Integrating Heterogeneous Web Services from an End User Perspective

Bipin Upadhyaya
bipin.upadhyaya@queensu.ca
Supervisor: Ying Zou
ying.zou@queensu.ca
Department of Electronic and Computer Engineering
Queen's University
Kingston, Ontario

ABSTRACT
Service composition combines a set of available Web services using control flows to create a more complex service. The service composition is a complex process which is challenging even for experienced users to discover and select appropriate service among a set of similar services. The available tools and techniques for service composition either are too complicated or require technical knowledge (such as script language). Moreover the heterogeneity in service description language and communication protocols make service composition more difficult. In our research, we propose an approach that allows non-technical users (i.e., end users) to easily compose heterogeneous Web resources with different communication protocol. The goal of this research is fulfilled on two stages: 1) identifying integrable Web resources from different Web applications and SOAP-based Web services; and 2) providing approaches to discover and compose Web resources that are capable of accomplishing user's activities. We propose a framework that assists a user to discover and compose services.

Categories and Subject Descriptors
H.3.5 [**Online Information Services**]: Web-based services Features – analysis of service interface, service discovery and service composition; H.5.2 User Interfaces, H.5.4 Hypermedia

General Terms
Service Composition, End-user development

Keywords
Service composition, service migration, service discovery.

1. Introduction
Web is a fundamental infrastructure to help an end user who has no extensive technical skills to search and perform different daily activities, such as buying flight tickets, and shopping on-line. Recent progress in Web services makes it possible to publish, locate, and invoke services across the Web. However, there are heterogeneous formats and standards used to describe service interfaces, such as Web service description language (WSDL) [25] and Web application description language (WADL) [26]. A SOAP-based Web service supports interoperable machine-to-machine interaction over a network. RESTful services are Web services that use a well-defined protocol such as HTTP. A RESTful service is exposed as a finite state machine; a client knows next state after it reaches a state unlike in SOAP-based Web service where client knows the entire states forehand [16]. A RESTful service is provided as a resource which is meaningful concept and has a URL associated with it [2]. A Web service (i.e., SOAP-based services or RESTful services) performs a specific task, such as booking a hotel or buying a flight ticket. An end user can combine different services to accomplish a daily activity (*e.g.*, planning a trip).

However, the different service description languages make an end user difficult to understand various types of services (*e.g.*, SOAP-based services and RESTful services). With the ever-increasing number of services published on the Internet (*e.g.*, Google has indexed 230,000 Web Service description languages (WSDL) documents), finding desired services is just similar to looking for a needle in a haystack. An end user cannot take the advantage to the wealth of services leaving them to use mostly Web based system. An end user cannot mix and match different services based on their preferences. Moreover, Web services provide limited support for an end user to compose when no single service can satisfy the functionality required by the end user. An end-user has to manually go through different services on different URL addresses to compose services. Imagine a scenario such as planning a holiday which includes both buying a flight ticket and booking a room. A user needs to decide on the duration of the stay in a hotel based on the on the availability and prices of flight tickets and hotels. An end user has to check if a hotel is available before buying a flight ticket and correspondingly check flight tickets before booking a hotel. An end user has to switch between service providers to perform these fundamental tasks. The situation becomes more complex when a user has to perform a task dependent on many different criteria.

In our work, we propose approaches that assist end users to compose services of heterogeneous formats. We aim to address the following challenges:

1) **Heterogeneity in service description languages**: Different services use various service description languages. WSDL is standard description for SOAP-based services. RESTful services have different description languages, such as WSDL, WADL, and Relational Link Language (RELL) [27]. Some of the service descriptions for REST services are in Web pages requiring human to read and implement the services. It is

Middleware 2012 Doctoral Symposium, December 3, 2012, Montreal, Quebec, Canada.

challenging for end users to understand and parse the different service description languages.

