OSA/Parlay and J2ME in Tandem:
Developing Innovative Services by Combining Intelligence of the Application Server and the Mobile Terminal

J. Domaszewicz, M. Rój, M. Kunikowski
domaszew@tele.pw.edu.pl, mroj@tele.pw.edu.pl, mkunikow@elka.pw.edu.pl
Institute of Telecommunications, Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland

G. Carroll
garethcarroll@openapisolutions.com
Open API Solutions
116 Brunswick Street, Canton, Cardiff, CF5 1LN, UK

Abstract – The paper explores synergies that can be achieved by combining two software technologies available to service developers: OSA/Parlay and Java 2 Micro Edition. The technologies are different (and complementary) in that the former is based on relatively few powerful application servers, while the latter on numerous resource-constrained terminals. Innovative services can be realized, if an application server and mobile terminals work together. MobileDate, a “pure” OSA/Parlay dating application, is described first. Next, a number of generic OSA/Parlay and J2ME in tandem architectures are presented. Each one assumes a specific way of interaction between the OSA/Parlay application and J2ME software (MIDlets). General examples of services that can benefit from applying the presented architectures are given. The paper is concluded with a fresh look on MobileDate: what enhancements can be added, if service-specific MIDlets are allowed to run on terminals of the service users.

1. Introduction

Recently, two open platforms for the development of mobile services have emerged. One of them, OSA/Parlay API [1-4], makes it possible to develop services (OSA/Parlay applications) to be run on dedicated, powerful application servers. The other one, Java 2 Micro Edition, J2ME, (most notably the CLDC configuration with the MIDP or IMP profile [5-7]) enables the creation of applications for resource-constrained mobile terminals. The both technologies are being introduced by vendors and telecommunications network operators. Deployment of OSA/Parlay gateways and applications has begun. Java-enabled mobile phones tend to dominate the market, and J2ME applications are gaining popularity. However, little work has been done on combining the two technologies to develop distributed services that include both OSA/Parlay server-based components and J2ME terminal-based components. The objective of the paper is to explore synergies that can be achieved by using the technologies in tandem.

The paper is organized as follows. An OSA/Parlay application of moderate complexity is presented in the second section. The application, called MobileDate, assumes no intelligence in the user’s terminal. The third section presents a number of generic OSA/Parlay and J2ME in tandem architectures. Each one assumes a specific way of interaction between the OSA/Parlay application and J2ME software (MIDlets). The next four sections explore, in turn, each of the architectures in more detail. The paper is concluded with a fresh look on MobileDate: what enhancements can be added, if service-specific MIDlets are allowed to run on terminals of the service users.

2. MobileDate: a case of a “pure” OSA/Parlay application

MobileDate [8] is an OSA/Parlay application developed by the Mobile and Embedded Applications Group at Warsaw University of Technology [9]. The main goal of MobileDate is to enable making new contacts between people of matching preferences. The service is location-based. When two compatible users happen to be close to each other, the service takes an appropriate action, depending on settings selected by the users. An action can be a third-party call connecting them (Fig. 1 (a)) or an SMS notification that a compatible party is nearby (Fig. 1 (b)). In the latter case, either of the parties can then call the other or send them an SMS. The service is anonymous. Users are distinguished by unique user identifiers. Parties can get in touch, i.e., send SMS messages or make a call, by using the other user’s identifier.

Two user interfaces to manage the service are available: one is based on SMS messages and the other one on a Web page. The user can register with the service, provide his description, specify preferences, manage service settings, and see service statistics.

MobileDate consists of the following four components:
• An OSA/Parlay Application, written in Java and implementing the service logic.
• An SQL database (for settings, user preferences, etc.)
A servlet/JSP-based Web application, responsible for the Web interface.

- A cellular network; at the development phase, Open API Solutions’ Application Test Suite (ATS) [10] was employed instead of a real network.

The service architecture is shown in Fig. 2.

We now leave MobileDate and present the key topic of the paper: a presentation of OSA/Parlay and J2ME in tandem architectures and their benefits. We return to MobileDate at the end of the paper, where we show how the general ideas can be applied to make this specific service more attractive.

3. OSA/Parlay and J2ME in tandem architectures

One can identify a number of generic patterns of interaction between OSA/Parlay and J2ME software components. They are presented in Fig. 3.

A pattern called (in this paper) Internet MIDlet-to-App is depicted in Fig. 3 (a). According to the pattern, an OSA/Parlay application and a MIDlet communicate by means of the Internet. The OSA/Parlay application acts as an HTTP server to which the MIDlet can connect as an HTTP client. HTTP client functionality is a part of even the most basic J2ME platform (CLDC 1.0 with MIDP 1.0).

