Developing a Framework for Mobile Positioning Client on PDA Platform

Alisa Devlić, Alan Graf and Darko Huljenić*
Department of Telecommunications, Faculty of Electrical Engineering & Computing
Unska 3, 10000 Zagreb, Croatia
E-mail: alice@fly.srk.fer.hr, Alan.Graf@inet.hr
Ericsson Nikola Tesla d.d*
Krapinska 45, 10000 Zagreb, Croatia
Phone: (385) 1 365 4548 E-mail: darko.huljenic@etk.ericsson.se

Abstract – Location based services are the most promising group of telecommunications services considered for development in nearest future. With new generations of mobile phones and their integration with PDAs, and better capabilities of mobile location technology provide introducing a new applications and services into exsisting and future mobile network architecture. This article deals with solutions concerning development of mobile positioning client on PDA platform. The application development framework using Java 2 Micro Edition (J2ME) is proposed. Application testing results and practical usability in the real environment are analysed.

I. INTRODUCTION

Location Based Service or LBS, is the ability to find the geographical location of the mobile device and provide services based on this location information. This ability to provide the user a customized service depending upon his geographical location could be used by telecommunication companies to ensure service platform for different content providers (for example restaurant owners, emergency services, etc.). The industry seems to agree that a position solution adheres to the service network architecture, where applications can access a positioning API that a service capability server provides. It means that the application developer would not need to know the details about the positioning method that is used (or even about the mobile system). The Mobile Positioning protocol (MPP) provides such an API and is already used by most of operators who offer LBS.

Mobile positioning client access position information from the Mobile Positioning Center (MPC) using MPP. The MPC works like a service enabler and executes positioning for the client providing him with a positioning result. The client should be lightweight application and all the processing should be executed on the server side.

The paper is organized as follows: basic LBS architecture and main development issue are described in Section 2, integrated development environment is defined in Section 3, while Section 4 deals with requirements and development of mobile positioning client for Palm OS. Section 5 concludes the paper.

II. LBS ARCHITECTURE

Mobile Positioning System (MPS) [1] fig.1 consists of:

- Mobile Positioning Centre (MPC) that functions as positioning server
- Software extensions for the mobile network

It does not require any modifications to standard mobile stations and terminals, so it is suitable for a wide range of location based services. MPC works as a positioning gateway that enables applications to access Mobile Station (MS) position information. Using such concept MPC hides an underlying technology which it uses when retrieving the position of MS. MPC is designed to be compatible with existing and future positioning methods, both GSM radio and GPS satellite and Assisted GPS (A-GPS). The great benefit is that the applications are independent of the used location technology. In this paper we will not discuss about details of each positioning method (see [2]). Cell Global Identity + Time Advance (CGI+TA) method is used in MPS Software Development Kit.

