Personalized Selection of Web Services for Mobile Environments: The m-Scroutz Solution

Evangelos Sakkopoulos, Poulia Adamopoulou, A. K. Tsakalidis
Computer Engineering and Informatics Dept, University of Patras, Greece
{ sakkopul, adamopop, tsak }@cti.gr

Spyros Sioutas
Informatics Dept, Ionian University, Greece
sioutas@ionio.gr

Yannis Manolopoulos
Informatics Dept, Aristotle University of Thessaloniki, Greece
manolopo@csd.auth.gr

ABSTRACT

In this paper we discuss the integration of QoS-aware search and personalization algorithms in order to discover effectual Web Services for the case of mobile web users. We present a number of novel ranking algorithms specially designed for mobile web architectures. To validate and evaluate the proposed algorithms we developed a fully working prototype for mobile-PDA devices. The prototype enables PDA users to access electronic shops and to retrieve information about their products, while going shopping. Comparative experimental results have shown encouraging results that prove m-scroutz to be effective.

Categories and Subject Descriptors

H.3.5 Online Information Services, K.4.4 Electronic Commerce [distributed commercial transactions], H.3.4 Systems and Software

General Terms

Algorithms, Performance, Experimentation,

Keywords

Mobile web, mobile web services, web services selection

1. INTRODUCTION

In recent years, mobile search services have gained a lot of interest in the mobile communication markets. A major drawback for mobile applications is the fact that mobile devices cannot process complex business logic due to their limited capabilities. The ability to access and consume web services over mobile devices would allow users to expand the capabilities of their device by providing the necessary processing power. Adopting Web Services as an implementation technology for mobile Service Oriented Architecture is fully justified, to prevent creating new non interoperable technology islands [7].

Although Web services have been proved extremely efficient in the creation of business applications, they exhibit a limited ability to utilize QoS profiles in the discovery process. An extend survey on web service discovery is done in [5]. The integration of QoS features in the description of Web Services is advantageous to both customers and providers. QoS features can determine which Web Service best serves the user requirements and preferences and therefore give providers a significant competitive merit. Awareness of non-functional features such as performance, reliability, security, and cost would enhance the automated WS selection and composition process. In the case of mobile Web Services, the need for qualitative search is extremely crucial since mobile environments bear additional constrains like limited bandwidth, unpredictable response time and high probability of packet loss [3]. As the number of mobile devices consuming Web services increases, there arises a need for QoS aware selection of mobile Web services. An even more compelling potential that would upgrade the mobile search experience is dynamic delivery of content according to the user's past behavior and preferences. Provision of personalized services to the users of mobile Internet devices such as mobile phones and Personal Digital Assistants (PDAs) is a field that is gaining increasing momentum.

In this work, we accomplish an efficient integration between mobile phone applications and Web Services in which we combine QoS parameters and user preferences in order to provide a personalized online search service for PDA users. The key idea is to take the user preferences into consideration and to take advantage of the fact that the mobile devices are particularly personal items that can reveal one's likes and dislikes. In our case we study and experiment further on integration of Mobile Web and Web Service utilizing implicit user feedback and time obsolescence parameters.

The developed prototype system, that implements our novel approach, enables PDA users to access electronic shops and retrieve information on their products, at any time and any place. The electronic shops are accessed with the use of web services which communicate with the mobile users' application. During the selection process, the user is able to define the weights of the QoS parameters, in order to retrieve a relevancy ranking of the available services based on their QoS characteristics. Moreover, our solution exposes a novel functionality by integrating the user's selection profile into the selection algorithm in order to achieve personalized results.

The paper is organized as follows: Section 2 presents the related work on the area of mobile Web Services, whereas Section 3 describes the proposed personalized selection algorithms. Section 4 outlines our system architecture and Section 5 focuses on the implementation of m-scroutz prototype. Section 6 presents the experiments and evaluation results. Section 7 concludes the paper with discussion and future steps.

2. RELATED WORK

A generalized perspective of the challenging task to devise a scoring function that combines the QoWS criteria in a righteous manner is presented in [6]. The aim is to produce a single score for each Web Service or execution plan by representing each WS as an m-dimensional vector, where m is the number of parameters taken into account. In [4] the proposed approach adaptively selects the most efficient Web Service among possible different alternatives with real-time, optimized and countable factors-parameters. These techniques though do not incorporate time based criteria which in the case of mobile devices are proved to be decisive.