A different pattern, called Direct MIDlet-to-App, is shown in Fig. 3 (b). Contrary to the previous one, this pattern involves the telecom network itself as a means of interaction. More specifically, SMS messages are exchanged between the OSA/Parlay application and the MIDlet. Coupled with the so-called Push Registry mechanism of MIDP 2.0, the SMS exchange may be very effective.

The next pattern is shown in Fig. 3 (c). It is called Internet MIDlet-to-Gateway, and it is different from all the others in that the MIDlet interacts with an OSA/Parlay gateway, not an application. In this case, the information flow is one way only; the MIDlet delivers information to the OSA/Parlay gateway, which uses it to enhance its SCF implementation.

Finally, in Fig. 3 (d), a MIDlet is a Parlay X application and interacts directly with a Parlay X gateway. In this Parlay X MIDlet pattern, the mobile phone becomes an OSA/Parlay application server. As Web services work over HTTP, they can be supported as a third party extension to the most basic J2ME platform. Also, a related API has already been defined by the Java Community Process [11].

The following four sections elaborate, in turn, on all of the above architectures. In particular, they present some classes of applications that can benefit from a given approach. It is worth to remember that some of them can benefit from more than one; they were placed in a section describing the supposedly most fitting architecture.

4. Internet MIDlet-to-App

The following subsections present some major application areas for the Internet MIDlet-to-App architecture; the list is not meant to be exhaustive.
4.1 Location-aware MIDlets

The terminal’s location can be made available to the MIDlet (which this way becomes location-aware). At this time, no support for location-aware MIDlets is included in most Java-enabled phones. Even though a related API has been defined (JSR-179 [12]), its incorporation into mobile phones by vendors may be some time away.

One can consider a third party, vendor-independent implementation of the JSR-179 API for all the Java-enabled phones. Specifically, the JSR-179 location inquiry calls can be mapped into HTTP requests, which are sent to an OSA/Parlay application. The application acts as a location server that retrieves the location using the Mobility SCF and sends it to the terminal. The information is then delivered by the JSR-179 calls to the MIDlet’s logic.

Moreover, as OSA/Parlay Mobility SCF allows localizing multiple terminals, the location information returned to a MIDlet can go beyond the JSR-179 API (which is limited to the location of the terminal hosting the MIDlet only). A single MIDlet can find out the location of an entire group of users. The functionality can be useful in team-oriented applications. Some possible scenarios are: (1) a field officer gathers individual locations of the unit members and (2) a game player detects all the team members in the nearby area.

4.2 OSA/Parlay Proxy

Another instance of the discussed architecture is based on a general purpose OSA/Parlay application, which we call an OSA/Parlay Proxy. In essence, the OSA/Parlay Proxy is a gateway translating a proprietary, MIDlet-to-Proxy protocol to the OSA/Parlay APIs. The protocol should be quite comprehensive (to allow a MIDlet to access most of telecom network functionality, as exposed by the OSA/Parlay APIs), yet simple (presumably HTTP-based, so that it could be run on a terminal). Using the OSA/Parlay Proxy, lightweight quasi-OSA/Parlay applications can be run as MIDlets.

Consider Rococo’s Conference Call Scheduling OSA/Parlay Application [13], which enhances Microsoft Outlook with the possibility to schedule
multiparty conferences directly from the Outlook program. A MIDlet using the OSA/Parlay Proxy can have similar functionality, i.e., it can allow the user to make and chair conferences directly from his phone.

4.3 Ubiquitous Computing

An important field, which can benefit from the Internet MIDlet-to-App pattern, is ubiquitous computing [14]. Ubiquitous computing (ubicomp) deals with the creation of so-called context-aware applications. A context-aware application uses sensors to acquire information about the current state of its surrounding environment and the user (the context), and acts upon the information, without active user involvement, to realize its functionality.

The context-aware application designer can take advantage of the mobile phone as a handy, intelligent computing node and, at the same time, of the telecom network functionality (as exposed by the OSA/Parlay APIs). The mobile phone, being carried along by the user most of the time, shares the location with him. This is highly desirable, as any context information acquired by the mobile phone from the environment is likely to apply to the user.

A context-aware application combining J2ME and OSA/Parlay could be structured as follows. A mobile phone (MIDlet) is used for context acquisition and preprocessing. The context acquisition can be achieved either by (1) attaching sensors directly to the phone, most likely through a serial port, or (2) using the phone’s Bluetooth interface (by means of the Bluetooth API for Java [15]) to enable communications with Bluetooth-equipped sensors.

Once the MIDlet acquires the sensor output, possibly from a number of different sensors, it can act as a so-called context synthesizer [16]. A context synthesizer transforms raw, low-level sensor data into higher-level, more abstract context information. The preprocessed context is then sent by the MIDlet to the OSA/Parlay application, which takes action, if needed. A number of SCFs (e.g., PAM, CC, UI, GMS) can be meaningfully used.