2) **Difficult to integrate heterogeneous Web services**: There are various Web services SOAP-based services require sophisticated tools to invoke them. Moreover, SOAP-based services are heavy-weighted services and encoding and decoding of XML-based SOAP messages consumes computation resources (i.e., battery, processor speed, and memory). RESTful services uses HTTP clients such as a browser. RESTful services describe a set of HTTP methods (GET, POST, DELETE, and PUT) for each resource. Each method has well defined semantics. RPC-based services use HTTP client but all method is tunnelled POST operation. A Web application uses forms and hyperlinks to perform a task. Each type of the Web services uses their own protocol. There is no standard way for communication and interaction among different types of Web services.

3) **Semantic gaps between service providers and users:** The precision of service discovery approaches is dependent on the understanding of the semantics of service description documents and end user's query. Current service discovery approaches [1, 3, 5] lack support for query formulation. Web service search engines provide little support for users to construct and reformulate their queries when the initial query fails. The available services act as a black box to users. A user has to conduct many trials in order to formulate an appropriate query to retrieve the desired services. The vocabulary adopted in service description is often used by developers in the software development domain. It can be very different from the ones used in users' queries [29]. Without a good understanding of the semantics of service descriptions and queries, a service discovery approach is likely to retrieve a large number of irrelevant Web services or fail to return any Web service.

4) Lack of support for end-users to compose services: Ko *et al.* [28] provides the challenges in end-user programming and its state of art. Current approach of service composition [15, 17, 18, 19, 20, 32] requires substantial technical knowledge making service composition difficult for non-technical users. The current approaches require end users to understand the concepts behind the tools or learn a language to be able to compose even a simple process [15, 17]. Mashup tools [19, 20] and their applications are focused on accessing, manipulating data and composing data flows (such as filtering, merging, and sorting data feeds). To use these tools effectively, the users need to know, not only how to program, but also how to use the different Web APIs from all services. Moreover, these service composition and mashups are limited to service of the same kind. The service composition techniques do not help the end-user to generate a flexible and agile process (i.e., ad-hoc process) to help end users with their day to day information needs.

In this thesis, we address the aforementioned challenges. To bridge the gap among the heterogeneous services and their descriptions, we propose a unified schema to represent the various Web services. We provide an approach to extract services from SOAP-based services and Web application and migrate them to RESTful services. To reduce the semantic gap between service providers and users, we provide an approach that helps a user to craft the queries for Web service search. Our approach does not require user`s to learn a new query language or a new tool to compose services. To simplify and make service composition easier for end users, we provide an approach to generate ad-hoc processes for service composition that bridges different implementation technologies and communication protocol.

This paper is structured as follows. Section 2 describes the overview of our approach. Section 3 discusses our initial evaluation. Finally Section 4 concludes and discusses the future work.

2. Overview of Our Approach

Our approach enables end users to compose heterogeneous services. Figure 1 gives an overall of the steps involved in our approach. We extract resources from SOAP-based services and Web applications. The extracted artefacts are represented in a unified schema. We use concepts which refer to entities, events and topics that are of interest to users to derive our approach for service discovery and composition. As shown in Figure 1, the first two steps (i.e., extract resources from SOAP-based services/ Web applications and transform service descriptions to the unified schema) are used by service providers. The following two steps (i.e., concept based service discovery and composition) can be adopted by an end user in service composition. In the rest of this section, we describe each step in more details.

