Data & Knowledge Engineering 72 (2012) 202–218

Contents lists available at SciVerse ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

A semantically enhanced service repository for user-centric service discovery and management

Jian Yu [a,*], Quan Z. Sheng [b], Jun Han [a], Yanbo Wu [b], Chengfei Liu [a]

[a] Faculty of Information & Communication Technologies, Swinburne University of Technology, P.O. Box 218, Hawthorn, VIC 3122, Australia
[b] School of Computer Science, The University of Adelaide, SA 5005, Australia

ARTICLE INFO

Article history:
Received 14 May 2010
Received in revised form 28 October 2011
Accepted 31 October 2011
Available online 10 November 2011

Keywords:
Service repository
User-centricity
Service ontology
Semantic services
Visualized service discovery

ABSTRACT

User centricity represents a new trend in the currently flourishing service oriented computing era. By upgrading end-users to *prosumers* (*producer* + *consumer*) and involving them in the process of service creation, both service consumers and service providers can benefit from a cheaper, faster, and better service provisioning. The EU-IST research project OPUCE (Open Platform for User-Centric Service Creation and Execution) aims at building a unique service environment by integrating recent advances in networking, communication and information technology where personalized services can be dynamically created and managed by prosumers. This paper particularly discusses the design and development of a service repository, which is at the very core of the OPUCE platform. The repository consists of two main components: a fully functioned XML registry supporting facet-based access to service descriptions, and a novel semantic service browser that supports prosumers who are not technically experienced to explore and discover services in an intuitive and visualized manner. We demonstrate the benefits of our design by conducting usability and performance studies.

1. Introduction

The service-oriented computing paradigm, which provides an effective means of application abstraction, integration and reuse with its loosely-coupled architecture, presently has a dominant position in developing Web-based information systems [28,56]. On the other hand, with the proliferation of Web 2.0 techniques and applications such as blogs, wikis, tagging systems and mashups [34,35], the notion of *user-centricity* has gained a significant momentum to put ordinary users in the leading role of delivering exciting and personalized content and services. The term *prosumer*—which was first coined by the futurist Alvin Toffler in 1980 to describe a type of consumer involved in the design and manufacture of products so they could be made to individual specification [52]—recently has been specifically promoted to define such a phenomenon.

The European Union sponsored IST-FP6 research project OPUCE[1] (Open Platform for User-Centric Service Creation and Execution) aims at advancing the current trend a step further by developing a unique platform for prosumers to create and publish mobile–accessible personalized services that integrate the capability of both telecom services and Web-based software services. For example, an OPUCE prosumer[2] could create a context-aware service monitoring her email account, which will send incoming email to her mobile phone as an SMS if she is in a meeting, or read them out as a voice call if she is driving.

* Corresponding author.
E-mail addresses: jianyu@swin.edu.au (J. Yu), qsheng@cs.adelaide.edu.au (Q.Z. Sheng), jhan@swin.edu.au (J. Han), yanbo.wu@adelaide.edu.au (Y. Wu), cliu@swin.edu.au (C. Liu).

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

[2] In this paper, an OPUCE prosumer is an end-user of OPUCE services, and she also uses the OPUCE platform to create OPUCE services.

0169-023X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2011.10.005

The service repository is a key component of the OPUCE platform. On the one hand, the repository is responsible for storing and managing all sorts of information related to services, i.e., service descriptions. On the other hand, the repository also needs to provide functions to facilitate the access and discovery of services. Some significant challenges need to be tackled in implementing such a service repository in a user-centric context. For example, how to facilitate the management of large amounts of service descriptions by prosumers who are not technically experienced? How to facilitate the exploration and discovery of services by prosumers?

The OPUCE service repository is our answer to the above-mentioned challenges. The significant features of the repository are twofold. Firstly, to facilitate prosumers in an effective management of service descriptions, we propose a *faceted* approach to describe services and to store service description facets. A dual-interface approach also enables a prosumer to access and manage service descriptions/facets either from personal computers through the Internet or from handheld devices through a mobile network. By faceted, we mean that service descriptions are logically divided into parts where each part can be stored and retrieved separately without interfering with other parts. For example, a service could have an interface facet and a composition logic facet from function perspective, and a provisioning facet, a deployment facet and a semantic facet from management perspective. Our faceted approach brings benefits on modularity and flexibility (i.e., new facets could be added freely and each facet could have its own description method and language).

Secondly, we leverage the semantic Web technology to design and implement a novel service browser called OPUCE Visual Semantic Service Browser (OPUCE Browser in short). Although semantic Web based techniques can dramatically improve the precision of information and service discovery [39,41,5], how to apply these techniques in a user-centric environment is still a challenge because common prosumers may not have the knowledge of any ontology definition language such as OWL [33] and may lack of a clear picture of the domain terminology, both of which are required to compose a valid semantic query statement. The OPUCE Browser supports prosumers to visually explore and discover OPUCE services using an intuitive point-and-click interface and does not require technical knowledge of ontology or semantic Web from them. The task of exploring and discovering services could be done simply with several mouse-clicks. To implement the OPUCE Browser, we developed an "eager" approach—in contrast to the "lazy" approach where the semantic service description knowledge base is searched at the time a service query is issued—that preprocesses a semantic service description whenever it is published and then presents the output information in a graphical user interface based on the ClusterMap Library[3]. Performance and usability study also have been conducted to demonstrate the scalability and effectiveness of the browser. It is worth noting that the focus of this paper is on the OPUCE Browser.

The rest of the paper is organized as follows. Section 2 gives a brief introduction to the OPUCE platform and the general architecture of the service repository. Section 3 discusses the faceted approach of the repository, focusing on the design of the OPUCE Service Description Language. Section 4 is dedicated to the OPUCE Visual Semantic Service Browser, where we detail the OPUCE Ontology, the semantic descriptions of OPUCE services, and the principles and implementation of this approach. Section 5 reports some evaluation results. In particular, we focus on the performance and usability of the browser. Finally, Section 6 discusses related work and Section 7 concludes the paper.

2. OPUCE platform overview

User centricity is a core theme of the OPUCE platform, which is reflected in the whole lifecycle of the service delivery process. As illustrated in Fig. 1, prosumers are actively involved in the whole service lifecycle: they take the role of both *service creator* and *service user* [40], and use the OPUCE platform not only to create, deploy, test, and share services, but also to recommend, adapt, and run services. It is worth noting that telecom operators and software service providers are responsible to encapsulate telecommunications and IT resources as *OPUCE base services*—the building blocks for prosumers to create *OPUCE composite services* [54].

The OPUCE platform consists of the following main components:

- *Portal* integrates the graphical user interfaces of different modules through which prosumers and administrators can perform management and service creation tasks. The web-based *Advanced Service Editor* is the main working space for prosumers to create OPUCE composite services.
- *Service Lifecycle Management* manages the entire lifecycle of services, including deployment, provisioning, and monitoring. It also hosts the *Service Description Translator*, which is responsible for translating the OPUCE composite service descriptions into executable WSBPEL (Web Services Business Process Execution Language) scripts.
- *Service Execution Environment* hosts and runs the executable code of both OPUCE base services and composite services. OPUCE base services can be implemented using different technologies such as JSLEE (Java Service Logic Execution Environment), J2EE (Java 2 Platform, Enterprise Edition), .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 *WSBPEL 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 OPUCE base services and will forward these notifications to the WSBPEL Engine which serves as the service logic execution engine [9].

[3] http://www.aduna-software.com/technologies/clustermap/overview.view.

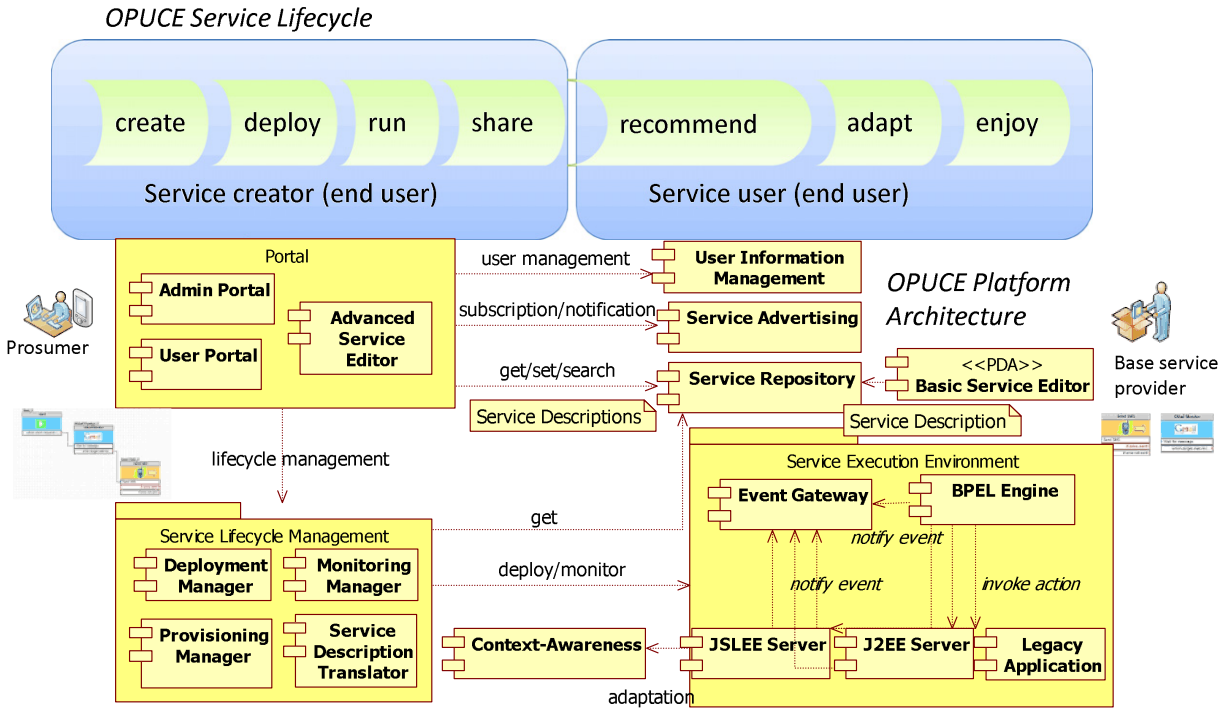


Fig. 1. OPUCE platform overview.

