Extending UDDI with Recommendations: An Association Analysis Approach

Andrea Powles and Shonali Krishnaswamy

School of Computer Science and Software Engineering
900 Dandenong Road, Monash University, Caulfield East, Victoria –3145, Australia.

Abstract. This paper presents a novel recommendation extension to UDDI that we term RUDDIS. Recommendations can have potential benefits to both providers and consumers of Web Services. We adopt a unique technique to making recommendations that applies association analysis rather than traditional collaborative filtering approach. We present the implementation and demonstrate the functioning of RUDDIS in an unobtrusive manner where the user has total control over the recommendation process.

1 Introduction

Recommendations are used in a wide variety of e-commerce applications such as Amazon.com. Recommendations are useful for both buyers and sellers. For sellers, they provide a means to highlight additional products and for buyers they provide a filtered list of options to consider.

A natural and intuitive extension to the use of recommender systems in e-commerce is the investigation of such recommendations within web services. With the increasing recognition of the commercial potential of the service oriented paradigm, it is conceivable that recommendations within services can be of significant benefit and use. Providing recommendations for Web Services has the potential to offer many benefits for accessibility and usability of Web Services for both users and providers. It could present users with alternative or additional Web Service selections thus improving the likelihood that a web service will be consumed and that useful Web Services are being provided to users. Web Service providers could see an increase in the use of the Web Services they are providing as they would see greater exposure to possible users.

This paper presents a first investigation into incorporating a recommendation component within the web services framework. We propose a plug-in component to UDDI that extends the functionality of the UDDI from a discovery mechanism to one that performs discovery and recommendations for web service search queries. The focus on UDDI is evidently due its role as the standardised directory service component within current web services framework. As the UDDI specification has been designed with extensibility as a priority it is limited to only a few set functions. There have been numerous models and implementations for extending the UDDI

Powles A. and Krishnaswamy S. (2005).
Extending UDDI with Recommendations: An Association Analysis Approach.
In *Proceedings of the Joint Workshop on Web Services and Model-Driven Enterprise Information Systems*, pages 67-76
DOI: 10.5220/0002561900670076
Copyright © SciTePress

specification to address existing limitations including Rashid (2003), Lyell (2003), Systinet (2004), and Pokraev (2003).

While there have been many extensions of UDDI, there is yet to be any investigation into the use of recommendations for Web Services. With the expected increase of Web Service use, it will be beneficial for the UDDI to be able to provide recommendations of Web Services. Web Services would gain additional exposure through a UDDI that provides recommendations and users or systems looking to integrate or consume a particular Web Service would benefit as they are provided with additional Web Services that could be of use to them.

In developing a recommendation framework for service oriented architectures – we had two possible options: Automated Collaborative Filtering (ACF) (Herlocker 2000) and content-based approaches (Sarwar 2004). Automated Collaborative Filtering (ACF) is a widely used process for recommending information, services or physical items that are of potential use for a person based on ratings provided by other "similar" users. In content-based approaches the focus is on usage patterns rather than having a user focus. In ACF, similarity of users is typically based on maintaining user profiles. ACF is used in a wide variety of applications such as e-commerce (typically agent-based systems such as (Guttman 1998), recommendations for books [Amazon.com], music [CDNow.com] and movies [MovieFinder.com]. The key distinguishing feature of ACF as opposed to "content-based" approaches to making recommendations is the incorporation of the user dimension. However, there are several major challenges in developing ACF systems [HKR00]: the difficulty in developing valid user profiles, the question of mapping user profiles to individual preferences and tastes for varied items and services, the application of user ratings that do not capture the rationale for the ratings provided, the dependence on user ratings that tend to be subjective, and the requirement for users to provide additional information and perform extra tasks in providing the ratings.

The entire premise of ACF rests on the notion that similarity between users can be captured and represented. This premise is certainly valid when the objects in question are movies, books or music - but becomes intractable when the question pertains to web services. Consider questions such as: what makes two users invoke a particular weather information service as opposed to another? Building such user models for services is a worthwhile consideration – however, it requires considerable user psychology and usage patterns to be available in order to develop such user models. Furthermore, the dependence on users providing ratings is very obtrusive and the uptake of this - given the very limited incentives in a service oriented environment - is also questionable.

