European Journal of Applied Sciences 9 (4): 178-183, 2017
ISSN 2079-2077

DOI: 10.5829/idosi.ejas.2017.178.183

Generating Dynamic GUI to Discover and Invoke Web Services from Android Mobile Devices

[1]A. Veeraiah and [2]P. Marichamy

[1]Assistant Professor, Department of Information Technology,
P.S.R Engineering College, Sivakasi-626 120, Tamil Nadu, India
[2]Dean (Academic), P.S.R Engineering College, Sivakasi-626 120, Tamil Nadu, India

Abstract: Modern trends in mobile computing and the popularity of mobile devices has motivated to access web services from mobile devices in order to extend their functionality and gain access to remote data. However, the particularities and limitations of mobile devices and the mobile environment pose great challenges for consuming web services. There are several issues and challenges that emerge from the fact that mobile devices have lower processing power, limited bandwidth, less memory and finite battery power when compared to desktops. Nowadays the researchers concentrate their research that target on mobile devices should aim to minimize their interactions with the network and reduce their resource consuming processing whenever possible. In this paper, there is a novel architecture is introduced for dynamically invoking web service methods from mobile devices with minimal user intervention that only involves entering a search phrase and values for the method parameters. The architecture overcomes technical challenges that involve consuming discovered services dynamically by introducing an agent for analyzing mood of the user's search phrase, a classifier for classifying the discovered web services according to their description and a man-in-the middle (MIM) server that provides a web service whose responsibility is to discover needed services and build the client-side proxies at runtime. The architecture moves to the MIM server energy-consuming tasks that would otherwise run on the mobile device. Such tasks involve communication with servers over the Internet, XML-parsing of files and on-the-fly compilation of source code. This approach evaluates the system performance to measure scalability as it relates to the capacity of the MIM server in handling mobile client requests.

Key words: Web service discovery · Web service classification · Dynamic invocation · Mobile devices · Mobile computing

INTRODUCTION

Service computing has become a cross- discipline that covers the science and technology of bridging the gap between business services and IT services. The underneath breaking technology suite includes Web services and service-oriented architecture (SOA),cloud computing, business consulting methodology and utilities, business process modeling, transformation and integration. This scope of Services Computing covers the whole life-cycle of services innovation research that includes business componentization, services modeling, services creation, services realization, services annotation, services deployment, services discovery, services composition, services delivery, service-to-service collaboration, services monitoring, services optimization, as well as services management. The goal of Services Computing is to enable IT services and computing technology to perform business services more efficiently and effectively.

Mobile computing is human–computer interaction by which a computer is expected to be transported during normal usage. Mobile computing involves mobile communication, mobile hardware and mobile software. Communication issues include ad hoc and infrastructure networks as well as communication properties, protocols, data formats and concrete technologies. Hardware includes mobile devices or device components. Mobile software deals with the characteristics and requirements of mobile applications.

Corresponding Author: A. Veeraiah, Assistant Professor, Department of Information Technology,
P.S.R Engineering College, Sivakasi-626 120, Tamil Nadu, India.

For a mobile device to access the web service there are two types of solutions either the mobile device will directly interact with the service when the WSDL file is parsed by the mobile devices and no commercial compilers have been developed to date for such devices? This lead to the introduction of the server between the mobile device and the internet based web service. The server will do the parsing and build proxy for the service requested by the user. This process of building proxy by the local server and send it to the mobile device looks like a local process and it is available for the mobile whenever it is needed. Battery power is the limited constraint of the mobile devices. It becomes critical whenever the mobile devices are used for communicating with the internet, remote systems for searching the user request in the systems and discovering user requested services and invocation of their methods. In summary the architecture allows mobile users to acquire needed data through dynamic invocation of web service methods by merely providing search phrases and method argument values into a dynamically generated GUI. The design is based on the framework defined in [1] the architecture realizes service invocation dynamically and the practical implementation of the interface as perceived by the user and returning the desired result.

There are two types of general techniques for accessing web services using SOAP and the REST approach. SOAP message could be sent to a website that has web services enabled with the parameters need for a search. The site would then return an XML formatted document with the resulting data. With the data being returned in a standardized machine parse able format, it can then be integrated directly in to a third party website or application. The process of web service invocation using SOAP messages involves that the client side proxy application encloses the user's request a form of method call by using the SOAP messages. This message is sent to the web server in which the requested service is present.This service will extract the call and executes the call. This execution of calls will produce the results and they are enclosed in the SOAP messages and sent to the requested client side proxy. This client side proxy will extract the message and send it to the requested client. Here the proxy is generated by generating the source file from the web service description language files which is present in the UDDI registries. This WSDL file consists of information that describes the web service, how to access it, the operations it performs, the types of parameters to be passed to each of the supported methods and the types of returned results. After the source file is generated it is compiled into a proxy class that is finally registered with the client application.