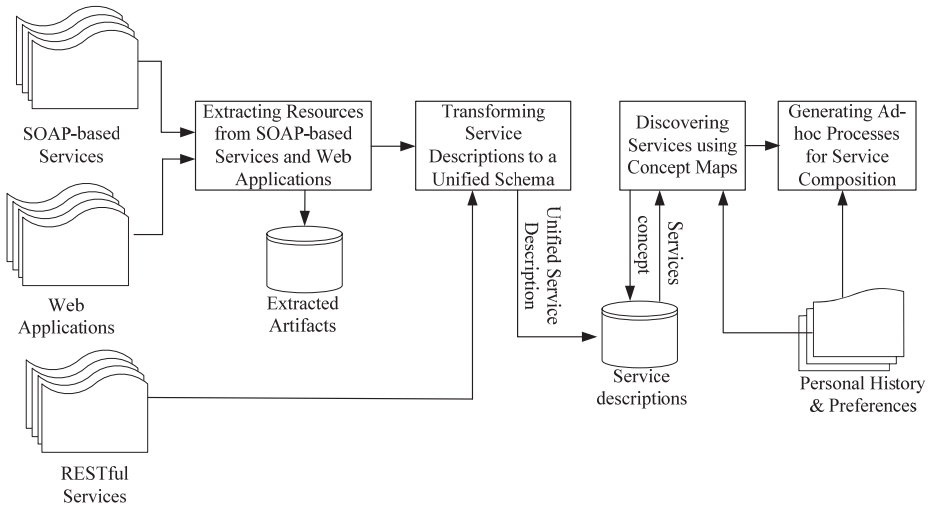


Figure 1: Overview of Our Proposed Approach

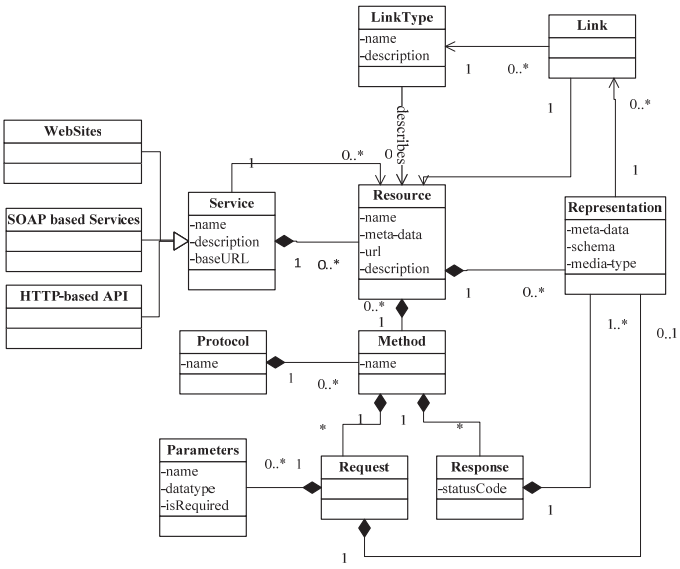


Figure 2: Unifying Service Description Documents

2.1 Transforming Service Descriptions to a Unified Schema

We provide a unified schema to represent heterogeneous service description languages used for various Web services (*e.g.*, Web sites, SOAP-based services, and HTTP-based APIs). The unified schema facilitates the discovery of various Web services with similar functionality. The uniform schema eases the use of services as a user does not have to know and understand different heterogeneous description formats.

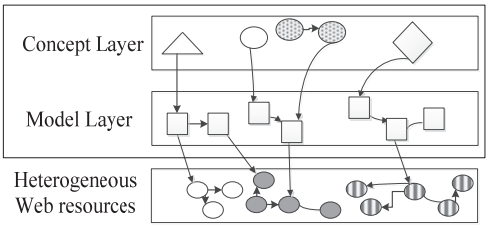
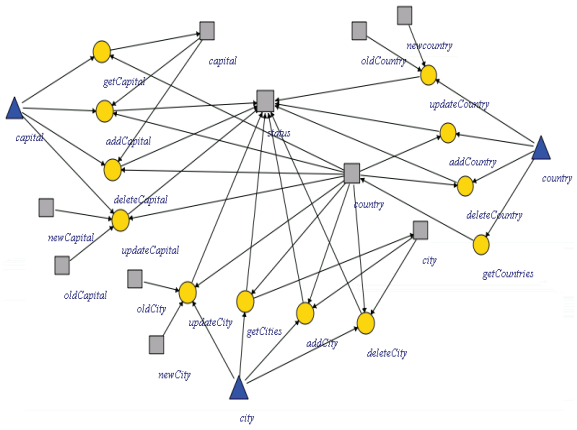
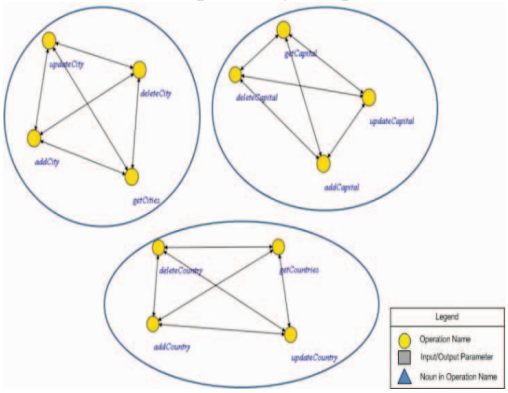


