roar @UEL
research open access repository

University of East London Institutional Repository: http://roar.uel.ac.uk

This paper is made available online in accordance with publisher policies. Please scroll down to view the document itself. Please refer to the repository record for this item and our policy information available from the repository home page for further information.

Author(s): Yu, Jian; Sheng, Quan Z.; Falcarin, Paolo
Article title: A visual semantic service browser supporting user-centric service composition
Year of publication: 2010
Citation: Yu, J., Sheng, Q.Z. and Falcarin, P. (2010) 'A visual semantic service browser supporting user-centric service composition'. In IEEE 24th International Conference on Advanced Information Networking and Applications (AINA 2010), Sch. of Comput. Sci., Univ. of Adelaide, Adelaide, SA, Australia, April 2010, pp. 244-251.
Link to published version: http://dx.doi.org/10.1109/AINA.2010.113
DOI: 10.1109/AINA.2010.113

2010 24th IEEE International Conference on Advanced Information Networking and Applications

A Visual Semantic Service Browser Supporting User-Centric Service Composition

Jian Yu
School of Computer Science
University of Adelaide
Adelaide, SA 5005, Australia
Email: jian.yu01@adelaide.edu.au

Quan Z. Sheng
School of Computer Science
University of Adelaide
Adelaide, SA 5005, Australia
Email: qsheng@cs.adelaide.edu.au

Paolo Falcarin
Dipartimento Automatica e Informatica
Politecnico di Torino
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
Email: paolo.falcarin@polito.it

***Abstract*—Follow the promising Web 2.0 paradigm, the telecommunications world also wants to implement the Telco 2.0 vision by inviting its users to actively participate in the creating and sharing of services accessible using handheld devices. The EU-IST research project OPUCE (Open Platform for User-Centric Service Creation and Execution) aims at providing end users with an innovative platform which allows an easy creation and delivery of personalized communication and information services. This paper introduces a novel visual semantic service browser built on top of the OPUCE service repository which enables intuitive visualized service exploring and discovery while requires no technical semantic Web knowledge from the user.**

I. Introduction

In recent years the needs of users concerning telecommunications services, especially mobile ones, have evolved rapidly, requiring the collaboration of an increasing number of network resources of various technologies and user data. On the other hand, users have been excluded from current service provisioning models, having few chances to personalize their services according to their needs [25].

Influenced by the Web 2.0 trend of user-centric content delivery, the Telco world tries to build its own Telco 2.0. The European Union sponsored IST-FP6 integrated research project OPUCE [1] (Open Platform for User-Centric Service Creation and Execution) represents the newest development following this vision. The OPUCE platform aims at developing a unique platform where end users could create and publish mobile services from a converged Web of information technologies and communications services. Taking a simple yet outstanding example, with the OPUCE platform, end users could easily create a useful personal service that monitors an email account and then sends the incoming mails out as SMS or text-to-speech transformed voice call to the user's mobile phone depending on its context.

A service repository is responsible for storing and managing all sorts of information related to services and then is at the very core of the OPUCE platform. since the user-centricity feature of the platform, apart from common data management needs, there are a few special requirements for the repository. For example, the repository is required to support technically non-experienced users so that they can on the one hand easily access and manage their own services, and on the hand conveniently discover services suitable for creating new value-added services.

In this paper, we discuss the visual semantic service browser as one of the key component of the OPUCE service repository. This browser leverage the semantic Web technology to enable end users to visually exploring and discovering OPUCE services in a intuitive and convenient manner.

The reset of the paper is organized as follows. Section 2 gives a brief introduction to the OPUCE platform and the role of the service repository in the whole architectural picture. Section 3 describes the technical details of the visual semantic service browser, including an ontology for describing OPUCE services, the underpinning principle, and also the implementation. Section 4 reports the evaluation of the service browser. Section 5 discusses related work and finally Section 6 concludes the paper.

