Université de Montréal

A versatile location-based service Java control for Pocket-pc.

par

Paul Bertrand

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Version courte du rapport de stage présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès Sciences (M.Sc.)
en informatique

Août, 2008

Université de Montréal
Faculté des études supérieures

Ce rapport de stage intitulé :

A versatile location-based service Java control for Pocket-pc.

présenté par :
Paul Bertrand

a été évalué par un jury composé des personnes suivantes :

POULIN, Pierre (Président-rapporteur)
GUEHENEUC, Yann-Gaël (Directeur de recherche)
HAFID, Abdelhakim (Membre du jury)

Abstract

With the democratization of Pocket-pc devices and their constant improvement in terms of performances, more and more companies equip their field technicians with such devices. A concrete application involves technicians who can use Pocket-pc to consult and organize their daily tasks synchronized with a central system. Along with its backend system (Netweaver), SAP also provides Pocket-pc software written in Java. As most of these technicians need to meet customers and go on sites, these systems need to be shipped with an embedded mapping control that must support any kind of mapping service independently from the technology used by this latter. The aim of this work is to propose a versatile location-based service Java control relying on a unique abstract interface that gives the possibility to adapt any kind of mapping service and to satisfy SAP mobile software development constraints. The proof of concept has been validated by implementing two services using different technologies (Google maps using AJAX and Microsoft Mappoint using Web services) and tested on two different devices: a Compaq Ipaq and a Fujitsu Siemens Pocket LOOX. The results obtained were very encouraging and conclusive enough to ship this control with SAP Pocket-pc systems. In conclusion, this project brings a concrete and flexible solution to mapping controls that can be integrated within any Java application running on Pocket-pc devices.

Keywords: LBS, Pocket-pc, SAP, Microsoft Mappoint, Google Maps, versatile Java control, strategy design pattern, multiple GIS support.

Résumé

Grâce à la démocratisation des appareils mobiles ainsi qu'à leur constante amélioration, nombreuses sont les compagnies qui fournissent à leurs techniciens des appareils mobiles de type Pocket-pc pour organiser leur travail et rester en contact avec leur entreprise. Grâce aux Pocket-pc et à la qualité des réseaux sans fils, les techniciens peuvent en tout temps rester connectés au système central de leur compagnie pour consulter, synchroniser et mettre à jour leurs tâches via des logiciels embarqués. SAP fournit en plus de son système central Netweaver, des logiciels écrits en Java destinés aux Pocket-pc capables de consulter, synchroniser et mettre à jour les informations contenues sur le système central. Sachant que la majorité des techniciens se déplacent chez des clients, ces logiciels doivent fournir une fonction de géo-localisation intégrée pour faciliter les déplacements. Le but de ce travail est de proposer un control Java de géo-localisation capable de supporter n'importe quel système d'information géographique (SIG) et d'être intégré dans n'importe quel logiciel SAP écrit en Java et destiné aux Pocket-pc. La validation a été réalisée via l'implémentation et le déploiement du contrôle sur deux Pocket-pc : un Compaq Ipaq et un Fujitsu Siemens Pocket LOOX. Le contrôle final peut utiliser deux SIG différents : Microsoft Mappoint, accessible depuis des services Web et Google Maps fourni via une API AJAX. Les résultats obtenus sont très encourageants et suffisants pour pouvoir considérer l'intégration du contrôle au sein des logiciels SAP Pocket-pc écrits en Java. En conclusion, ce travail apporte une solution prouvée concrète, flexible, réutilisable et modifiable aux contrôles Java pour Pocket-PC dédiés à la géo-localisation.

Mots-clés : LBS, Pocket-pc, SAP, Microsoft Mappoint, Google Maps, contrôle Java versatile, géo-localisation, patron de conception stratégie, support de multiples SIG.

Table of Content

List of Figures

Chapter 1

Introduction

For the last 10 years, continuous improvements of wireless networks have encouraged more and more people to adopt mobile devices. Each day, they carry billions of transactions and messages. With time, these networks get more secure, cover more area, transport more data and therefore bring more connectivity among people. Wireless networks can be accessed and used by a wide diversity of devices including mobile devices such as Smartphone, PDA, Pocket-pc, and Blackberry. Just like wireless networks, these devices get more powerful every year and consequently offer more functionality. People can now, at any time and almost anywhere, consult their emails, browse Web sites, take photos, record videos, use a GPS system, create documents, and much more. Industries and companies naturally also acknowledged mobile technologies assets and their adoption significantly increased companies' productivity. For instance, a survey conducted by Nucleus Research showed that for 230 users, "the average mobile device user works 15 percent more each day" [18]. Because we are living in a fast moving economy, information needs to constantly travel in and outside a company. Operations must be done in real time and employees shall remain inter-connected to ensure proper information propagation. SAP provides applications and solutions for companies to harmonize, unify, and facilitate information exchange through mobile devices. Our work focuses on one of these applications called User Centric Design (UCD) and more specifically on its Geographical Information System control (GIS) responsible for providing Location-based Service (LBS). UCD is a Java project that has been initiated in May 2007 and aims at delivering efficient data display for on-field technicians using Pocket-pc devices. This application consists of a work-order list from which the technician can

