SERVICE-ORIENTED ARCHITECTURE TO MOBILE PHONES

Elena Sánchez-Nielsen, Sandra Martín-Ruiz, Jorge Rodríguez-Pedrianes
Dpto. E.I.O. y Computación – Escuela Técnica Superior de Ingeniería Informática
La Laguna University, 38271 La Laguna, Spain

Keywords: Service oriented architecture, Mobile phones, Dynamical services invocation.

Abstract: New business opportunities are being enlarged by mobile phones. Electronic commerce on mobile phones is an emerging expression of internet electronic commerce to provide sophisticated services to mobile users. The Web services technology has been proposed to support electronic commerce on wired networks. However, no significant research has been done to allow mobile phones acting as Web services requestor at runtime without prior knowledge of available services. In this paper, we propose a service oriented architecture to mobile phones. We introduce service managers as proxies between services providers and mobile devices in order to solve the limitations faced by direct access from mobile devices to Web services. We investigate the use of dynamic invocation interface as communication mechanism between service providers and service managers in order to compute service description and invocation at runtime. With this approach, we allow that service providers to create, update and change services at anytime and mobile users may locate new services without knowing the accessible services. We have implemented a first prototype with open source tools and basic service scenarios.

1 INTRODUCTION

Mobile phones are enlarging new business opportunities. Examples of mobile services (m-services) include access to basic services (e.g., search engine tools, language translation facilities, newspaper reports, weather forecast,…), mobile shopping (e.g., booking flights, reserving rental cars, restaurants,…), mobile banking (e.g., billing of services, buying stocks and contacting banks through mobile devices) and m-government .

Having pre-installed services on users' mobile phones is an option that cannot be considered in an open environment with multiple users with different needs. Therefore, online delivery of services from providers to mobile users and dynamical service discovery infrastructure (where provider services can announce their presence and mobile users can locate these services) is more appropriate in dynamic contexts than pre-installing services (Pilioura et al. 2003, p.28-36). At present, Web Services technology (Gustavo et al. 2004) is used as approach to support services on wired networks (e-services) as well as integrating enterprise applications in a heterogeneous environment. However, the available technology to discover, access and invoke Web services directly from mobile devices lacks some important features. First, the using of static stub based communication between a mobile client application and Web services framework implies a stub appended to the client at compile time for each Web service to be invoked. Second, available technology using open source software does not allow users to access directly to UDDI registry from mobile phones. This context requires that all services must be specified at design time and no new services can be incorporated later (at runtime) by service providers. The other solution consists of downloading a new application to client device when new services are available. With the purpose of providing availability service and dynamical discovery to mobile users at anytime and anywhere, in this paper, we focus on dynamic binding for mobile devices. In particular, we propose a manager entity with a service registry as an intermediate layer between service providers and mobile users. This entity is responsible of how to make service information available and delivery services to mobile users at anytime. We investigate the use of dynamic invocation interface as communication mechanism between service providers and service managers in order to compute service description and invocation at runtime. We propose XML-encoded data exchange between service managers and mobile phones to describe: (i) Web services

Sánchez-Nielsen E., Martín-Ruiz S. and Rodríguez-Pedrianes J. (2006).
SERVICE-ORIENTED ARCHITECTURE TO MOBILE PHONES.
In *Proceedings of the International Conference on e-Business*, pages 57-62
DOI: 10.5220/0001425000570062

descriptions, (ii) operation invocations and (iii) searching at UDDI registry. With the framework proposed, service providers and new services can easily be added at anytime without updating the application of users' mobile phones when new services are incorporated. With the purpose of reducing technology fragmentation and enhancing interoperability, the solution has been developed using open source software, standards and specifications.

The remainder of this paper is organized as follows. Section 2 describes the related work about Web services technology. Section 3 outlines the architecture proposed. Section 4 describes prototype implementation and experimental results with basic services and UDDI registry. Section 5 gives concluding remarks.