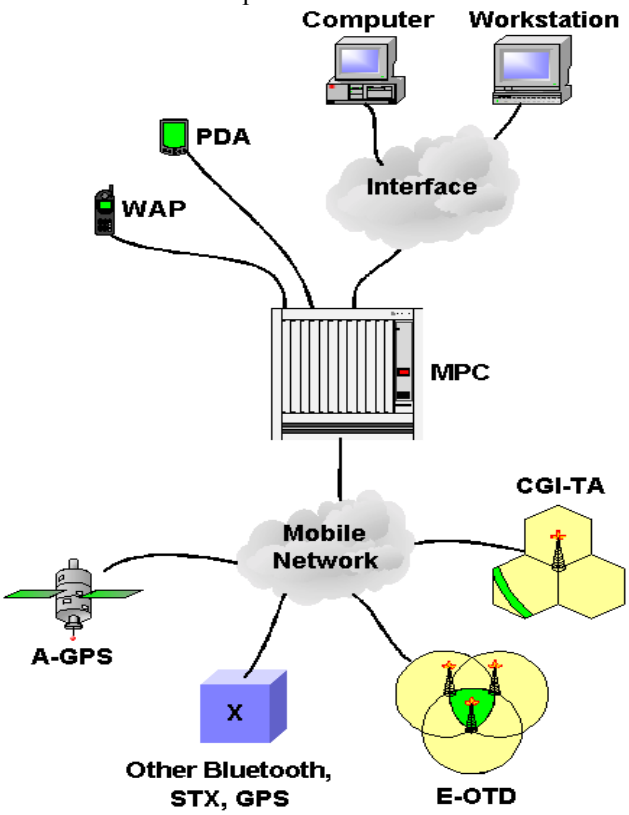

Fig. 1. Concept of Mobile Positioning System

The communication between an application and the MPC relies on the Mobile Positioning Protocol (MPP). MPP is an application-level protocol for GSM positioning. It is implemented on top of HTTP, thus making the MPC

available from any platform with TCP/IP capabilities. HTTP is a request/response type of protocol involving a server and a client. In this context the client is referred to as the Location Service Client (LCS client) and the server as the MPC. The MPP defines an URL that positioning applications use to request the position of MS. As a response to the positioning request, MPC delivers an answer telling the positioning application where the MS is located. This information may be sensitive and MPC supports HTTP over SSL ensuring a secure transfer of location information as well as user data. MPC supports both SSL version 2 and 3. User privacy is arranged in a way that only authenticated users can access location information.

III. INTEGRATED DEVELOPMENT ENVIROMENT

Mobile Positioning System Software Development Kit (MPSSDK) [3] is environment that contains the tools needed for development, demonstrating and testing positioning applications. It consists of:

- MPC Emulator – a servlet based engine running on a Web server used for testing applications
- Map Tool – creates simulated mobile network based on scanned or vectorized map of area. The size of the cell varies from less than100 meters in dense urban up to several kilometers in rural areas.
- Java class library – used for easy developing applications with little or no knowledge about protocols involved
- Demo program showing its basic functionalities, and User Guides Protocol documentation that includes instructions on how to use Emulator, MPC Map Tool and Java classes, fundamentals of mobile networks and MPP specification.

MpsSdkDemo is a demo program of MPSSDK and for its execution requires Apache Server and Jserv running [4]. First, we import a digitized map into MPC Map Tool and create a cell pattern of area based on type of particular part of area, imported roads and users' routes. There are two types of cells: CircleSector and OmniSector.

Finally MPC Map Tool generates text (txt) files:

- AuthFile.txt – contains authentification parameters necessary for successful positioning request: name of LCS client, password and MSISDN.
- Routfile.txt – contains routes for mobile stations. Route is presented as virtual path of mobile station through particular area covered by cells described in CellData.txt. MS has a simulated movement along the route depending on the system time providing its position that is constantly changing, and the distance between MS and base station with maximum value of radius of cell.
- CellData.txt – contains data about cells: Cell ID, longitude, latitude, cell type, radius and direction in degrees.
- Confdata.txt – contains some parameters providing more realistic network conditions: delay time, max MS to position, maximum delay per request etc.

MpSSdkDemo sends positioning request to MPC Emulator that based on txt files, determines the location of the mobile user and sends it back as the positioning result.

IV. MOBILE POSITIONING CLIENT FOR PALM OS

In this section we will cover the fundamentals of designing and implementing a mobile positioning client for Palm OS that is currently under development in Mobility Lab at FER. When developing an application there are some necessary steps to consider:

A. *Requirements Analysis*

Requirements Analysis is one of the most important parts of the entire project. It is also one of the most common reasons why most software projects fail. From the business perspective, poor requirements analysis usually leads to lack of functionality, or in the worst case building the wrong system. Here we will discuss various types of requirements in terms of creating our application.