- *User Information Manager* stores information about users—including prosumers and pure end-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 the *Context-Awareness Components*.
- *Service Advertising* 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.
- *Context-Awareness* 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 behavior and/or the data handled in order to adapt the service to context of the each user.
- *Service Repository*'s architecture is illustrated in Fig. 2, consisting of four main components: i) the *Faceted Repository* that serves as a common registry for publishing and retrieving service facets, ii) the *Visual Semantic Service Browser* that provides a graphical service exploration and discovery environment, iii) the *XDM/XCAP Interface* (XML Document Management/XML Configuration Access Protocol) that enables accessing the repository from handheld devices, and iv) the *Web Service Interface* that enables other components of the OPUCE platform to access the repository in a distributed and standardized manner. In the next section, we will briefly introduce the Faceted Repository and the XDM/XCAP Interface, and we will describe the Visual Semantic Service Browser in detail in Section 4.

3. Facet-based approach for describing OPUCE services

The faceted approach of the OPUCE service repository is reflected in the design of the service description language in the first place. A facet is a view over a service property that provides a partial description of a service [12]. Each facet addresses a specific aspect of a service, describing either functional, non-functional, or management properties. For example, an OPUCE service may have functional facets (e.g., logic facet and interface facet), non-functional facets (e.g., behavioral constraints facet and security facet), and management facets (e.g., accounting facet and provisioning facet) [53]. The main advantages of the faceted approach are its modularity and flexibility in the sense that new facets can be added on-the-fly as long as it is necessary. Furthermore, each facet can be described using a different language, either standardized or customized.

Fig. 3 shows the structure of the OPUCE service description language [53]. The left side of the figure represents the master document of the service, which includes general purpose metadata, or *information about one or more aspects*,[4] of the service, such as `service name`,[5] `version`, `creator name`, or `unique identifier`. This master document also contains a list of the facets

[4] http://en.wikipedia.org/wiki/Metadata.
[5] `Service name` could be in different natural languages, e.g., `Send SMS` in English, or `Envio de SMSs` in Spanish.

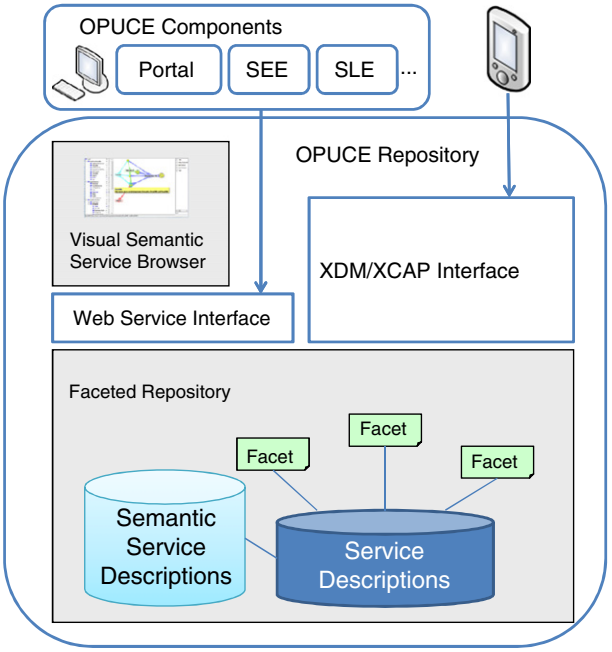


Fig. 2. Architecture of the OPUCE repository.

included in the service description. Each facet contains its type and a link to the document where the complete facet description is. The right side of the figure represents the structure of a facet description document. Facet description documents are saved separately, so they need an identifier that must be the same as that of the service they belong to. The facet specification language contains the format of the language used in the facet specification data. For example, the facet representing the logic of an OPUCE service is described in the WSBPEL language, thus the facet specification language contains a reference to the WSBPEL language and the facet specification contains the WSBPEL script.

The provisioning facet is an indispensable part of the description for an OPUCE base service. It contains important setup information on how to make a base service up and running. In Fig. 4, we have demonstrated how to specify the provisioning facet for a base service called `SendSMS`. In this provisioning facet, the `TargetID` tag gives a unique name to the base service to be provisioned. The value of the `ExecutionMode` tag could either be `synchronous` or `asynchronous`. The `rdfFile` tag specifies the file used in

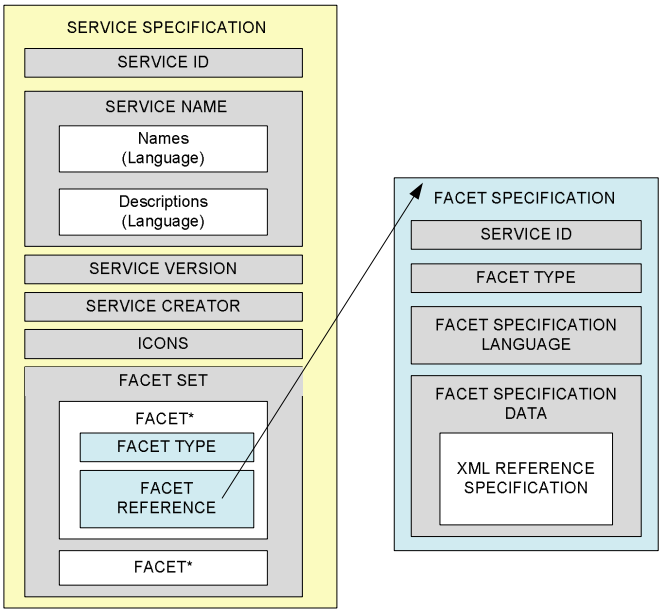


Fig. 3. The structure of the OPUCE service description language.

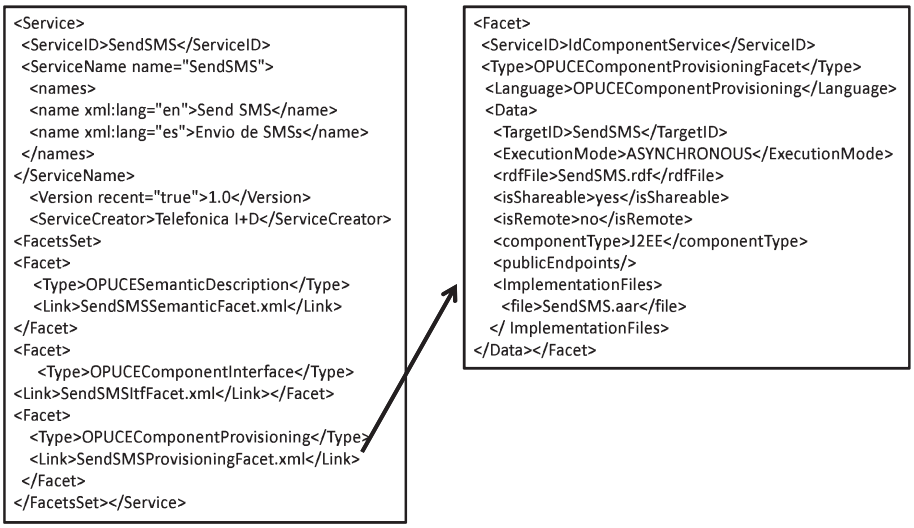


Fig. 4. The provisioning facet and master document for the `SendSMS` base service.

deployment. `isShareable` indicates whether to create a new instance of the base service, and `isRemote` indicates whether to install the base service on the remote handheld device or on the server. `componentType` specifies the component technology that is used to implement this base service. The `publicEndpoints` tag specifies the external access point for the base service, and final `serviceImplementation` gives the details of implementation such as the executable file archive to be deployed.

The Faceted Repository module of the OPUCE service repository is built on top of ebXML Registry,[6] which is a powerful tool for managing XML data. The ebXML Registry in fact consists of two parts: an XML data storage and a registry. On top of the storage, we define and implement the faceted access interfaces based on Web service protocol to facilitate remote publishing and retrieving of service description facets. The registry part manages metadata using a Registry Information Model (RIM). Basic service metadata such as creator, version, and tags are registered in the RIM to support basic keyword-based search.