II. OPUCE platform overview

User-centricity is the dominant feature of the OPUCE platform, which is reflected in the whole life-cycle of the service provisioning process. As illustrated in Figure 1, operators and service providers encapsulate diverse telecommunications and IT resources as OPUCE base services; and prosumers use the OPUCE platform not only to create value-added services, but also to manage, execute, share, and adapt them. In general the OPUCE platform comprises the following main components:

- Advanced Service Editor: It is a Web-based graphical service editor for prosumers to create services and also to deploy and manage services. . Service Lifecycle Management: It manages the entire lifecycle of all services, including deployment, provisioning, and monitoring. It also hosts the Service Description Translator, which is responsible for translating the user-created OPUCE composite service description into executable BPEL [6] scripts.
- Service Execution Environment: It hosts and runs the real executable code of both OPUCE base services

[1] http://www.opuce.eu/

1550-445X/10 $26.00 © 2010 IEEE
DOI 10.1109/AINA.2010.113

IEEE computer society

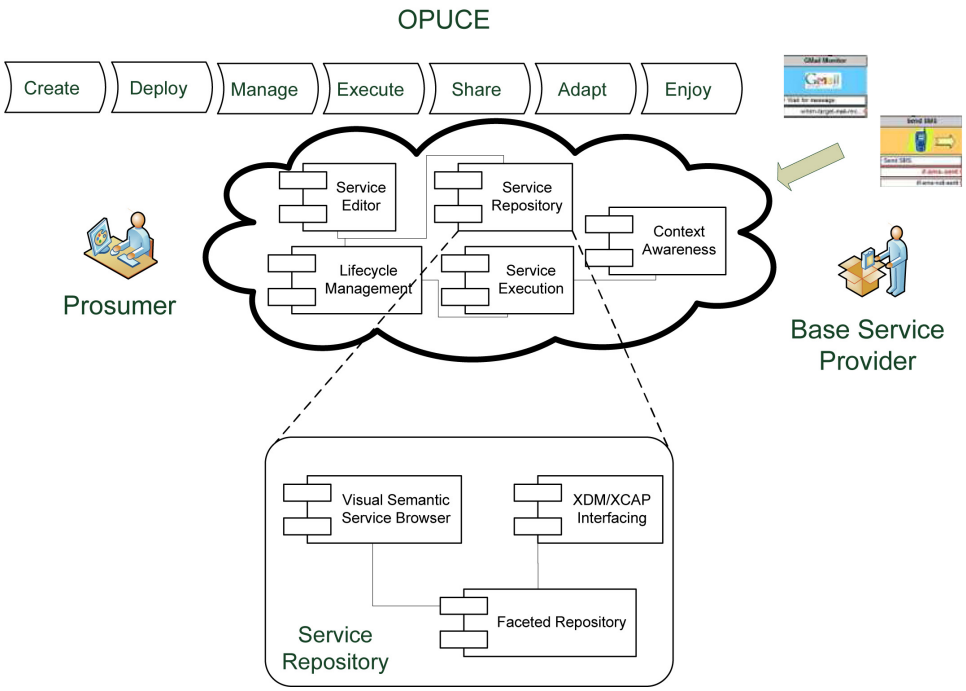

Figure 1. The OPUCE platform

and user services. OPUCE base services can be implemented using different technologies such as JSLEE, J2EE, .NET or even legacy technologies as long as they are encapsulated as WSDL-interfaced Web services. The main components of the Service Execution Environment are the Event Gateway and the BPEL Engine. The Event Gateway reflects the event-driven nature of the telecommunications applications. It is the endpoint for all the event notifications generated by the base services and will forward these notifications to the BPEL Engine which serves as the service logic execution engine.

- User Information Manager - It stores information about users. Specifically, five groups of user information, including User Profile, User Context (such as location, presence, device capability, network condition), Service Usage, Device Usage, and User Preferences are kept to be used by the Service Advertising and Context-Awareness components.
- Service Advertising - It recommends services to end users based on both explicit user subscription to service categories or keywords, and intelligent matching of user profiles with service descriptions.
- Service Repository - It stores the descriptions of both the OPUCE base services and user services. The repository comes with a powerful search capability supporting both keyword-based and semantic search.
- Context-Awareness - It allows the dynamic adaptation of services according to the information retrieved from the profiles within the User Information Manager. It performs the necessary changes in the service logic and/or the data handled in order to adapt the service to the context of each user.

