International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-2, Issue-9)

Research Article | **September 2013**

Service Mash up Using Knowledge Discovery in Services

***R.K.Rekha , A.Arence Flemi**
**Assistant Professor,*
Department of Computer Science,
Vel Tech Multi Tech Engineering College, India

Abstract—

Service mash up is the act of integrating the resulting data of two complementary software services into a common picture. Such an approach is promising with respect to the discovery of new types of knowledge. However, before service mash up routines can be executed, it is necessary to predict which services (of an open repository) are viable candidates. Similar to Knowledge Discovery in Databases (KDD), we introduce the Knowledge Discovery in Services (KDS) process that identifies mash up candidates. In this work, the KDS process is specialized to address a repository of open services that do not contain semantic annotations. In these situations, specialized techniques are required to determine equivalences among open services with reasonable precision. This paper introduces a bottom-up process for KDS that adapts to the environment of services for which it operates.

Keywords— service mash up, web-based services, filtering, clustering, and categorization

1. INTRODUCTION

SERVICE-ORIENTED COMPUTING (SOC) supports the notion of thousands (even millions) of modular software capabilities discoverable and usable by disparate organizations. Underlying SOC are specifications, such as the Web Service Description Language (WSDL), and the invocation methods supported by SOAP or REST that provide machine-interpretable interfaces and interactions. Such open approaches provide an environment where atomic modules (i.e., web services) can be discovered and composed on demand. This type of interaction, in a business setting, is optimal because it allows the underlying business units or divisions to act autonomously while, at the same time, facilitating active collaboration at a layer of granularity that each stakeholder can dictate and control. As such, a business unit can discover relevant capabilities then consume and compose them in terms of their existing capabilities to create entirely new offerings.

A service mash up is the simultaneous execution of two or more services to create an integrated data provision with a more complete description about some object or task. For example, web services from a web search company such as Google Corporation that provides mapping capability can be integrated with capabilities from a shipping business such as the United Parcel Service (UPS). The resulting mash up could visualize the path of parcels that get lost in the delivery process.

This example is illustrated in Fig. 1. This integration of web services outputs is the general idea behind service mash up. While composition involves linking services into a higher level workflow or business-oriented process, service mash up is closer related to running services in parallel and integrating the resulting data.

There are three major challenges addressed in this work.

1. In environments where web service-based semantic definitions are not available, high-precision syntactical approaches must be in place to infer equivalences among services using direct and indirect information from service specifications (Equivalence Processing).
2. Characteristics that make two of more services capable of integration or mash up (Clustering) must be well under-stood and adaptable as the nature of service repositories evolve.
3. Of the services that have sufficient equivalence to support integration or mash up, the subset that actually provides value-added information to end users must be identified (Categorization/Filtering).

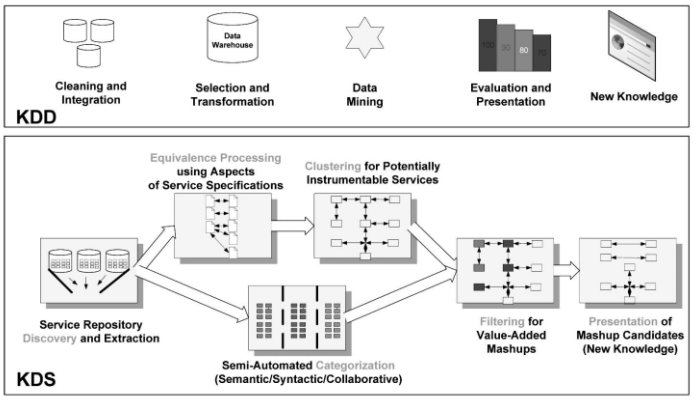
2. KDS AND RELATED APPROACHES

Although the area of data integration has had a long-standing background, the use of these techniques to accomplish mash up has only just recently started to be addressed. A majority of the work in this area addresses the tools and environments that support the visualization and presentation of mash up.