To facilitate accessing the repository from handheld devices, we also develop the XDM/XCAP Interface module [55] that uses the XDM/XCAP protocol defined by OMA[7] as a wrapper around the ebXML Registry based repository. In this module, the XDM server provides a SIP (Session Initiation Protocol) event framework interface, which is supported by most modern smart phones, to subscribe/notify changes in XML documents.

4. The OPUCE visual semantic service browser

Efficiently supporting prosumers in exploring and discovering services is a key requirement in designing the OPUCE service repository. Herein exploring services is the process of navigating through available services and acquiring important knowledge of them, while discovering services is the process of locating desired services that satisfy the criteria specified by the user. Both tasks heavily rely on how the characteristics of services are represented, organized, and rendered. In computer science, the term *ontology* has numerous descriptions and definitions. For example, it is defined as "*a formal*, *explicit specification of a shared conceptualisation*." [21], "*a computer model of some portion of the world*." [23], "*a shared and common understanding of a domain that can be communicated between people and heterogeneous and distributed systems*." [17], and more formally "*a logical theory accounting for the intended meaning of a formal vocabulary*, i.e. *its ontological commitment to a particular conceptualization of the world*." [22]. In the context of this paper, we use the definition given by Arvidsson and Flycht-Eriksson: "*An ontology provides a shared vocabulary*, *which can be used to model a domain—that is*, *the type of objects* (*classes*) *and/or concepts that exist*, *and their properties and relations*" [4], which gives an insight to the common constructs of the Web ontology language OWL. In the service-oriented computing area, using ontologies to describe the formal semantics of Web services has proofed its effectiveness in high-precision service discovery [41,25] and automatic service composition [51]. To facilitate service exploration and discovery, we define the OPUCE Ontology based on OWL-DL, a sub-language 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 [33]. Reasoning on OWL-DL is decidable and supported by various DL reasoning engines such as Pallet[8] and Racer.[9] The ability of OWL-DL to define new classes based on Boolean combinations (e.g., intersection and union) and property restrictions (e.g., existential restriction) is essential in our service discovery approach.

[6] www.oasis-open.org/committees/regrep/.
[7] http://www.openmobilealliance.org/
[8] http://clarkparsia.com/pellet/.
[9] http://www.racer-systems.com/.

However, ontologies only answer part of our questions on how to represent characteristics of services. The exploration and discovery 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 for specific information [18]. Existing works on enhancing Web service discovery framework—such as UDDI (Universal Description Discovery and Integration)—with semantic features [25,47] cannot solve the problem since users still need to write the query in a dedicated ontology language, or to fill a template with terms defined in an ontology when searching services.

To cross this hurdle, we have developed the OPUCE Visual Semantic Service Browser (OPUCE Browser in short) that provides an intuitive visual interface for users to easily carry out the tasks of service exploration and discovery. As the most significant feature, the browser does not require its users to have any prior knowledge of ontology languages or domain terms, which is pertinent to the theme of the OPUCE project: user centricity. In this section, we first introduce the OPUCE Ontology and then describe the technical details of the OPUCE Browser.

4.1. 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 by considering the significant service characteristics that should be included in the context of OPUCE. To promote user involvement and to keep the ontology a *live* knowledge base, user-defined folksnomies can be included in the ontology to tag services. It is worth noting that because the OPUCE Ontology is encoded in OWL-DL, it can be easily integrated into generic service upper ontology such as OWL-S [31].

In general, the service properties defined in the ontology can be divided into *functional properties*, *non-functional properties*, and *tagging properties*. Next we describe the OWL object properties associated with the `Service` concept in the OPUCE Ontology.

4.1.1. Functional properties

- *hasInput* and *hasOutput*: Just like most service semantic models such as OWL-S and FUSION [25], `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.
- *hasFunctionality*: In [44], 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.

4.1.2. Non-functional properties

Non-functional properties may relate to quality of services (QoS), policy compliance, adherence to technical standards or protocols, or categorization within a classification scheme [25]. 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 attributes such as security, performance, and reliability are always a major concern of modern information systems including service-oriented systems [19]. Since much work has been done on defining a formal QoS ontology for services [42,14], 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 information technology domains. 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 select a VOIP call service or a PSTN call service based on her current situation.

4.1.3. Tagging properties

- *hasTag*: Unlike the above properties which are used to describe a service with formally defined concepts, the `hasTag` property is meant to be used by a prosumer to freely tag a service created by herself in a personalized manner. It is worth noting that just like any Web 2.0 tagging system, all the tags added by one prosumer are shared among all prosumers, and to avoid potential conflicts between formally defined ontology and user-defined tags, all the tags cannot be used to describe the other properties in the OPUCE ontology.

Except for the above properties of the `Service` concept, two subclasses of `Service`, namely `BaseService` and `CompositeService`, are defined to separate the base service provided to prosumers as building blocks and the prosumer-created OPUCE services.

Fig. 5 shows a snapshot of the OPUCE Ontology and the semantic description of an OPUCE base service `GetLocation`.

Currently the OPUCE Ontology knowledge base contains about 870 concepts in the above discussed property categories. These concepts are manually constructed by Telecom domain experts in the project following the four-phase methodology (requirements analysis, building, implementation, and evaluation and maintenance) [38] based on the requirements document for more than one hundred services. These experts are from major Telecom companies in Europe and some of them have been involved in several European Union projects. They are very familiar with telecom businesses and services, and they also have

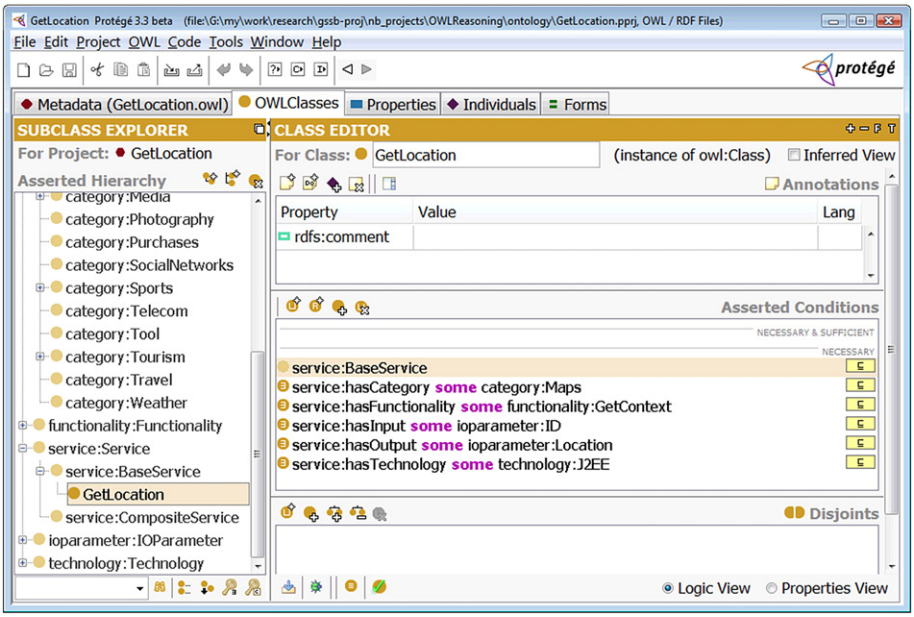


Fig. 5. A snapshot of the semantic description of the `GetLocation` service in *Protégé*.

technical background in knowledge representation using OWL and software engineering knowledge such as UML. It is worth noting that the verbs in the service functionality property were extracted based on the concept of *interaction patterns* [10,7] in which a key verb is used in representing a frequently used interaction pattern, and all the other synonymous verbs appeared in the requirements are replaced by this key verb. For example the key verb *send* was selected as a concept in the ontology to represent its synonymous verbs such as *transmit*, *give*, and *post*.

4.2. Describing OPUCE services

To describe the characteristics of a service, a new OWL class is constructed as the service's semantic model (we call it OPUCE semantic service) using OWL-DL class constructors and the OPUCE Ontology. As illustrated in Fig. 6, we could 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 base or a composite service, and then restricting its functional, non-functional, and tagging properties one by one. For example, a `SendSMS` base service can be described as:

$$\begin{aligned} \mathit{SendSMS} \equiv\ & \mathit{BaseService} \sqcap \leq 2\mathit{hasInput} \sqcap \geq 2\mathit{hasInput} \sqcap \\ & \exists \mathit{hasInput}(\mathit{Text}) \sqcap \exists \mathit{hasInput}(\mathit{MobilePhoneNumber}) \sqcap \\ & \exists \mathit{hasFunctionality}(\mathit{Send}) \sqcap \\ & \exists \mathit{hasCategory}(\mathit{Telecom}) \sqcap \exists \mathit{hasTechnology}(\mathit{J2EE}) \sqcap \\ & \exists \mathit{hasTag}(\mathit{multilingual}). \end{aligned}$$

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

Fig. 6. OWL class constructors.