Figure 3: Extracting Concepts from Different Heterogeneous Web Resources

Figure 2 describes a unified resource schema that represents heterogeneous Web resources (*e.g.*, Web Forms, SOAP-based services, and HTTP-based APIs) in a single format. The schema closely follows the Web architecture standard [11] used by RESTful services. A service is defined as an aggregation of different resources that can be identified using a Universal Resource Identifier (URI). A resource represents an entity from the real world whose state is exposed and can be changed via accessing the URI. Resources are accessed using methods defined by a protocol. Resources are manipulated through representations portrayed according to a media type (*e.g.* HTML, Atom, etc.) and some metadata. A representation represents the state of the client's interaction within the application and contains links [24] that are required to change the client's state (*e.g.*, a submit form). A method contains the request and response message. For example in the HTTP protocol, the methods can be GET, POST, PUT, and DELETE. A GET request for a URI notifies the service provider to retrieve the data. A PUT request indicates that a service provider should update the old data with the data sent in the request. A DELETE request indicates that the data of a service to be deleted by a service provider. A POST request is used to create a new resource. A method can have at least one request and several optional response messages. The request can include a set of additional parameters to be sent to the resource. A response message is linked with a representation that returns the state of the resource after a method invocation. The response uses different status code to represent faulty responses (*e.g.*, incorrect parameters, and server errors). The protocol of the method specifies the different types of status code that defines how the response should be interpreted. For example, HTTP status code 200 represents that the request has succeeded. The information returned with the response is dependent on the method used in the request. For example, when the method, GET, is invoked, an entity corresponding to the requested resource is sent in the response.



(a)Dependency Graph



(b) Clusters of Operations

Figure 4: Clustering Operations based Dependency Graph

2.2 Extracting Resources from SOAP-based Services and Web Applications

RESTful services are hypermedia based services that can be invoked and composed using HTTP clients (such as a Web browser). For the Web resources that are not RESTful services, we develop techniques to migrate them to RESTful services. We extract resources from SOAP-based services and Web applications. Our approach allows different services to interact using the same application protocol (i.e., HTTP). The user agent does not require sophisticated tools to invoke the services. Figure 3 abstract the Web services into three layers: a heterogeneous Web resource layer which hosts different types Web services available in various service providers; a model layer which

captures the business processes that integrate services in the Web resource lay; and a concept layer which abstracts the semantic meanings of the functionality provided a Web resource and can be understood by end-users. We extract Web resources for different heterogeneous Web services and represent these services in the unified schema using RESTful services.

To migrate SOAP-based services, we analyze the WSDL document of a SOAP-based service and build a dependency graph which describes the relations between operation names, input and output parameters of an operation defined in the interface of a SOAP-based service as shown in Figure 4a. We identify and group similar operations from the dependency graph as shown in Figure 4b. Each cluster of operations is analyzed to identify resources and the HTTP methods associated with the resource. Furthermore, we manually refine the identified resources and HTTP methods. Once the resources are validated and verified by a user, the wrappers and configuration files are automatically generated. We identify resources from a SOAP-based Web service by analyzing its service description and mapping the contained operations to resources and HTTP methods. We provide a semi-automatic technique to migrate a SOAP-based service to RESTful services. We use cluster analysis and natural language processing to identify resources and the associated HTTP methods. Our previous work [12] provides more detailed discussion about resource extraction. In the future, we plan to identify reusable tasks (*e.g.*, user registration, searching) from Web applications and represent these tasks as Web resources. Extracting reusable tasks involves understanding the client side [7, 9] representation in a Web page.