Related Work: Several approaches have been developed for enabling dynamic invocation of web services in mobile devices and analysis of performance of the MIM server. These are either conceptual solutions or the proposed architectures. The paper is based on the second type but differs from the proposed schemes in that it is complete, fully dynamic, employs generic technologies and is not restricted to a particular platform. However the survey that follows depicts how the proposed solutions in the literature either only allow non dynamic access to deployed web services from mobile devices, or provide dynamic access to services, but not from mobile devices. Starting with the first class of solutions mentioned above, an application for Peer-to-Peer (P2P) web services is suggested in [2] for use in ad hoc networks. In this article, a heterogeneous environment is being composed of mobile nodes with computing and communication capabilities. Each peer node equally acts as both, server and client. These nodes provide their services to other nodes in the distributed environment and they are able to use remote services. Here the server based implementation is focused which enables the mobile devices to provide and publish their services. In this system P2P web services are implemented in java enabled mobile devices that were used to analyze the memory usage and response time of the SOAP server. In [1] a system was presented to study the resource consumption and performance of providing web services on mobile phones. The analysis of a prototype which was developed using personal java on a Sony Ericson P800 phone shows that the implementation is able to handle up to eight concurrent users with reasonable time delays. In [3] two architectures were presented for high and low e mobile devices using Java 2 Platform Micro Edition(J2ME),Web service API(WSA) and short messaging service(SMS),which define a specific architecture for each case. WSA integrates basic support for web services invocation and XML parsing in to the runtime environment of the mobile device. A network application that resides on a WSA enabled device must include a stub, which is generated by a tool on a development workstation and then deployed to the device. The application can then employ the stub and the runtime to parse XML SOAP messages using JAXP and consumes the service using JAXRPC.

To access web services using SMS, a gateway that translates short messages to HTTP requests was used to connect the SMS center to an application server, which in turn translates HTTP request or response to or from SOAP messages. Every time a web service needs to be accessed programmed and configuration is required in order to create a stub or to set up communication using SMS. In another type of approaches servers are used in the middle between the mobile devices and web servers. In [4] it was argued that the normal web service architecture that employs requestor, a broker and service provider alone not allow for web service invocation dynamically that is suitable for mobile devices. It is suggested that a service gateway must be included between the requestor and the rest of the architecture. In such a solution it was recognized that additional code must be included on both the mobile and service side. In this paper results were analyzed from simulation shows that service access using gateway in the architecture is faster compared to that of service access using KSOAP protocol. In [5] Halteran and Pawar discusses the requirements for nomadic mobile service provisioning and proposes the mobile service platform (MSP) as a supporting infrastructure and middleware which extends the service oriented architecture paradigm to the mobile device. The MSP design is based on the Jini surrogate architecture specification [6] which enables devices that can not directly participate in a Jini network to join a Jini network with the aid of a third party. MSP consists of an HTTP Interconnect protocol to meet the specifications of Jini surrogate architecture and provides a custom set of APIs to develop and deploy a nomadic mobile service. Invocation of web methods dynamically involves development of few platforms technologies. The DynWsLib library [7] stands for a.NET library which enables to dynamically invoke XML web services at runtime. This generates the client side proxy and it does not need WSDL file during the proxy design time. However, according to one developer's forum, this library worked consistently with services running on the local server, but not always trying to reference remote services. Moreover this library which is no longer supported does not provide an effective mechanism to handle specific types during the proxy generation process. In [8] the work proposed was to design of automatic generation of the multimodal abstract user interface. This paper presents a novel architecture for discovery and invocation of mobile Web services through automatically generated abstract multimodal user interface for these services. A prototype has been developed to auto-generate user interface based on XForms and VoiceXml from a WDSL file. In this proposed architecture, the discovered Web services are invoked dynamically with a transparent mechanism. Moreover, the proposed architecture is a component-based architecture that provides its core functionality as Web services.

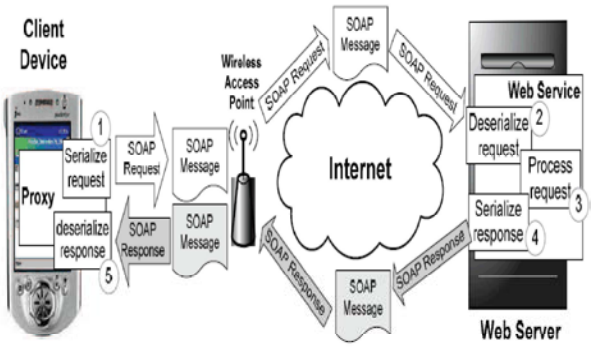

Fig. 2.1: Architecture for accessing SOAP based web service from Mobile

Finally, in the work [4] where Man In the Middle server is used in between the requestor and the service provider which does the maximum work of generation of proxy class by the process of text matching with the cached web service descriptions that are sent periodically from the UDDI registries by periodical requests from the MIM server. If the requested service is not present in the cached files then the server will send request for URLs to the web server and the web server will send the WSDL files. With these files the matching file is selected and the source code is generated. Then the source code is compiled by using libraries and build the client side proxy and then it is shipped to the mobile device. This client side proxy is responsible for generating the GUI dynamically which provides the device results requested by user in the mobile device.