As for the repository itself, it can be divided into three modules: the `Faceted Repository` that serves as a common registry for publishing and retrieving service facets [26] ; the `Visual Semantic Service Browser` that provides intuitive service exploring and discovering functionalities; and the `XDM/XCAP Interfacing` module that enables accessing the repository from handheld devices.

III. The OPUCE Visual Semantic Service Browser

A. Overview

One of the key functionalities of the service repository is to support two important user tasks: *exploring services* and *discovering services*. Herein exploring is the process of navigating through available services and acquiring important knowledge of them. And discovering is the process of locating desired services that satisfy the criteria specified by the user. Both of the two tasks heavily rely on how the characteristics of services are represented, organized, and rendered. An ontology is a formal explicit description of concepts, or classes in a domain of discourse [15]. Various features and attributes of a class can be described with the properties of the class. In the service-oriented computing area, using ontologies to describe the formal semantics of

Web services has proven its effectiveness in high-precision service discovery [17], [11] and automatic service composition [24]. To facilitate service exploring and discovering, we define the OPUCE Ontology which is based on OWL-DL, a sublanguage of the Web Ontology Language (OWL) W3C standard that is named for its correspondence with description logics. OWL-DL achieves a balance between expressiveness and computational completeness [14]. Reasoning on OWL-DL is decidable and supported by various DL reasoning engines, e.g. Pallet [2] and Racer [3]. And the ability of OWL-DL to define new classes based on Boolean combinations, such as intersection and union, and property restrictions, such as existential restriction, is essential in our service discovering approach.

Ontologies only answer part of our questions on how to represent characteristics of services. The exploring and discovering of large information spaces is still a difficult task, especially if the user is not familiar with the terminology used to describe information and the query language used to search specific information [9]. Work on enhancing Web service discovery framework—such as UDDI—with semantic features [11], [21] cannot solve the problem since the user still needs to write the query in a dedicated ontology language, or to fill a template with terms defined in an ontology to find services. To cross this hurdle, we develop the OPUCE Visual Semantic Service Browser (OPUCE Browser in short) that provides intuitive visual interfaces for users to easily carry out the tasks of service exploring and discovering. As the most significant feature, the browser does not require users to have any prior knowledge of ontology languages or domain terms, which highly pronounces the theme of the OPUCE project: user-centricity.

In the next subsections, we first introduce the OPUCE Ontology and then describe the technical details of the OPUCE Browser.

B. The OPUCE ontology

The OPUCE Ontology is an OWL-DL based semantic model that serves as the basis for representing and reasoning upon characteristics of services. It is designed considering the significant service characteristics that should be included in the context of OPUCE. In general, the service properties defined in the ontology can be divided into *functional properties* and *non-functional properties*. Next we describe the OWL object properties associated with the `Service` concept in the OPUCE Ontology.

Functional Properties.

- `hasInput` and `hasOputput`:Just like most service semantic models such as OWL-S [13] and FUSION [11], inputs and outputs (IO) are introduced to represent the set of parameters that a service expects to receive and the set of parameters that a service will produce if invoked. Furthermore, these two properties are grouped as the sub-properties of the `hasParameter` property. It is worth noting that we do not include the other popular properties preconditions and effects (PE) that represent the state-wise conditions that need to hold before and after the service invocation. PE are mainly used in the situation of automatic service composition and thus out of the scope of OPUCE.
- `hasFunctionality`: Functionality represents the 'functional ability' property of services. In [19], functional abilities are modeled as a list of actions using pairs of <verb, noun>. Since there is overlapping between the nouns and the service IO parameters (e.g. message, location, context), we only use verbs to capture the service functionalities. The range of `hasFunctionality` is the `Functionality` concept, and a list of subclass of `Functionality`, such as `Send`, `Search`, `Translate`, are defined in the ontology. So it is flexible either using a single verb (e.g. send) or using a verb with an IO concept (e.g. send message) to define a service functionality.

Non-Functional Properties.