Some efforts have been made in the direction of the integration between mobile phone applications and Web Services. The challenges related to the introduction of Web Services in the wireless domain are discussed in [7]. The authors of [2] discuss the combination of Web Services technology and mobile terminals technology, though they mainly focus on the feasibility of using mobile terminals as Web Service providers. Authors of [3] propose a Mobile Web Service Availability Checking Model an approach to manage measure and deliver the required QoS metrics in a mobile environment, and formulate the overall availability status to mobile client. Our work is different as previous studies do not include user preferences and time dependent parameters for the case of mobile web and web services integration.

3. RANKING ALGORITHMS

One of the prevalent technical challenges regarding Web services discovery is the client's ability to customize the process and discover Web services that meet not only his functional but also qualitative requirements. A proposed solution to this problem [1] introduces the Web Services Relevancy Function (WsRF), used for measuring the relevancy ranking of a particular Web service based on QoS metrics.

Our solution is designed to enable the user to select the most suitable Web Service based on two criteria: a) the values of the QoS parameters, and b) his/her personal preferences. Following this concept, we present a number of novel ranking metrics/algorithms.

The first algorithm performs a relevancy ranking of search results based on the normalized sum of the QoS parameters values of the available Web Services. The qualitative attributes which characterize web services of the shops are (i) Accessibility, (ii) Availability (i.e. The probability a service is operational at a time instance), (iii) Capacity, (iv) Cost (i.e. The amount of money a client has to pay to the service provider in order to use a web service), (v) Interoperability, (vi) Response Time (i.e. The response time measures the expected delay in seconds between the moment when a service request is sent and the moment when the service is rendered.), (vii) Scalability (i.e. The ability to maintain an efficient and stable performance as the number of subscribers increases.) and (viii) Throughput (i.e. The average number of web service operations executed per second.). New criteria though can be added without significant changes in the underlying computation mechanism. The second algorithm (WSRF_UPRF) combines the QoS values, presented above, with the user choices that are logged and stored into the user's selection profile. The third algorithm (WSRF_HL) uses the concept of half-life in order to personalize and post - rank the search results. The half-life of a quantity whose value decreases with time is the interval required for the quantity to decay to half of its initial value. The last algorithm (WSRF_LOGUPRF) utilizes logarithmic decay upon the weight that previous choices of the user have in order to post rank the final results.

The user preferences are depicted introducing the function *exists(i)* into the proposed algorithms for a current Web Service under evaluation. The function *exists(i)* for *k* user choices receives values according to the following definition:

$$exists_{WebServiceX}(i)=\begin{cases} \text{0, when the user does not choose the current Web Service} \\ \text{1, when the user chooses the current Web Service} \end{cases}$$

As a result the ranking value of each participating Web Service in the results of a search query can be computed according to the proposed algorithms of table 1.

Table 1. Ranking matrics/algorithms

1. WSRF_UPRF
WSRF_UPRF=WSRF1* $\sum_{1}^{k} 2^i * exists(i) \Big/ 2^i - 1$
2. WSRF_HL
WSRF_HL=WSRF1* $\sum_{i=1}^{i=k}\left(0,5^{i/hl} * exists(i)\right) \Big/ \sum_{i=1}^{i=k}\left(0,5^{i/hl}\right)$
3. WSRF_LOGUPRF
WSRF_LOGUPRF=WSRF1* $\dfrac{\sum_{i=1}^{i=k+1}\left(\log_2(i+1) * exists(i)\right)}{\sum_{i=1}^{i=k+1}\left(\log_2(i+1)\right)}$

4. SOA: OVERVIEW

The overview of the architectural components is discussed in this section. In order to validate and apply the proposed new algorithms presented in the previous section Service Oriented Architecture (SOA) has been employed. Figure 1 depicts the architectural design and shows how web services are connected with the mobile/PDA and the backend supporting components. First of all we present the set of Web Services that have been designed and implemented:

Web Service DynamicWS. Dynamic invocation during runtime is defined as the process of choosing, binding and invoking a Web Service without having to know the exact description and endpoint of the web service, or to create a client-side proxy class during design/compile time. Mobile/Pda applications and Web Service software libraries do not support fully dynamic invocation during execution on the mobile client application. This web service is devised in order to implement fully dynamic invocation of web services from the mobile/PDA clients during runtime. In order to receive the results of the invoked web service, we provide as parameters the WSDL, the service name, and the method name of the web service which we want to invoke, and to use the classes of the DynWsLib library.