access various kinds of information related to each work-order: customer, geographical information (given through the LBS control) such as location of the customer, spare parts needed, work history, job priority, etc. This application is embedded on a Pocket-pc and connects to a backend system to fetch and send data.

UCD stands on top of SAP Service Composite (see figure 1.1). Service Composite enables SAP customers to define which SAP components they want to use and by composing them together, they are free to organize, sort, manage and prioritize data.

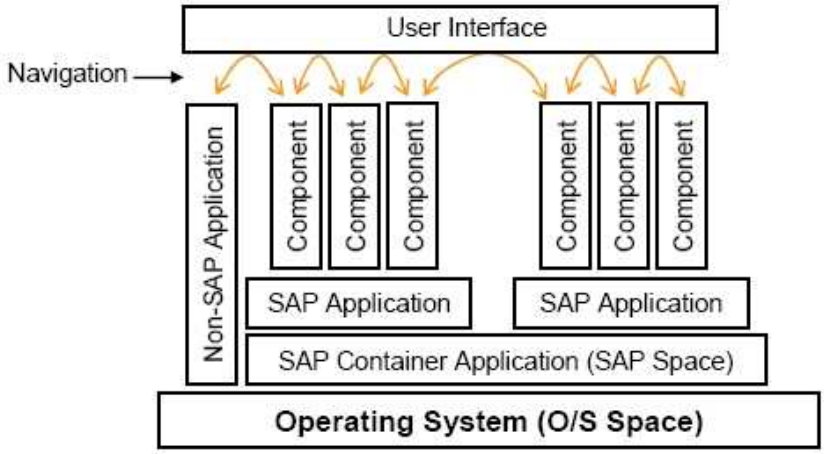

Figure 1. 1. Service composite architecture.

The data retrieved from Service Composite is displayed on the Pocket-pc using the View Inspect and Act concept (VIA). VIA is a user interface design pattern specially created for mobile devices. Each screen of the application either belongs to a V, I or A pattern:

1. View: each screen labeled as a V pattern is used to provide general information. For example, the work-order list we mentioned before is defined as V pattern.
2. Inspect: each screen labeled as an I pattern is used to provide more detailed information. For example, a typical I screen is the one displayed after the user

has clicked on a work-order from the list. There, he can find more information related to the customer, the task, etc.

3. Act: each screen labeled as an A pattern is used to record inputs from a user. For example, the signature capture screen is an A pattern.

These screens and the information they contain are arranged as a stack structure that reflects the granularity of the information (see figures 1.2 and 1.3).

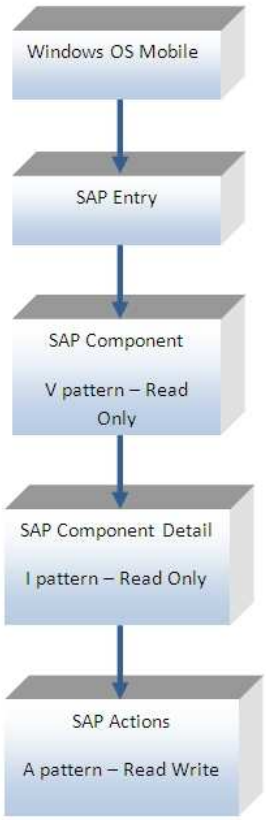

Figure 1. 2. UCD stack architecture.

