WEB Services for Ubiquitous Mobile Device Applications

Mihai Barbos
IT&C Department
SC IPA SA
Bucharest, Romania
e-mail: mihaibarbos@ipa.ro

Eugen Pop
IT&C Department
SC IPA SA
Bucharest, Romania
e-mail: epop@ipa.ro

Abstract— A WEB service is as a self-describing, self-contained software module available via a network, such as the Internet, which completes tasks, solves problems, or conducts transactions on behalf of a user or application. WEB services constitute a distributed computer infrastructure made up of many different interacting application modules trying to communicate over private or public networks (including the Internet and WEB) to virtually form a single logical system. Mobile WEB services target embedded devices. In other words, they enable handheld devices to interact with servers in a standardized way, regardless of operating systems, platforms, and programming languages. WEB services provide good opportunities for developing ubiquitous mobile client applications, allowing the delivery of information to users anytime and from anywhere. This paper brings to focus some general considerations required in designing and building WEB services that target ubiquitous mobile applications as consumers. The research work presented here was carried out under the UbiPOL FP7 research project, which aims to develop a ubiquitous platform for policy making, funded within the grant agreement nr. 248010. The main achievement of this work is the analysis of several WEB service frameworks and choosing the most appropriate one for developing the UbiPOL platform.

Keywords - WEB services; mobile devices; ubiquitous applications

I. INTRODUCTION

The "always on" vision of mobile Internet access has became a reality with the nearly ubiquitous coverage provided by cellular networks. Communications speeds have also increased significantly with the advent of 3G mobile networks that enable data rates supportive of real time video. In addition, using the competing access technologies such as wireless, the user can experience Internet data rates similar to those available with broadband connectivity in the fixed networks [4].

The increasing use of mobile terminals and infrastructure makes it possible the communications and information access from any location at any time. The convergence of mobile and WEB service technologies enables new services and business models, and accelerates the development of mobile and fixed Internet technologies [1]. The mobile industry is poised to take advantage of the benefits of interoperability that WEB services provide [2]. Interoperable messages can reduce the time and costs of business integration, creating opportunities for the adoption of WEB services technologies.

The present paper describes the technology and the building blocks needed to be put together in mobile networks, that can wirelessly delivered WEB content [7]. Included are extensive coverage of the network elements, languages used to represent browser content, communication protocols, network services and related software components that are used in the operation of such networks [5], [8]. The UbiPOL platform services will include user location tracking, security schemes, content personalization approaches, privacy mechanisms, etc. .

While on the move, a mobile user faces many challenges such as mobile terminal's limited screen size, restricted input capabilities, battery power constrains and air time costs. There is where knowledge of a user context can be leveraged to drive and personalize the interaction between user and the Internet server, so as to ease the communication exchange and focus the delivery of WEB content to the user.

First, the paper presents and analyzes the tendencies and characteristics of the mobile device market. Then, the JAVA ME platform is presented as a very useful and performance tool for mobile application development. In fact, JAVA ME seems to be the nost ubiquitous application platform for mobile devices, which complies with the UbiPOL objectives. Following these considerations, the WEB services clients in JAVA ME are also presented. Several Java WEB service stacks and frameworks that will be considered in the implementation of UbiPOL are presented next. Finally, a section for conclusions is present at the end of the paper.

II. THE MOBILE DEVICE MARKET

The mobile devices market trends are closely related to the significant increase of the bandwidth necessity and data traffic on the cellular or wireless channels. Nowadays, the network and service infrastructure domain is facing many important changes like:

- Explosive growth of data rates and capacities;

 ISBN: 978-1-61208-105-2

- The emergence of massive data and content delivery and consumption;
- Evolution towards a converged architecture that has dramatically increased the permutations and combinations of services and usages between people, devices, media, and even between real and virtual worlds;
- User involvement in defining her/his communication sphere has led to much more customization, personalization and users becoming content producers.

An explosion in number of mobile Internet devices is expected, evolving towards trillions of connected devices, M2M, within the Internet of objects.