Web Service ShopsCatalogue. This web service consists provides a method that gets as input the name of a shop, executes a query in the application database and returns the shop's WS.

Web Service ShopProducts. This web services is used in order to retrieve the desired products from the online shop. It gets as input the product category and subcategory and returns a dataset with the corresponding products of the shop. Moreover, we use a backend database and a mobile database which is stored on the memory of the mobile device.

m-scroutz backend database. The main database in which are stored information on the users, the products categories an subcategories, the available shops, the URLs of the shops' web pages and the QoS parameters. Moreover, the products which were returned by the shops according to the last users' requests are stored here.

m-scroutz database. This database is developed with SQL Server Mobile and is deployed on the Windows Mobile 6. Thus, it is locally stored on the memory card of the mobile device to cache user choices locally. Exchange of data with the main application updates the local data, when network connectivity is available and/or at user's request using implicit replication.

5. m-SCROUTZ IMPLEMENTATION

5.1 Dynamic Web Service Invocation

One of the premises of web services is that the user can dynamically discover new services and invoke them. As the set of available Web services may not be known a priori may change frequently or service requester requirements and preferences may change, the problem of dynamic Web service selection is fundamental.

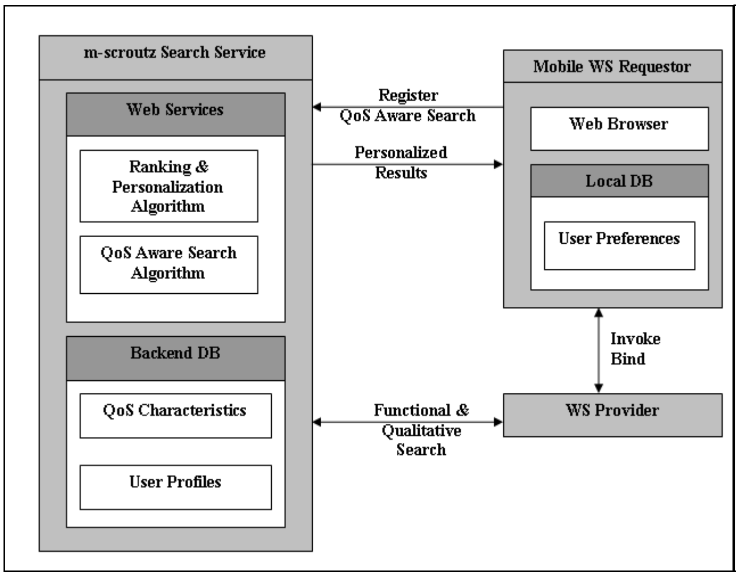
An effective solution to this problem is proxy pattern which wraps code around software elements. This code deals with update without affecting the proxied software element or its clients. There are several utilities and classes which enable developers to dynamically create web service proxies. For example, a frequently used utility is the .Net WebServiceStudio which is based on the WSDL description of the service. The user provides a WSDL endpoint and WebServiceStudio fetches the WSDL, generates a .NET proxy from the WSDL and displays the list of methods available. The user can choose any method, provide the required input parameters and invoke the web service. Next the SOAP request is sent to the server and the response is parsed to display the return value. DynamicProxy is another tool that allows developers to create dynamic WCF clients at runtime by specifying the WSDL URI of the service without depending on the precompiled proxy or configuration. It uses the MetadataResolver to download the metadata from the service and WsdlImporter to create the contract and binding at runtime. The web service endpoint comprises the required information about the address, binding, contract and behavior in order to find and interact with the service of that endpoint. Moreover, with the use of the WsdlImporter creator, a WsdlImporter object is created for a set of metadata and information about the contract, the endpoint and the binding is extracted at execution time. The compiled dynamic proxy can be used to invoke the operations on the service using reflection. Apart from proxies, another technology, the one of dynamic interfaces is proved to be suitable. Instead of programming software that invokes operations, software just passes arguments and lets the platform decide which operation is best suitable for the given arguments.