There are few different categories of requirements:

- Functional requirements
  - User must provide his MSISDN to ensure successful positioning
  - The User must be able to provide his username and password to ensure privacy of the data
  - Application has to respond with approximate user longitude and latitude as a result of positioning
- System requirements
  - The application has to run on a Palm OS device
  - 33 MHz Processor
  - 8 MB of memory
  - 160 x 160 pixel display
  - 6 continuous hours of battery life
  - GSM Springboard module with 9600 baud modem
- Performance requirements
  - The positioning request and reply has to be as simple as possible to keep communication at minimum due to the slow wireless modem
  - Most processing has to be executed on the server side, and the result should be then passed to the client.
  - Attempt the use of GPRS modem to reduce the costs of the time spent online and faster download speeds.

B. *Design using UML*

Today UML is becoming de facto standard for developing applications and complex systems. It is also one of the Best Practices [5] of software development to model visually. Visual models provide the much needed simplification as well as help to share the vision of the entire project among the members of the development team.

There are two main categories of UML diagrams:

- Structural diagrams – capture the structure of the application or system under development, and are particularly useful for discovering the dependencies between components.
- Behavioral diagrams – used to visualize, specify, construct and document dynamic aspects of the system.

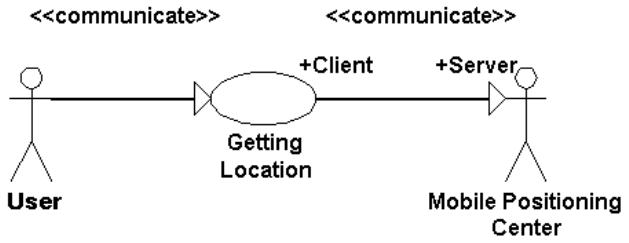

Fig. 2. Use Case diagram

For the purposes of developing our application we used several types of diagrams. In the beginning it was necessary to state the system functionality, and therefore we created a simple Use-Case diagram as seen on Fig. 2. It explains the basic user interaction with the system giving a brief insight into the problem space. From the diagram it is obvious that our application should be developed as a client application to be used in conjunction with MPC emulator, that comes as the part of MPSSDK. To give a basic idea of the application's behavior, a Statechart diagram shown in Fig. 3. was created, showing the Use-Case realization in form of states and transitions.

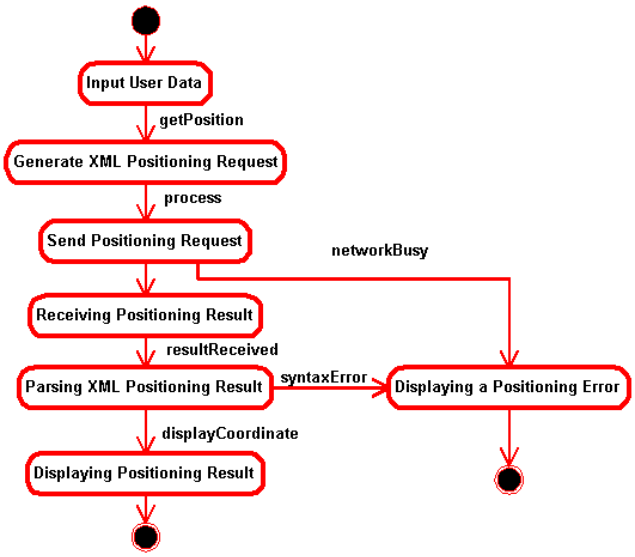

Fig. 3. Statechart diagram

After inputting his mobile phone number, username and password, user initiates the getPosition procedure. Then the application generates positioning request, forwards it to server running an MPC emulator, and receives a positioning result that is displayed on the device display. In case of an error, an error message is displayed to notify the user that his positioning request was unsuccessful. For the purposes of implementation, a more detailed approach was required. The Sequence diagram shown in Fig.4 represents the application at runtime. After starting an application the graphical user interface is displayed. User inputs his data into the form, and pushes the getPosition button. The application then stores user data, and calls the method for generating positioning request in form of XML document as specified in [1]. After receiving the string containing XML request, the gui calls a method for processing the request that opens the HTTP connection, sends request via the POST method, receives the result and returns it as a string containing the XML positioning result. For the purposes of extracting user's Coordinate a kXML parser was used, which will be explained in more detail in the next chapter. After the successful reception, the coordinate is displayed on the device screen in World Geodetic System 1984 (WGS-84) format [10].