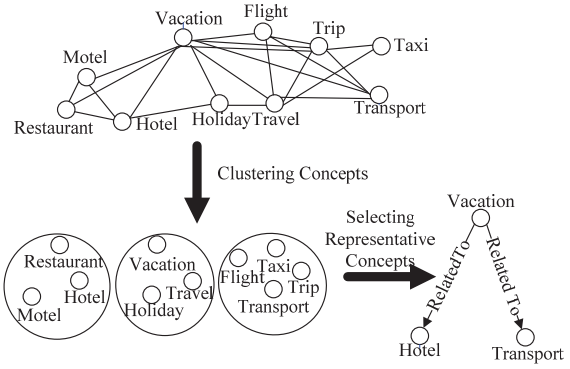


Figure 5: A Process for Creating a Concept Map

2.3 Discovering Services using Concept Maps

Service discovery techniques help a user to identify and select services based on the query that he or she submitted. Our proposed approach consists of two major steps: 1) service indexing which extracts concepts from service description documents and cluster services using the concepts conveyed in the service description; and 2) service retrieval that extracts concepts from a user's query, guides users to formulate queries and returns the services associated with the concepts. The service indexing is an offline procedure which is invoked once for each service. The service retrieval is an online step that is executed for each query. In the service retrieval step, we provide a mechanism to recommend concepts and allow a user to navigate through the concepts associated with services.

As shown in Figure 4, we model heterogeneous services and then extract concepts related each service in the concept layer. A concept is a semantic notion or a keyword for describing a subject (*e.g.*, "traveling", "weather" or "taxi reservation"). It can be a single word, an idiom, a restricted collocation or a free combination of words. We use noun phrases extracted from the service documents as concepts. We create a concept map to help a user visually select concepts to formulate their query. A concept map is a set of concepts identified from a user's query and their relations. The concepts are represented as nodes. A relation between two concepts is derived from repositories (such as DBPedia [21], ConceptNet [14] and WordNet [13]) and illustrated as an edge linking two concepts. To create a concept map, we cross-reference concepts in a query with semantically related concepts in the service repository. For example if the concept "vacation" is present in a user's query, we retrieve concepts related to "vacation" from ConceptNet, provided that there are services associated with the retrieved concepts. For example the concept "vacation" is related to {motel, hotel, restaurant, flight, trip, taxi, transport, travel, and holiday} as shown in Figure 5. There may be a large number of related concepts which can be overwhelming to users. We cluster related concepts by measuring their word similarity. Each cluster is represented by a representative concept. We find a representative concept in a concept cluster. We represent the selected representative concepts and their relations as a concept map. Figure 5 shows the process of creating a concept map. An end user submits a query and our approach recommends the more specific sub-concepts and option concepts related to ones available in user's query. Thus an end user can articulate more specific query and retrieve more precise result. We will propose an approach to index services based on the concepts shared between services. Our approach will guide users to formulate queries and group the retrieved results. Our approach can minimize the human effort to find a specific service and bridge the semantic gap between users and service providers by assisting the users in formulating queries and recommending services related to a user's query. Moreover, our approach uses user histories and preferences helps to personalize the process of service discovery. The history and preference will help to rank the relevant services.