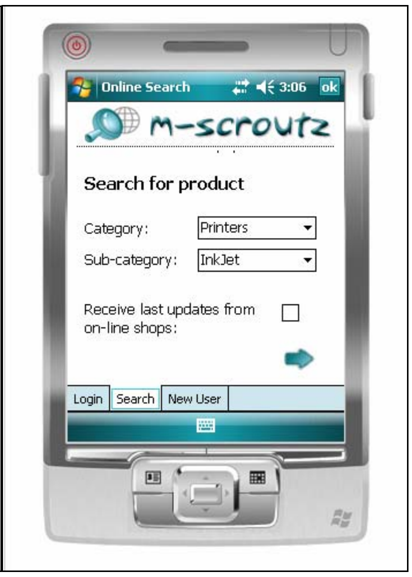
In m-scroutz, dynamic invocation of methods provides the advantage of easy and full scale extendibility, not only in the case of addition of one or more new shops' Web services, but also in the case of modification of an existing Web service. In both cases, we are not obligated to add the new web service, recompile and reload the modified application in all window mobiles that use it.

In order to programmatically invoke web services, assuming that we have already obtained the URL to the WSDL document of interest, we used the .NET DynWsLib. This library is used for the dynamic invocation of web services without a client-side proxy class at design time. Moreover, it is not necessary to know the exact description and endpoint of the web service during design and compilation time. The developer provides the URI of the WSDL and defines the desired binding and method.

5.2 Presentation of the prototype

m-scroutz search adaptation solution was implemented in .NET framework 2.0 using framework 2.0 using C# and Web Services by Visual Studio 2005 Technologies. Moreover, we used Device Emulator, a virtual hardware platform that mimics the behavior of a Windows Mobile–based hardware platform. In order to create user selection profiles we provide a user registration and authentication functionality. Registered users first have to successfully login to the application in order to search for shops and products. The PDA user initially selects the requested product category and subcategory (Figure 2a). Following, he sets the preferred weights of the QoS parameters, which range within the normalized values of 0 up to 1, based on his/her requirements. Finally, the user selects the ranking algorithm of his preference. The system then invokes a web service which implements the ranking algorithm and displays the results of the available shops' web services listed in suitability order.





Search for product

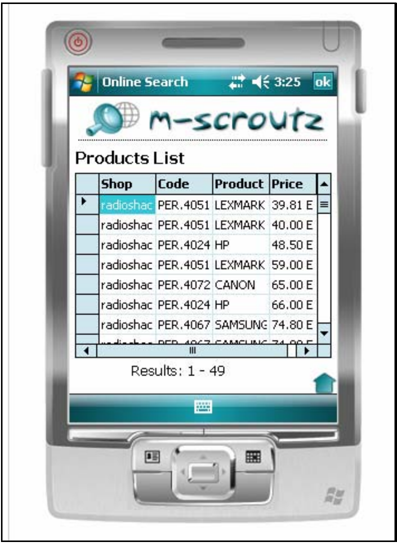
Category: Printers

Sub-category: InkJet

Receive last updates from on-line shops: ☐

Figure 1. System architecture: overview

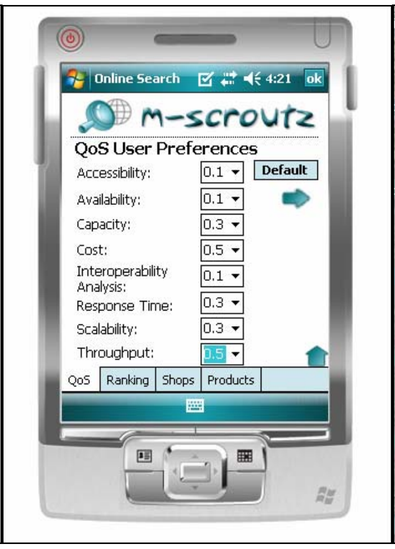
Figure 2. a) User selection criteria, b) Local product list stored in the cache of the mobile/PDA



Products List

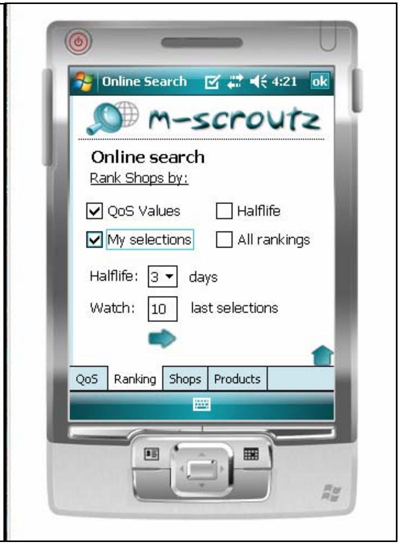
Shop	Code	Product	Price
radioshac	PER.4051	LEXMARK	39.81 E
radioshac	PER.4051	LEXMARK	40.00 E
radioshac	PER.4024	HP	48.50 E
radioshac	PER.4051	LEXMARK	59.00 E
radioshac	PER.4072	CANON	65.00 E
radioshac	PER.4024	HP	66.00 E
radioshac	PER.4067	SAMSUNG	74.80 E