Therefore, this project aims to address these issues of ACF by using a content-based approach. This project proposes the use of association analysis [WiE99] to support recommendations in the web service environment. Association analysis or association rule mining is widely used as a data mining technique in the retail industry to perform tasks such as Market-Basket Analysis to search for interesting customer habits by looking at associations (Witten 1999). The classical example is the one where a store was reported to have discovered that people buying nappies tend also to buy beer. It is also used in applications in marketing, store layout, customer segmentation, medicine and finance. However, the value of using the concept of association analysis in a wider context is slowly emerging with applications in content-based image retrieval [MSP04]. The primary aim of association analysis is to

discover groups of items that occur together. Given a set of transactions {T}, each containing a subset of items from an item set $\{i_1, i_2, \ldots, i_m\}$, the focus is on the discovery of association relationships or correlations among a set of items. The strength of such associations is expressed by means measures known as *support* (i.e. the probability of a set of items occurring together ($P(i_j \cup i_k)$) and *confidence* (i.e the conditional probability of a set of items (i_k) appearing given that a set of items (i_j) exists ($P(i_k \mid i_j)$)). The *support* indicates the frequency, while *confidence* denotes the strength of the association. In the context of service oriented environment, this technique alleviates many of the disadvantages highlighted with ACF, while retaining the strength that it is also like ACF derived from a user-centric basis. In association analysis, the transaction is derived from user activity – that is the user determines how services are invoked in conjunction with each other. This is the fundamental basis for performing an association analysis. On the other hand, it does not presume to build or rely on user profiles and identification of similarity between users, which is inherently challenging in the context of service oriented environments at this stage of its evolution. However, it may be foreseen that in future when such widespread user models and interactions are available ACF maybe used to in conjunction with content-based approaches (Pennock 2001)enhance results obtained through techniques such as association analysis. Furthermore, the occurrence of objects / items in a transaction is an easily documented event and there is no additional overhead in getting users to rate the objects / items they use. This "preferential rating" may easily be established through implicit means such as frequency and duration of usage by the same user.

We also note that the use of data mining techniques for recommendations in e-commerce (Schafer 2001) has been validated. The paper is organized as follows. In section 2 we present the design considerations and architecture of our UDDI extension to perform recommendations – RUDDIS. Section 3 presents the implementation of RUDDIS. Section 4 demonstrates its functioning using both a local and external UDDI. Finally section 5 concludes this paper.

2 Recommendations in UDDI (RUDDIS)

We are proposing the use of UDDI as a Recommender system. We term this model as Recommender Universal Description, Discovery and Integration System (RUDDIS). RUDDIS will consist of a UDDI registry encompassing a Recommendation component. This section examines the considerations and issues for the RUDDIS model. An UDDI that includes a Recommendation component should contain the following features:

- The model should conform to the UDDI specification and have minimal or no impact on the existing Web Services stack. The specification integrity being maintained is vital to the entire infrastructure and purpose of Web Services. Should the integrity of the specification be violated then interoperable nature strived for by Web Services may be foregone.
- Minimal effort should be required from the user to utilise RUDDIS compared to utilising a standard UDDI.

- The model should support the provision of useful and meaningful recommendations.

The key concern to be deliberated with regards to the design for the UDDI framework that includes a recommendation component is how to keep the UDDI compliant with the specification. The UDDI API Specification document provided by OASIS (Bellwood 2002) describes the programming interface and expected behaviours of all instances of the UDDI registry. When enhancing UDDI it is crucial to keep the standards set by this specification document. The UDDI data structures in the specification provide a framework for the description of basic service information, and an extensible mechanism to specify detailed service access information using any standard description language. Web Services are based on open standards which is the key to its heterogeneity. Altering the standards could damage the ability of others being able to use the Web Service stack. In this context, it is essential for RUDDIS to keep the recommendation and the UDDI components separate to ensure the compliance of the existing standards. The recommendation component and the UDDI component will be able to plug into each other via the calls made to and from RUDDIS. This way the UDDI will not require any internal modification and will maintain its integrity. This will allow the UDDI to function as normal. Service providers still wishing to register Web Services in the UDDI can do so as per usual. Clients wanting to search for Web Services without being provided with recommendations can do so with the RUDDIS model. With this transparency being modelled the user may never know they are using an extended UDDI.