The above statement indicates that i) SendSMS is a base service, ii) it has exactly two inputs: one is a Text type and the other a MobilePhoneNumber type, iii) its functionality is Send, iv) it belongs to the Telecom domain, vi) one of its associated technologies is J2EE, and vii) it supports multilingual inputs.

4.3. Challenges and requirements for user-centric discovery of OPUCE services

With all the OPUCE semantic services stored 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 matching list since it is subsumed by the above class.

Obviously, discovering services by directly querying the DL reasoner presents the following challenges for a user:

- She must know the query language (in our case OWL-DL);
- She must be familiar with the domain terminology;
- Some eligible 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, such as the class $\exists hasCategory(Telecom)$ and the class $\forall hasCategory(Telecom)$, cannot match each other since they do not have a subsumption relationship [33].

Although some semantic matchmaking approaches such as [25,47] use query templates to alleviate the first and the third challenges, they still require the user to have knowledge of the domain terminology to fill the template.

Inspired by the flourishing research area of ontology visualization [24], we set out to design a user-centric visualized browser to overcome the above-mentioned challenges and to help prosumers effectively explore and discover OPUCE services. The essential requirements of this browser are summarized as follows:

- *Specifying query statements by visual selection*. Instead of asking the user to manually *compose* query conditions and statements, the query statement should be automatically composed by the browser with the user just *select* predefined query conditions, which are rendered by the browser, according to the OPUCE Ontology framework. For example, if the user is able to select (instead of typing) Telecom under the hasCategory service property and select send under the hasFunctionality service property, then all the services that satisfy the condition $\exists hasCategory(Telecom) \sqcap \exists hasFunctionality(Send)$ should be returned.
- *Visualize the discovery process*. We observe that the process of discovering desired services is also a process of stepwise refinement of query condition clauses: For example, to find telecom services that send text, the user first query all the telecom services, and then refine the condition by specifying the service functionality as send; and the user may continue this process to refine the query condition by adding or removing condition clauses. The browser should provide such support by visualize the refinement process, so that the user can see the *connection* between each step, instead of just rending a single final result.

It is worth noting that these requirements are not intended to be comprehensive but rather aiming at capturing the specific needs of OPUCE service prosumers. In the next subsection we discuss the detailed design of our visual semantic service browser aiming at tackling the above-mentioned requirements and challenges, and also the rationality behind such design.

4.4. Visually exploring and discovering OPUCE services

From the above analysis, we can find 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. The task of discovering services can be broken down into: first getting services that can be subsumed by each property restriction class, and then calculating the intersection of all these services. For example, the query—*finding telecom services that send text*—can be further refined to getting services that are the intersection of the following sets of services:

- Services subsumed by the class $\exists hasCategory(Telecom)$,
- Services subsumed by the class $\exists hasFunctionality(Send)$, and
- Services subsumed by the class $\exists hasInput(Text)$.

According to the above principle and the requirement of predefining all the query condition clauses, instead of using a "lazy" query approach where the semantic service description knowledge base is searched when a service query is issued, the OPUCE Browser adopts an "eager" approach to preprocess every newly published semantic service description and calculates the services subsumed by each property restriction class in the description. After all the property restriction classes together with the subsumed services are collected, they are displayed in an organized graphical layout for visual service exploration and discovery. Using the browser, a prosumer can visually check the services subsumed by any property restriction class, and calculate the intersection of services subsumed by property restriction classes on the fly. All of this can be done by a prosumer in a point-and-click fashion.

The algorithm illustrated in Fig. 7 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 ('$\sqcap$') from its semantic

```
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
```

Fig. 7. Algorithm for generating property restriction classes.

description; and then we go through all the extracted classes. If a class is a property restriction, the DL reasoner will be consulted to get the class's subsumed services.

4.4.1. The graphical interface

As shown in Fig. 8, the browser is divided into three parts. The left pane is used to satisfy the first requirement specified in Section 4.3: rending all the property restriction classes for selection. An indented list is used to display the hierarchy of restriction classes. Although there are other ontology visualization methods such as node-link and tree, zoomable, space-filling, focus + context, 3D information landscapes [24], we choose indented list because of its familiarity to the user: the same concept has been widely used in file browsers, and exploring services with an 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 [24]. In case the indented list becomes too long for effective browsing, the user may use the right pane as a shortcut: whenever the user inputs a keyword, all the property restriction classes whose name contains this keyword will be displayed and ready for selection. Since service properties have been predefined in the OPUCE Ontology, it is self-evident to use a service property as top level elements and attach the associated property restrictions as its children. For example, the `hasCategory` node contains eight child nodes (which are also leaf nodes), and each node represents a property restriction class. On the indented list, we do not expose the subtle difference between existential restrictions and universal restrictions to the user. In fact, a leaf node represents the union between existential restrictions and universal restrictions. For example, the leaf node `hasCategory/Telecom` represents the property restriction class $\exists hasCategory(Telecom) \sqcup \forall hasCategory(Telecom)$. Clearly, such design is specifically for solving the third challenge listed in Section 4.3. In order to know what services are subsumed by a property restriction class, a user just needs to select the check box next to a node. For example, in Fig. 8, if the `hasCategory/Telecom` leaf node is selected, a set containing 74 services—all of them belong to the

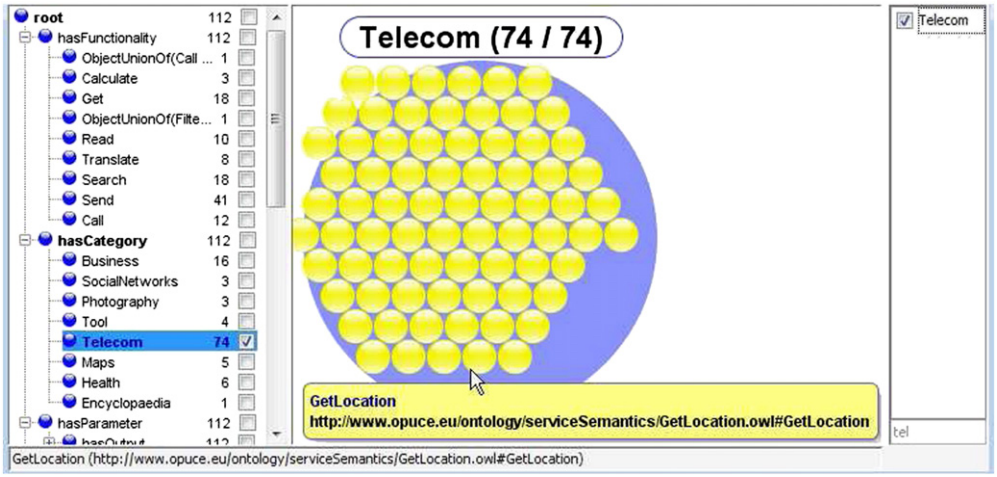


Fig. 8. Graphical interface of the OPUCE Browser.

telecom category—will be displayed in the middle pane with each service rendered as a small circle within the set. It is worth noting that even before the user performs any action, the number of services subsumed by a property restriction has been displayed next to it to give the user a general idea of how many services each property restriction class has (see Fig. 8).

From the middle pane, the user can explore what services are actually inside a property restriction class. If the user hovers the mouse pointer over a service, a tooltip containing the url that brings the user to the detailed service description will appear. Next, we explain how the user can use the browser to do service exploration and discovery tasks, and also explain how the second requirement discussed in Section 4.3 is implemented.

4.4.2. Exploration and discovery

A user could explore OPUCE services through the following actions:

- Browse the list of property restrictions in the left pane and select one for further examination in the middle pane;
- Browse services contained in a property restriction class in the middle pane and check detailed service description following the service url;
- Locate a property restriction class node by entering its keyword in the right pane.

A discovery process is in fact the combination of several exploring steps, which can be illustrated by the statechart in Fig. 9: When the user starts a discovery task, she begins by browsing the property restriction list; when she finds the interesting node and selects this node, a set containing all the services subsumed by this node is displayed for the user to explore; if the user does not find the desired services, she can select more nodes from the property restriction list, and then the selected nodes/sets and also the intersection of these nodes/sets are displayed dynamically for the user to inspect; in this process, the user may also change her selection criteria by deselecting some nodes. For example, supposing that a user wants to discover telecom services that can send text. First of all, this user selects the `Telecom` node in the `hasCategory` list (see Fig. 8). Next, the user refines the search criteria by selecting the `Send` node in the `hasFunctionality` list. The browser reacts to this action by displaying the `hasCategory/Telecom` set, the `hasFunctionality/Send` set, and also the intersection of these two sets (we call it *intersection set*) in the middle pane. As shown in Fig. 10(a), the `hasCategory/Telecom` set contains 74 services, the `hasFunctionality/Send` set contains 41 services, and the intersection set contains 25 services, which can be interpreted as there are 25 telecom services whose functionality is `Send`. The user then selects the `hasInput/Text` node. As illustrated in Fig. 10(b), the number of services in the intersection set reduces to seven. Since the number of discovered services is at a manageable scale, at this stage, the user may continue to refine the discovery process by selecting more property restriction nodes or just browse the details of the seven services to select the most suitable one. The details of the discovered services (e.g., the URL) can also be displayed in a HTML list for effective browsing. It is worth noting that the approach of discovering services using stepwise intersection sets follows the common principle of information retrieval: we start with a broad search condition, and if the result set is too large, we narrow down the criteria by including more search conditions. The graphical representation is similar to Venn Diagrams, which are a common tool widely used in many areas of our daily life.

