the professors in the process of reading and grading student homework assignments. The add-in helps the users to quickly understand the context of the full document, the details of the author and the purpose of the document, in order to help them evaluate and grade the assignment faster and better.

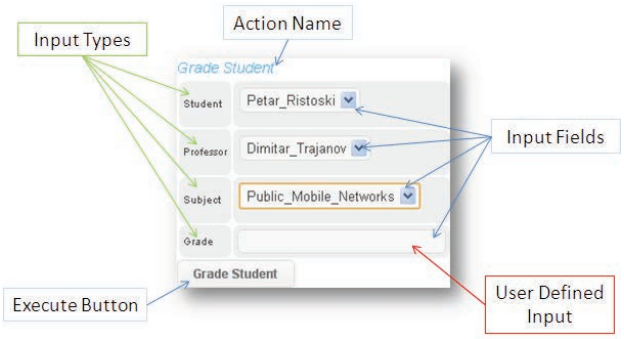

Figure 8: Action Form.

When the user selects an action from the Gmail or Work plug-in, an action form is generated. Figure 8 shows an example of a single action form. The form displays the action name and is populated with fields for all of the input types necessary for execution of the action. The user can manually choose the value for each of the inputs from the list of possible values. In some action forms, there may additionally be a specific input type, called User Defined Input, which must be entered manually by the user. These types of inputs are used when the value of the specific input type cannot be extracted from the knowledgebase or the context of the text, but is user dependent. As shown on Figure 8, the professor needs to enter the grade manually, in order to grade the student. After he clicks the execute button, the preexisting web service for grading a student for a given course – now semantically annotated within our system – is invoked and the grade is entered into the e-services system of the Faculty.

We also developed a Microsoft Office Outlook add-in, shown on Figure 9. Much like the Gmail gadget, this add-in analyzes the content of e-mail messages. The identified entities are listed, and all of the actions associated with them are displayed in a drop-down list under the Semantic Sky ribbon. When the user selects an action, a form similar to the one on Figure 8 is generated on the bottom of the Outlook mail Inspector window, providing a fast and easy way to execute the appropriate action within the context of the e-mail message.

4. CONCLUSION

The main goal of this system is the development of the novel and innovative framework which enables connectivity and integration, not only of different cloud services, but also of local data placed on the user machine. The developed system provides the opportunity for automation of the use of different services. It also offers an engine which detects and ranks the

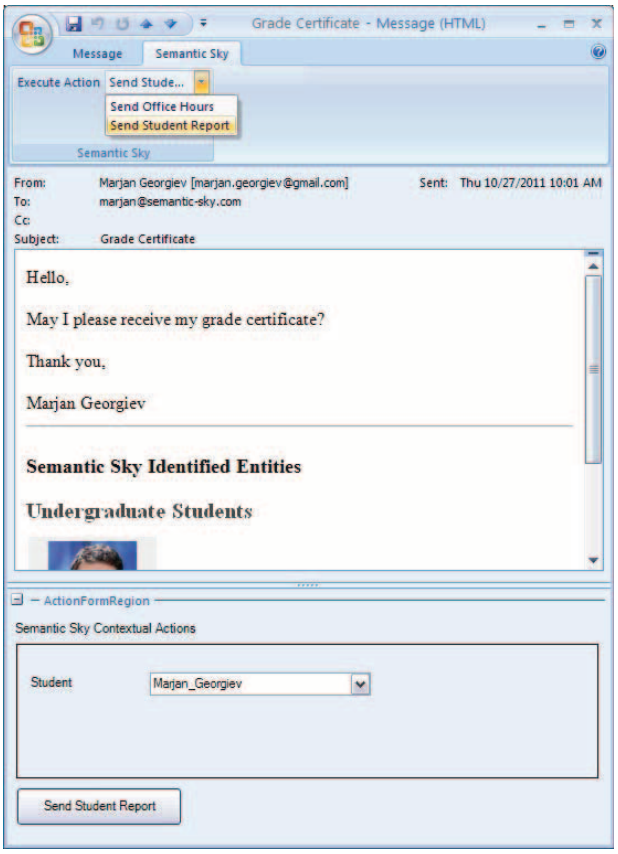

Figure 9: Microsoft Office Outlook add-in.

possible actions which can be executed within a given context, over the identified concepts.

The system is designed and implemented as a modular architecture, which allows it to be very flexible and easily extendable with new modules. It is platform- and technology-independent, which means the users have no restriction while implementing application plug-ins for it.

In the process of the development of the system, an efficient algorithm for automated context extraction from text, which extracts the most relevant semantic entities and concepts from the user data, has been developed. This process reduces the time necessary for data entry when taking an action, because the entities are already detected and aligned as inputs for the appropriate action.

Also, an algorithm has been developed for automatic action suggestion, based on the user context. It includes creating a composition of web services and automatic execution of the selected actions. This process can sometimes provide the user with actions which he is not aware existed within his work environment, and can help him be more productive. The automatic detection and composition of services into a single action means that longer sequences of processes can be executed with just one interaction from the user, saving him valuable time.

We also developed a methodology for generic visualization of the extracted entities and concepts, and user interface generation.

The integration of this solution within everyday systems and applications, especially enterprise systems, will improve their usability and integration with other systems and services, and will improve the overall performance of the users. The features of the system reduce the time needed by the end users to discover and take the appropriate action over the data they are working with within a given context.

This solution aims towards unification of both the knowledge and the actions which reside within many diverse and isolated systems, present within the work environment of a common user.