Non-functional properties may relate to quality of services (QoS), policy compliance, adherence to technical standards or protocols, or categorisation within a classification scheme [11]. We define three non-functional properties for Service:

- `hasCategory`: This property attaches services with some semantically represented classification scheme for course-grained filtering.
- `hasQoS`: QoS represents features such as performance, reliability, security, integrity, and cost. Since much work has been done on defining a formal QoS ontology for services [18], [6], we use this property as a placeholder for further extension and integration.
- `hasTechnology`: This property identifies the technical standards or protocols for implementing services. The reason why we define this property is that OPUCE is a platform spanning both the telecommunications and the information technology domain. A clear statement of what technology is used by a certain service helps users get a better understanding of the service. For example, a user may either select a VOIP call service or a PSTN call service based on its current situation.

Except for the above properties of the Service concept, two subclasses of Service, `BaseService` and `CompositeService`, are defined to separate the basic service provided to users as building blocks and the value-added services created using the OPUCE platform.

[2]Pellet: http://clarkparsia.com/pellet/

[3]Racer: http://www.racer-systems.com/

Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer
complementOf	$\neg C$	¬Male
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}
allValuesFrom	$\forall P.C$	∀hasChild.Doctor
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer
maxCardinality	$\leqslant nP$	⩽1hasChild
minCardinality	$\geqslant nP$	⩾2hasChild

Figure 2. OWL class constructors

C. Describing and querying OPUCE services

To describe the characteristics of a service, we can build a new OWL class as the service's semantic model (we call it OPUCE semantic service) using OWL-DL class constructors and the OPUCE Ontology.

As illustrated in Figure 2, we can construct a new class from existing classes, properties and individuals by:

- applying set operators including intersection, union, and complement on classes;
- explicitly and exhaustively enumerating the individuals that are members of the new class;
- restricting a property of the class: the range of the property either has all the individuals from a specific class (universal restriction) or has some individuals in a specific class (existential restriction);
- or restricting the cardinality of a property.

Using the OWL-DL class constructors, an OPUCE semantic service can be built by first specifying whether it is a basic or a composite service, and then restricting its functional and non-functional properties one by one. For example, a `SendSMS` basic service can be described as:

$$SendSMS \equiv BaseService \sqcap \leq 2hasInput \sqcap \geq 2hasInput \sqcap \exists hasInput(Text) \sqcap \exists hasInput(MobilePhoneNumber) \sqcap \exists hasFunctionality(Send) \sqcap \exists hasCategory(Telecom) \sqcap \exists hasTechnology(J2EE).$$

The above statement says that `SendSMS` is a basic service, and it has exactly two inputs: one is a `Text` type and the other a `MobilePhoneNumber` type, and its functionality is `Send`, and it belongs to the `Telecom` domain, and one of its associated technology is `J2EE`.

With all the OPUCE semantic services store in a knowledge base, we can discover services using a DL reasoner. For example, if we want to find all the `Telecom` services with functionality `Send`, we can write the query in an OWL-DL class as:

$$\exists hasCategory(Telecom) \sqcap \exists hasFunctionality(Send).$$

And then send it to the DL reasoner for matchmaking. Clearly `SendSMS` is on the match list since it is subsumed by the above class.

As we can find, discovering services by directly querying the DL reasoner has the following obstacles for a user:

1) She must know the query language (in our case OWL-DL).
2) She must be familiar with the domain terminology.
3) Some intended services may not be discovered because of the subtle semantic differences of OWL-DL statements. For example, existential restrictions and universal restrictions are two different property restriction class constructors, and two classes only differ in restriction type, e.g. $\exists hasCategogy(Telecom)$ and $\forall hasCategogy(Telecom)$, cannot match each other since they do not have a subsumption relationship [14].

Although some semantic matchmaking approaches such as [11] and [21] use query templates to alleviate Problem 1 and 3, they still require the user to have knowledge of the domain terminology to fill the template. In the next subsection we discuss our visual semantic service browser that aims at solving the above-stated problems and giving the users an intuitive visual service discovering and exploring environment.

D. Visually exploring and discovering OPUCE services

Through the above analysis, we can reach the observation that the characteristics of a service are derived mainly by adding restrictions to its properties, and the intersection (or logical 'AND') of these restrictions gives a refined definition/semantics to this service. And the task of discovering services amounts to finding services that satisfy some property restrictions, e.g. *discovering `Telecom` (`hasCategory` restriction) services that can `send` (`hasFunctionality` restriction) `text` (`hasInput` restriction).*

Based on the above principle, the OPUCE Browser first collects all the property restriction classes and the services each class subsumes, then graphically display them for visual exploring and discovering.