An example is related to taking care of elderly or sick people living alone. Higher-level context information could be “no activity around the house has been detected for the last two hours.” This conclusion can be based on the lack of events like opening the refrigerator, turning on TV, etc. (assuming the appliances are equipped with respective sensors and wireless connectivity to the phone). The OSA/Parlay application is run by a home nursing care provider. The application can choose, according to some possibly complex criteria, an available nurse, and make a call connecting her to the patient.

4.4 Nation-wide environmental monitoring

Another opportunity can be exploited by employing a big number of terminals and make them participate in a centrally-supervised project. The idea resembles the Internet-based SETI@Home initiative, in which thousands of users share their personal computers to achieve a common goal. Specifically, the Internet MIDlet-to-App pattern can be used for applications producing all kinds of nation-wide maps showing geographical distribution of some quantity (e.g., pollution, allergens, water levels in flood situations) in real-time. What a terminal contributes in this case is a locally acquired value of the quantity being monitored, not computing power (although some MIDlet-based preprocessing may be useful). A MIDlet can use sensors; however, in the basic version, it can just query the user: “What is the temperature outside your home?” Once the data is submitted to an OSA/Parlay application, it adds a location to it (using the Mobility SCF). All the location-stamped data items are then collected, and a nation-wide map is produced.

5. Direct MIDlet-to-App

In this pattern, the MIDlet and the application use wireless messaging to communicate. A MIDlet can send and receive SMS messages using the Wireless Messaging API, WMA [17]. At the OSA/Parlay application’s side, one can employ the Generic Messaging SCF.

Actually, an SMS message sent to a MIDlet requires an application port number [18] to be specified. Here we find an inconsistence: a commonly used E.164 address plan does not support port addressing. That is why another address plan has to be used. Our suggestion is to use the URL address plan and specify the address as follows: “sms://<address>:<port>,” which is compatible with both WMA and Parlay. We strongly suggest that the port addressing issue be clarified in the next version of the OSA/Parlay specification.

5.1 MIDlet-originated messaging

MIDlet-initiated messaging can be employed, for instance, to configure settings and preferences. If the amount of data is low, this method can also be used in ubiquitous computing and nation-wide environmental monitoring applications described above.

5.2 Application-originated messaging

In this subsection we assume that the OSA/Parlay application can indeed specify a port when sending a message.

The OSA/Parlay application can send a message to a MIDlet even if the MIDlet is not running. This can be achieved with the MIDP 2.0 Push Registry mechanism. When an appropriate registration entry is present in the
Push Registry, the terminal’s Java system launches a MIDlet upon receiving an SMS directed to a specific port. This is a powerful mechanism enabling the OSA/Parlay application to initiate a terminal-based activity.

Here is an example of application-originated messaging. The application can instruct the MIDlet what action to take. For instance, an “electronic guide” (an OSA/Parlay application) periodically determines the user location. If a location changes, the application sends an SMS to a MIDlet, instructing it which part of the map of the city should be displayed at the terminal. Since the map data have to be present in advance, the service can be divided into two phases: (1) offline, when the content data is downloaded to the phone (for instance through a Bluetooth connection at the airport) and (2) online, when the relevant information is displayed (activated) by an incoming SMS. This approach, which we call, activation of offline content, saves much bandwidth, even if the content takes much space (an activation message is very brief).

6. Internet MIDlet-to-Gateway

In this case the terminal does not cooperate with the OSA/Parlay application but with an OSA/Parlay gateway (specifically, an SCF implementation).

Terminal Capabilities SCF is responsible for giving the OSA/Parlay application the capabilities of the terminal. Currently, a so-called UAProf [19] description is used. Apparently, Terminal Capabilities SCF has not been widely implemented so far. One way to do it is to have a MIDlet that reports terminal capabilities to an OSA/Parlay gateway, according to the Internet MIDlet-to-Gateway pattern. Depending on the type of a capability, the MIDlet can find out about it (1) through J2ME system calls, (2) by asking the user to manually provide the information, or (3) by wirelessly (e.g., using Bluetooth) detecting other devices present in the neighborhood (this way a terminal can become a complex “super-device”).

7. Parlay X MIDlet

One way to enable a MIDlet to access telecom network functionality (as exposed by the OSA/Parlay APIs) is to implement an OSA/Parlay Proxy application, as described above. A drawback of that approach is the need to use a proprietary MIDlet-to-Proxy protocol. Another, more direct and open way to achieve the same goal is to use Parlay X and make the MIDlet itself a Parlay X application (we call it a Parlay X MIDlet). With the current level of resources (especially memory) available in high-end mobile phones, a MIDlet can handle the Web services protocols. Due to potential problems with preserving the IP address of the terminal (maintaining a GPRS session), the preferred Parlay X APIs are those, for which the MIDlet acts as a Web services client, and the gateway acts as a Web services server. Fortunately, most Parlay X APIs fall into this category.