2 RELATED WORK

The Web services paradigm (Gustavo et al. 2004), (Vinoski 2002, p.89-91) offers and consumes software as services. Interactions among Web services involve three types of participants: service provider, service requestor and service registry. Service providers are the owners that offer services. They define descriptions of their services and publish them in the service registry. Service requestors use a find operation to locate services of interest. The registry returns the description of each relevant service. The requestor uses this description to invoke the corresponding Web Service. Three standardization initiatives XML-based are used in order to support interactions among Web services: WSDL (Web Services Description Language n.d.), UDDI (UDDI n.d.) and SOAP (Simple Object Access Protocol n.d.).

Using SOAP-based interaction, services can exchange messages by means of standardized conventions to turn a service invocation into an XML message, to exchange the message, and to turn the XML message back into an actual service invocation. Through WSDL, a designer specifies the programming interface of a Web service. Four type of messages SOAP are possible: RPC/encoded, RPC/literal, document/literal and document/encoded. Using Web Services paradigm, the client makes a procedure call of a Web service in the same way it invokes a local call. According to client applications have access to WSDL file at compile time or runtime, invocation of Web services can be carried out by means of:

- Static Stub: a procedure call of a client application is an invocation of a proxy procedure located in a stub appended to the client at compile time. As a result, a client can invoke methods of a Web service directly via the stub. The advantage of this model is that it is simple and easy to code. The disadvantage is that the slightest change of Web service definition leads to the stub being useless and a generation of a new stub. Therefore, the stub based approach is only appropriate in static contexts, where services are not removed and updated by service providers.

- Dynamic proxy: in this case, the client application calls a remote procedure through a dynamic proxy that is created at runtime. The dynamic proxy needs to be re-instated whenever the service endpoint interfaces change.

- Dynamic invocation interface (DII): this model enables dynamic invocation of Web services without having to know interface details at compile time. With this approach, a client application can query for a service it has never heard of and build on the fly a call to that service. As a result, an application is able to invoke a service that was not known prior to runtime: it can dynamically download the appropriate WSDL file, parse it, and construct all the elements required to use the service.

3 SERVICE ORIENTED APPROACH

Traditional service oriented architecture using stub-based model as communication mechanism between client mobile applications and provider services is not appropriate to support mobile services in dynamic contexts. There are two main limitations: (i) every service needs to be coded in the client application assuming a detailed knowledge of each service that will be invoked at runtime and (ii) a stub for each service provided needs to be appended to the client application at compile time in the mobile device. This situation involves that all services must be described (network address, operations to provide, parameter…) at the design-time and no new services can be added after compile time. This context means that service providers cannot create, update and change services at anytime and that mobile users can only access to pre-defined services at their mobile phones. In order to solve the problems faced by traditional Web services architecture to mobile users, we propose a framework with an intermediate entity between service providers and service clients. This entity is represented by a service manager that operates as a client of the distributed network of Web services offered by the different service providers and as server to mobile phones.

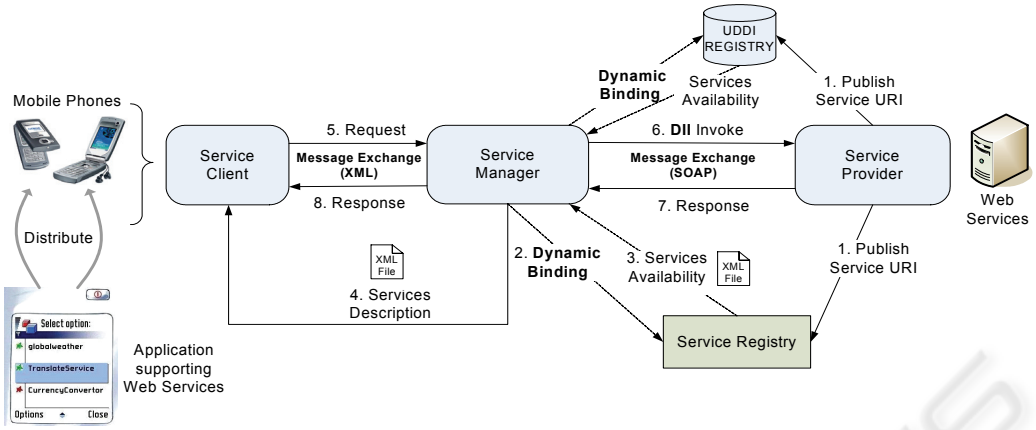


Figure 1: Service oriented architecture for mobile phones.