Proposed Work: The system architecture for the proposed system based on SOAP is shown in the Fig.3.The proposed system contains the two main components. The first component is a Mobile application that making a request to consume the web service. This application is installed in the mobile handset and also it act as a client. The other component is Man in Middle (MIM) server. The responsibilities of MIM are discovering the web services based on the mobile device request and creating proxy classes for that discovered

services. After getting the proxy, a mobile device can invoke a particular method of the web service and get the desired results. The MIM server has five components to accomplish the completed task. They are 1. Text Matcher, 2. Web Service Classier and Indexer, 3. Periodic Web Service Downloader, 4. Web Service Downloader and Parser and 5. Proxy Builder.

Text Matcher: This is the key component that is used for dynamic discovery mechanism. The services are maintained in XML based structure. Once the discovery agent found out appropriate the service for the request, it extracts the service description (operations provided, input and output parameters…) from the service registry. When the Web service is requested by the user the Web service is searched in the local database maintained by the periodic WS downloader process. If the requested Web Service is found in the database the URL is taken from the database. Otherwise the web service is searched in the UDDI and added to the database. Fast String Searching algorithm is used for searching the web service in the UDDI. This agent also uses the Wordnet dictionary tool to determine the mood of the input phrase which is given by the user.

Classifier and Indexer: After discovering the amicable service by the discovery agent, it sends the description of the service to the indexing and classification agent. The indexing and classification agent will classify the services and store it into the local database according to their key word and domain by MIM server. Classifying mechanism is used to enhance the searching capability. Classifier is used to classify the services stored in local cache. Local cache concept is introduced so that users can learn from past experiences. The local database is used to store the web services. It consists of categorized repositories. Web services that are discovered by matching engine are returned to the requester. It also stores the service reference in local cache using indexing mechanism for future reference. Web services are classified or categorized into different categories in local database. The classification is done based on different domain types for example weather, bank, travel, stock exchange etc. For each category different repositories is maintained. This mechanism introduces fast and efficient retrieval of web services. The agent is based on multithreading and opens thread for each registry listed with itself. Multiple agents have been used in our proposed framework and each agent will search desired service from its listed registries and returned the results to matching engine. This agent involves discovery of the requested web services from the service registry using the information provided by the service consumer. For example, if the search string is that zip code for India means that the agent will make an index with the key word “zip code” in the database. The index that points to the countries in the world that having the zip code. When the next time if the request comes like that zip code for China, the discovery agent gets result from database with help of indexing, because all the services which related to zip code are classified under that index. This agent makes the searching process where the services are already stored in the local database. Whenever a new comes, this agent will check the key word and stored it under the index if already available in the database else it creates a new index and store it under that.

Periodic WS Downloader: The Periodic WS Downloader program downloads the newly added WS Descriptions and the corresponding WSDL file URL from the service registry UDDI(Universal Description Discovery and Integration) Registry. UDDI is the public registry for web services it contains the information such as Web Service description, WSDL file’s URL, Company’s contact details,etc., The downloaded WS description and WSDL file’s URL are stored in a local database.

Web Service Downloader: The WS Downloader program downloads the newly added WS Descriptions and the corresponding WSDL file URL from the UDDI (Universal Description Discovery and Integration) Registry. UDDI is the public registry for web services it contains the information such as Web Service description, WSDL file’s URL, Company’s contact details,etc., The downloaded WS description and WSDL file’s URL are stored in a local database.

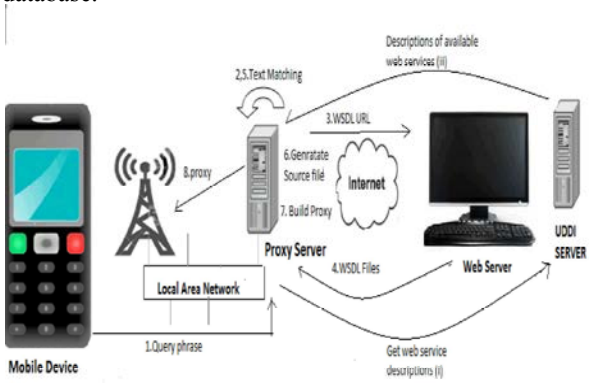

Fig. 3.1: System Architecture

Proxy Builder: Client side proxy classes are generated by using the WS-Import tool then these classes can be included in the source code by using simple import statements. The proxy class contain the details found by the WSDL parser module are then send to the mobile application for GUI generation and directly invoke the web service.