The algorithm illustrated in Figure 3 formalizes the process of generating the set of property restriction classes: for each semantic service, we first recursively extract all the component classes separated by the intersection operator ('⊓') from its semantic description; and then we go through all the extracted classes. If a class is a property restriction, then the DL reasoner will be consulted to get the class's subsumed services.

Next we describe the graphical interface of the OPUCE Browser. As illustrated in Figure 4, the browser is divided into three panes. The left pane is an indented list containing all the property restriction classes; since service properties have been predefined in the OPUCE Ontology, it is self-evident to use them as top-level elements and attaching the associated restrictions under. For every element on the indented list, there is a number on its right indicating how many services it contains. For example,

```
PROC GENERATE-PROPERTY-RESTRICTION-SET()
Input: the set of semantic services S, where s_i.DL_Desc is its semantic description string
Input: the DL Reasoner DLR
Output: the set of Property Restriction Classes P, where P.S is the set of services that can be subsumed by P
Auxiliary: sets of DL-Class V and T
Begin
  V,T,P← ∅
  for each s_i in S do
    V← EXTRACT-INTERSECTED-CLASS(s_i.DL_Desc)
    T←T∪V
  end for
  for each t_i in T do
    if t_i is a Property Restriction Class then
      t_i.S← DLR.FIND-SUBSUMPTION(t_i)
      P←P ∪ {t_i}
    end if
  end for
  DISPAY(P)
End
```

Figure 3. Algorithm for generating property restrictions

under the `hasCategory` property, the `Telecom` restriction has 7 services. It is worth noting that we do not expose the difference between existential restrictions and universal restrictions to users, so in fact the `Telecom` restriction represents the class: $\exists hasCategogy(Telecom) \sqcup \forall hasCategogy(Telecom)$. To discover services, a user may browse through the list; and when a desired property restriction is found, she may select this restriction by clicking on the checkbox attached and this restriction will be shown in the middle pane as a piece of colored cloud with small balls inside indicating its subsumed services; and if we put the mouse cursor on top of a ball, a tooltip will be shown indicating a URL linking to the semantic description of this service. If more than one property restrictions are selected, each one will be shown in the middle pane as a piece of cloud but in different colors. And the overlapping of clouds visually indicates the intersection of property restrictions. For example in Figure 4, the middle pane shows three property restrictions: `hasCategory(Telecom)`, `hasInput(text)` and `hasFunctionality(Send)`; and the intersection of these restrictions, which answers the discovering question of *what Telecom base services can send text message*, is visually shown as the overlapping of four pieces of cloud: `Telecom`, `Text`, `Send`, and `BaseService` that contains two services: `SendSMS` and `SendIM`. The right pane is an enhanced feature to facilitate fast locating elements on the indented list: instead of browsing, if a user enters a 'keyword' in the right-bottom input-box, all the elements containing this 'keyword' will be listed for selection.

In summary, the OPUCE Browser provides a visual environment to facilitate service exploring and discovering. Without knowing any ontology language and domain terminology, a user can discover a desired service based on its properties by several clicks.

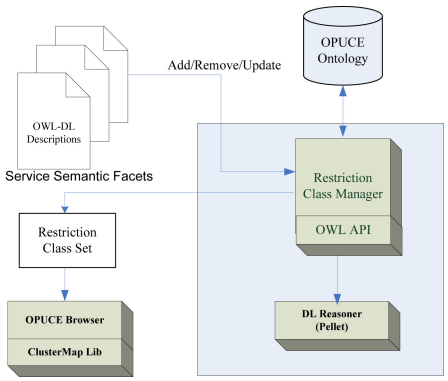

Figure 5. OPUCE Browser Architecture

E. Implementation

Figure 5 shows the architecture of the OPUCE Browser. On the server side, The core part is the Restriction Class Manager that is built on top of OWL API [4], which is a Java interface and implementation for OWL. And Pellet [5] is used as the OWL-DL reasoner. The Restriction Class Manager provides interfaces for the user to add, update, and remove semantic service descriptions. Whenever a new semantic service is added, updated, or removed, the manager will consult Pellet and update the Restriction Class Set, which is the input of the OPUCE Browser GUI. The GUI is built on top of ClusterMap Library [6], a visual technique which has been used in several Semantic web applications [9]. A snapshot of the GUI in Figure 4 has been discusses in the previous subsection.