2.4 Generating Ad-hoc Processes for Service Composition

Service composition is a process to combine collaborating services to obtain higher functionality. We perform service composition by generating ad-hoc processes which provides flexibility in modeling in day to day activities [30]. Mining service repositories and execution logs provide templates to generate ad-hoc processes [4]. The uniqueness of ad-hoc processes is to support users not only following the repetitive routines as specified in workflows but also unique non-recurring situations. An example for generating an ad-hoc process is to plan a trip. A user states his preferences, *e.g.* comedy movie and dinning in a restaurant with Indian cuisine. An agent in an ad-hoc process reserves a table in a restaurant and a ticket for a movie. In this scenario a user wants to find a cinema showing comedy and an authentic Indian restaurant. For this purpose, an agent contacts a movie recommendation service in order to discover a comedy movie, as well as a yellow page directory to select an Indian restaurant. Afterwards, the agent searches for a cinema which plays the selected movie and uses a recommendation service to ensure that the selected restaurant has a satisfactory rating. Finally, the agent returns a restaurant and cinema combination to a user. As it can be seen in this example, this newly generated ad-hoc process can support a user in their everyday life where

situations are unique and usually not appear in the same way more than once. Based on user requirements, we will design a service composition engine that generates such ad-hoc processes, which integrate individual services in order to provide the desired functionality. Ad-hoc processes may require negotiations and human judgment, or a change in structure or participants as the work progresses. We will define the notion of a work item to specify a set of tasks along with their data flow for fulfilling a transaction. Xiao et al [31] have proposed an ad-hoc process for service composition based on tag suggested by a user. Our approach extends Xiao et al [31] approach by using a concept-based composition of heterogeneous services. A concept is different from tag and represents semantic information related to a people, place or thing same to all users.

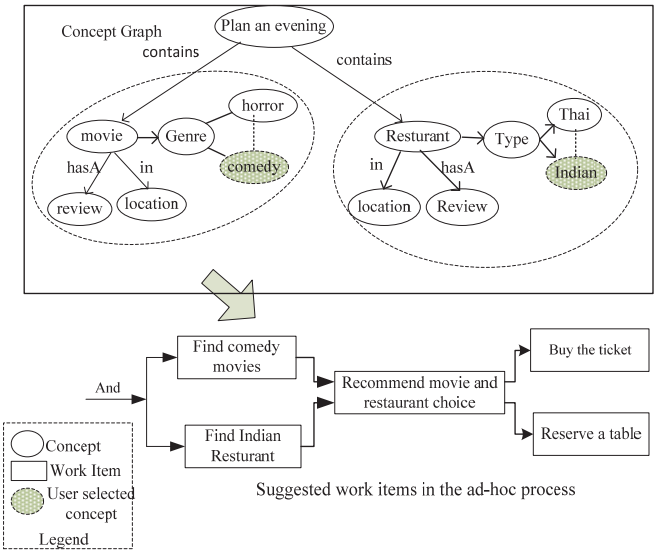


Figure 6: An Example for End-User Service Composition

Figure 6 presents an example of a mapping relation in the concepts and processes the ad-hoc process. In Figure 6, the user selected concept is turned into work items such as "Find comedy movie" and "Find Indian Restaurant". Our approach finds the suitable combination and asks for user confirmation. In the whole process, a user doesn't have to know how and which services are invoked. A user just mashes one or more concepts and provide his preferences. We will provide a framework that allows a user to connect different concepts to fulfill the goal. We will use a goal description to find a matching ontology to extend the semantic meanings of the goal. We will group Web services with similar functionalities and design an algorithm to identify tasks for the ad-hoc process. We will provide a language represent an ad-hoc process. The research challenges in this step will be the process of hiding the technical details from a user.

3. Evaluation

In this section we discuss some of our preliminary evaluation of our approach. We will discuss the effectiveness of our approach to identify resources from SOAP-based services and our plan for evaluating future work.