Figure 2 illustrates the architecture proposed. In the following sections, we describe the components illustrated in Figure 2, the interactions among the different components and the XML infrastructure proposed as service manager and format of exchange data between service managers and mobile client applications.

3.1 Service Providers and Clients

Service providers are the owners that offer different services. They define descriptions of their services using WSDL specifications (Web Services Description Language n.d.). Service clients are mobile phones-oriented users interested in standard services and searching of facilities, when it is needed without prior knowledge of available services and bringing these facilities to their devices in a transparent way.

3.2 Service Managers

Service managers act as a mediator layer between service providers and mobile clients. They are responsible for information flow among both components. A service manager is a Web service entity that uses dynamic binding to compute service descriptions and dynamic invocation interface (DII) to query for services to service providers. With the use of DII, we allow service managers to invoke Web services without knowing their communication interface at compile time. As a result, we obtain several advantages: (i) invocations of Web services not known prior can be computed by the service manager, (ii) service providers can create, update and change Web services at runtime, (iii) no static stub generated manually for the service manager at compile time is required and (iv) a single stub appended to a Java ME application is required. This appended stub corresponds to the service manager. According to the structure of marketplace, one or multiple service managers can be supported. The use of a single service manager involves a centralized marketplace. If multiple service managers are used, different operators or third parties can be incorporated at anytime, where each one can support different service providers.

The integration of service managers into a service oriented architecture leads to mobile client applications to only interact with these components and no with the different service providers. This way, a single stub corresponding to the service manager is needed to be appended to the client application and no several stubs corresponding to the different services available on the marketplace.

Client applications that reside in mobile devices only interact with a service manager.

Interactions between mobile devices and service providers using a service manager entity consist of the following processes:

-**Start up:** When the service manager starts up, it processes a service registry. This registry is a structure that enables service providers to store their list of URL address (URI) of accessible services made available. The service manager maintains a XML based structure as registry. The structure of this registry is described in section 3.3.1. Dynamic binding is used by the service manager in order to obtain the service descriptions at runtime.

-**Service delivery descriptions:** the description (operations provided, parameters…) of available Web services is sent from service manager to mobile client according to the XML document described in section 3.3.2.

-**Request service:** once mobile clients have received the descriptions of available services, they send requests of services of their interest.

-Service invocation: service manager receives a request encoded as an XML message with the necessary information (Web service name, selected operation, parameter values introduced…) from a mobile device when a user is interested in some service. Dynamic invocation is used by service manager in order to invoke Web services functionalities to service providers.
-Results transmission: the service manager sends the information encoded as an XML message to the mobile user, when it receives the response of the corresponding service provider. This information is shown on the screen display of the mobile device.
-UDDI services: mobile clients can also demand services supported by UDDI registry. In this context, a client makes a request to the service manager using keyword in order to discover a particular service at UDDI registry. Following, the service manager uses dynamic binding to discover services at UDDI registry that match with the user search criterion. The description of these services is sent from service manager to the mobile application. The user selects the service of its interest and finally service manager process this request at the same way as the request and invocation of services described previously.

3.3 UDDI Registry and XML Based Infrastructure

UDDI service directory can also be used by mobile users in order to locate new services. Service discovery is computed at runtime by the service manager, once the user has sent their request of new services in UDDI registry. A uniform infrastructure using XML-encoded data exchange is used with two purposes: (i) to define the service registry structure and (ii) establish the communication between the mobile phone and the service manager.

3.3.1 Service Registry

The XML structure of the service registry enables service providers to store their network addresses URL of WSDL documents. The format of the XML document is the following:

```
<?xml version="1.0" encoding="windows-1252"?>
<WSDLaddresses>
    <version>1.2</version>
    <wsdl>http://api.google.com/GoogleSearch.wsdl</
    wsdl>
      …
    <wsdl>http://www.webservicex.com/TranslateServi
    ce.asm?WSDL</wsdl>
<WSDLaddresses>
```

In this document, the tag *WSDLaddresses* contains two types of elements: *version* and *wsdl*. *version* is the current version of the Web Services set provided to the service manager. This element is used in order to update new services to mobile phones. *wsdl* is the URL address where is located the WSDL document of a Web Service offered. There are so many *wsdl* elements like Web Services offered.

