Facilitating Enterprise Service Discovery for Non-Technical Business Users

Marcus Roy, Basem Suleiman, and Ingo Weber

SAP Research, Sydney NSW 2060, Australia
School of Computer Science and Engineering, UNSW, Sydney NSW 2052, Australia
{m.roy,basem.suleiman}@sap.com,ingo.weber@cse.unsw.edu.au

Abstract. Enterprise Services (ES) are Web services with which enterprise applications expose a subset of their functionality. Due to the often high number of different ES, as well as the complex nature of their names, it is difficult for non-technical business users to discover services in ES repositories. However, most of this complexity stems from a SOA governance-driven service design process that is essential to the development of harmonized and long-lasting ES. Based on the example of SAP's ES, we describe a representational model that consolidates existing models and patterns used during the service design process. We created an iterative search approach that uses this consolidated metadata. The evaluation of the approach with real business users, based on a prototypical implementation, demonstrates that our iterative search is more efficient and effective than the currently offered search.

1 Introduction

Business process automation enables a cost-effective, agile and rapid composition and execution of new applications to address on-demand and changing organizational requirements. It leverages a business user's expertise to perform the modeling of business processes by specifying relevant activities that in turn are implemented by experienced developers reusing existing Enterprise Services (ES) in a Service-Oriented Architecture (SOA). In IT organizations, ES represent non-public Web Services intended for internal developers, partners and customers to reuse enterprise-specific data and functionality from existing legacy applications stored in company-internal repositories, e.g. SAP Enterprise Service Repository and Registry (ESR) or IBM WebSphere Service Registry and Repository (WSRR). They are idiosyncratically designed to represent specific parts of integrated and enterprise-internal business processes and legacy applications leading to long, technical and less comprehensive signatures (in this work we refer to service interface and operation names only) as shown in Examples (1.1- 1.3) of an "Sales Order" related SAP ES. These examples also underline the difficulty to discover ES as described by Beaton et al. [3, 2]. Generally, when it comes to the naming of ES, it is a challenge to give unique and easy-to-understand names to a large number of ES, e.g. SAP Business Suite offers more than 4000 ES (08/2010). Most of its complexity has been introduced deliberately during its development life-cycle to ensure long-lasting, unique and easy-to-manage ES from a software

engineering perspective. For instance, the ES shown in Example (1.1) is quite detailed and less prone to duplication. It describes a read operation of a Sales Order, stored in an ERP system, which is identified by an ID expected as part of the request. Most of these intrinsic characteristics are inferred from a standardized service design process applied prior to the service implementation as part of a SOA governance process that guides and governs the development of ES. It is indispensable for large enterprises to follow such a globally agreed on SOA governance process.

`SalesOrderERPByIDQueryResponse_In` (1.1)

`SalesOrderCreateRequestConfirmation_In` (1.2)

`SalesOrderBasicDataByBuyerAndBasicDataQueryResponse_In` (1.3)

In the following, we used the example of an SAP service design process to illustrate the use of agreed-on methodologies and proprietary models that can be adopted to what we call a *service* and *data model*. Principally, such a design process is not limited to SAP, but can also be found with others companies (not the focus here). We then introduced a representational model that integrates both service and data models as first-class components including their relationship. Based on this model, we created a metadata repository based on a list of ES and their respective metadata that has been extracted from multiple sources, e.g. corporate web pages and semi-structured documents. This metadata hereby refers to entities in the data and service model respectively. Note that the focus of this paper is not on yet another service description technology. We are not competing with existing technologies, e.g. RDF or OWL-S, instead we could utilize them to represent such metadata as annotations. We then implemented the conceptual solution of our iterative search in form of a prototype. We performed a first evaluation of this prototype with real business users, which demonstrates the feasibility of our approach to enhance the current search for ES.

In the remainder, we explain the service design process in Section 2 and introduce our representational model in Section 3. Section 4 details on our iterative search and prototype. In Section 5, we evaluate our search and prototype, refer to related work in Section 6 and conclude the paper in Section 7.

2 Service Design Process

In this section, we describe the application of a design process for the service development. Based on the example of SAP's ES, we detail on the existing data and service model used to develop ES as shown in Examples 1.1-1.3.