Dynamic GUI Generation: The Dynamic GUI Generation process generates the GUI dynamically by using the information retrieved from the WSDL parser such as List of operations, Input and Output parameters and their types. The Parameter name is displayed as a Textview in the GUI and the type of the Input parameter is displayed as the hint in the EditText. Then the Web Service is invoked dynamically and the result is displayed in the other GUI which is generated exactly as the Input GUI. The obtained results values are set as the Text in the Textbox value.

Screen Shots:

Fig. 5.1: Application screenshots: Search String (first), input parameter (second), results (last).

Implementation: The web service client on the mobile device was implemented using the J2ME android SDK and the java programming language. The application was installed on an Celkon A89 running the Android Mobile operating system and set up to communicate with a wireless access point using Wireless LAN. The MIM server was implemented on a Dell Insprion laptop with a dual core Processor and 1 GB of memory. The UDDI for this work is Xmethods.com. This UDDI has provided many public accessible web services. Boyer – Moore algorithm is used for matching the input phrase with web service descriptions. Classification is implemented with simple java code.

The first screen helps the users to provide their search phrase. This is the entry point of the mobile client application. The second screen displays opted web service for the user's search string with the input parameters. The last screen outputs the result for the discovered web service.

Analysis:

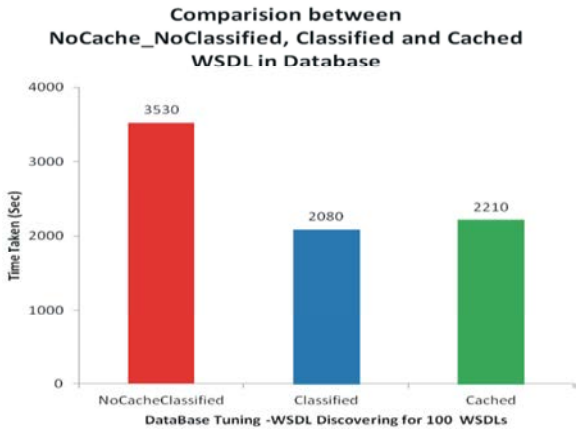

Fig. 6.1: Time Consuming Comparison Chart

The above chart shows the experimental results of the application for generating proxy classes for 100 different discovered web services. In this work, three different methods are used to generate the proxy classes for the discovered web services. The first method is the discovered web services are not cached and classified. It consumes more time to generate the proxy. The Cached web services method takes a moderate time to generate that. But in our proposed work shows that less time consumption for generating the proxy class and dynamic user interface. From the result we conclude that the classification method shows the better performance compare than two others.

CONCLUSION

The presented architecture makes it possible for mobile device users to dynamically invoke web service methods that meet their needs. The implemented solution shows better performance while discovering the web services. It overcomes the problems in the existing systems. It works consistently for remote web services also. The time consumption of searching the web service in UDDI is reduced by using the Fast String Searching algorithm and also Classification algorithm. Thus our proposed solution working better when compared to the all other related works for invoking the remote web services. This work will be well support for discovering individual web services but limited to the composite web services. As a Future work, the MIM Server can be implemented with the functionality with the intelligence to identify a set of services whose collective functionality can serve the user's request.

REFERENCES

1. Boyer, R. and J. Moore, 1977. A Fast String Searching Algorithm, Comm. ACM, 20: 762-772
2. Chakravarti, A., G. Baumgartner and M. Lauria, 2005. The Organic Grid: Self-Organizing Computation on a Peer-to-Peer Network,"IEEE Trans. Systems, Man and Cybernetics, 35(3): 373-384, May 2005.
3. Dattatreya, G., 2008. Performance Analysis of Queuing and Computer Networks. CRC Press, 2008.
4. CodePlex, ProxyFactory Home Page, www.codeplex.com/ ProxyFactory, 2011.
5. Chatti, M., S. Srirama, D. Kensche and Y. Cao, 2006. Mobile Web Services for Collaborative Learning, Proc. IEEE Int'l Workshop Wireless Mobile and Ubiquitous Technology in Education, pp: 129-133. Nov. 2006.
6. Fielding, R. and R. Taylor, 2002. Principled Design of the Modern Web Architecture, ACM Trans. Internet Technology, 2(2): 115-150.
7. Flinn, J. and M. Satyanarayanan, 1999. PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications," Proc. IEEE Second Workshop Mobile Computer Systems and Applications, pp: 1999.
8. Duda, I., M. Aleksy and T. Butter, 2005. Architectures for Mobile Device Integration into Service-Oriented Architectures, Proc. Int'l Conf. Mobile Business (ICMB '05), 2005.