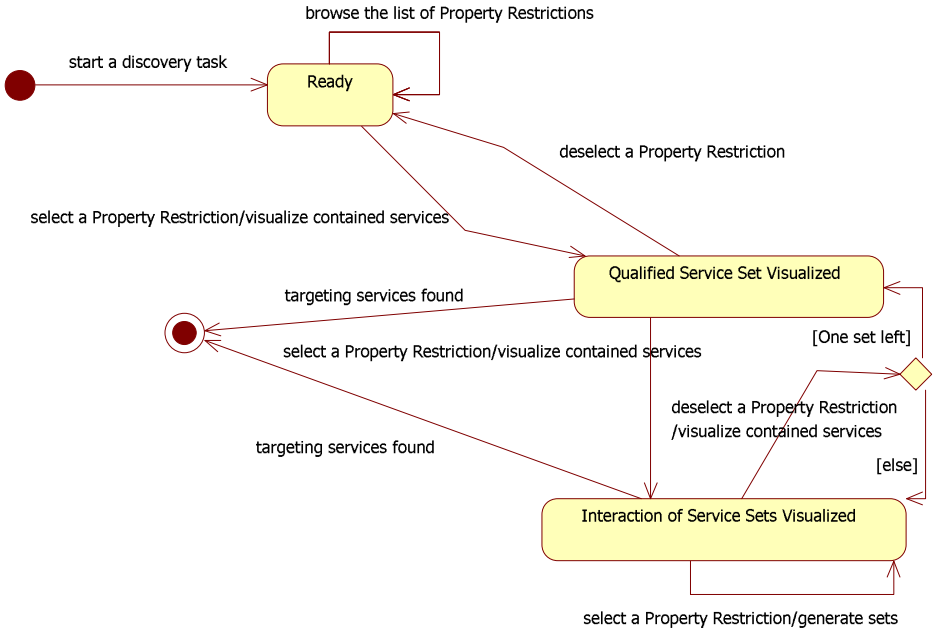


Fig. 9. The discovery process statechart.

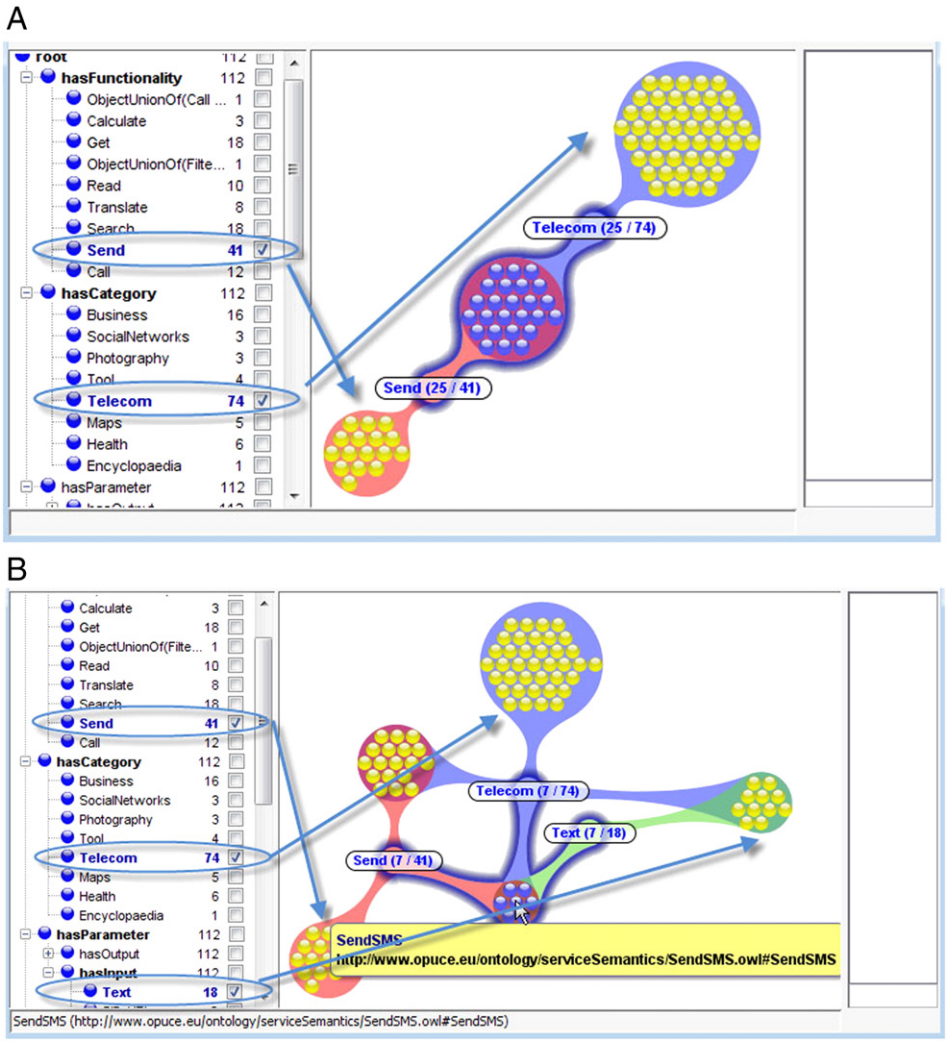


Fig. 10. Discovering services using the OPUCE Browser.

In summary, the OPUCE Browser provides a graphical environment to facilitate service exploration and discovery. Without knowing any ontology language and domain terminology, a prosumer can conveniently discover desired services based on its properties in an effective point-and-click fashion.

5. Evaluation

The service repository discussed in this paper has been up and running in the OPUCE platform. It contains around 112 real life services with manually annotated semantic descriptions. Accessing and managing service descriptions using the repository have proved to be efficient thanks to the faceted approach where service descriptions are retrieved and saved based on facets, instead of a whole complete service description.

Fig. 11 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,[10] which is a Java interface and implementation for OWL. Pellet is used as the OWL-DL reasoner. The Restriction Class Manager provides interfaces for the user to add, remove, and update semantic service descriptions, which are the content of the semantic facets of services. Whenever the semantic facet of a service is added, removed, or updated, 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,[11] a visual technique that has been used in several semantic Web applications [18].

We have conducted several experiments to validate the effectiveness of the OPUCE Browser. First, we studied the performance of the browser server in terms of the time needed to publish a service semantic description. Because usually service descriptions

[10] http://owlapi.sourceforge.net/.
[11] http://www.aduna-software.com/technologies/clustermap/overview.view.

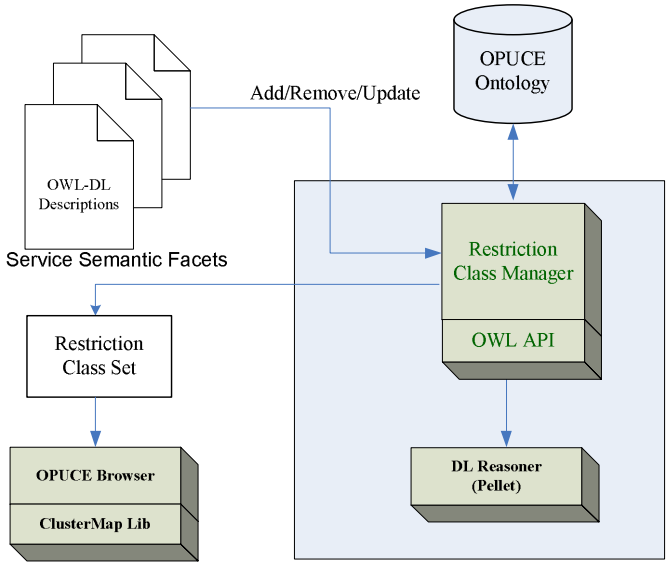


Fig. 11. OPUCE Browser architecture.

are manually published by service providers and we use an "eager" approach to process the description and generate property restriction classes at publishing time, it is necessary to study the server's response time to a normal publishing request and if the server is scalable in dealing with large-size service descriptions.

More importantly, we conducted two usability studies: the first one tested the efficiency of users in using the OPUCE Browser. The purpose of this study is to validate the user-centric claim of the browser: whether common users can complete typical service discovery tasks with the browser and how efficiently they perform. The second study is to collect subjective feedback from the users based on a standard questionnaire-based survey in terms of user perceptions of *usefulness*, *ease of use*, *ease of learning* and *satisfaction* about the browser.

5.1. Performance study

The Restriction Class Manager module is the core component of the browser in processing service semantics and also generating the restriction class set as the input for the GUI. The performance of the OPUCE Browser largely depends on this module. We tested the performance of the Restriction Class Manager module on its `Add`, `Remove`, and `Update` functions. This experiment is conducted on a desktop computer with Intel(R) Core(TM)2 Quad CPU@2.50 GHz and 3 G of RAM, and the Java version is 1.6.0_10-beta. The OPUCE Ontology knowledge base has 870 concepts, and the class descriptions under test are generated by randomly selecting properties and concepts from the ontology and then connect them using the `intersection` constructor of OWL.