In the introduction, we generally referred to a SOA governance-driven service design process [6, 1] that is deployed to guide and ensure the development of harmonized and business-aligned ES. It is also used to create a mutual understanding about the definition and meaning of ES among stakeholders and partners. Such a design process is required and executed recurrently prior to any definition and implementation phase. Using the example of an SAP service

design process, developers are guided in their tasks to first design ES by specifying high-level and business-related concepts that abstractly describe the purpose and scope of the prospective ES.

Most of these design efforts are supported by a modeling tool to visually compose an abstract service definition as shown in Figure 1. Such an application enables developers to arrange and order abstract shapes representing existing key business entities (red shape) as well as the preferred but predefined way of accessing them (blue shape). Eventually, this visual representation is transformed into an abstract service definition, i.e. coarse WSDL template, detailing only on service interfaces and operations.

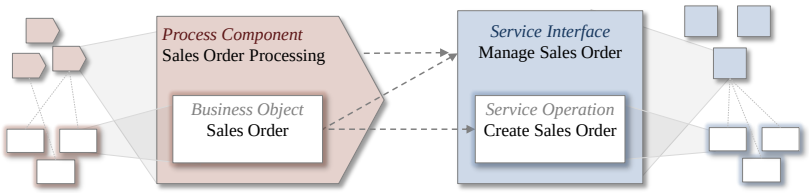

Fig. 1. Visual model of an abstract service definition during service design

Although a modeling tool helps to visually generate an abstract ES definition based on entities from the data and service model, this information is typically not available to a search environment (though it's hidden in documentation). Therefore, Figure 2 illustrates earlier mentioned models, i.e. data and service model, and their relationship, which embraces key entities of information that can me made available to a search. For simplicity's sake, we only focused on showing a relevant subset of model entities that are considered significant for the definition of service interface and operation names.

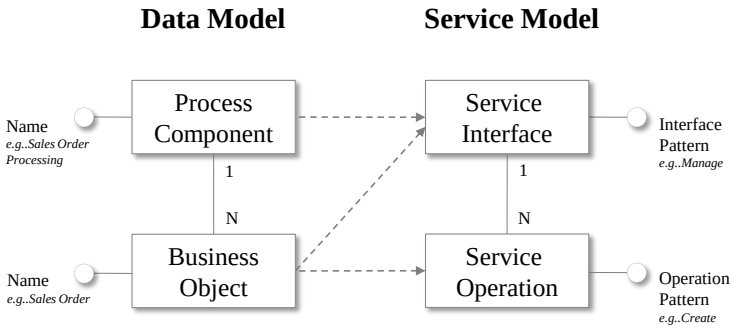

Fig. 2. Subset of the data and service model shown as ER diagram

In Figure 2, a Process Component "Sales Order Processing" represents a technical concept similar to a department inside a company that groups all Sales Order related activities (e.g. Create, Approve, Notify etc.). Within such a Process Component, a Sales Order Business Object denotes a central business entity that stores and maintains all relevant information about a real Sales Order

(e.g. Buyer, Seller etc.). Figure 2 also shows service model entities, i.e. Service Interface and Operation, that are associated to related data model entities, i.e. Process Component and/or Business Object. Based on both model entities and their association, potential signatures for service interface (Table 1) and operation (Table 2) are defined. In Table 1, the service interface implicitly describes a grouping of related service operations that manage a Sales Order of a Sales Order Processing activity.

Service Interface	
Process Component	`Sales Order Processing`
Business Object	`Sales Order`
Interface Pattern	`Manage`
Service Interface	`SalesOrderProcessingManageSalesOrder`

Table 1. Construction of the service interface name

Service Operation	
Business Object	`Sales Order`
Interface Pattern	`Create`
Service Operation	`SalesOrderCreateRequestConfirmation_In`

Table 2. Construction of the service operation name

The use of the keyword "manage" in the service interface signature is based on an interface pattern (as shown in Fig. 2) that determines selective service operations. Hence, service operations listed under a "manage" interface have to be either a "create", "read", "update", "change", "check" or "cancel" operation (called *operation pattern*). In Table 2, a service operation is defined based on a "Sales Order" Business Object and the "create" operation pattern, which is implied by the "manage" interface pattern of Table 1.

3 Leveraging Service Design Principles

The example of the previous section illustrated the design of an abstract definition for service interface and operation names based on the service and data model model. In this section, we describe how to utilize the knowledge of this service design process. Firstly, we designed a representational model (Section 3.1) by consolidating the data and service model. Second, we created a metadata repository defined by our representational model and populated it with information extracted from multiple sources (Section 3.2). Both representational model and metadata repository represent the basis of the iterative search as explained in Section 4.