3.3.2 Web Service Description

The following format of XML document is sent from the service manager to the mobile phone, when the description of Web Services available and/or location of new services in UDDI are required by the mobile user:

```
<webservices version="1.0">
   <service name="Calculator">
   wsdl="http://81.45.231.68:8080/Server/Calculato
   r.wsdl">
      <porType localPart="Calculator"
      namespaceURI="http://calculator.com">
        <operation name="add">
            <title> Add operation</title>
            <description>To compute add operation
            between two numbers</description>
            <parameters>
               <parameter name="Operator1"
               type="number">
               <parameter name="Operator2"
               type="number">
            </parameters>
            <return>true</return>
        </operation>
        …
        <operation name="subtract">
            …
        </operation>
      </porType>
        …
      <porType>
      …
      </portType>
   </service>
   <service>
   …
   </service>
</webservices>
```

The root of the document is the *webservices* tag. It represents the start of the set of Web Services. The attribute *version* corresponds to the current version of Web Services descriptions set provided to the mobile device. The use of this attribute allows

users check their set of descriptions with the service manager and downloading a new version, when new services have been added. Each Web Service is described with two attributes: *name* and *wsdl*. Different port types can be associated to a specific Web Service through the element *portType*. This element contains three attributes: *localPart*, *namespaceURI* and *operation*. A operation is described by the following elements: *title* (title of the operation), *description* (description of the operation), *parameters* (input parameters of the operation), and *return* (true is returned if the operation returns a value, false in other case).

3.3.3 Operation Invocation

The structure of the XML document sent from service manager to the mobile device when a Web service operation is invoked is the following:

```
<results>
   <result>Search time:0.5 seconds</result>
   <result>
      <result>Department Store: XYZ</result>
      <result>
         <result>
            <result>Name: wood table</result>
            <result>Price: 50 euros</result>
         </result>
      …
      </result>
   </result>
</results>
```

The different results are enclosed between *result* tags. If a complex type is returned (such as arrays and structures), the different fields of this type are enclosed into different *result* tags.

3.3.4 UDDI Search

A service manager sends to the mobile phone, the following XML document, when a user requests a search for new Web Services in UDDI registry:

```
<UDDI>
   <webservice
   wsdl="http://81.45.231.68:8080/Server/Calculato
   r.wsdl">
      <description>="To compute add, subtract
      operations…"/>
   …
</UDDI>
```

The different Web Services located are enclosed by a *webservice* tag. *wsdl* and *description* attributes represent the URL address of the WSDL document and the Web Service description.

4 IMPLEMENTATION

In order to test the Web services framework for mobile phones, we have implemented and tested basic service scenarios. The following service scenarios have been developed using a single service manager entity: (1) searching with Google engine, (2) text translation from one language to another, (3) newspaper reports, (4) temperature converter, (5) weather forecast, (6) calculator and (7) dynamic searching with UDDI.

The framework has been implemented using the following open source software: Apache Tomcat 5.0.28 for application server, J2ME Wireless Toolkit (WTK) for developing wireless applications and designed to run on cell phones, and Eclipse 3.1 development platform with WTP (Web Tools Platform) plug-in for building software and developing Web applications. Axis and UDDI4J Java libraries have been used as SOAP motor and Java implementation of UDDI protocol. The client application has been implemented as a MIDlet using J2ME Wireless Toolkit. That is, a Java application developed with MIDP profile and CLDC configuration. We have tested the Java application on the Sun emulator and mobile emulators of commercial trademarks.

In order to invoke Web services from a J2ME application, the mobile devices must support the Java Specification Request 172 (JSR-172). At present, however, JSR-172 has no support to UDDI specification in a J2ME application.