As illustrated in Fig. 12, for a common class description containing 5–10 property restrictions, it takes less than 400 ms to add it to the Restriction Class Set. The time increases almost linearly when the number of property restrictions of a class increases from 150 to 1000. As we can see, considering 1000 property restrictions is already a very large set for a single service description to have, at this scale, the CPU time used by `Add` is only 515 ms. For the `Remove` function, it keeps at a constant 65 ms no matter what the number of property restrictions are. Finally, for the `Update` function, it is implemented by calling the `Remove` and `Add` functions in sequence; so its CPU time is calculated by summing the time used on `Remove` and `Add`. It is worth noting that because we use an "eager" approach as we have discussed in Section 4.4, from the user interaction point of view, when a user uses the OPUCE Browser, the browser does not need to interact with the DL engine at all because the information on property restrictions and its associated services has been generated and updated after each add/update/remove operation on the service descriptions.

5.2. Usability study

The OPUCE Browser contributes to usability mainly through its visual service exploration and discovery mechanism. The Browser GUI has been proved to be intuitive for people who are not technically experienced. Indeed, exploring services with an indented list is natural to most end users since they have been accustomed to it in everyday tasks, like scanning the content at the beginning of a book or writing down a list of tasks they have to perform [24]. Discovering services can be done visually in a point-and-click fashion. As a result, end users are not required anymore to have a high level of technical expertise or even basic knowledge of ontology and semantic Web.

To test the efficiency of users in using the OPUCE Browser, we developed three typical discovery tasks and recorded the time needed to complete each task. Participants in the study were six computer science undergraduate students, with two "expert

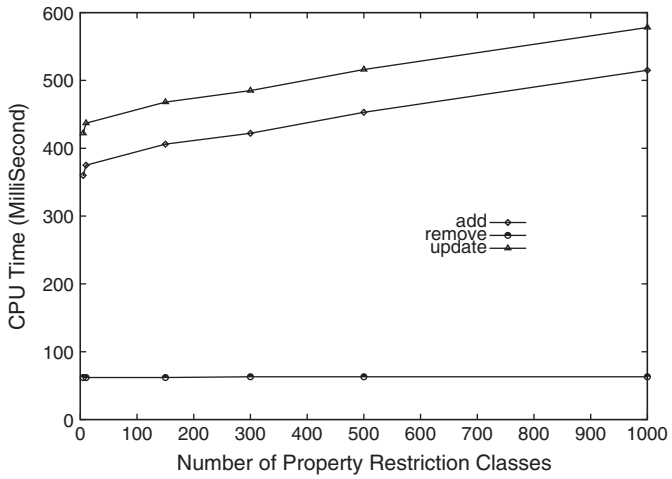


Fig. 12. Performance of the Restriction Class Manager module.

users" who had knowledge in semantic Web and had been involved in the development of the OPUCE Browser, and the other four "novice users" who had no previous knowledge of the semantic Web technology and the OPUCE project.

It is worth mentioning that using a small number of participants in our experiment can be justified by [37,36], in which the authors propose a mathematical model on the effectiveness of usability experiments and demonstrate that running multiple tests with a small number of users is more effective than running a single test with a large number of users. The reason why less expert users are participated in the study is that the main focus of this study is on novice users, and the results of expert users are just used as a benchmark to derive the relative learnability of novice users (we will detail this point in the following paragraphs).

Each participant was given a brief tutorial on how to use the OPUCE Browser. They then were asked to complete the following three typical discovery tasks one by one using the OPUCE Browser: i) find telecom services that send messages, ii) find services that search medical centers closest to a user's current location, and iii) find services that can find photos of a specific location and is implemented in J2EE.

The time each participant used to complete each task and the average time for expert and novice users to complete a task are presented in Table 1. In the table, we also show the learnability, which indicates how quick a user can learn to use a user interface [48], for each task. In this study, we adopt the approach used in [57] to calculate the learnability by comparing the time needed for an expert to complete the same task relative to the time needed for a novice user to complete a task. For example, if an expert takes, on average, 50 s to complete a task, while a novice user takes 100 s to complete the same task, then the learnability of the user interface is 0.5.

From this table, first we can see that all the users, including experts and novices, complete all the tasks successfully, and their performances in using the OPUCE Browser are promising: all the experts are able to finish any task within 1 min, and all the novices are able to finish any task within 2 min. Second, the novices are improving their performances as they do more tasks, which can be indicated by the learnability of the browser in each task: it increases from 0.40 (41/103.75) in the first task to 0.66 (53/80.75) in the third task.

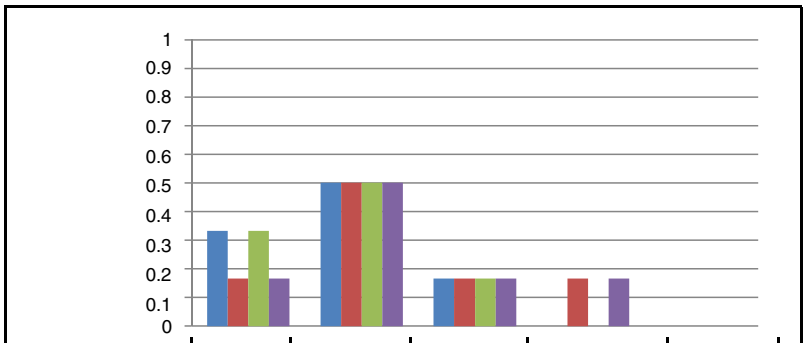
Next, we used a standard usability questionnaire [29] to evaluate user perceptions of useful, ease of use, ease of learning, and satisfaction of the OPUCE Browser. A set of three questions is used to address each of these characteristics, and a five part Likert scale was used for each question.

All the six participants completed the questionnaire and the results are shown in Fig. 13. The results are very positive with strong agreement over the usefulness (over 80% strongly agree or agree), the ease of use (over 65%), ease of learning (over 80%) and satisfaction (over 65%).

Table 1
Results of the efficiency study.

Subject	Task 1 (s)	Task 2 (s)	Task 3 (s)
Expert 1	40	34	51
Expert 2	42	32	55
Average (Expert)	**41**	**33**	**53**
Novice 1	110	66	82
Novice 2	98	71	84
Novice 3	112	65	80
Novice 4	95	55	77
Average (Novice)	**103.75**	**64.25**	**80.75**
Learnability	**0.40**	**0.51**	**0.66**

The bold text is derived/aggregated data.



	Strongly Agree	Agree	Neither	Disagree	Strongly Disagree
Usefulness	1/3	1/2	1/6	0	0
Ease of use	1/6	1/2	1/6	1/6	0
Ease of Learning	1/3	1/2	1/6	0	0
Satisfaction	1/6	1/2	1/6	1/6	0

Fig. 13. Usability survey results.

During the usability survey, one of the participants mentioned a potential weak point of the browser: the user interface may look cluttering if hundreds of services are displayed simultaneously. We argue that this problem may turn much less significant when the user gets used to the OPUCE Browser: On the one hand, the services that a user needs to care about are always in the intersection set of the selected property restriction sets. Take the snapshot in Fig. 10(b) as an example, although there are more than a hundred services displayed on the interface, the user only needs to manage the seven services in the intersection set which is always placed in the center of the interface. On the other hand, if the size of the intersection set is still large, the user may reduce it by including more property restrictions.

6. Related work and discussion

With the prevalence of the service-oriented computing paradigm in both Internet and telecom domains, research on repositories for convenient and efficient discovery and management of services has gained a significant momentum recently. Specifications such as UDDI [11] and ebXML Registry [8] 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 has been reported in the literature aiming at enhancing service repositories with ontology and semantic discovery facilities. In [46], the authors discussed the major shortcomings of the state-of-the-art Web service repositories such as limited browsing and discovering capabilities and suggest solutions using semantic Web related techniques.

To enable semantic service discovery, the first thing we need is a semantic service description language. OWL-S [31] is an OWL based generic upper ontology for describing the profile, process logic, and grounding of services, which is characterized by rich expressiveness [43]. The OPUCE Ontology is not a generic upper ontology, instead, it is a simple domain specific extension based on actual requirements in the OPUCE project. In current version of the OPUCE Ontology, we directly use OWL-DL as the base language since most of the constructs of OWL-S, such as those in `ServiceProcess` and `ServiceGrounding` are not directly related to our work. However, it is worth mentioning that we can easily import some constructs from OWL-S, such as those in `ServiceProfile`, into the OPUCE Ontology if further requirements appear.