Results: 1 - 49



QoS User Preferences

Accessibility:	0.1
Availability:	0.1
Capacity:	0.3
Cost:	0.5
Interoperability Analysis:	0.1
Response Time:	0.3
Scalability:	0.3
Throughput:	0.5



Online search

Rank Shops by:

☑ QoS Values ☐ Halflife

☑ My selections ☐ All rankings

Halflife: 3 days

Watch: 10 last selections

Figure 3. a) QoS user preferences, b) Ranking parameters

When the user selects a shop from the ordered list, the system calls a web service which dynamically invokes the shop's web service at runtime. This functionality aims to increase the extendibility of the application in new operational environments without requiring continuous edition updates. Each time the user selects a shop, the corresponding web service is invoked and the corresponding products are returned. These products are stored in the local database of the mobile device (Figure 2b), while previous logged user choices/preferences concerning the specific shop, product category and subcategory are updated using the recent choices. Moreover, we have developed a synchronization procedure of the local mobile database each time the user starts the application.

Each time the user has the ability to view either the last online updates of products of the selected shop, or the shop's products stored in the local database, as shown in Figure 2b. In the second case, the functionality can be proved particularly useful when a user applies for the same product from a specific shop in small time periods, for example during the same day. The possibility that the shop's product catalogue has been renewed is relatively small and therefore it is sufficient to view the products of the previous access which are stored in the local database. Finally, the user is able to compare the products and prices of the available shops in order to decide and obtain the most advantageous.

In the prototype the user may choose the appropriate QoS in order to prioritize his preferences, e.g. give high priority to the accessibility of the store and the response time of its Web Service because he/she might be in a hurry to buy something regardless of its price (Figure 3a). Furthermore the system is fully customizable to take into account both the QoS values and previous choices and preferences of the user to further personalize the results (Fig. 3b).

6. EXPERIMENTAL EVALUATION

In order to evaluate the proposed ranking algorithms in section 3, we developed ten (10) web services for real life online shops that match the mobile client's functional requirements. We had a 100 users searching for web services and consuming services providing thus feedback to the ranking algorithms providing synthetically the user's choices following different distributions.

We compare the proposed algorithms with respect to the average number of steps (Precision) required to make a search result more precise (higher in the ranking list) from the last ranking position to the first one. In this scenario, we consider a search result that has the lowest relevancy to the user QoS requirements, to be preferred by the user in all of his search queries.

We assign to QoS parameters random values which are generated according to a) the mean deviation of Normal, b) Random, and c) Zipf Distribution. QoS parameters values range (table 2). All QoS parameters are set to have equal weights.

Table 2. QoS parameters value range

QoS Parameter	Values Range
Accessibility	0-100
Availability	20- 100
Capacity	20- 100
Cost	1-10
Interoperability	20- 100
Response Time	100-2000
Scalability	20-100
Throughput	1-20

We used randomly generated QoS values to calculate the WS Relevancy Function. In the next step we consider that the user has selected to use the least relevant web service and we calculate the ranking order for the algorithms according to this preference. We repeat this step for 100 times and each time we update the user profile and recalculate the ordering using the mean values. We are interested in the converge rate of the ranking algorithms results to the user preferences.

In the following figures we present the average number of steps required by each ranking algorithm to succeed in upgrading the least relevant web service to the first ranking position according to user selection profile. In Figure 2 and Figure 3 the values of the web services QoS parameters are generated with the use of a Random Distribution.

In the first **case 1**, the calculation of the average number of steps does not count the unsuccessful test runs of the algorithms while in the second **case 2** when an algorithm fails in the first 100 steps to order the last web service in the first search result position we set the required number of steps equal to 100. Algorithm WSRF_HL is tested for a set of 6 different half life periods (5, 10, 15, 20, 25, and 30).