IV. EVALUATION

The OPUCE Browser together with the service repository has been up and running in the OPUCE platform.

[4]http://owlapi.sourceforge.net/
[5]http://clarkparsia.com/pellet/
[6]http://www.aduna-software.com/technologies/clustermap/overview.view

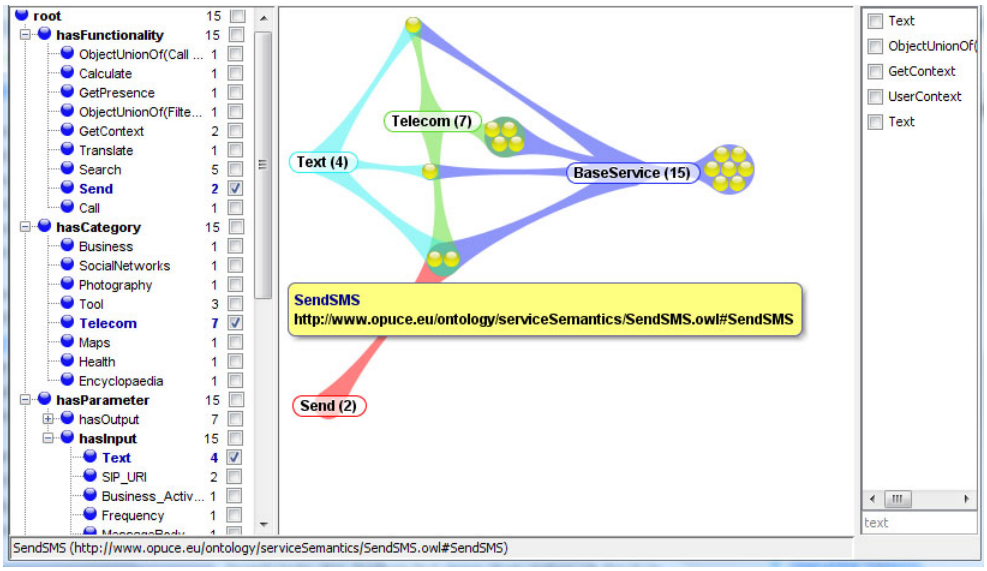

Figure 4. OPUCE Browser Snapshot

To test and evaluate the repository, we defined totally 15 OPUCE services including `SendSMS`, `MailMonitor`, and `PhotoSearch` etc. The repository is efficient and convenient in terms of performance and ease of use. For the OPUCE Browser, it is worth noting that the process of exploring and discovering services using the OPUCE Browser GUI is very fast since during the process there is no need to interact with the DL engine: the information on property restriction classes and the associated services each class contains has been generated and updated after each add/update/remove operation on semantic service descriptions. Such design is similar to the 'eager' approach discussed in [11], which takes place at publication-time.

Enabling end users to manage service descriptions on handheld devices dramatically brings convenience and then promotes faster creation and delivery of services to other users. The OPUCE Browser contributes to usability mainly through its visual service exploring and discovering mechanism. The Browser GUI proved to be intuitive for technically non-experienced people: exploring services with indented list is natural to most end users since they have been accustomed to it in everyday tasks, like scanning the contents at the beginning of a book or writing down a list of tasks they have to perform [10]; discovering services can be done visually with several clicks of the mouse. To do so, end users are not required to have a high level of technical expertise or even basic knowledge of ontology and semantic Web.

Apart from the above apparent benefits, we also identified one weaknesses of the current OPUCE Browser: clearly, when generating the restriction classes from OWL-DL, only selected OWL-DL class constructors are considered, which means the expressiveness of the OPUCE Browser is weaker than OWL-DL and the information graphically rendered is a projection of the original semantic services. But since the OPUCE Ontology has defined a framework with great emphasis on service properties, the OPUCE Browser still can capture the essential information of OPUCE semantic services.