Another two DL-based service ontologies can be found in [27] and [6]. In [27], the authors proposed a DAML-S [3] based service ontology to represent concepts in the e-commerce domain such as *advertisements*, *service queries*, *sales*, and *delivery*. Similar to the OPUCE Ontology, some domain specific properties such as `providedBy` and `requestedBy` are defined to extend the base generic framework. In [6], the authors proposed a three-layer service ontology—consisting of *Concrete Services*, *Abstract Services*, and *Service Categories*—to organize services at different levels of abstraction. Concrete Services and Abstract Services are described by means of operations (`hasOperation` property) and I/O parameters (`hasInput` and `hasOutput` properties), and Service Categories organize services into taxonomies (using the `hasCategory` property). Comparing with the OPUCE Ontology, the `hasOperation` property is similar to the `hasFunctionality` property in the OPUCE Ontology, and all the other three properties are also part of the OPUCE Ontology. It is worth noting that similar to our work, services in [6] are also mainly described using pattern of connecting existential restriction classes with intersections.

WSML [26] is another powerful semantic service description language that has a formal language basis. Another line of work on semantic service description language is based on extending WSDL with semantic annotations. For example, WSDL-S [2] and its recent development SAWSDL [16] are two initiatives that add semantics to the abstract part of a WSDL declaration. Another

annotation language PS-WSDL also extends the implementation part of WSDL by adding information about the geographic scope, the provider, and the QoS properties of the service [43].

As far as service registry enrichment with semantic information is concerned, many approaches have been proposed. In [15], the authors enriched 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 [41] where the authors proposed a matchmaking engine inside the UDDI registry to match service capability descriptions encoded in DAML-S Profile [3]. Its follow-up work reported in [50] used OWL-S Profile and also improved the matchmaking algorithm. In another two works, applying DAML-Son UDDI was reported in [1], and applying OWL-S on UDDI was reported in [30]. Recently, with the emerging of WSDL-S and SAWSDL, in [49], the authors mapped WSDL-S semantic annotations to UDDI. In [25], the authors proposed the FUSION Semantic Registry that augments the UDDI's service publication and discovery facilities based on SAWSDL and OWL-DL. In [43], the authors proposed the PYRAMID-S framework that uses PS-WSDL and uses hybrid peer-to-peer topology to organize heterogeneous service registries. In contrast, the OPUCE repository uses the ebXML registry as a basic storage and adds a separate semantic layer on top of it. The OPUCE Browser focuses on providing a visual interface to facilitate the exploration and discovery of semantic services by common prosumers, while none of the above-mentioned approach has provided such function.

Our visual service browser is built on top of ClusterMap Library. Although ClusterMap Library has been used in several applications such as the DOPE Browser for exploring large online resources in the domain of drugs and diseases and AutoFocus for managing personal information sources [18], to the best of our knowledge, the OPUCE Browser is the first application that uses this library in the area of service discovery.

Finally, although the OPUCE Browser is an OWL-DL based service discovery framework, the expressiveness is actually weaker than OWL-DL because when generating restriction classes from semantic description of services, only selected OWL-DL class constructors—including `intersectionOf`, `allValueFrom`, and `someValueFrom`—are considered, which means the information graphically rendered is a projection of the original semantic services. However, since the OPUCE Ontology has defined a framework with great emphasis on service properties, and using the pattern of connecting existential restriction classes with intersections is the recommended approach to describe OPUCE services, the OPUCE Browser can capture the essential information of OPUCE semantic services. As we have discussed above in this section, the same pattern was also used in [6] to describe their semantic services.

7. Conclusion

The proliferation of Web 2.0 techniques and applications (e.g., mashups) is fostering the emergence of user-centricity where ordinary users are in a leading position to not only consume, but also deliver personalized content and services. One of the most difficult problems however is discovering and managing large amount of services by these users who are generally not technically experienced. In this paper, we have presented our first hand experience gained from a European Union sponsored research project OPUCE. We specially focus on the OPUCE service repository and one of its core components called OPUCE Browser. In line with the user-centric theme of the OPUCE project, the repository supports accessing facets of service descriptions from handheld devices. Most importantly, we have enhanced the repository with a visual service browser by adopting ontology and semantic Web technology. The browser is intuitive and efficient in the sense that it does not require users 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-click.

In the future, we plan to investigate automatic ontology derivation techniques such as Formal Concept Analysis [20] and then apply these techniques in the repository to alleviate the burden of manual ontology construction and semantic annotation. We also plan to apply our visualized service discovery technique on service ontologies other than the OPUCE Ontology, for example the one defined in [6], to test its effectiveness. Considering that multi-ontology environments are one of the important trends in the Web 2.0 world, we plan to enhance the browser with a mapping engine to map other service ontologies to the OPUCE Ontology with the help of ontology matching techniques [32]. It is worth noting that because such mapping happens at publishing time, it will not bring performance impact to exploration and discovery tasks. Finally, considering research on context-aware service discovery is gaining a significant momentum [13,45], we plan to make the OPUCE Browser runnable on mobile devices and add context-awareness feature to the browser.

Acknowledgments

This work is partly funded by the research project OPUCE, under the 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] Rama Akkiraju, Richard Goodwin, Prashant Doshi, Sascha Roeder, A method for semantically enhancing the service discovery capabilities of UDDI, Proc. of the Workshop on Information Integration on the Web (IIWeb'03), 2003, pp. 87–92.
[2] Rama Akkiraju, et al., Web Service Semantics—WSDL-S. W3C Member Submission, http://www.w3.org/Submission/WSDL-S November 2005.
[3] Anupriya Ankolekar, et al., DAML-S: web service description for the semantic web, Proc. of the International Semantic Web Conference (ISWC'02), LNCS, vol. 2342, Springer, 2002, pp. 348–363.