3.1 Evaluation of Resource Identification from SOAP-based Services

To validate our approach for identifying resources from SOAP-based Web services, we collect 61 WSDL documents from various categories, such as finance, government, travel/tourism, and e-commerce. For each operation in WSDL, we identify the resources and then manually verify the accuracy of resource identification process. Table I lists the summary of our approach. We identified resources and HTTP methods for each operation in WSDL. We analyzed the WSDL documents manually to examine the accuracy of the identified resources and the corresponding HTTP methods. Table I summarizes the results of accuracy of the identified resources. The accuracy is 74% showing that our approach can successfully identify the resources in most of the cases. There are 410 operations in those 61 WSDL documents, for which there are 284 resources. Therefore, each resource may not always have four operations exposed for the users. We found most of Web services in our study to be read centric as most of the resources expose retrieval HTTP-verb (i.e., GET).

Table I: Summary of RESTful Resource identification

Number of WSDL documents Analyzed	61
Total Number of Operations	410
Number of Resources Identified	284
Misidentified Resources	46
Total Number of Resources	320
Average Accuracy	0.74

We found the performance of RESTful service better than SOAP-based services [12]. Even with the delay introduced by the wrappers, the performance of RESTful services is faster than of the corresponding SOAP-based services. The faster response time for the RESTful services favor mobile and thin clients. It is clear from the results that RESTful services have performance benefits compared to WSDL/SOAP-based services.

3.2 Evaluation for Future Work

For the future work, we plan to evaluate the precision of our approach to extract services from Web applications. We will conduct case studies using real world Web application. Similarly, we plan to evaluate the effectiveness of our service discovery approach based on user queries in terms of precision and recall. We will compare the performance of our approach with traditional approaches. We plan to perform a user study to measure the effectiveness of concept recommendation and query formulation in service retrieval. We will use real world service description languages, such as WSDL, and WADL in our case study.

We will conduct a user study to evaluate our approach to generate ad-hoc processes. We will examine the correctness; easiness and the duration taken by a user to generate ad-hoc processes for a given goal (*e.g.*, plan a trip). The correctness checks how precisely our approach can perform a given task. The easiness evaluates how easy it is for a user to generate an ad-hoc process. The duration measure the times used by a user to generate an ad-hoc process. We will also examine the user satisfaction of the service composition process.

4. Conclusion

In this thesis, we aim to address the problems faced by end users in service composition. Our proposed approaches unify service description languages, identify resources, help users in services discovery and service composition. We present a meta-

model to unify different heterogeneous service description languages. A unified schema is capable of representing different kinds of services. We propose an approach for a concept-based analysis for guiding users in service discovery. Our service discovery approach helps a user to bridge the semantic gap between service providers and user queries. Our work will reduce the burden for end users to remember and parse different service description languages. Additionally we use a single communication standard to invoke heterogeneous Web services. Our approach allows the end users to compose services to fulfill their day to day activities.

In future, we plan to identify reusable business functionality from Web applications. We will implement an approach to automatically generate ad-hoc processes based on concepts available in user's task description. We will design and develop a framework that allows a user to perform service composition of an ad-hoc process given a goal.

REFERENCES

[1] X. Dong, A. Halevy, J. Madhavan, E. Nemes and J. Zhang. Similarity search for Web services, In Proc. of the 13th international conference on very large data bases, 2004.

[2] R. T. Fielding. Architectural styles and the design of network-based software architectures, Doctoral Dissertation, University of California, Irvine, 2000.

[3] E. Al-Masri and Q.H. Mahmoud. WSCE: A Crawler engine for large-scale discovery of Web services. In Proc. of the IEEE International Conference on Web Services (ICWS), pp.1104-1111, 2007.

[4] B. Upadhyaya, R. Tang and Y. Zou. An approach for mining service composition patterns from execution logs. Journal of Software Evolution and Process; DOI: 10.1002/smr.1565, 2011.

[5] F. Liu, Y. Shi, J. Yu, T. Wang and J. Wu. Measuring similarity of Web services Based on WSDL. In Proc. of the ICWS, pp. 155-162, 2010.

[6] H. M. Sneed and S. H. Sneed. Creating Web services from legacy host programs. In 5th International Workshop on Web Site Evolution (WSE), pp. 59–65, 2003.