Globally, mobile data traffic will double every year through 2014, increasing 39 times between 2009 and 2014. Mobile data traffic will grow at a compound annual growth rate (CAGR) of 108 percent between 2009 and 2014, reaching 3.6 exabytes per month by 2014 [24]. The UbiPOl project aims to use this opportunity to involve the citizens in the policy making process.

According to Gartner Inc. Report, the worldwide mobile phone sales totalled 286.1 million units in the second quarter of 2009, a 6.1 per cent decrease from the second quarter of 2008, according to Gartner, Inc. [3] (see Table 1). Smartphone sales surpassed 40 million units, a 27 per cent increase from the same period last year, representing the fastest - growing segment of the mobile - devices market (see Table 2).

TABLE I. WORLDWIDE MOBILE TERMINAL SALES END USERS IN 2Q09 (THOUSANDS OF UNITS)

Company	2Q09 Sales	2Q09 Market Share (%)	2Q08 Sales	2Q08 Market Share (%)
Nokia	105,413.3	36.8	120,353.3	39.5
Samsung	55,430.2	19.3	46,376.0	15.2
LG	30,497.0	10.7	26,698.9	8.8
Motorola	15,947.8	5.6	30,371.8	10.0
Sony Ericsson	13,574.2	4.7	22,951.7	7.5
Others	65,260.2	23.0	57,970.6	19.0
Total	**286,122.7**	**100**	**304,722.3**	**100**

TABLE II. WORLDWIDE SMARTPHONE SALES TO END USERS IN 2Q09 (THOUSANDS OF UNITS)

Company	**2Q09 Sales**	**2Q09 Market Share (%)**	**2Q08 Sales**	**2Q08 Market Share (%)**
Nokia	18,441.0	45.0	15,297.9	47.4
Research In Motion	7,678.9	18.7	5,594.2	17.3
Apple	5,434.7	13.3	892.5	2.8
HTC	2,471.0	6.0	1,330.8	4.1
Fujitsu	1,249.0	3.0	1,071.5	3.3
Others	5,688.2	13.9	8,085.8	25.1
Total	**40,962.8**	**100.0**	**32,272.7**	**100.0**

As mobile devices continue to converge with consumer electronics, vendors find themselves (and their devices) locked in a battle over similar market segments, similar buyer demographics and similar product concepts. Mobile users have a wide choice of capable device types to fulfill their mobile communication and computing needs. Studies highlighed key device segment trends and market considerations across many mobile device categories, including mobile handsets and handset accessories. Since wireless connectivity continues to be incorporated into new device segments, the impact of emerging services and technologies is in close relation to their market potential in mobile devices [4].

The remarkable technical performances of the mobile devices make them available for a great variety of applications: social networking, consumer and, business applications, content & delivery platforms, messaging, browsers, operating systems (OS), and users interfaces (UI).

III. JAVA ME "THE MOST UBIQUITOUS APPLICATION PLATFORM FOR MOBILE DEVICES

Java Platform, Micro Edition (Java ME) provides a robust, flexible environment for applications running on mobile and other embedded devices—mobile phones, personal digital assistants (PDAs), TV set - top boxes, and printers. Java ME includes flexible user interfaces, robust security, built - in network protocols, and support for networked and offline applications that can be downloaded dynamically [5]. Applications based on Java ME are portable across many devices, yet leverage each device's native capabilities.

In order to provide ubiquitous deployment across different mobile operating systems and device types the Java ME platform was selected for development of front end components for the UbiPOL platform [6]. An inventory of mobile application development tools was realized. For each mobile application development tool, the following issues were analyzed and specified: system's requirements, installation procedures, available device emulators, Java ME API, etc.

The following tools are components of the mobile application development environment for UbiPOL:

- Java ME SDK 3.0, [15],[16];
- Nokia S60 3rd Edition SDK for Java, [17];
- BlackBerry JDE 5.0 [21];
- LG SDK 1.5 for the Java ME Platform ; [18];
- MOTODEV SDK for Java ME v3.0 [19];
- Samsung Java SDK 1.1.2, [20];
- Sony Ericsson SDK 2.5.0.6 for JavaME Platform[22].