2.1 KDS and KDD

The four phases of KDD, cleaning and integration, selection and transformation, data mining, and evaluation and presentation can be compared to the KDS phases of discovery, equivalence processing, clustering, categorization, filtering, and presentation .The KDS process is shown together with the KDD process. The Cleaning and Integration step in the KDD process assumes that noise must be removed from initial data in parallel with the integration of multiple data sets.

ARCHITECTURE:



In contrast, the integration of multiple repositories in KDS's Discovery phase is aided by common standards for storing services (e.g., Universal, Description, Discovery, and Integration (UDDI)). Discovery in KDS also relates to the extraction of parts and other descriptive service-based information (such as service name, operation name, type name, descriptive fields, etc.) from the service specification. Even before considering equivalent services, this step attempts to identify equivalent parts

Similar to KDD's Selection phase, the KDS's Categorization phase attempts to group services that use similar terms into like categories. For example, services that use a large number of financial terms may be grouped into a set of services characteristic of banking operations. In our approach, a category is less defined by a title or operation but more defined by a bag of characteristic strings taken from its underlying service specifications.

Once equivalences are made between specific parts, services can be linked together by their output and input messages during the Equivalence processing and clustering phases. In this step, a list of potential mash up can be determined solely based on the fact that certain services share equivalent parts. The Filtering phase attempts to identify the mash up that are indeed useful to a particular stake-holder.

2.2 Determining Equivalences in Software Services (Equivalence Processing)

Approaches to service mash up are similar to the fundamental techniques for the discovery and composition of web services. Two common approaches to composition, consumption and perhaps mash up are semantic and syntactic techniques. Semantic approaches generally support the integration of web services by exploiting the semantic description of their functionality using ontological approaches

Here, in the experimentation with service mash up, we operate with non-semantically annotated services. A recent approach attempts to create semantics from extensible Markup Language (XML)-based specifications. Similarly, syntactical approaches are the focus of this work. However, future approaches will require the combination of lightweight syntactical approaches and heavier semantic reasoning.

2.3 Determining Service Provisions that are Complementary (Clustering)

The Clustering phase attempts to define groups of services that may be complementary and eligible for mash up.

2.4 Grouping Services and Determining Value (Categorization and Filtering)

This approach also attempts to understand when two complementary services add value to a user. This determination tends to be relegated to semiautomatic approaches. This approach attempts to fully automate this determination. The major questions that we seek to answer are "Are there specific categories of services that typically add value when integrated?" and "Can it be assumed that a service that integrates with a large number of services (i.e., congenial services) are likely to add value for new mash up. Since service mash up is such a new area, there were no closely related projects investigating these questions in the service-oriented computing domain.

3. EQUIVALENCE PROCESSING

3.1 Discovering Trends in Web Service Message Naming
3.2 Using Natural Language Approaches to Exploit Trends in Message Matching
3.3 Various Approaches for Syntactical Message Matching
3.4 Evaluating Similarity Approaches on Open Web Services

3.5 Assessment of the Similarity Approaches on Controlled Web Services.

4. CLUSTERING INSTRUMENTABLE SERVICES

An effective mash up blends two services that are not similar in function, but rather similar in the messages that they provide. In this way, identifying a similar message part, either input or output, between two services is the first step in our approach to predicting a service mash up. This approach combines syntactical methods with human naming tendencies to increase the probability of the messages having equivalent meanings. After finding similarity between web service messages, we place services with similar messages into clusters. Services can be associated to more than one cluster, but later they will have be attached to only one category. In the final phase, we attempt to predict the clusters that will add value to the end users.

4.1 Determining Instrumental Services

The approach to finding instrument ability starts with the selection of a web service operation.

4.2 Evaluating Instrument ability at Various Levels

Web services are represented by a combination of strings within their specifications.

KDS promotes the bottom-up evaluation of a repository of web services to discover new information via mash up. As such, these experiments are important in determining how the suite of KDS tools should be customized. In many cases, the choice of assessment level (i.e., service level, operation level, or part level) is dictated by the nature of the repository that is under assessment. Although categorization is most effective at the service level, initial clustering requires the use of input/output parts. Future experiments will determine hybrid approaches that blend levels based on the specific KDS objectives.