3.1 A Representational Model

All information stored in our metadata repository is defined according to a representational model consisting of the data model extended with entities of the

service model. Originally, both models are used in rather different context meaning that the knowledge about the relationship between both models is not given by default, and thus, not obvious to anyone trying to discover ES. We therefore consolidated both models by integrating them and describe their relationship as the association of Business Object to Service Interface (Fig. 3). Having both models and their relationship defined as first-class components in our representational model has the advantage of naturally exploiting services based on their underlying data model and vice versa. This means we can describe a service based on its concrete relation to specific data model entities or in turn find any services that are related to a particular data model entity.

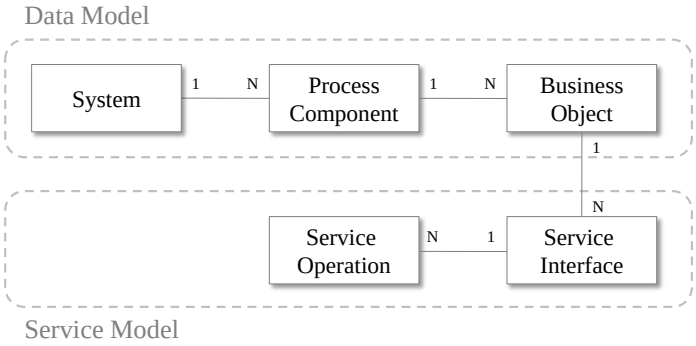

Fig. 3. Enterprise Service Representation Model

3.2 Extraction of Service Design Entities

In order to create the basis for our ES Search approach, which we present in the next section, we created a metadata repository that represents a list of all available SAP ES that can be found on SAP's Enterprise Service Workplace[1] (ESW). The ESW is a central place used by developers and consultants to search for and view detailed information about ES. It includes documentation, examples, and technical descriptions, e.g. WSDL.

Starting from this list of ES, we enriched the entries with additional metadata according to our representation model, by extracting information from multiple sources as follows. Firstly, we extracted data model-related information from a set of reports (e.g. Excel spreadsheets) that were exported from an operational content management system. Secondly, we extracted service model data directly from the ESW: we used simple page scraping techniques to extract data from the web site of each Enterprise Service.

We then used basic inference techniques and regular expressions on the extracted data, so as to detect model entities in the service interface and operation signature. Subsequently we mapped the respective service and service model entities to the data model based on relationship as defined in our representation

[1] http://www.sdn.sap.com/irj/bpx/esworkplace

model and created the association in our metadata repository. We also recognized interface and operation patterns and annotated the corresponding service model entities.

4 Enterprise Service Search

In this section, we describe the concept of the iterative search and its prototypical implementation. The search uses the repository from the previous section.

4.1 Iterative Search Approach

In order to significantly improve the search, we first analyzed how users search and what they enter to find a particular ES. This analysis has been conducted on anonymized log files of search queries entered by users over a period of six months. In detail, the examined log files contained search strings that have been categorized according to entities in our model, if applicable. Finally, we accumulated recurrent categories to rank their frequency of occurrence (not in scope of this work). One key finding was that users rarely enter more than three keywords. Secondly, they frequently entered simple keywords representing central business entities that formed the basis of the desired ES, e.g. "Sales Order" for a "Create Sales Order" ES etc. Based on these observations, we created an iterative search that allows users to choose from a set of predefined search options rather than allowing them to freely enter search text. Each search option conceptually relates to entities in our model whereas each selection of a search option identifies a node in the metadata repository. Starting from this node, a list of respective ES can be inferred by following up on any consecutive parent-child-relationship down to the Service Operation level as illustrated in Figure 4.

Using such a search-by-selection technique better leverages the underlying data model from a twofold perspective. Firstly, potential search options (as illustrated in Figure 4) can be directly connected to related entities in our representation model without the need to understand and map a potentially volatile human search text. Secondly, we can use our metadata repository to populate search options from which the user can choose. With each selection, the content of any consecutive search option is determined while the list of potential ES candidates gets iteratively filtered. In the end, the more information is provided by the user, the more effective is the filtering and the smaller is the result set of potential ES (enabling a subsequent browsing as a final search step). Our evaluation in Section 5 demonstrates the effectiveness and efficiency if this approach.