The SDK from SUN was chosen as the default development tool for UbiPOL mobile applications identified as necessary. Compiling, building, running, testing and debugging mobile applications will be done using this tool. The other tools have been deployed only for test purpose, in order to ensure UbiPOL mobile applications compatibility with other device types from leading market manufacturers.

The Java ME Platform SDK 3.0 form SUN provides device emulation, a standalone development environment, and a set of utilities for rapid development of Java ME

 ISBN: 978-1-61208-105-2

applications. On Windows, Java ME SDK 3.0 is the successor of the Java Wireless Toolkit 2.5.2 and Java Toolkit 1.0 for CDC. The Java ME SDK 3.0 is available for Windows XP and Vista 32 - bit, and for Mac OS.

The Java ME SDK 3.0 includes several emulators to allow running, testing and debugging applications under different scenarios (different device screen size, different input device configuration – key board, touch screen, both –, external events generator support –location events, device orientation change –, number of colours, different Java ME API support etc.) The device emulators available in the SUN's Java ME SDK are: ClamshellCldcPhone1, DefaultCldcJtwiPhone1, DefaultCldcJtwiPhone2, Default CldcMsaPhone1, DefaultCldcMsaPhone2, DefaultCldc Phone1, DefaultCldcPhone2, DefaultFxPhone1, DefaultFx TouchPhone.

The Java ME SDK from SUN offers a great number of Java ME API's, configurations and profiles as follows: configurations (CLDC 1.0, CLDC - 1.1), profiles (MIDP - 2.0), Java ME API's (ex. JSR 172 – WEB Services Specifications, JSR 179 – Location API, and so on), available for each emulator included in the SDK from SUN.

IV. WEB SERVICES CLIENTS IN JAVA ME

The J2ME WEB services API (WSA) extends the Java2 Platform, ME to support WEB services, and was developed within the Java Community Process, as JSR 172. The API's two optional packages standardize two areas of functionality that are crucial to clients of UbiPOL WEB services: remote service invocation and XML parsing.

WSA is designed to work with J2ME profiles based on either the Connected Device Configuration (CDC) or the Connected Limited Device Configuration (CLDC 1.0 or CLDC 1.1). The remote invocation API is based on a strict subset of J2SE's Java API for XML - Based RPC (JAX - RPC 1.1), with some Remote Method Invocation (RMI) classes included to satisfy JAX - RPC dependencies. The XML - parsing API is based on a strict subset of the Simple API for XML, version 2 (SAX2).

The core specifications and application - level protocols that define WEB services are promoted by the WEB Services Interoperability Organization (WS - I), and governed by the World Wide Web Consortium (W3C) and the Organization for the Advancement of Structured Information Standards (OASIS) [8]. The four key standards that specify how to create, deploy, find, and use WEB services, are presented in the following table.

TABLE III. WEB SERVICES STANDRADS

WEB services standards	Description
Simple Object Access Protocol (SOAP) 1.1	Defines transport and data encoding
WEB Services Definition Language (WSDL) 1.1	Defines how remote services are described
Universal Description, Discovery, & Integration (UDDI) 2.0	Defines how remote services are discovered
Extensible Markup Language (XML) 1.0, and XML Schema	Defines the Extensible Markup Language (XML) and XML Schema

The goal of WSA is to integrate fundamental support for WEB services invocation and XML parsing into the device's runtime environment, so developers don't have to embed such functionality in each application.

To make the interpretations of the standards easy, WS - I has defined a set of conformance rules called the WS - I Basic Profile, version 1.0. JSR 172 conforms to the Basic Profile. JSR 172 specifies standardized client - side technology to enable J2ME applications to consume remote services on typical WEB services architectures, as Figure 1 illustrates:

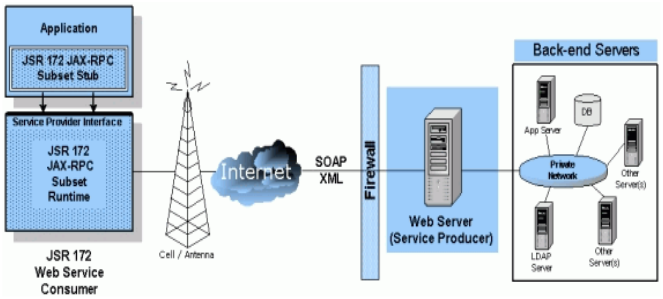

Figure 1. J2ME in a typical WEB service architecture

At a high level, this WEB service architecture has three elements:

- A network-aware application residing on a WSA-enabled wireless device. The application includes a JSR 172 stub that uses the JSR 172 runtime to communicate with the network.