In order for RUDDIS to provide useful recommendations we investigate the use of Recommendations using Market Basket Analysis. Market Basket Analysis is mainly used for data mining in the retail industry for discovering association rules between items in the data (Witten 1999). For example if we have a video shop that has a database of every hire transaction ever made over the history of the store. Each transaction contains customer details, the videos hired, how many and the video type e.g. Comedy, Romance, Horror, Drama or Action. As we mine through this data we find that in the cases where there was more than one video hired, selecting type Drama, 50% of the time also implies a comedy video was also hired. Then rule can be described as "drama" ➔ "comedy". Knowing such information can be very useful. In this case the store manager could place the Drama and Comedy sections closer together or introduce a special promotion for the two types when hired together. Association analysis is a relatively simple yet effective analysis tool and should be able to be implemented into a Recommendation System algorithm with ease. The Apriori algorithm (Witten 1999) or its many variants and enhancements are widely used as an effective implementation tool to support association analysis. It is simple and is computationally efficient. We propose to use Apriori for facilitation recommendation in RUDDIS.

We now examine how to ensure the model supports the provision of useful accurate recommendations. For the recommendations to be accurate, data being used to generate the rules needs to be accurate and up to date. RUDDIS is concerned with firstly how to obtain the data that will be used to generate the association rules then secondly, often the rules will be refreshed. Refreshing of the rules will require extra processing by the system which may slow down the performance. There is the need to weigh the importance of the performance of the system against the provision of the most accurate recommendations. We establish that to obtain the data required to

generate the association rules the users interactions with RUDDIS will be recorded. The details of all the queries made by users will be saved into a RUDDIS usage database. When the rules require refreshing the data from the RUDDIS usage database will be run through the Apriori algorithm to generate the rules. Also established is that the user should be able to determine the frequency of updating the rules. This way the user has control over the performance of RUDDIS as updating requires extra processing power. We also believe that it is essential to design this recommendation component such that it can be situated at the client or UDDI server for maximum flexibility.

2.1 RUDDIS Architecture

A scenario that could take place with the use of RUDDIS, is that a possible user is interested in searching for Web Services to do with planning a family trip to the east coast. Using RUDDIS, the user is looking up Web Services on airline flights to get there. The RUDDIS usage database contains all the Web Service requests made to RUDDIS and to which session it belonged. The RUDDIS usage database is utilised when recommendation rules need to be found. When our user enters in the query "flights", it is recorded in the RUDDIS usage database. Any other Web Service requests made by the same user at the one time will also be recorded under the same session. Also occurring in RUDDIS, is the data from the RUDDIS usage database being run through the Apriori algorithm producing a set of association rules. RUDDIS then seeks out any rules supporting "flights", if there are rules for this query item existing, the supporting rule with the strongest confidence level is found and the association item in that rule is extracted. So there maybe two rules for our query found such as "flights" ➔ "car hire" and "flights" ➔ "accommodation". Which ever rule of the two has the strongest confidence level for example the "flights" ➔ "accommodation" rule, gets the associated item extracted, in this example, "accommodation". The original query "flights" is then queried in the UDDI registry which contains the details of all registered Web Services. The association item extracted, "accommodation" is then also queried in the UDDI registry. For either of the two queries any Web Services found, are compiled and presented to the user, who may then decided to proceed integrating the Web Services.

The following five elements illustrated in Figure 1 have been identified as being required to carry out the tasks needed to be accomplished by RUDDIS.

The Manager Component: Is in control of handling all the interactions with the interface. As a Web Service request comes through the Manager Component evaluates the environment options selected by the user and the query item. It then directs the requests being made to the appropriate components. Any items returning from the other components are managed and acted upon by the Manager Component.