4.2 Search Prototype and Example

We have implemented the iterative search as a Web application (cf. Figure 4) that uses our representational model and metadata repository. In the top half of the application (Part A), the user can choose from previously mentioned search options. For each selection, a list of potential ES will be updated and displayed in the bottom part of the application (Part B). Users can skip search options if they are unsure about their significance or if the search is already successful.

As an example, say a business user wants to find an Enterprise Service that "creates a Sales Order". First and without any selection, a complete list of service operations is shown (> 4000 ES). In a next step, the user selects "Sales Order Processing" from a list of available Process Components (Option 1/Figure 4), which decrements the result by an order of magnitude as only ES are displayed that belong to that particular department. In a second step, the user selects a "Sales Order" from a list of Business Objects determined by the previously chosen Process Component (Option 2/Figure 4), which reduces the result set by another order of magnitude. By that time, users are already able to browse the significant smaller list of ES to find possible matches. In the case of uncertainty, the user can continue to choose from the remaining set of options by detailing on the intention what to do with the Sales Order. The intention to create a Sales Order implies a "manage" pattern (Option 3/Figure 4), which reduces the search result to 11 ES as partly shown in Part B of Figure 4. This search result correctly contains the desired `SalesOrderCreateRequestConfirmation_In` ES, which has been given the synonym "Create Sales Order" for reasons of readability.

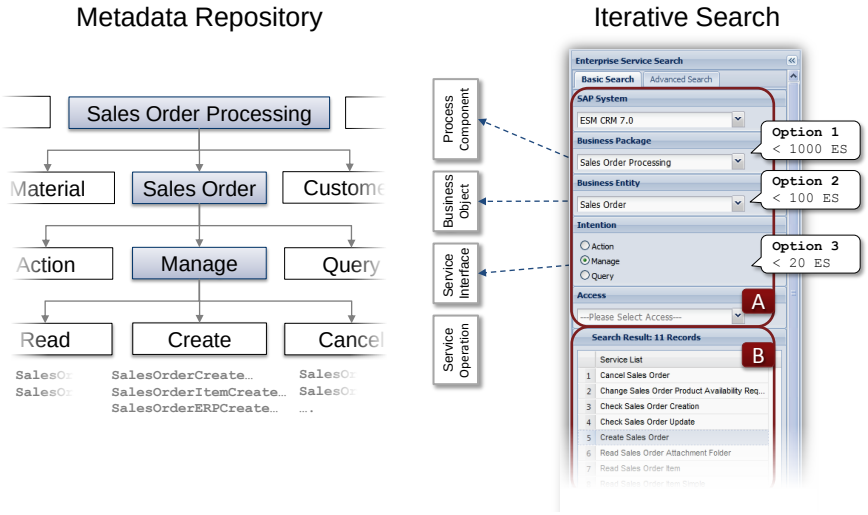

Fig. 4. Correlation of our representational model to iterative search prototype

5 Evaluation

We evaluated our discovery approach by exposing the above-described iterative search prototype to a group of test users, as discussed in this section. Since the time for searching and displaying relevant ES only takes 1-2 seconds, performance seems to be no big concern in our approach and has not been evaluated further.

5.1 Experimental Setup

Five business users from SAP business consulting and support departments have been briefly introduced to the ESW search and our iterative search prototype.

The participants had never used either one of them. They also have been given a four-step "Create Sales Order" business process (see Fig. 5) with brief description of the process activities. They then have been asked to find the relevant ES for each activity of the given business process by using (1) our iterative search prototype and (2) the ESW – in that order, so the iterative search would not benefit from a possible learning curve. Each participant was given 30 minutes to complete the task.

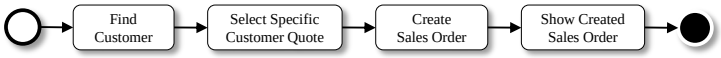

Fig. 5. "Create Sales Order" Business Process used in the experiment

5.2 Data Collection and Analysis

The participants notified us whenever they thought they found a service for one of the activities. During each experiment (1) and (2) for each of the five participants, we recorded the time spent to find each ES. We also observed the participants' behavior in terms of how easy or hard it was for them to use the respective application to find the relevant services. After each experiment, we asked the participants about their experience with each of the investigated tools. We recorded all their feedback for future improvements. The summary of the results collected from our experiments are grouped into two main categories: First, the search time represents the overall time (in seconds) spent by each participant to find all relevant ES and second, the total number of correct ES found by each participant (out of 4 services).