- The wireless networks, the Internet and the corresponding communication and data - encoding protocols, including binary protocols, HTTP, and SOAP/XML.

- A web server, acting as the service producer, typically behind one or more firewalls and a proxy gateway; the web server often provides access to back - end applications and servers on a private network [7].

A typical JSR 172 based application is a smart client based on the Mobile Information Device Profile (MIDP) or the Personal Basis Profile (PBP), with business - specific logic, user interface, persistence logic, and life - cycle and application - state management. To handle XML documents, the application can employ the JAXP subset API. To consume WEB services, it can use the JAX - RPC subset API, employing JSR 172 stubs and the runtime. In devices such as cell phones, typically the application and the JSR 172 stub reside in the device's memory, while all the JSR 172 elements, along with the underlying profile and configuration, are embedded in the device itself.

At the center of JSR 172 operations is the runtime, with its service provider interface, which enables the stubs to perform all the tasks associated with invoking an RPC service endpoint:

- Set properties specific to an RPC invocation;
- Describe the RPC invocation input and return values;

 ISBN: 978-1-61208-105-2

- Encode input values;
- Invoke the RPC service endpoint;
-decode and return to the application any values that the service endpoint returns.

In the Figure 2 is presented how a typical JSR 172 based application is organized.

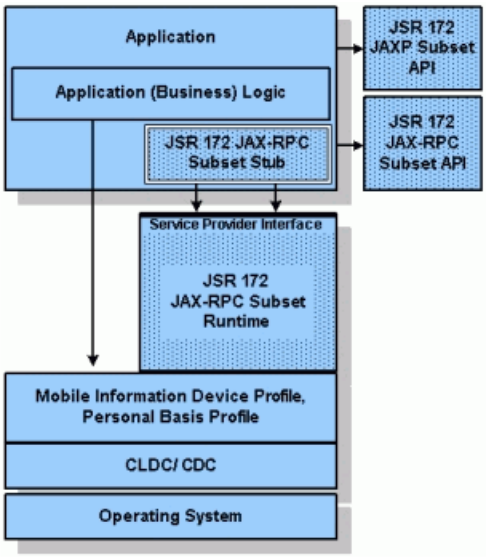

Figure 2. A typical JSR 172 based application architecture

There are many WEB services useful for developing Java ME client applications. Among them, Google Maps API WEB services is a collection of HTTP interfaces to Google services providing geographic data for maps client applications. Google Maps offers REST services that allow accessing its data with simple HTTP requests, so they can be easily integrated into mobile applications [13].

The developer must sign up, [23] to get a key (API_KEY, a simple string) that he/she will use for all the queries to Google Map services.

The static maps WEB service allows retrieving single images that can be used in mobile applications, because it not requires Javascript or any dynamic page loading.

The static maps service supports different image formats (png32, GIF, JPG) and customizable image size that can be used for all purposes. The developer must retrieve an URL with an HTTP request, that must include some parameters like: the map's center geographical coordinates (latitude and longitude), the image format and size, the zoom level (the zoom range is from 0 to a maxim level of 19) and the API_KEY [14].

Geocoding is the process of converting an usual address into geographical coordinates, that can be used, for example, to place markers or position the map. The Google Geocoding API is another REST service that provides access to a geocoder via an HTTP request, so it can be integrated into mobile applications. Additionally, "reverse geocoding" can be performed, that means the converse operation (turning coordinates into addresses).

Location API (JSR - 179) allows Java ME applications to get the user geographic location. The implementation of this technique in the device can be based on a GPS technology, using the mobile phone network (with Cell ID), etc.