V. Related work

With the prevalence of Service-Oriented Computing paradigm in both the Internet and telecom domains, research on repositories for convenient and efficient managing and discovering of services has gained momentum. Specifications such as UDDI [5] and ebXML Registry [4] provide fundamental support for registering, discovering and integrating services. Recognizing the limited capabilities in offering accurate service discovery facilities of these specifications, a rich body of work have been reported in the literature aiming at enhancing service repositories with ontology and semantic discovery facilities. In [20], the authors discuss the major shortcomings of state-of-the-art Web service repositories such as limited browsing and discovering capabilities and suggest solutions using Semantic Web related techniques. In [16], the authors propose a semantic repository for discovering and selecting services in the e-Government domain.

In [7], the authors enriches ebXML registries with OWL by mapping various constructs of OWL to ebXML classification hierarchies and then services can be queried through standardized ebXML query facilities. A first work on adding semantic matching capability to UDDI appears in [17] where the authors propose a matchmaking engine inside the UDDI registry to match service capability descriptions encoded in DAML-S [3] Profile. It's follow-up work reported in [23] uses OWL-S Profile and also improves the matchmaking

algorithm. Another two works applying DAML-S or OWL-S on UDDI are reported in [1] and [12]. Recently, with the emerging of WSDL-S [2] and SAWSDL [8], in [22], the authors map WSDL-S semantic annotations to UDDI and in [11], the authors propose the FUSION Semantic Registry which augments the UDDI's service publication and discovery facilities based on SAWSDL and OWL-DL.

In contrast, our work are based on ebXML Registry and OWL-DL with great emphasis on providing an intuitive visual interface for facilitating discovering and also exploring of semantic services.

Our visual service browser is built on top of ClusterMap Library. In [9], the authors report three applications—the DOPE Browser for exploring large online resources in the domain of drugs and diseases, a peer-to-peer knowledge management platform SWAP, and AutoFocus for managing personal information sources—that also build the user interface based on ClusterMap Library.

VI. Conclusion

In this paper, we have presented the OPUCE visual semantic service browser inside the service repository. In line with the user-centric theme of the OPUCE project, the repository supports accessing facets of service descriptions from handheld devices. And most importantly, we enhance the repository with a visual service browser by adopting ontology and semantic Web technology. The browser is intuitive and efficient in a way that it does not require the user to have any knowledge of ontology or semantic Web and the task of exploring and discovering services can be easily done with several steps of mouse-clicking.

In the future, we plan to investigate and integrate ontology learning techniques into the repository to alleviate the burden of manual ontology authoring. We also plan to apply our visualization technique on other service ontologies other than the OPUCE Ontology to test its effectiveness.

Acknowledgments

This work is partly funded by research project OPUCE, under Information Society Technologies (IST) priority of the 6th Framework Program of the European Community, Contract No. 34101. We thank all our partners in the project for their valuable comments to this paper.

References

[1] R. Akkiraju, R. Goodwin, P. Doshi, and S. Roeder. A method for semantically enhancing the service discovery capabilities of uddi. In *2003 Workshop on Information Integration on the Web (IIWeb)*, pages 87–92, 2003.

[2] R. Akkiraju et al. Web service semantics - WSDL-S. W3C Member Submission, November 2005.

[3] Anupriya Ankolekar et al. DAML-S: Web service description for the semantic web. In *2002 International Semantic Web Conference (ISWC)*, volume 2342 of *LNCS*, pages 348–363. Springer, 2002.

[4] K. Breininger, F. Najmi, and N. Stojanovic (eds.). The ebXML registry repository version 3.0.1. OASIS, Febuary 2007.

[5] L. Clement, A. Hately, C. von Riegen, and T. Rogers (eds.). Uddi version 3.0.2 specification. UDDI Specification Committee, October 2004.

[6] Glen Dobson, Russell Lock, and Ian Sommerville. QoSOnt: an ontology for QoS in service-centric systems. In *2005 UK e-Science All Hands Meeting*, pages 80–87, 2005.