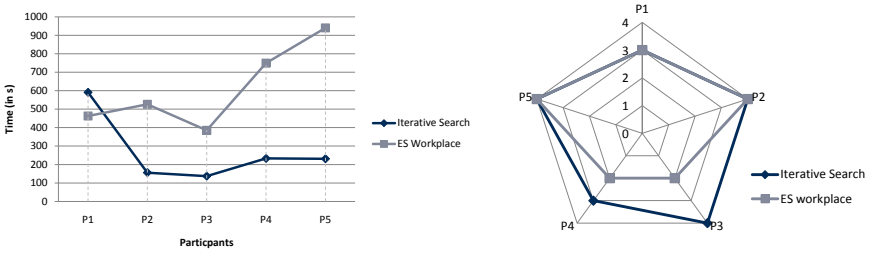

Fig. 6. Evaluation results, per participant, using the ESW vs. our iterative search. **Left:** search time; **Right:** number of correctly discovered services.

The result, as depicted in Fig. 6 (left) , shows that the search time spent by almost all participants, except for participant P1, using our iterative search prototype is noticeably less than the time required by the same participants using the ESW. For instance, participants P4 and P5 spent only about a third of the ESW search time using our iterative search to find the four ES. With respect to participant P1, we noticed some degree of uncertainty when using our prototype, compared to the other participants. We also noticed that he

found some ES on the ESW by coincidence. In summary, the average search time spent by all participants using our iterative search is more than half of the time spent by the same participants using the ESW. Although the number of asked participants is not significant large, the results still clearly indicate an improvement of our iterative search approach over the existing ESW in terms of search time.

The second evaluation criterion is the number of correct ES found by participants using both search applications. As shown in Fig. 6 (right), 60% of participants were able to find all four correct ES by using our iterative search prototype compared to 40% of participants using the ESW search. The minimum number of ES found is two on ESW vs. three with our iterative search. Although 40% of the users have not found *all* four correct services (sometimes as simple as a "Create Sales Order"), they at least found three out of four ES and therefore did not completely fail. On the basis of these initial results, we can say that our approach has the potential to iteratively improve the search, with room for further improvements. Although the number of five users can be considered marginal to comprehensively validate our search, we believe it yet has shown the potential to improve the discovery of Enterprise Services over the existing search.

6 Related Work

In order to position our work, we classified current service retrieval techniques into linguistic, semantic and hybrid approaches.

Linguistic-based approaches are mainly based on textual analysis of service description such as a WSDL using clustering such as [5, 8, 17] techniques. For instance, the Woogle [5] search engine uses a similarity search for operations by grouping operation parameters into meaningful concepts.

Semantic approaches are based on extending service descriptions with formal models to capture their underlying meaning. The resulting so-called Semantic Web Services (SWS) [11] relate service properties to concepts belonging to ontologies – roughly speaking, formal conceptual models of the terms in a given domain, including relationships and constraints between terms. The discovery approach is usually to match concepts of queries against SWS descriptions, using logic-based inference. In [4], the service discovery algorithm matches service request against the services ontology using description logic. Similarly, [10, 12] present service matchmaking approaches to enable discovery of ontology-based service descriptions based on DAML-S. The matching technique in [9] relates service inputs and outputs using OWL ontologies to capture the meaning of terms in service descriptions. Also, [14] describes the formalization of a client query as a goal that is matched to the goal of a service. However, semantic discovery requires (i) an agreed-upon common ontology, (ii) complete semantic service descriptions, which can require considerable manual effort during service development, and (iii) semantic models for queries.

Hybrid approaches, e.g. [13, 15, 18, 7, 16] use the ontology and semantic techniques to enhance linguistic service discovery. For example, the Seekda [13] public

search engine relies on crawling and data mining techniques of Web API repositories such as ProgrammableWeb (`http://www.programmableweb.com/`) in order to automate the annotation of APIs with metadata that can be used for a semantic search. The Opossum [16] search engine also crawls the Web for WSDL descriptions and transforms them into a generic ontological-based model. The service properties are then automatically enriched with concepts from existing ontologies of different domains (e.g. finance, e-commerce). Search is done by matching concepts of the query against the ontological-based model of the service. This is done by considering the linguistic and structural properties of ontology.