V. WEB SERVICES FRAMEWORKS AND PROTOCOLS

WEB services are being widely deployed to facilitate interoperability across different hardware and software implementation, machine architectures and application programming interfaces (API's) [9].

Creating effective mobile WEB services requires an architecture that addresses issues related to identity management, security, the machine readable description of WEB services and methods for discovering WEB services instances.

The XML Protocol work is the foundation for a WEB Service framework within which automated, decentralized services can be defined, deployed, manipulated and evolved in an automated fashion. This framework provides a structure for integration and a foundation for protocols that will support the needs of such service - oriented applications. The goal is a scalable, layered architecture, one that can appropriately meet the needs of both simple and extremely robust high - volume deployments. As with other Web technologies, the focus is on enabling ubiquitous interconnectivity of entities and organizations dispersed throughout the world. The WEB services framework focuses on supporting application - to - application integration between entities having disjoint platforms, management, infrastructures and trust domains. Using the framework, a model for describing, discovering and exchanging information can be realized, that is independent of application implementations and the platforms on which applications are developed and deployed.

The UbiPOL platform will be realized using frameworks and stacks that allow the WEB services implementation using Java language. As implementation options for UbiPOL platform the following WEB service frameworks have been considered: Axis 1.x, Axis2, CXF, Glue, JBossWS, XFire (1.2), Metro, OracleAS 10g.

Their general features and WS related JSR standards, as a criteria list approach, are presented synthetically in the following tables.

Apache Axis is an open source, XML based WEB service framework, It consists of a Java and a C++ implementation of the SOAP server, and various utilities and APIs.

Apache Axis 2 is a core engine for WEB services. It is a complete re - design and re - write of the widely used Apache Axis SOAP stack.

Apache CXF is an open source services framework, suitable for building and developing services using frontend programming APIs, like JAX - WS and JAX - RS. These services can comply with protocols such as SOAP, XML/HTTP, RESTful HTTP, or CORBA.

GLUE Java WEB services, delivered by Mind Electric, is a WEB service toolkit for Java programmers that offers an easy way to implement SOAP messaging.

For UbiPOL, **the Glue server** can have two parts: a Java class that performs the business logic and a Java class

 ISBN: 978-1-61208-105-2

that starts GLUE's HTTP Server and publishes the business logic object as a SOAP service.

JBossWS is a WEB service framework developed as part of the JBoss Application Server. It implements the JAX - WS specification that defines a programming model and run - time architecture for implementing WEB services in Java [12].

XFire is an open source Java SOAP framework built on a high performance, streaming XML model. XFire includes support for WEB service standards, an easy to use API, Spring integration, JBI support, and plugable bindings.

The Metro WEB services stack is an open source tool developed by Sun Microsystems. It incorporates the reference implementations of the JAXB 2.x data - binding and JAX - WS 2.x WEB services standards, along with other XML - related Java standards.

Oracle Application Server 10g R3 WEB Services provides a new runtime infrastructure supporting J2EE 1.4 WEB services. Figure 3 provides an architectural overview of this new infrastructure [15].

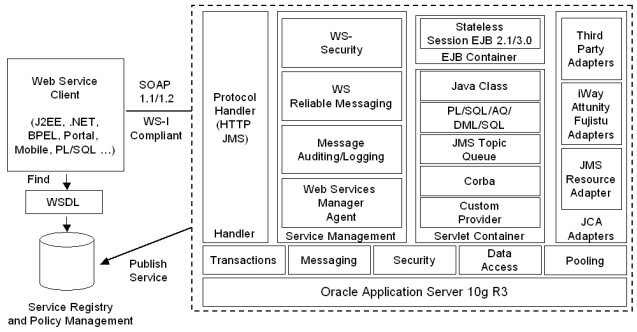

Figure 3. Oracle Application server 10g R3 (10.1.3.0.0) WEB Services Framework

Their general features and WS related JSR standards, as a criteria list approach, are presented synthetically in the following tables.