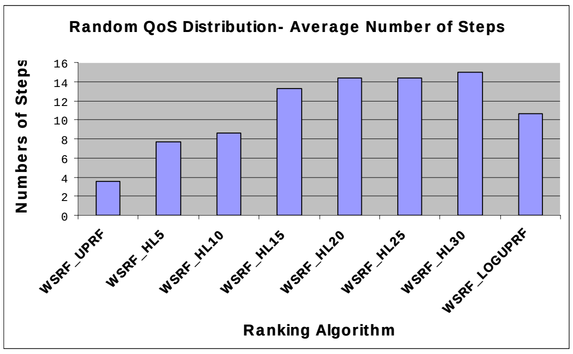


Figure 2. Average number of steps- Random QoS values distribution case 1

The WSFR_UPRF algorithm in both cases requires a considerably small number of steps. Algorithm WSRF_LOGUPRF also appears to be quite sufficient, while algorithm WSRF_HL is the least affected by user preferences. The WSRF_HL performance is improved as HL period is increasing.

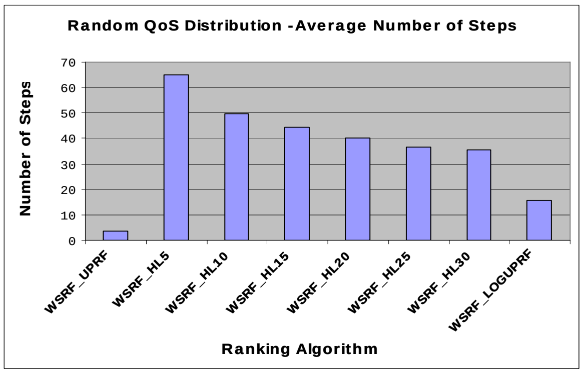


Figure 3. Average number of steps- Random QoS values distribution case 2

In Figure 4 and Figure 5 the values of the web services QoS parameters are generated with the use of a Normal Distribution. In comparison with the results of the randomly distributed QoS values, the performance of all algorithms in the random distribution is noticeably improved.

In Figure 6 and Figure 7 the values of the web services QoS parameters are generated with the use of a Zipf Distribution. We observe that for this distribution the number of necessary steps for the WSRF_HL5 algorithm is extremely high.. WSRF_UPRF has the best performance as shown in the experiments using all different distributions.

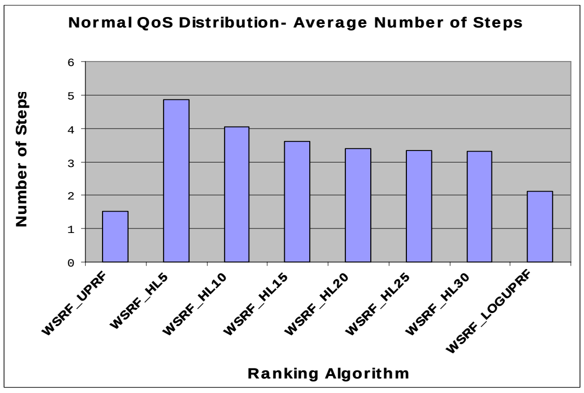


Figure 4. Average number of steps- Normal QoS values distribution case 1

In the next figures (figs. 12, 13, 14) we present the progress of the ranking position in the case that a user selects the same web service for 30 consecutive times/steps (in an experimental execution). Initially, the user preferred web service is supposed to be ordered in the last position (10th) based on an initial bad WSRF. At each step the specific web service is marked in the user profile as selected and reordered according to the three ranking algorithms. This evaluation experiments intends to show the tendency that each ranking algorithm has to integrate and depict on the ranking list recent consecutive same user choices/ preferences. We see that WSRF_HL5 (the highest curve) is the most "resistant" to previous consecutive user choices, and this means that the algorithm takes into consideration user choices in a slow rate.

On the other end WSRF_UPRF (the lowest curve) integrates quickly user preferences and adapts almost instantly to his/her behavior. In this set of experiments WSRF_LOGUPRF provides an average approach in which user preferences are taken into account in a progressive manner.

7. CONCLUSIONS & FUTURE STEPS

In this work we propose novel solutions towards the integration of QoS-aware search and personalization algorithms in order to discover effectual Web Services for the case of mobile web users. We present a service oriented architectural approach specially designed for mobile web architectures for fully online dynamic invocation during execution of the client runtime. A number of novel ranking algorithms are proposed that take into consideration the mobile web user's preferences and quality of service characteristics of web services available.