Our approach is different from the above since we base it on concepts that stem from an organizational SOA governance-driven service design process which guides the development of the ES. In a way, the collection of terms that are instances of the data and service model elements can be seen as a specialized ontology. The corresponding vocabulary contains rather business terms than technical concepts. As such, discovery in this context cannot be put on a par with discovery of arbitrary Web services on the Web. Instead, ES are very much predictable as they comply to internal design principles, which are created once and applied to all existing and future ES.

7 Conclusion and Future Work

In this paper, we investigated how SOA governance artifacts can be used during service discovery. We first explained why governance is necessary when large amounts of services are developed, and showcased the artifacts that can result from governance at the example of the more than 4000 Enterprise Services from SAP. At the core of the information of interest in this context are the *service* and *data models*, which are used to abstractly define ES interface and operation names. For our discovery approach, we created a representational model offering a consolidated view on both models. Based on this combined model and its associations, we created a metadata repository of all SAP ES, and enriched it with the metadata extracted from multiple (real-world) sources. We then developed an iterative search based on this metadata repository. An initial evaluation of our prototype with a group of test users demonstrated the feasibility and effectiveness of our approach.

In the future, we plan to more generally investigate the complex nature of governance-driven service design. We believe this will lead to richer ES descriptions, and ultimately a more sophisticated and effective search technique. Finally, we will continue evaluate our techniques with real users as the work progresses.

References

1. D. J. Artus. SOA Realization: Service Design Principles. `http://www.ibm.com/developerworks/webservices/library/ws-soa-design/`, February 2006.
2. J. Beaton, S. Y. Jeong, Y. Xie, J. Stylos, and B. A. Myers. Usability Challenges for Enterprise Service-oriented Architecture apis. In *VLHCC*. IEEE, 2008.

3. J. K. Beaton, B. A. Myers, J. Stylos, S. Y. S. Jeong, and Y. C. Xie. Usability evaluation for enterprise SOA APIs. In *SDSOA '08*. ACM, 2008.
4. B. Benatallah, M. Hacid, A. Leger, C. Rey, and F. Toumani. On Automating Web Services Discovery. *VLDB Journal*, 14(1):84–96, 2005.
5. X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity Search for Web Services. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 372–383. VLDB Endowment, 2004.
6. T. Erl. *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall Professional Technical Reference, 2005.
7. G. Fenza, V. Loia, and S. Senatore. A Hybrid Approach to Semantic Web Services matchmaking. *Int. J. Approx. Reasoning*, 48(3):808–828, 2008.
8. A. Funk and K. Bontcheva. Ontology-based Categorization of Web Services with Machine Learning. In *LREC'10*, Valletta, Malta, may 2010. ELRA.
9. D. Hull, E. Zolin, A. Bovykin, I. Horrocks, U. Sattler, and R. Stevens. Deciding Semantic Matching of Stateless Services. In *AAAI*, 2006.
10. L. Li and I. Horrocks. A Software Framework for Matchmaking based on Semantic Web Technology. In *WWW'03*, 2003.
11. S. A. McIlraith, T. C. Son, and H. Zeng. Semantic Web Services. *IEEE Intelligent Systems*, 16(2):46–53, 2001.
12. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic Matching of Web Service Capabilities. In *ISWC'02*, 2002.
13. N. Steinmetz, H. Lausen, and M. Brunner. Web Service Search on Large Scale. In *ICSOC/ServiceWave*, pages 437–444, 2009.
14. M. Stollberg, U. Keller, H. Lausen, and S. Heymans. Two-Phase Web Service Discovery based on Rich Functional Descriptions. In *ESWC*, pages 99–113, 2007.
15. E. Toch, A. Gal, I. Reinhartz-Berger, and D. Dori. A Semantic Approach to Approximate Service Retrieval. *ACM Trans. Inter. Tech.*, 8(1):2, 2007.
16. E. Toch, I. Reinhartz-berger, A. Gal, and D. Dori. OPOSSUM: Bridging the Gap Between Web Services and the Semantic Web. In *NGITS, Vol. 4032 of LNCS*, pages 357–358. Springer, 2006.
17. J. Wu and Z. Wu. Similarity-based Web Service Matchmaking. In *SCC '05*, pages 287–294, Washington, DC, USA, 2005. IEEE Computer Society.
18. Y. Zhang, B.-Y. Liu, and H. Wang. A Method of Web Service Discovery Based on Semantic Message Bipartite Matching for Remote Medical System. *J. Theor. Appl. Electron. Commer. Res.*, 4(2):79–87, 2009.