5. CATEGORIZATION

The idea of categorizing a repository or set of information is not a new one. It is useful to be able to target a certain type of information within a larger collection. Here, as with the search for instrumental services, we work from a predefined set of services. Upon creating our repository of 545 services, we classified each service into one of nine categories. These categories followed closely the categories defined in related work [10]. As a first step, services were manually stratified into categories based on a human inspection of the name of the service and a brief analysis of its WSDL specification. The result is a reasonable, human-generated classification of services. This categorization is an aspect of the filtering that quantifies the value of a particular mash up prediction. However, in production environments, human-generated categories would not be feasible.

5.1 Automatic Categorization of Services

Since a human-generated categorization approach would be impractical in an open services environment, we introduce an automated approach entitled Categorization On Pairing (COP). This approach clusters services into categories based on the similarity of their specifications.

The COP method created many more categories, but the Service Similarity factor shows that each category was more closely related. Perhaps promising is the fact that the percentage of unique descriptors for the COP method was only slightly higher than the manual approach. These results suggest that the COP method appropriately creates many tightly defined categories.

5.2 Categorizing Services by Varying Similarity

The COP method produces a large number of categories, however, at times; certain domains may require a smaller or user-designated number of categories.

6. FILTERING BY EXPLOITING CATEGORIES

The Filtering phase within the KDS process attempts to predict which mash up predictions will add value for a particular user. Although we intend to use human-generated data to filter our predictions in the future.

7. PERFORMANCE ASSESSMENT

To assess the variance in performance across the prediction methods, we recorded the average time for each method over five different executions.

8. CONCLUSION

Existing web services on the Internet to gather insight about how services are developed and how service-based messages are named. As a result, we developed several natural language processing approaches (i.e., TSM-L, TSM-P, and TSM-LP) that mirror the nature of the open web services. In this work, we explored how these approaches could be applied to the domain of predicting service mash up. Results show that the TSM-L method provides the largest percentage of valid predictions. In addition, the recommendation performance is favourable with regard to making real-time recommendations. The TSM-L method is most effective on message names that utilize abbreviations. This suggests that the open repository contains many abbreviations as a part of service messages.

In future: we plan to assess the ability to predict service mash up using a combination of inputs and outputs.

Future application is the creation of a distributed web services development environment that leverages the knowledge of existing services.

Another future project would be the integration of our approach for use as a front-end to the emerging service mash up editors and visualization environments.

ACKNOWLEDGMENT

I would like to thank my Dean, HOD and Faculty members of my department for their cooperation and encouragement towards us to develop this innovative idea. And we personally thank our colleagues to finish this paper successfully.

REFERENCES

[1] Amazon Web Services, www. amazon.com/gp/aws/ landing.html, 2011.
[2] B. Benatallah, M. Dumas, and O.Z. Sheng, "Facilitating the Rapid Development and Scalable Orchestration of Composite Web Services," Distributed and Parallel Databases, vol. 15, no. 1, pp. 5-37, Jan. 2005.
[3] M.B. Blake, "Knowledge Discovery in Services," IEEE Internet Computing, vol. 13, no. 2, pp. 88-91, Mar. 2009.
[4] M.B. Blake and M.F. Nowlan, "Predicting Service Mash up Candidates Using Enhanced Syntactical Message Management," Proc. Int'l Conf. Services Computing, July 2008.
[5] M.B. Blake and M.F. Nowlan, "Taming Web Services from the Wild," IEEE Internet Computing, vol. 12, no. 5, pp. 62-69, Sept./Oct. 2008.
[6] M.B. Blake, K.C. Tsui, and A. Wombacher, "The EEE-05 Challenge: A New Web Service Discovery and Composition Competition," Proc. IEEE Int'l Conf. E-Technology, E-Commerce, and E-Services, Mar. 2005.
[7] A. Bosca, A. Ferrato, D. Corno, I. Congui, and G. Valetto, "Composing Web Services on the Basis of Natural Language Requests," Proc. Third IEEE Int'l Conf. Web Services (ICWS '05), pp. 817-818, June 2005.