TABLE IV. WEB SERVICES FRAMEWORK GENERAL FEATURES

Feature	Axis 1.x, 2	Cxf	Glue	JBossWS XFire 1,2 Metro	Oracle AS 10g
Basic Profile Compliant	☑	☑	☑	☑	☑
Easily Create Services from POJOs	☑	☑	☑	☑	☑
Open Source	☑	☑	☑	☑	
RPC-Encoding	☑	☒	☑	☑ M1	☑
Spring Support	☑	☑	☒	☒	☒
REST Support	☒	☑	☑	☒	☑
IDEA Plugins	☒	☑	☑	☑	ANT
Eclipse Plugins	☒	☑	☒	☑	ANT
NetBeans Plugins	☒	☒	STP	☒	ANT
JDeveloper	☒	☒	☒	☒	☑
Hot Deployment	☒	☑	☒	☑	☑
Soap 1.1	☑	☑	☑	☑	☑
Soap 1.2	☑	☑	☑	☑	☑
Streaming XML (StAX based)	☒	☑	☑	☒	☒
WSDL 1.1 –Code (Client)	☑	☑	☑	☑	☑
WSDL 1.1 –Code (Server)	☑	☑	☑	☑	☑
WSDL 2.0 –Code (Client)	☒	☑	☑	☒	☒
WSDL2.0 – Code (Server)	☒	☑	☒	☒	☒
Client-side Asynchrony	☑	☑	☑	☒	BPEL
Server-side Asynchrony	☑	☑	☑	☒	BPEL
Policy-driven code generation	☒	☑	☑	☒	☒

TABLE V. WS RELATED JSR STANDRADS

Feature	Axis 1.x, 2	Cxf	Glue	JBossWS XFire 1,2 Metro	Oracle AS 10g
JAX-RPC	☑	☑	☑	☑	☑
JAX-WS	☑ A22	☑	☑	☑ M3	☑
JAX-RS	☑	☑	☑	☑	
JSR 181	☑	☒	☑	☑	☑
JSR 181 on Java 1.4	☑	☑	☒	☒	☒
SAAJ (1.2/1.3)	☒	☑	☑	☒	☑
JSR 109	☒	☑	☑	☑	ANT
JBI	☒	☑	☒	☑	ANT

Generally, all the previously WEB services Frameworks comply with the following transport protocols: HTTP, JMS, SOAP/JMS Spec, Jabber, SMTP/POP3, TCP.

The notes from Tables IV and V highlight some issues relevant to our objectives, as follows:

 ISBN: 978-1-61208-105-2

- M1 - in case of the Metro WEB service framework, the Remote Procedure Call encoding is available only through the JAX-RPC 1.1 APIs and removed JBI JSR;
- M3 - in case of Metro WS, JAX-RPC 1.1, JAX-WS 2.0 and JAX-WS 2.1 RI are combined together in Metro as well as JAXB 2.0 and JAXB 2.1. JAX-WS 2.0 and JAXB 2.0 functionality is available in Java SE 6 as well;
- A22 –in case of Axis 2 WS, there is not JAX-WS TCK compliant due to lack of JAX-WS tooling;
- BPEL – allow services interaction to be described easily and thoroughly by the WS – BPEL, an XML based programming language;
- ANT – is a Java based software tool useful for automating software build processes.

CXF supports a very wide range of connection possibilities, by using the Camel transport.

The specification from the Tables IV and V show that Metro is in conformance with the WS-I Basic Profile and it supports SOAP 1.1 messages relayed over HTTP as transport. It also provides support for WSDL 1.1 and XML.

Metro is an open source web service stack that is a part of the GlassFish project. It is included in Glassfish V3 and it is available under the CDDL and GPLv2 licence. Since Metro is compliant with the WS-I Basic Profile and the requirements of JSR 172, web services deployed with it can target Java Me applications running on mobile phones as service consumers. That is why Metro seems an appropriate web service stack option for UbiPOL.

OSGi technology is the dynamic module system for Java™. The OSGi Service Platform provides functionality to Java that makes Java the premier environment for software integration and thus for development [11].

ProSyst, an OSGi and Java pioneer, is a company entirely focused on open standards technology and was most actively involved in helping to create the OSGi specifications. It offers OSGi technology based products and services for Mobile Devices market [12].