A prospective application area is work management for small teams of mobile workers. The Parlay X MIDlet runs on the team manager’s mobile phone and automates some management tasks. For example, a new service request is received from a customer. The manager enters the customer’s address to the MIDlet. The MIDlet, based on the current locations and availability of the workers, picks the most appropriate one and makes a call connecting him to the customer.

A major advantage of the Parlay X MIDlet architecture seems to be the ease of deployment. A small company could run a Parlay X application streamlining its business processes, with little more than a higher-end mobile phone and a contract with a gateway operator. Also, one can envision a market for inexpensive, off-the-shelf Parlay X MIDlets.

8. MobileDate revisited

MobileDate can become much more attractive and easy to use if enhanced with a MIDlet at the terminal. One more general remark can be made here before proceeding. Some OSA/Parlay applications are not aware of the existence of cooperating MIDlets, while others are tightly coupled with them. We call the former MIDlet-neutral, and the latter MIDlet-aware. For example, MobileDate, as described in Section 2, is MIDlet-neutral. A MIDlet-neutral application can be more widely deployed, no matter whether the terminal is J2ME-enabled. On the other hand, MIDlet-aware applications can offer more advanced functionality. We begin with enhancements that allow MobileDate to remain MIDlet-neutral.

One of them is a MIDlet that replaces the awkward SMS-based user interface. In the current version of MobileDate, the user has to compose SMS messages manually, using a special syntax. Also, for the sake of simplicity, the SMS-based interface allows only a subset of operations possible with the Web interface. With a MIDlet’s GUI, the user will be able to manage the service with the phone almost as easily as through the Web page.

The MIDlet can use its storage to save all relevant MobileDate information (a user identifier list, names, comments, photos, and a database of received messages) and allow the user to retrieve it later.

A MIDlet can compose an MMS message in a special MobileDate format. As more terminals are released that
have a camera in them, the MMS can include a current picture of the user, along with some MobileDate-specific data. The OSA/Parlay application can then forward it to the other user, just like an SMS. The message is then interpreted by the peer MIDlet in the other user’s terminal.

Now we turn to enhancements that require MobileDate to become MIDlet-aware. Assume MobileDate prepares and sends MIDlet-interpreted SMS or MMS messages. (Using Push Registry, MIDlet execution could be triggered by an incoming message notifying that a compatible user is nearby.) A message can be processed by the MIDlet in a number of different ways, depending on the type of its content. The MIDlet can, e.g., (1) play a ring tone that depends on the “compatibility level” of the nearby user or the distance to her, (2) display a picture of the nearby user along with a map showing her current location (the location could be kept up to date by subsequent messages), (3) initiate an instant messaging session. If a substantial amount of data has to be available to the MIDlet, activation of offline content can be used.

Here is another example of MobileDate-to-MIDlet messaging. In the current version, the MobileDate activity is proximity-triggered. However, some users may be interested in the “density” of matching parties in a big area (e.g., a section of a city). The radius of the area can be selected by the user and a request can be sent to the OSA/Parlay application using a MIDlet. The application responds with a message containing coordinates of high density areas. The MIDlet makes it possible to browse through the city map with high-density, “romantic” areas highlighted.

9. Conclusions

A number of OSA/Parlay and J2ME in tandem architectures and their applications are presented. Clearly, significant synergies can be achieved, if the two technologies are allowed to work together.

The above ideas point to a possible improvement of the existing OSA/Parlay simulators. Currently, such tools include a model of the user's terminal. However, in most cases, the terminal is relatively “dumb.” In order to enable the development of OSA/Parlay and J2ME in tandem services, the terminal should be Java-enabled.

The problem of OSA/Parlay-initiated port-addressed SMS messages is identified and a possible specification improvement is suggested.

Acknowledgements

The authors would like to gratefully acknowledge the contribution of Derek Edwards, Ian Lucas and Tim Tobin of Open API Solutions.

References

2. J. Zuidweg, Next Generation Intelligent Networks, Artech House, 2002
4. M. Rój, J. Domaszwicz, Simple Service Developed with Ericsson OSA/Parlay Simulator, MOST (Mobile Open Society through Wireless Telecommunications), 2002
6. JSR-118 Expert Group, Mobile Information Device Profile v2.0, Java Community Process, 2000
11. JSR-172 Expert Group, J2ME Web Services, Java Community Process, 2004
15. JSR-82 Expert Group, Java APIs for Bluetooth, Java Community Process, 2002
19. Open Mobile Alliance, User Agent Profile ver. 1.1, http://www.openmobilealliance.org/tech/docs/