[7] S. Oney and B. Myers. FireCrystal: Understanding interactive behaviors in dynamic Web pages. IEEE Symposium on Visual Languages and Human-Centric Computing, pp. 105-109, 2009.

[8] G. Lewis, E. Morris, and D. Smith. Analyzing the reuse potential of migrating legacy components to a service-oriented architecture. In Proc. of the 10th Conference on Software Maintenance and Reengineering, pp. 16-23, 2006.

[9] P. Li and E. Wohlstadter. Script InSight: Using models to explore JavaScript code from the browser view. In Proc. of the 9th International Conference on Web Engineering. pp. 260 – 274, 2009.

[10] April 2012 Web Server Survey, http://news.netcraft.com/archives/category/web-server-survey/

[11] Web Architecture, http://www.w3.org/standards/webarch/

[12] B. Upadhyaya, Y. Zou, H. Xiao, J. Ng and A. Lau. Migration of SOAP-based services to RESTful services, In Proc. of the 13th IEEE International Symposium on WSE, pp. 105-114, 2011.

[13] G. A. Miller. WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11 pp. 39-41.

[14] H. Liu and P. Singh. ConceptNet — A practical commonsense reasoning toolkit. BT Technology Journal, Vol. 22, No. 4, pp. 211-226, 2004.

[15] O. Diaz, S. Perez and I. Paz. Providing personalized mashups within the context of existing Web applications. In Proc. of International Conference on Web Information Systems Eng., pp. 493-502, 2007.

[16] C. Pautasso, O. Zimmermann and F. Leymann. Restful web services vs. "big" web services: making the right architectural decision. In Proc. of the 17th international conference on WWW, pp. 805-814, 2008.

[17] M. Maximilien, H. Wilkison, N. Desai and S. Tai. A Domain Specific Language for Web APIs and Services Mashups. In Proc. of the IEEE ICSC, pp. 13-26, 2007.

[18] N. Mehandjiev, A. Namoune, U. Wajid, L. Macaulay and A. Sutcliffe. End User Service Composition: Perceptions and Requirements. In Proc. of the IEEE 8th European Conference on Web Services, pp. 139-146, 2010.

[19] Yahoo! Pipes, http://pipes.yahoo.com/pipes

[20] Microsoft Popfly, www.popfly.ms/microsoft-popfly

[21] DBpedia, http://dbpedia.org

[22] Business Process Execution Language, http://www.ibm.com/developerworks/library/specification/ws-bpel/

[23] W3C. HTTP Specification. www.w3.org/protocols.

[24] Link Relations, www.iana.org/assignments/link-relations

[25] Web Service Definition Language , www.w3.org/TR/wsdl

[26] WADL, www.w3.org/Submission/wadl/

[27] R. Alarcón and E. Wilde. RESTler: crawling RESTful services. In Proc. of the 19th international conference on WWW, pp. 1051-1025, 2010.

[28] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers, M. B. Rosson, G. Rothermel, M. Shaw and S. Wiedenbeck. The state of the art in end-user software engineering. ACM Computing Survey, 43, 3, April 2011.

[29] L. Calderon-Benavides, C. Gonzalez-Caro and R. Baeza-Yates. Towards a deeper understanding of the user's query intent. In Query Representation and Understanding. A workshop at the SIGIR, 2010.

[30] H. Xiao, B. Upadhyaya, F. Khomh, Y. Zou, J. Ng and A. Lau. An automatic approach for extracting process knowledge from the Web. In Proc. of ICWS. pp. 315-322, 2011.

[31] H. Xiao, Y. Zou, R. Tang, J. Ng and L. Nigul. An automatic approach for ontology-driven service composition, IEEE International Conference on SOCA, pp. 1-9, 2009.

[32] C. Pautasso. A Flexible System for Visual Service Composition. Doctoral Dissertation. Department of Computer Science, ETH Zurich, Diss. No. 15608, July 2004.

All URLs are last accessed on 5th October, 2012