VI. CONCLUSIONS AND FUTURE WORK

The following general advantages indicate WEB services as a suitable solution for UbiPOL:

WEB services are platform and language independent: WEB services provide interoperability between various software applications running on disparate platforms. They are not tide to any operating system, development platform or programming language. WEB services allow different applications from different sources to communicate with each other, without time - consuming or custom coding. This is because WEB services use open standards and protocols like SOAP, WDSL and XML. After deployment, this could provide other developers with the opportunity to include UbiPOL WEB services in their custom applications and deliver UbiPOL content and data merged with their own.

One solution for handling interaction between Java ME applications running on mobile phones and UbiPOL servers are WEB services. WEB services will be the logic tier components of the scalable UbiPOL system architecture.

There are several APIs that provide WEB service client support for the Java Me platform: WSA, WINGFOOT SOAP client, kSOAP2. The WSA specification was developed within the Java Community Process as JSR 172. The J2ME WEB Services API (WSA) extends the Java 2 Platform, Micro Edition to support WEB services. This is why it is included in many Java ME platform implementations by mobile phone manufacturers. WSA provides out of the box support for Java ME WEB service client applications on many mobile phones available on the market.

In order to develop Java ME WEB service client applications, UbiPOL must rely on an available API. Because WSA (JSR 172) provides out of the box support for Java ME WEB service client applications on mobile phones, it is the most appropriate option for now. Other third party APIs require including those APIs in the distribution file. This would lead to larger distribution files, which is not recommended. The other options may be implemented if they prove to be necessary.

WSA (JSR 172) conforms to the WS - I Basic Profile WEB service specification. It requires the use of SOAP 1.1, WDSL 1.1 and XML 1.0. The WEB service framework or stack that will be used for the deployment of UbiPOL WEB services on the server must comply with the same specification as WSA in order to ensure compatibility and interoperability.

Several WEB service frameworks were analysed: Axis 1.x, Axis2, CXF, Glue, JBossWS, XFire (1.2), Metro, OracleAS 10g.

The application server identified as required for the implementation of UbiPOL is Glassfish. Metro is the WEB service application stack bundled with Glassfish. It is in conformance with the WS - I Basic Profile WEB service specification. This makes Metro the best available option for now.

WEB services will be logic tire components of the UbiPOL system architecture, enabling the interaction between UbiPOL Java ME applications (presentation tier) and UbiPOL database servers (data tier) [17]. UbiPOL WEB services will be in conformance with the WS - I Basic Profile 1.0 specification. SOAP 1.1 conveyed over HTTP will be used for messages sent between Java ME applications and WEB services. As SOAP is an XML based protocol, the data conveyed between Java ME applications and UbiPOL WEB services will be XML serialised for both request and response messages. UbiPOL WEB services will make use of WDSL 1.1 to provide a “machine - processable” description of the operations supported, enabling client side code generation in many IDEs. The Metro WEB service stack and the Glassfish application server will be used to deploy UbiPOL WEB services. The WSA (JSR 172) API will be used in the implementation of UbiPOL Java ME WEB service client applications.

ACKNOWLEDGMENT

This paper represents part of the work realized by the authors within the UbiPOL FP7 European Union research

project, dealing with e - participation of citizens in policy making and funded by grant agreement nr. 248010.

REFERENCES

[1] A. Pashtan, "Mobile WEB Services" Cambridge University Press 2005 pp. 5 – 30.

[2] M. Papazoglou, "What Are WEB Services?" WEB Services: Principles and Technology. Harlow, England Pearson/Prentice Hall, 2008. pp. 22- 32.

[3] Gartner Press Release on Worldwide Mobile Phones Sales http://www.gartner.com/it/page.jsp?id=1126812, [accessed: 30.08.2010] 20 August 2010

[4] P. M. Woo, K.Y. Seok, and Kyong - Ho "Migrating WEB Services in Mobile and Wireless Environments" International Journal of WEB Services Research, Volume 6, Number 2, April - June 2009 (pp. 1 - 19).