[7] A. Dogac, Y. Kabak, and G.B. Laleci. Enriching ebXML registries with OWL ontologies for efficient service discovery. In *14th International Workshop on Research Issues on Data Engineering: Web Services for e-Commerce and e-Government Applications*, pages 69–76, 2004.

[8] J. Farrell and H. Lausen (eds.). Semantic annotations for WSDL and XML Schema. W3C Recommendation, August 2007.

[9] Christiaan Fluit, Marta Sabou, and Frank van Harmelen. *Ontology-Based Information Visulization: Toward Semantic Web Applications*, chapter 3, pages 45–58. Springer, 2004.

[10] Akrivi Katifori and Constantin Halatsis. Ontology visualization methods—a survey. *ACM Computing Surveys*, 39(4):1–43, 2007.

[11] Dimitrios Kourtesis and Iraklis Paraskakis. Combining SAWSDL, OWL-DL and UDDI for semantically enhanced web service discovery. In S. Bechhofer et al., editor, *5th European Semantic Web Conference (ESWC 2008)*, volume 5021 of *LNCS*, pages 614–628. Springer-Verlag Berlin Heidelberg, 2008.

[12] J. Luo, B. Montrose, A. Kim, A. Khashnobish, and M. Kang. Adding owl-s support to the existing uddi infrastructure. In *2006 IEEE International Conference on Web Services (ICWS)*, pages 153–162, 2006.

[13] David Martin et al. OWL-S: Semantic markup for web services. W3C Member Submission, November 2004.

[14] D.L. Mcguiness and F. van Harmelen. OWL web ontology language overview. W3C Recommendation, February 2004.

[15] N.F. Noy and D.L. McGuiness. Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory Technical Report, 2001.

[16] Matteo Palmonari, Gianlugii Viscusi, and Carlo Batini. A semantic repository approach to improve the goverment to business relationship. *Data and Knowledge Engineering*, 65:485–511, 2008.

[17] M. Paolucci, T. Kawamura, T.R. Payne, and K. Sycara. Semantic matching of web service capabilities. In I. Horrocks and J. Hendler, editors, *2002 International Semantic Web Conference (ISWC)*, volume 2342 of *LNCS*, pages 333–347. Springer, 2002.

[18] I.V. Papaioannou, D.T. Tsesmetzis, I.G. Roussaki, and M.E. Anagnostou. A QoS ontology language for Web-services. In *20th International Conference on Advanced Information Networking and Applications (AINA 2006)*, pages 101–106, 2006.

[19] R. Prieto-Diaz and P. Freeman. Classifying software for reusability. *IEEE Software*, 4(1):6–16, January 1987.

[20] Marta Sabou and Jeff Pan. Towards semantcally enhanced web service repositories. *Journal of Web Semantics*, Science, Services and Agents on the World Wide Web 5:142–150, 2007.

[21] Amit P. Sheth, Karthik Gomadam, and Ajith Ranabahu. Semantics enhanced services: METEOR-S, SAWSDL and SA-REST. *IEEE Data Engineering Bulletin*, 31(3):8–12, 2008.

[22] K. Sivashanmugam, K. Verma, A. Sheth, and J. Miller. Adding semantics to web services standards. In *2003 International Conference on Web Services (ICWS)*, pages 395–401, 2003.

[23] Naveen Srinivasan, Massimo Paolucci, and Katia Sycara. Adding OWL-S to UDDI, implementation and throughput. In *1st Intl. Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, pages 6–9, 2004.

[24] Katia Sycara, Massimo Paolucci, Anupriya Ankolekar, and Naveen Srinivasan. Automated discovery, interaction and composition of semantic web services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1):27–46, December 2003.

[25] Jian Yu, Paolo Falcarin, Jose M. del Alamo, Juergen Sienel, Quan Z. Sheng, and Jose F. Mejia. A user-centric mobile service creation approach converging telco and it services. In *8th International Conference on Mobile Business*, pages 238–242, 2009.

[26] Jian Yu, Paolo Falcarin, Sancho Rego, Isabel Ordas, Eduardo Martins, Quan Sun, Ruben Trapero, and Quan Z. Sheng. Xdm-compatible service repository for user-centric service creation and discovery. In *7th IEEE International Conference on Web Services*, pages 992–999, 2009.