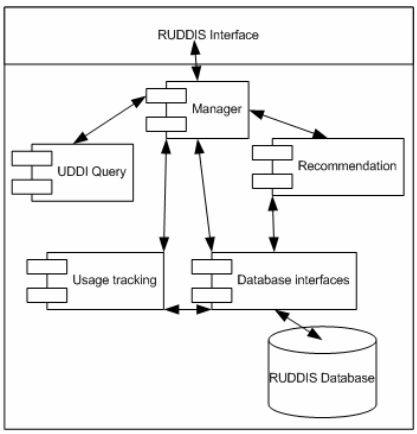

Fig. 1. RUDDIS Architecture

UDDI Query Component: Is in control of interacting with the UDDI registry which stores all the Web Services details. When passed a query item by the Manager Component it encapsulates the query into the appropriate format used for inquiries to the UDDI. It is supported by the UDDI API client framework which assists in the discovery of Web Services when requests are made. Any Web Services retrieved are then passed through to the Manager Component to deliver back to the user.
Database interfaces Component: Is in control of monitoring any databases within RUDDIS. Primarily this will be the RUDDIS usage database, but enables the provision of additional databases to be added to the system if this flexibility is required. Interactions between the Manager, Recommendation, Usage tracking and Databases interfaces Components occur when the tracking data is being recorded and when the RUDDIS usage database data is needed to run through the Apriori algorithm.

Recommendation Component: Is in control of discovering the association rules in RUDDIS and finding the strongest rule for the Web Service query being made by the user. If any results are found they are then passed back to the Manager Component to forward onto the UDDI Query Component, who sees if any correlating Web Services exist in the registry. It ensures the processes of extracting the data from the RUDDIS usage database and running the Apriori algorithm to generate the association rules.

Usage tracking Component: Is in control of ensuing that the users session and all the Web Services requested during the session are recorded. This data will be stored in the RUDDIS usage database through the use of the Database interfaces Component.

Each of the components have there own task which they are responsible for, but are required to communicate with each other to accomplish providing the recommendations to the user.

3 RUDDIS Implementation

A preliminary investigation of available UDDI implementations was required to select one for implementation. In order to determine which UDDI registry to utilise the following criteria was used in assessments. Based on the investigation we selected the use of the open source UDDI juddi supported by the use of UDDI4J (UDDI for Java) as the client. This selection was also based on recommendations being made for these two technologies being used together (UDDI.org, Hess 2004, Jung 2003).

The recommendation component in RUDDIS requires the ability to process the data from the Usage database using the Apriori algorithm. WEKA stands for the Waikato Environment for Knowledge Analysis. It provides practical machine learning tools and techniques with Java implementations (Witten 1999). WEKA contains an Apriori implementation which can be used to run the usage data through to find association rules.

The implementation was built using Java. The RUDDIS implementation was built as a web application using a combination of Java Server Pages (JSP) and Java Servlets running on the Jakarta Tomcat Server. For the describing of Web Services, they are categorised into two types, businesses and services. In RUDDIS the assumption is made that what we do for business can also be applied to services. For the implementation we have only the inquiry of businesses in the UDDI registry.

The Graphical User Interface (GUI) was implemented for RUDDIS takes the form of a web application to be used in a web browser that then accesses the juddi registry that resides on a server. The interface of RUDDIS was tailored to look like a standard UDDI interface with just some minor enhancements to assist with the recommendation section of the application and the facilitation of the selecting of various environment options. The interface is aimed to comply with the look and feel of existing public registries. Figure 2 provides a screen shot of the RUDDIS GUI.

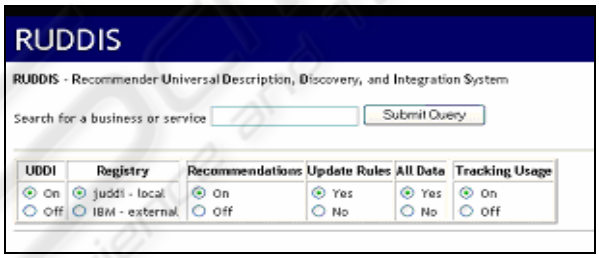

Fig. 2. RUDDIS GUI

4 RUDDIS At Work

The primary function of RUDDIS is to extend the UDDI to comprise of the ability to make recommendations of Web Services. Aside from this RUDDIS also allows the user to make various selections with regards to the recommendations being made. The section demonstrates the functioning of RUDDIS. This section illustrates the

feasibility of our approach and various options provided to the user to control the operation of RUDDIS such that the user has total control and the recommender system is as unobtrusive as possible. We now illustrate the options are as follows:

- A comparison of querying a local UDDI registry to querying an external registry for Web Services. The user is presented with the selection of Registry being either "juddi – internal" or "IBM – external". When juddi is selected the juddi UDDI registry on the local host is queried for Web Services. When IBM is selected the IBM test registry on an external server is queried for Web Services.
- A comparison of RUDDIS providing recommendations to RUDDIS working as a typical UDDI by not providing any recommendations. The user is presented with the selection of Recommendations being either "on" or "off". When "on" is selected, RUDDIS will attempt to provide any recommended Web Services found in the UDDI registry. When "off", RUDDIS will function as a typical UDDI providing only the Web Services found for the query item and not attempt to provide any recommendations.

For the purpose of assisting in the evaluation of RUDDIS, the juddi registry database was populated with an assortment of 315 web service names. The appropriate usage data was synthetically generated and used to populate the usage database to assist in the provision of recommendations. The usage database contains around 135 different sessions, each containing a number of Web Service requests. The RUDDIS usage database has been set up to assure that some rules supporting different scenarios will be generated. The evaluation data was used to generate the ARFF file required by WEKA. This lists the 24 rules discovered once the usage data is processed by WEKA. One of the rules generated supports the evaluation query of "skiing" ➔ "hire" meaning that a user looking for a Web Service on skiing would be highly likely to also want to look for Web Services on "hire". Other associations that could be useful that support this query are "lift passes", "snow reports" and "ski lessons". Rather than the user having to remember that these are items they could be interested in using, RUDDIS can offer them as recommendations.

The purpose of evaluating the difference between running with and without recommendations, is to compare RUDDIS in both scenarios. Not only do we want to see that associations analysis recommendations can be made with the methods that have been selected, but what the impact is on a typical UDDI in providing recommendations. When recommendations were turned on and "Skiing" was entered in as the Web Service query the following Web Service results produced included a recommended services list: *board hire, car hire, hire costs, ski hire, taxi hire, toboggan hire and hire snow gear.*

What is observed in the previous results is that RUDDIS searches through the rules and finds that for skiing, the strongest association rule contained the result of "hire". Under the Web Services Found heading, the Web Services retrieved from the registry that contain "skiing" are displayed. Under the Recommended Web Services heading any Web Services from the registry that contain "hire" are displayed. These were retrieved using find_business from UDDI specification. When recommendations are sected off the Web Service results are exactly the same as when recommendations are switched "on", except obviously no recommendations are provided. From examining these we can observe that the UDDI can be extended to provide Web Service recommendations. Also that it can be implemented in such a way that it can be requested ensure no recommendations are made.

We also evaluated to establish RUDDIS' ability to access an external UDDI registry. In this case the IBM test registry was used. Again the same query was entered. RUDDIS searched through the rules and found that for skiing, "hire" was the strongest rule result. It found no Web Services in the registry using the find_business library that contained "skiing". Under the Recommended Web Services heading any Web Services from the registry that contained hire are displayed. In this case there was one with the name of "Saphire". RUDDIS is dependant on what Web Services are registered in the external UDDI, so there were no Web Services existing in the IBM test registry that suited the query "skiing".

It can be determined from the above results that RUDDIS is successfully able to access an alternative external registry to the juddi UDDI on the local server and provide recommendations. The main difference is the Web Services retrieved as this is obviously a IBM test registry that contains a different set of registered Web Services.

5 Conclusions and Future Work

We have proposed and developed a recommendation extension to UDDI that we term RUDDIS. Recommendations can have potential benefits to both providers and consumers of Web Services. We have also adopted a novel approach to making recommendations that applies association analysis rather than traditional collaborative filtering approach. We have implemented and demonstrated the functioning of RUDDIS in an unobtrusive manner where the user has total control over the recommendation process. Further, we make no changes to the existing UDDI and the recommendation component acts as a plug-in that can used locally or at the server side.

We recognise that while we have highlighted the usefulness of this approach and demonstrated its practical feasibility – in order to fully validate such a model user trials that collect real data are essential. We recognise this as a limitation of our work so far. We plan to address these in at least a simulated context given that access to real usage data at this stage of web services research and development is not feasible. Furthermore, it is essential to determine the search space issues associated with a large list of recommendations. This notwithstanding, this paper takes the first step towards bringing the widely and successfully used concepts of recommendation in e-commerce to area of service oriented computing.