[5] D. Lizcano, J. Soriano, M. Reyes and J. Hierro "A user - centric approach for developing and deploying service front - ends in the future internet of services" International Journal of WEB and Grid Services (IJWGS) Vol 5, Issue 2 – 2009, pag. 155 - 191, doi: 10.1504/ IJWGS. 2009.027572

[6] A. Yamazaki, A. Koyama, J. Arai and L. Barolli "Design and implementation of a ubiquitous health monitoring system" International Journal of WEB and Grid Services (IJWGS) Volume 5, Issue 4 – 2009, pag. 339 - 355, doi: 10.1504/IJWGS.2009.030263

[7] E. Cerami, "Introduction to WEB Services."WEB Services Essentials, O'Reilly & Assoc., USA 2002, ISBN 0 – 596 - 0 224 - 6, pp. 10 - 50.

[8] G. Alonso and F. Casati "WEB Services - Concepts, Architectures and Applications", Springer – Verlag – Berlin, Heidelberg 2004 ISBN 3 – 540 - 44008 - 9, pp. 124 - 149.

[9] L Richardson and S Ruby "RESTful WEB Services" Farnham O'Reilly 2007 pp. 47 - 67

[10] "Appache WS Wiki"; http://wiki.apache.org/ws StackComparison; [accessed: 30.08.2010];

[11] "OSGi Technology"; http://www.osgi.org/About/Technology; [accessed: 30.08.2010];

[12] "ProSyst is an OSGi and Java Pioneer"; http://www.prosyst.co; [accessed: 30.08.2010];

[13] "GoogleMap API WEB Services" http://code.google.com/apis/maps/documentation/webservices/index.html; [accessed: 30.08.2010];

[14] "How to use Google Map Data in Mobile Applications" http://wiki.forum.nokia.com/index.php/How_to_use_Google_Maps_data_in_mobile_application; [accessed: 30.08.2010];

[15] "Java ME SDK 3.0" http://www.oracle.com/technetwork/java/javame/downloads/ sdk30-jsp-139759.html; [accessed: 30.08.2010];

[16] "Java ME SDK 3.0 released" http://weblogs.java.net/blog/ 2009/04/22/java-me-platform-sdk-30-released-goodbye-wtk-hello-java-me-sdk-part-2; [accessed: 30.08.2010];

[17] "Nokia S60 3rd Edition SDK for Java" http://www.forum.nokia.com/info/sw.nokia.com/id/6e772b17-604b-4081-999c-31f1f0dc2dbb/S60; [accessed: 30.08.2010];

[18] "LG SDK 1.5 for the Java ME Platform" http://developer lgmobile.com/lge.mdn.tnd.RetrieveTNDInfo.dev?modType=T&objectType=T&menuClassCode=&saveFileName=&resourceNo=TND00000294&selectedType=&tabIndex=1#none; [accessed: 30.08.2010];

[19] "]MOTODEV Studio for Java ME: Downloads" http://developer.motorola.com/docstools/motodevstudio/javame/download [accessed: 30.08.2010];

[20] "New Samsung Java SDK 1.1.2 - release 17th Nov 2009" http://innovator.samsungmobile.com/down/cnts/toolSDK.detail.view.do?platformId=3&cntsId=5640&listReturnUrl=http://innovator.samsungmobile.com:80/down/cnts/toolSDK.list.do%3FplatformId%3D3; [accessed 30.08.2010];

[21] "What's new in BlackBerry Java application development 5.0" http://www.blackberry.com/developers/docs/5.0.0api/index.html; [accessed: 30.08.2010];

[22] " Sony Ericsson SDK for the Java ME platform 2.5.0.6" http://dwgs3.sonyericsson.com/wportal/devworld/downloads/download/dw-99962-semcjavamecldcsdk2506?cc=gb&lc=en; [accessed: 30.08.2010];

[23] "Sign up for the Google Maps Api" http://code.google.com/apis/maps/signup.html; [accessed: 30.08.2010];

[24] " Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2009 - 2014" http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html; [accessed: 30.08.2010];

 ISBN: 978-1-61208-105-2