5. REFERENCES

[1] D. Almaer. Creating Web Services with Apache Axis. *Disponible en*, 2002.

[2] J. Angele, M. Erdmann, and D. Wenke. Ontology-Based Knowledge Management in Automotive Engineering Scenarios. *Ontology Management*, pages 245-264, 2008.

[3] J. Angele. OntoBrokerMature and approved semantic middleware. *Semantic Web*, 2008.

[4] The Anzo Data Collaboration Server: Anzo Architecture, 2012. Revised June 2012, from Cambridge Semantics: http://www.cambridgesemantics.com/technology/architecture

[5] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, A view of cloud computing. *Communications of the ACM*, 53(4):50-58, 2010.

[6] Babylon-Enterprise: Overview, 2011. Revised 2011, from Babylon Ltd.: http://www.babylon-enterprise.com/

[7] C. Bizer and R. Cyganiak. D2R Server - Publishing Relational Databases on the Semantic Web. In *5th International Semantic Web Conference*, page 26, 2006.

[8] K. Budinoski, M. Jovanovik, R. Stojanov, and D. Trajanov. An Application for Semantic Annotation of Web Services. In *7th International Conference for Informatics and Information Technology - CIIT 2010*. Institute of Informatics, Skopje, Macedonia, 2010.

[9] T. Erl. Service-Oriented architecture: Concepts, Technology, and Design. *Prentice Hall PTR*, 2005.

[10] J. Farrell and H. Lausen. Semantic Annotations for WSDL and XML Schema. *W3c recommendation*, 28, 2007.

[11] Friend of a Friend (FOAF), April 2011, http://www.foaf-project.org/

[12] F. Garcia-Sanchez, E. Fernandez-Breis, R. Valencia-Garcia, E. Jimenez, J. Gomez, J. Torres-Nicno, and D. Martinez-Maqueda. Adding Semantics to Software-as-a-Service and Cloud Computing. *WSEAS Transactions on Computers*, 9(2):154-163, 2010.

[13] Google Developers: Gmail contextual gadget, 2011, Revised May 2011, from Google: https://developers.google.com/google-apps/gmail/contextual_gadgets

[14] A. Jakarta. Apache Lucene – A high-performance, full-featured text search engine library, 2004.

[15] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes. Sky Computing. *Internet Computing, IEEE*, 13(5):43-51, 2009.

[16] J. Kopecky, K. Gomadam, and T. Vitvar. hRESTS: An HTML Microformat for Describing RESTful Web Services. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference, IEEE,* volume 1, pages 619-625, 2008.

[17] J. Kopecky, T. Vitvar, D. Fensel, and K. Gomadam. hRESTS & microWSMO. *Technical report, available at* http://cms-wg. sti2.org/TR/d12, 2009.

[18] T. B. Lee, J. Hendler, O. Lassila. The Semantic Web. *Scientic American*, 284(5):34-43, 2001.

[19] M. Maleshkova, C. Pedrinaci, and J. Domingue. Supporting the Creation of Semantic RESTful Service Descriptions. *8th International Semantic Web Conference (ISWC 2009)*, 2009.

[20] B. Di Martino, D. Petcu, R. Cossu, P. Goncalves, T. M_ahr, and M. Loichate. Building a mOSAIC of Clouds. In *Euro-Par 2010 Parallel Processing Workshops*, pages 571-578. Springer, 2011.

[21] P. Mika and G. Tummarello. Web Semantics in the Clouds. *Intelligent Systems, IEEE*, 23(5):82-87, 2008.

[22] M. Mitrevski, M. Jovanovik, R. Stojanov, and D. Trajanov. Open University Data. In *9th International Conference for Informatics and Information Technology*. Faculty of Computer Science and Engineering, Skopje, Macedonia, 2012.

[23] F. Moscato, R. Aversa, B. Di Martino, D. Petcu, M. Rak, and S. Venticinque. An Ontology for the Cloud in mOSAIC. In *Cloud Computing: Methodology, System and Applications, Taylor&Francis Group, to be published*, 2009.

[24] M. Ngyen, C. Basca, A. Bernstein, Speeding up on-disk RDF index lookup using B+Hash Trees. In *Proceedings of The 6th International Workshop On Scalable Semantic Web Knowledge Base Systems*, 2010.

[25] N. Noy, M. Sintek, S. Decker, M. Crubezy, R. Fergerson, and M. Musen. Creating Semantic Web Contents with Protege-2000. *Intelligent Systems, IEEE*, 16(2):60-71, 2001.

[26] D. Petcu, C. Craciun, M. Neagul, S. Panica, B. Di Martino, S. Venticinque, M. Rak, and R. Aversa. Architecturing a Sky Computing Platform. In *Towards a Service-Based Internet. ServiceWave 2010 Workshops*, pages 1-13. Springer, 2011.

[27] A. Reelsen, Play Framework Cookbook. *Packt Publishing Ltd.*, 2011.

[28] L. Sauermann, G. Grimnes, M. Kiesel, C. Fluit, H. Maus, D. Heim, D. Nadeem, B. Horak, and A. Dengel. Semantic Desktop 2.0: The Gnowsis Experience. *The Semantic Web-ISWC 2006*, pages 887-900, 2006.

[29] L. Youse, M. Butrico, and D. Da Silva. Toward a Unified Ontology of Cloud Computing. In *Grid Computing Environments Workshop*, 2008. GCE'08, pages 1-10. IEEE, 2008.