References

1. Guttman, R, H., Moukas, A, G., and Maes, P., (1998), Agent-mediated electronic commerce: A survey, *Knowledge Eng. Rev.*, vol. 13, no. 2, pp. 147--159, 1998.
2. Herlocker, J, L. Konstan, J, A., and Reidl, J., (2000), Explaining Collaborative Filtering Recommendations, Proceedings of the 2000 ACM conference on Computer supported cooperative work**,** Philadelphia, Pennsylvania, USA**,** pp: 241 - 250
3. Hess Andreas, (19th March 2004), "How to set up your own UDDI registry" Andreas Hess [online] Available: http://moguntia.ucd.ie/programming/uddi/ [Accessed 2 June 2004]

4. Jung, Christoph. (15th October 2003), "Discovering and Publishing Web Services with JBoss.net" JBoss: Professional Open Source [online] Available: http://www.jboss.org/developers/guides/jboss.net/uddi [Accessed: 2nd June 2004].
5. Lyell M, Rosen L, Casagni-Sinkins M , Norris D, (2003), "On Software Agents and Web Services: Usage and Design Concepts and Issues", The MITRE Corporation [online] Available: http://www.agentus.com/WSABE2003/program/lyell.pdf [Accessed: 24th June 2004].
6. Padovitz, A., Krishnaswamy, S., and Loke, S, W., (2003), Towards Efficient and Smart Selection of Web Service Providers Before Activation , Proceedings of the Workshop on Web Services and Agent-based Engineering (WSABE 2003), [online] Available: http://www.agentus.com/WSABE2003/program/shonali.pdf [Accessed: 24th June 2004].
7. Pennock David, Lawrence Steve, Popescul Alexandrin, and Ungar Lyle, "Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments",In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-2001),* Morgan Kaufmann, San Francisco, 2001, pp. 437-444.
8. Pokraev, S. Koolwaaij, J. Wibbels, M. (2003, Aug, 4), "Extending UDDI with context-aware features based on semantic service descriptions", Xerox Corporation [online], Available: https://doc.telin.nl/dscgi/ds.py/Get/File-30562/ UDDI__paper__camera_ready.pdf [Accessed 4 August 2003].
9. Rashid,A.ShaikhAli,O.F.Rana,David.W.Walker (2003)"UDDIe: An Extended Registry for Web Services" [online], Department of Computer Science Cardiff University, UK Available: http://www.wesc.ac.uk/projects/uddie/uddie/papers/saint03.pdf [Accessed: 24th June 2004].
10. Sarwar B, Karypis G, Konstan J, Riedl J, (10th May 2001), "Item-based Collaborative Filtering Recommendation Algorithms", GroupLens Research Group/Army HPC Research Center [online] Available: http://www10.org/cdrom/papers/519/ [Accessed: 16th July 2004].
11. Schafer J Ben, Joseph A, Konstan, John Riedl, (2001 January - April), "E-Commerce Recommendation Applications", Data Mining and Knowledge Discovery, 5 (1-2): 115-153
12. Muller Henning, Squire David, Pun Thierry, (8th November 2003), "Learning from User Behaviour in Image Retrieval: Application of the Market Basket Analysis", *International Journal of Computer Vision*, vol. 56, no.(1/2/3), pp. 65-77.
13. "Systinet" (2004) New Features in Systinet UDDI registry 5.0, Systinet [online] Availbale: http://www.systinet.com/download/whats_new_in_wasp_uddi_5.0.pdf [Accessed: 29th July 2004]
14. "UDDI.org: UDDI Products and Components" (2nd April 2004) UDDI.org [online] Available: http://www.uddi.org/solutions.html [Accessed 2 June 2004].
15. Windley Phillip, (10th July 2003), Managing the Web services flow, InfoWorld [online] Available: http://www.computerworld.com.au/index.php?id=1765450771&fp=16&fpid=0 [Accesssed: 29th July 2004]
16. Witten Ian H, Frank Eibe, (October 1999), "Data Mining" *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations,* Morgan Kaufmann, p105-111.