The application developed to mobile users presents different menus with different options, e.g.,: invocation, update, delete and searching of new Web services. Figure 2 illustrates the main menu of the client application with three different options:

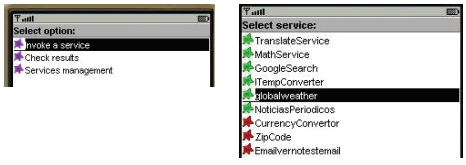


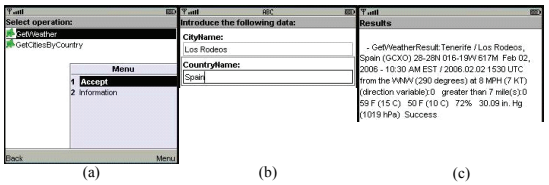
Figure 2: Main menu of the J2ME application and specific menu of "Invoke a service".

- Invoke a service: the set of available services to the mobile device is shown to the user. Two set of different services are available: (i) standard services registered by the service manager entity (represented by a green star) and (ii) services searched in UDDI

(represented by a red star). Right image of figure 2 depicts such functionality. After a service is selected, the set of operations supported is shown to the user. Figure 3 illustrates a weather forecast service on Sun's J2ME emulator.

- Check results: allows users to check the results from the different invocations achieved. A list with the different invoked services by the user is saved with the following attributes: operation, parameters values, invocation date and result computed. This way, user can read results of previous services invoked at anytime.

- Services management: allows users to select different management tasks such as: (i) update Web Services from service managers, (ii) searching services in UDDI and (iii) remove services from the mobile phone.



(a) (b) (c)

Figure 3: Global weather forecast service: (a) Selection of the operation to invoke, (b) Introduction of the parameter values, and (c) Results of the invoked operation.

4.1 Experimental Results

At present, we found that open source development tools for building, deploying and testing production quality work well together. Using a dynamic binding based approach and MIDlet's as client applications allows end users to download a single time the application directly to their device over-the-air or via their PC. No update is required, when new services are provided. However, we have found the following drawbacks when J2ME technology is used: (1) Java Specification Request 172 (JSR-172) required for invocating Web services from a mobile J2ME application does not support UDDI specification and SOAP encoded messages, (2) there is only support to static stubs, so new stubs must be manually generated when new services are incorporated at runtime and clients must download a new application in order to incorporate the new services. We have solved both problems by the use of service managers. (3) The use of UDDI registry provides high time responses and also many of the services published at UDDI registry are not correctly published. Thus, all the services consulted through UDDI registry must be verified by the service manager, before the results are sent to the mobile user. We have solved this problem verifying the services required to UDDI registry by the service manager entity and (4) specific implementations must be developed in order to support complex types when dynamic invocation interface is used.

5 CONCLUSIONS

In this paper, we address the discovery and invocation of Web services from mobile phones at runtime. We introduce service managers that act as clients over the network of services and as servers to the mobile devices. We propose the use of dynamic binding in order to compute at runtime Web services descriptions and invoke services selected by mobile users. We provide a uniform infrastructure using XML-encoded data exchange between service managers and mobile devices. With this approach, we delegate the business logic to service managers, solving the limitations faced by direct access from mobile devices to Web services. The use of dynamic binding allows that providers to create, update and change services at anytime and mobile users may locate new services without knowing the accessible services. However, the use of dynamic binding implies a bigger time to access to remote SOAP services in relation to the use of static stubs or dynamic proxies.

REFERENCES

Gustavo, A., Casati, F., Kuno H. and Machiraju, V., 2004. *Web Services: concepts, archictectures and applications.* Springer-Verlag publications, Berlin.

UDDI. Universal Description, Discovery, and Integration. Retrieved from http://www.uddi.org/

Vinoski, S, 2002. *Web Services Interactions Models*, Part 1: Current Practice. In IEEE Internet Computing, Vol 6, Nº 3, pp. 89-91.

Simple Object Access Protocol. (SOAP). W3C. World Wide Web Consortium. Retrieved from http://www.w3.org/TR/soap/

Web Services Description Language (WSDL). W3C. World Wide Web Consortium. Retrieved from http://www.w3.org/TR/wsdl

Pilioura T, Tsalgatidou A. and Hadjiefthymiades S, 2003. *Scenarios of using Web Services in M-Commerce*. In ACM SIGecom Exchanges, Vol. 3, Nº 3, pp. 28-36.