[4] F. Arvidsson, A. Flycht-Eriksson, Ontologies I, http://www.ida.liu.se/janma/SemWeb/Slides/ontologies1.pdf 2008.
[5] Sonia Bergamaschi, Francesco Guerra, Mirko Orsini, Claudio Sartori, Maurizio Vincini, A semantic approach to ETL technologies, Data & Knowledge Engineering 70 (2011) 717–731 (8).
[6] Devis Bianchini, Valeria Antonellis, Michele Melchiori, Flexible semantic-based service matchmaking and discovery, World Wide Web 11 (June 2008) 227–251.
[7] R. Biddle, J. Noble, E. Tempero, Patterns for essential use case bodies, Proceedings of the 2002 Conference on Pattern Languages of Programs (CRPIT'02), 2003, pp. 85–98.
[8] Kathryn Breininger, Farrukh Najmi, Nikola Stojanovic (Eds.), The ebXML Registry Repository Version 3.0.1, February 2007 http://www.oasis-open.org/committees/download.php/23648/regrep-3.0.1-cd3.zip.
[9] David Cipolla, Fabrizio Cosso, Matteo Demartini, Marc Drewniok, Francesco Moggia, Paola Renditore, Jurgen Sienel, Web service based asynchronous service execution environment, Proc. of the Service-Oriented Computing—ICSOC 2007 Workshops, 2007, pp. 304–316.
[10] L. Constantine, A.D.L. Lockwood, Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design, ACM Press, 1999.
[11] Fransico Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, Sanjiva Weerawarana, Unraveling the web services web: an introduction to SOAP, WSDL, and UDDI, IEEE Internet Computing 6 (2) (March 2002) 86–93.
[12] SeCSE Project Deliverable, A1.D2.1—State of the Art—Service Engineering, http://www.secse-project.eu/wp-content/uploads/2007/08/a1d1-state-of-the-art-service-engineering.zip 2005.
[13] Stefan Dietze, Alessio Gugliotta, John Domingue, Towards context-aware semantic web service discovery through conceptual situation spaces, Proc. of the 2008 International Workshop on Context Enabled Source and Service Selection, Integration and Adaptation: Organized with the 17th International World Wide Web Conference (WWW 2008), CSSSIA'08, ACM, New York, NY, USA, 2008, pp. 6:1–6:8.
[14] Glen Dobson, Russell Lock, Ian Sommerville, QoSOnt: an ontology for QoS in service-centric systems, UK e-Science All Hands Meeting, 2005, pp. 80–87.
[15] Asuman Dogac, Yildiray Kabak, Gokce Laleci, Enriching ebXML registries with OWL ontologies for efficient service discovery, Proc. of the 14th International Workshop on Research Issues on Data Engineering (RIDE'04), 2004, pp. 69–76.
[16] Joel Farrell, Holger Lausen (Eds.), Semantic annotations for WSDL and XML schema, W3C Recommendation, August 2007 http://www.w3.org/TR/sawsdl.
[17] Dieter Fensel, Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer, 2001.
[18] Christiaan Fluit, Marta Sabou, Frank van Harmelen, Ontology-based information visualization: toward semantic web applications, Visualizing the Semantic Web, Springer, 2004, pp. 45–58.
[19] Vandana Gandotra, Archana A. Singhal, Punam Bedi, Layered security architecture for threat management using multi-agent system, SIGSOFT Software Engineering 36 (5) (2011) 1–11.
[20] B. Ganter, G. Stumme, R. Wille (Eds.), Formal Concept Analysis, Foundations and Applications, Springer, 2005.
[21] T. Gruber, A translation approach to portable ontology specifications, Knowledge Acquisition 5 (2) (1993) 199–220.
[22] Nicola Guarino, Formal Ontology and Information Systems, IOS Press, 1998, pp. 3–15.
[23] M.N. Huhns, M.P. Singh, Ontologies for agents, IEEE Internet Computing 1 (6) (1997) 81–83.
[24] Akrivi Katifori, Constantin Halatsis, Ontology visualization methods—a survey, ACM Computing Surveys 39 (4) (2007) 1–43.
[25] Kourtesis Kourtesisz, Iraklis Paraskakis, Combining SAWSDL, OWL-DL and UDDI for semantically enhanced web service discovery, in: S. Bechhofer, et al., (Eds.), Proc. of the 5th European Semantic Web Conference (ESWC'08), LNCS, vol. 5021, Springer-Verlag, Berlin Heidelberg, 2008, pp. 614–628.
[26] H. Lausen, J. de Bruijn, A. Polleres, D. Fensel, The WSML rule languages for the semantic web, Proc. of the W3C Workshop on Rule Languages for Interoperability, 2005.
[27] Lei Li, Ian Horrocks, A software framework for matchmaking based on semantic web technology, Proc. of the 12th International Conference on World Wide Web (WWW'03), 2003, pp. 331–339.
[28] Nikolaos Loutas, Vassilios Peristeras, Konstantinos Tarabanis, Towards a reference service model for the web of services, Data & Knowledge Engineering 70 (9) (2011) 753–774.
[29] A.M. Lund, Measuring usability with the USE questionnaire, The Usability SIG Newsletter 8 (2) (2001).
[30] Jim Luo, Bruce Montrose, Anya Kim, Amit Khashnobish, Myong Kang, Adding OWL-S support to the existing UDDI infrastructure, Proc. of the IEEE International Conference on Web Services (ICWS'06), 2006, pp. 153–162.
[31] David Martin, et al., OWL-S: semantic markup for web services, W3C Member Submission, November 2004 http://www.w3.org/Submission/OWL-S.
[32] Jorge Martinez-Gil, Jose F. Aldana-Montes, Reverse ontology matching, ACM SIGMOD Record 39 (4) (2010) 5–11.
[33] Deborah L. Mcguiness, Frank van Harmelen, OWL web ontology language overview, W3C Recommendation, February 2004 www.w3.org/TR/owl-features.
[34] John Musser, Tim Oreilly, Web 2.0 Principles and Best Practices, O'Reilly Media, 2006.
[35] Anne H.H. Ngu, Michael P. Carlson, Quan Z. Sheng, Hye-Young Paik, Semantic-based mashup of composite applications, IEEE Transactions on Services Computing 3 (1) (2010) 2–15.
[36] Jakob Nielsen, Why You Only Need to Test with 5 Users, http://www.useit.com/alertbox/20000319.html March 2000 (visited 07/03/2010).
[37] Jakob Nielsen, Thomas K. Landauer, A mathematical model of the finding of usability problems, Proc. of the ACM INTERCHI'93 Conference, 1993, pp. 206–213.
[38] A. Ohgren, K. Sandkuhl, Towards a methodology for ontology development in small and medium-sized enterprises, IADIS Conference on Applied Computing, 2005.
[39] Jesus Oliva, Jose I. Serrano, Maria D. Castillo, Angel Iglesias, SyMSS: a syntax-based measure for short-text semantic similarity, Data & Knowledge Engineering 70 (4) (2011) 390–405.
[40] OPUCE Project Team, OPUCE Booklet, http://www.opuce.tid.es/docs/OPUCE%20booklet_v1.7.pdf 2008.
[41] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, Katia Sycara, Semantic matching of web service capabilities, Proc. of the International Semantic Web Conference (ISWC'02), LNCS, vol. 2342, Springer, 2002, pp. 333–347.
[42] Ioannis V. Papaioannou, Dimitrios T. Tsesmetzis, Ioanna G. Roussaki, Miltiades E. Anagnostou, A QoS ontology language for web services, Proc. of the 20th International Conference on Advanced Information Networking and Applications (AINA'06), 2006, pp. 101–106.
[43] Thomi Pilioura, Aphrodite Tsalgatidou, Unified publication and discovery of semantic web services, ACM Transactions on The Web 3 (3) (June 2009) 11–44.
[44] R. Prieto-Diaz, P. Freeman, Classifying software for reusability, IEEE Software 4 (1) (January 1987) 6–16.
[45] Wenge Rong, Kecheng Liu, A survey of context aware web service discovery: from user's perspective, Proc. of the 5th IEEE International Symposium on Service Oriented System Engineering, 2010.
[46] Marta Sabou, Jeff Pan, Towards semantically enhanced web service repositories, Journal of Web Semantics, Science, Services and Agents on the World Wide Web 5 (2007) 142–150.
[47] Amit P. Sheth, Karthik Gomadam, Ajith Ranabahu, Semantics enhanced services: METEOR-S, SAWSDL and SA-REST, IEEE Data Engineering Bulletin 31 (3) (2008) 8–12.
[48] Ben Shneiderman, Designing the User Interface, Addison-Wesley, 1998.
[49] Kaarthik Sivashanmugam, Kunal Verma, Amit Sheth, John Miller, Adding semantics to web services standards, Proc. of the International Conference on Web Services (ICWS'03), 2003, pp. 395–401.
[50] Naveen Srinivasan, Massimo Paolucci, Katia Sycara, Adding OWL-S to UDDI, implementation and throughput, Proc. of the 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC'04), 2004, pp. 6–9.
[51] Katia Sycara, Massimo Paolucci, Anupriya Ankolekar, Naveen Srinivasan, Automated discovery, interaction and composition of semantic web services, Web Semantics: Science, Services and Agents on the World Wide Web 1 (1) (December 2003) 27–46.
[52] Alvin Toffler, The Third Wave, Bantam, 1984.
[53] Ruben Trapero, et al., (Eds.), OPUCE Deliverable 3.1: Service, Service Lifecycle and Service Components Specification, 2008 http://www.opuce.tid.es/docs/OPUCE_D3%201_1.pdf.
[54] Jian Yu, Paolo Falcarin, Jose M. Alamo, Jurgen Sienel, Quan Z. Sheng, Jose F. Mejia, A user-centric mobile service creation approach converging Telco and IT services, Proc. of the 8th International Conference on Mobile Business (ICMB'09), 2009, pp. 238–242.

[55] Jian Yu, Paolo Falcarin, Sancho Rego, Isabel Ordás, Eduardo Martins, Quan Sun, Rubén Trapero, Quan Z. Sheng, XDM-compatible service repository for user-centric service creation and discovery, Proc. of IEEE International Conference on Web Services (ICWS'09), 2009, pp. 992–999.
[56] Jian Yu, Quan Z. Sheng, Joshua K.Y. Swee, Model-driven development of adaptive service-based systems with aspects and rules, Proc. of 11th International Conference on Web Information Systems Engineering (WISE 2010), 2010, pp. 548–563.
[57] Ying Zou, Qi Zhang, Xulin Zhao, Improving the usability of e-commerce applications using business processes, IEEE Transaction Software Engineering 33 (December 2007) 837–855.

Jian Yu received the PhD degree in computer theory and software from Peking University, Beijing, China. He is Lecturer and Research Fellow in the Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia. His current research interests include service-oriented computing, pervasive computing, semantic Web, and adaptive systems. He was a task leader in European Union FP6 IST project OPUCE. He is the author of more than 35 publications.

Quan Z. Sheng received the PhD degree in computer science from the University of New South Wales, Sydney, Australia. He is a senior lecturer in the School of Computer Science at the University of Adelaide. His research interests include service-oriented architectures, distributed computing, and pervasive computing. He is the recipient of Microsoft Research Fellowship in 2003. He is the author of more than 80 publications. He is a member of the IEEE and the ACM.

Jun Han is a Professor of Software Engineering at Swinburne University of Technology, Melbourne, Australia. He is also a research leader with Australia's Cooperative Research Centre in Smart Services (Smart Services CRC) and Cooperative Research Centre in Advanced Automotive Technology (AutoCRC). His research interests include software architecture, software system qualities, adaptive and context-aware software systems, and services engineering and management.

Yanbo Wu is now a PhD candidate in the University of Adelaide. His research interests include distributed databases, RFID and Internet of Things, and Cloud Computing.

Chengfei Liu is a Professor and the Leader of the Knowledge and Data Intensive Systems focus area in the Swinburne University Centre for Computing and Engineering Software Systems, the Faculty of Information and Communication Technologies, Swinburne University of Technology, Australia. He received the BS, MS and PhD degrees from Nanjing University, China in 1983, 1985 and 1988, respectively, all in Computer Science. Prior to joining Swinburne, he taught at the University of South Australia and the University of Technology Sydney, and was a Senior Research Scientist at Cooperative Research Centre for Distributed Systems Technology, Australia. He also held visiting positions at the Chinese University of Hong Kong, the University of Aizu in Japan, and IBM Silicon Valley Lab in USA. He has published more than 150 peer-reviewed papers in various journals and conference proceedings and has served on technical program committees and organizing committees of more than 80 international conferences or workshops in the areas of database systems, workflow systems and Web information systems. His current research interests include XML data management and query processing, keyword search for structured data, RDF databases, data provenance, transaction management for advanced database applications, Web service transactions, and workflow management.