C. *Implementation in J2ME*

J2ME provides a standard platform for developing wireless devices services [6]. For the profile of *low-end consumer devices*, such as cell phones, two-way pagers and personal organizers Java Community Process (JCP) has developed *Connected Limited Device Configuration* (CLDC) and *Mobile Information Device Profile* (MIDP). These devices have very simple user interfaces, minimum memory budgets starting at about 128 kilobytes, and low bandwidth, intermittent network connections. CLDC is base technology that defines the core virtual machine features and libraries that all small Java technology powered devices will share. Core component of J2ME is KVM (Kilobyte Virtual Machine). It is compatible with Java Virtual Machine, but with some restrictions: there is no floating point support on CLDC devices, libraries in CLDC are limited and it can't afford to use J2SE security model. MIDP is running on the top of CLDC. A MIDP application is called a MIDlet. A MIDlet is a class that extends the class javax.microedition.midlet.MIDlet. Class javax.microedition.midlet.MIDlet defines three abstract methods: startApp, pauseApp and destroyApp, which must be defined by all MIDlets. During the lifetime of a MIDlet (Fig.5) it may be in one of three distinct states, with well defined rules that govern the transitions between these states: Paused, Active and Destroyed.

The central abstraction of the MIDP user interface is the *screen*. Each MIDP application has a *Display* on which a single screen is shown.

There are two types of APIs:

- High-level, portable API (Screen) – high level objects that encapsulate a complete user interface component (classes Alert, List, TextBox or Form). Applications which use this API should work on all devices. It has no direct access to device features (color, sizes, input).