The ranking algorithms keep track and utilize logged user choices of Web Services during previous search attempts, which are kept at the mobile device serving as a preference profile. The use of half-life concept (WSRF_HL) that decays the value of previous choices appears to be useful in cases where the ranking is designed for minor changes by the user preferences. The logarithmic based decay (WSRF_LOGUPRF) is an average solution, while WSRF_UPRF depicts most quickly the user changes in preferences. A fully working prototype for mobile-PDA devices has been developed. It implements full functionality for the case of a shopping assistant that consumes a number of e-shop catalogues. The design and architecture of the solution is based on SOA concepts and it is developed with Web Services. Experimental evaluation has shown that the algorithms and the system are efficient for real life working mobile Web environments.

Future steps include the evolution of the algorithms to take into account additional implicit user feedback of the final products chosen and not only the e-shops and services. This is particularly efficient case for e-businesses implementations based on lightweight RESTful mobile Web Services.

8. ACKNOWLEDGEMENTS

This work is partially supported by the research project "Infrastructure and Development of Broadband Application Services Provision in the Region of Western Greece (A3)" of the Regional Innovation Pole of Western Greece.

9. REFERENCES

[1] E. Al-Masri and Q. H. Mahmoud, "Discovering the Best Web Service", WWW 2007, Canada, May 2007, pp. 1257 – 1258.

[2] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile Web Service Provisioning", Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, Guadeloupe, 2006, p. 120

[3] Kee-Leong Tan, S.M.F.D. Syed Mustapha, "Measuring Availability of Mobile Web Services", International Conference on Semantic Web & Web Services, Nevada, USA, 2006, pp. 137-142.

[4] Ch. Makris, Y. Panagis, E. Sakkopoulos, and A. Tsakalidis, "Efficient and adaptive discovery techniques of Web Services handling large data sets", *Journal of Systems and Software*, Elsevier, Volume 79, Issue 4, April 2006, pp. 480-495.

[5] J. D. Garofalakis, Y. Panagis, E. Sakkopoulos, A. Tsakalidis, "Contemporary Web Service Discovery Mechanisms", *Journal of Web Engineering*, Rinton Press, Vol.5, No3, September 2006, pp. 265-290.

[6] Liu et al., 2004 Liu, Y., Ngu, A.H.H., Zeng, L., 2004. "Qos computation and policing in dynamic Web Service selection". In: Feldman, S.I., Uretsky, M., Najork, M., Wills, C.E. (Eds.), Proceedings of the 13th international conference on World Wide Web-Alternate Track Papers & Posters, WWW 2004, New York, NY, USA, May 2004, pp. 66–73.

[7] R. Tergujeff, J.Haajanen, J. Leppänen, and S. Toivonen, "Mobile SOA: Service Orientation on Lightweight Mobile Devices", IEEE International Conference on Web Services, USA, 2007, pp. 1224 – 1225.

[8] T. Pilioura, S. Hadjiefthymiades, A. Tsalgatidou, M. Spanoudakis, "Using Web Services for supporting the users of wireless devices", *Decision Support Systems*, Volume 43, 2007, pp. 77– 94

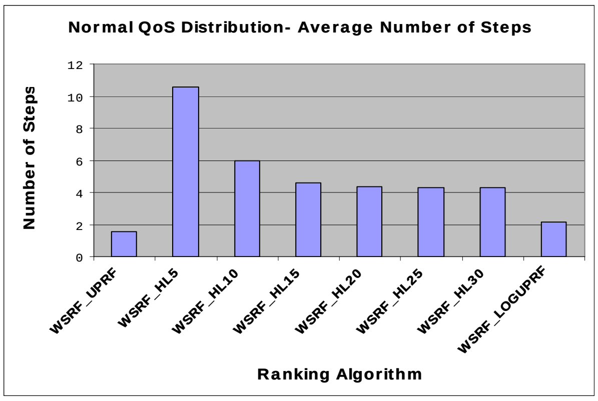


Figure 5. Avg number of steps- Normal QoS values case 2

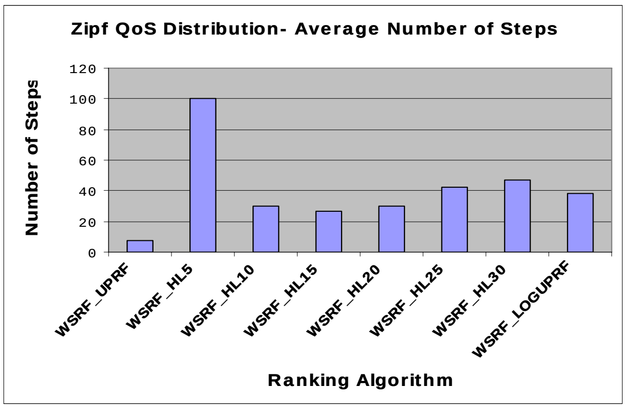


Figure 6. Avg number of steps- Zipf QoS values case 1

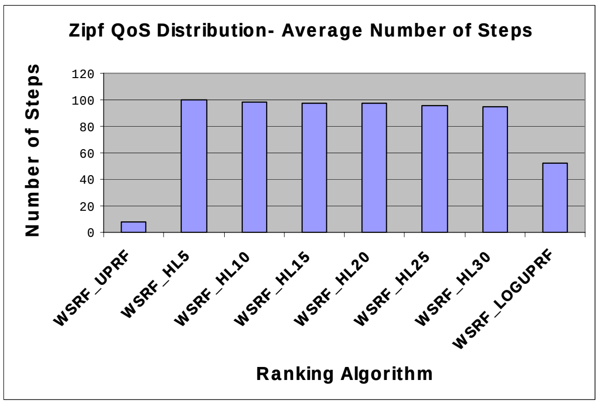


Figure 7. Avg number of steps- Zipf QoS values case 2

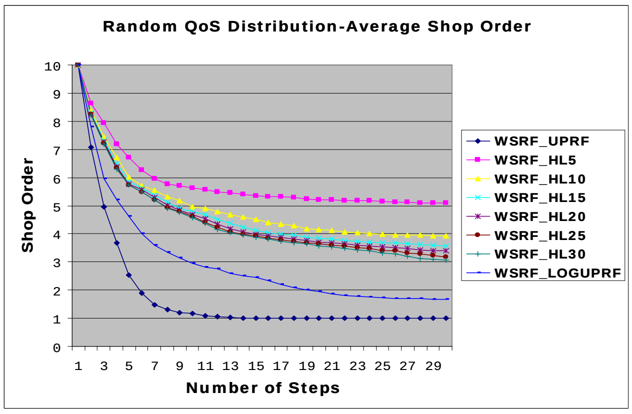


Figure 8. Avg shop order- Random QoS

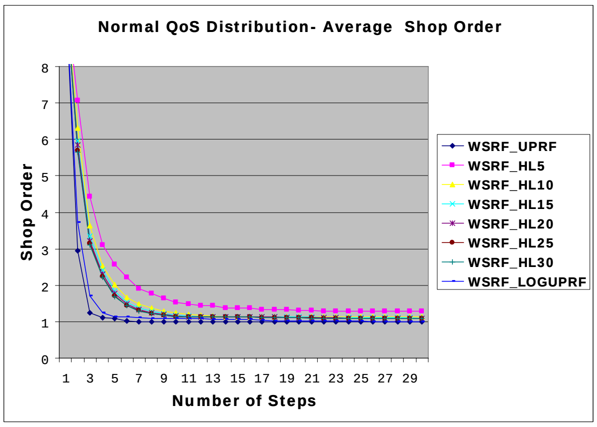


Figure 9. Avg shop order- Normal QoS

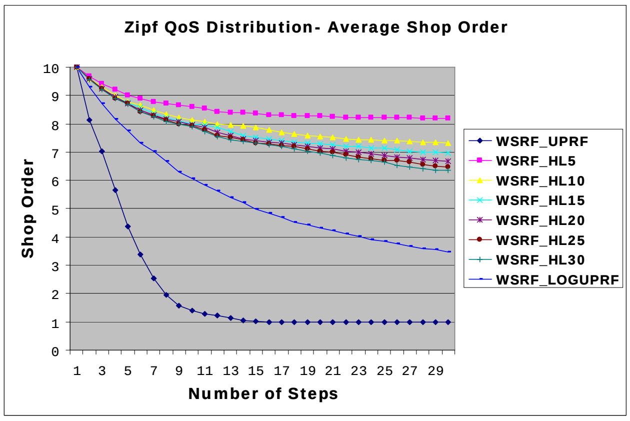


Figure 10. Avg shop order- Zipf QoS