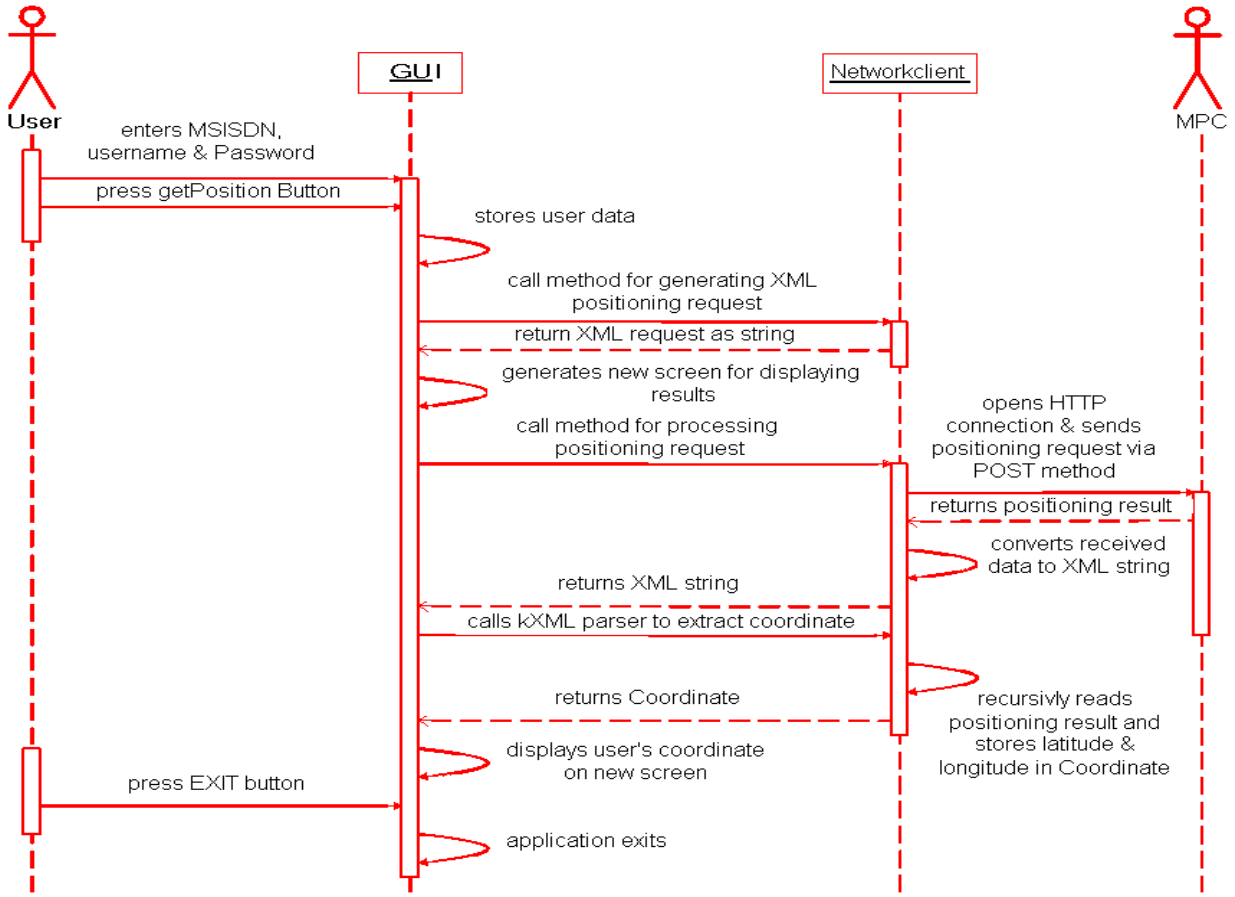

Fig. 4. Sequence diagram

- Low-level API (Canvas) – low-level objects that allow the application to provide the graphics and handle input. Developers may compromise portability for better user experience.

The user interface of client is screen oriented and is shown on a fig.6. MIDP UI elements: Form, TextField, Ticker and Command are used [7].

The MIDlet has a well-defined life cycle:

- Retrieved from server to device
- Installed on device
- Run on device (paused, active, destroyed states)
- Upgraded with new version
- Removed by user

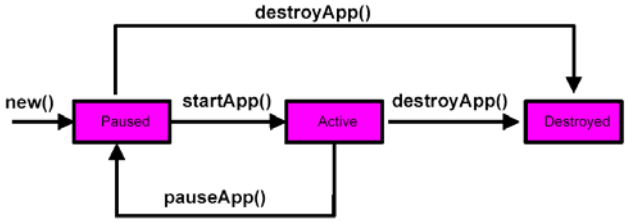

Fig. 5. MIDlet's lifecycle

Device-specific application management software performs lifecycle operations. MIDP defines mechanisms which support the lifecycle operations: Jar and Jad file formats, MIDlet descriptors, javax.microedition.midlet package API. Midlet class files and resources are deployed in JAR files. Each MIDlet jar file may be accompanied by an application descriptor. MIDlet descriptors are stored in jad files. They allow application management software to verify that the MIDlet suite is suitable before downloading. They also provide configuration-specific attributes and packaging multiple MIDlets into MIDlet Suite.

MIDP networking extends CLDC connectivity support. It supports a subset of HTTP protocol, which can be implemented using IP protocols such as TCP/IP. Non-IP protocols (WAP and I-mode) can also be implemented using a Gateway to HTTP access. HttpConnection defines the necessary methods and constants for a HTTP connection. Connection exist in one of following states:

- Setup – Connection has not been made to the server. The methods setRequestMethod() and setRequestProperty() have been called
- Connected – Connection has been made. Request parameters have been set. Response is expected.
- Closed – Connection has been closed. Methods, if called, will throw IOException.

MIDP networking allows access to data formats such as XML, WML, etc. XML parsing is a heavy string manipulation and it has to happen on MIDP devices. KXML is a XML parser written exclusively for J2ME/CLDC/MIDP which supports SAX (Simple API for XML) [8] and DOM (Document Object Model) parsing [9]. XML data (positioning result) is provided by the

servlet in MPC Emulator which is known by URL. In our case that is: http://mulder.tel.fer.hr:80/newRequest. XML data is obtained as streaming data (in bytes) and has to be converted to String for further processing. SAX parsing (event driven) is done for a project.

D. Application development output

After starting an application on Palm OS Emulator, the user interface containing the form for inputting user data is displayed as shown on Fig.6. When the user has entered his phone number, username and password, he initiates the positioning procedure by pressing on the Position button.

Fig. 6. Positioning request on Palm OS Emulator

Upon successful positioning the result displayed in the new screen as shown on fig.7. representing user's location in form of coordinate coded in WGS-84 format.

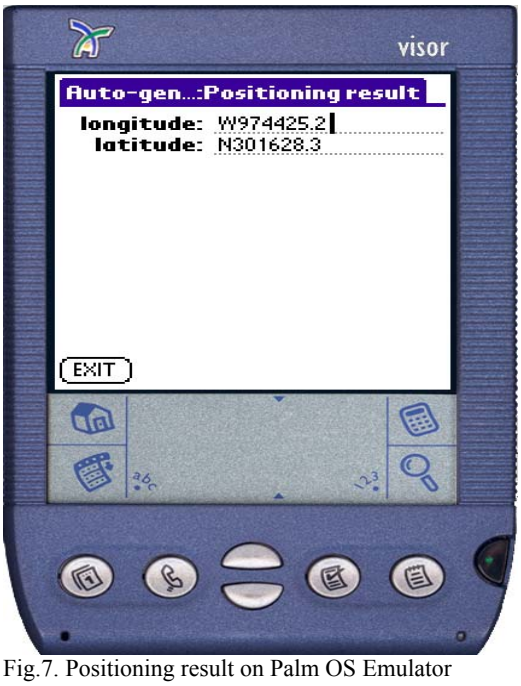

Fig.7. Positioning result on Palm OS Emulator

In any moment user can exit the application by pressing EXIT button.

This application could be a base for upgrading with access to GIS and/or other databases in order to create more complex LBS such as finding the closest restaurant, hotel, or even another mobile user. This development is used for preparing framework with basic elements for variety of prospective applications.

V. CONCLUSION

Developing a framework for mobile positioning client on PDA platform was inspired by availability of J2ME development environment and increasing number of MIDP applications on small communication devices, limited by memory and processing power and with the support for HTTP. With the rapid development of next generation networks and end-user devices as well as location aware computing, it is becoming obvious, that the location based services will have one of the leading roles in the mobile communications of the future.

VI. ACKNOWLEDGEMENT

The paper represents results of R&D work started during Summer Camp organized by Ericsson Nikola Tesla and Faculty of Electrical Engineering and Computing. The discussed solution was developed and implemented in Mobility Lab, Department of Telecommunications, Faculty of Electrical Engineering and Computing, and tested on Palm OS Emulator and Handspring Visor Prism with VisorPhone module.

VII. REFERENCES

[1] Ericsson's MPC Emulator and Mobile Positioning Protocol User's guide, http://www.ericsson.se/developerszone
[2] C.Andersson, "GPRS and 3G Wireless Applications", Wiley, 2001.
[3] Detailed information about the Mobile Positioning System, http://www.ericsson.se/mps/
[4] A. Devlic and A. Graf, "Application based on Mobile Positioning", Ericsson/FER Summer Research Camp, 2001.
[5] T. Quatrani, "Visual modeling with rational rose and UML", Addison-Wesley, 1997.
[6] S.Raju, "Java programming for Wireless devices using J2ME – CLDC/MIDP APIs", Sun Microsystems, http://www.microjava.com
[7] R. Riggs, A. Tailvasaari and M. VandenBrink, "Programming Wireless Devices with the Java 2 Platform, Micro Edition", Addison-Wesley, Boston, 2001.
[8] About SAX, http://www.saxproject.org/
[9] Document Object Model (DOM) Level 1 Specification, http://www.w3org/TR/REC-DOM-Level-1/
[10] World Geodetic System 1984, http://www.wgs-84.com/