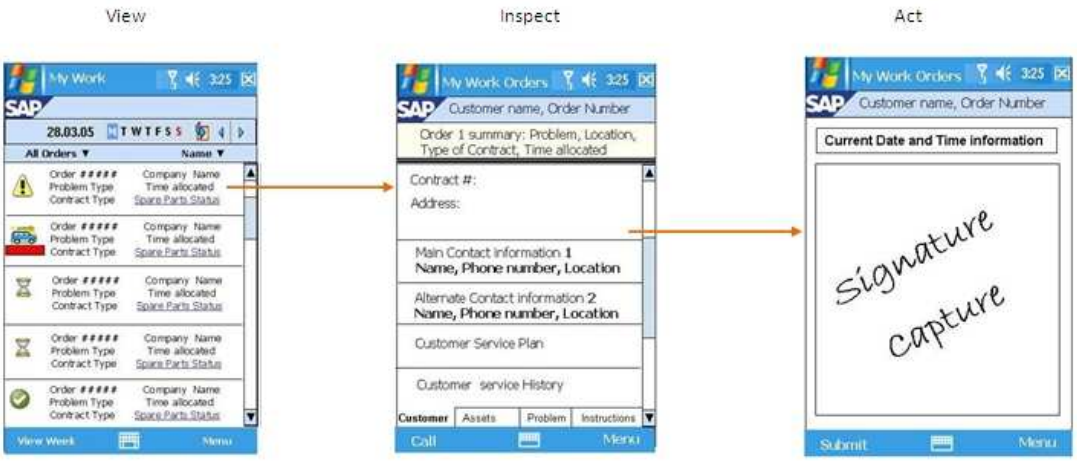

Figure 1. 3.UCD display and workflow example on Pocket-pc.

Stack arrangements are pretty standard for Pocket-pc as they cannot display more than one window at a time.

Mapping solutions recently gained a tremendous popularity as they have been made accessible by different providers (Google, Microsoft, Yahoo, and others) for free to anyone who can access the Internet. These solutions offer a rich set of functionalities such as multiple visualization capacities (satellite, terrain, and map), user friendly interfaces, address finding, business finding, driving directions, etc. In addition, a large majority of these mapping providers opened their functionalities to developers to build custom mapping applications. However, the way to access these functionalities greatly differs among providers. For instance, Mappoint is being offered through Web services while Google uses AJAX. Both have very different implementations but share common purposes. Web services are remote function calls that can be consumed from a wide diversity of programming languages (C++, Java, Python, Ruby, etc.). On the other side, AJAX (Asynchronous JavaScript and XML) also enables remote function calling but is limited to Web browsers as it requires a JavaScript environment.

This work consists in designing a mapping Java control integrated within UCD (as an ‘I’ pattern) that can adapt to any GIS provider independently from the interface it uses (Web Services, AJAX, RMI) and capable of LBS tasks such as routing and address finding. By enabling our control with any GIS provider, SAP future customers will be given the choice rather than being forced to use a particular mapping solution (most of them come with a license). None of the mapping solutions proposed in the literature or on the market offer such flexibility. Some are capable of continuously browsing maps stored on a server but cannot locate addresses while some others embed GPS and LBS functionalities but render maps as spatial data and depend on a specific GIS server. Therefore, our contribution consists in designing and implementing a Java control capable of retrieving, requesting, and displaying information from any GIS provider. Our proof of concept consists in being capable of finding any location-based on a geographical address from two different GIS providers and display the map on the Pocket-pc.

This work is presented through three parts. The first one deals with the requirements and exposes the problems. The second part presents the previous works, the state of art, the existing products. The third part introduces our solution while the last part consists of the future work and the conclusion.

Chapter 2

Problem: Adapt to Several GIS Providers

Even though Pocket-pc devices are more and more powerful, they still cannot compete with regular desktop stations. Consequently, developing applications for Pocket-pc can be very challenging due to the limited memory, CPU, display, chipsets, and energy. All these limitations are sources of constraints when writing an application. This section covers the business requirements of our work along with issues they may raise. A state of the art is also given covering the existing business solutions and literature.

2.1. Requirements

The following list represents the requirements. It will be referred as the RC list in the remaining sections of the report for further references. It states that the control must:

1. Be written in Java (JDK1.3 compliant) and use eSWT as a graphical API;
2. Be easy to integrate into any other application (for reuse purpose);
3. Be capable of locating customers and display their position on a map;
4. Be capable of zooming in and zooming out;

5. Be capable of finding routes between multiple addresses, display the path on a map, and propose directions;
6. Display interactive icons for locations which provide specific information about the selected location;
7. Let the user drag the map to explore the surroundings;
8. Be capable of finding specific location types around a location such as restaurants or hotels;
9. Support any kind of GIS provider so that customers are given a choice;
10. Be easy to extend in terms of functionalities;
11. Display the maps on a tap and drag component similar to Google Maps, MapQuest and Yahoo Maps. This component must be very responsive to make the touch and feel comfortable;
12. Take into consideration device limitations especially in terms of network access.

Based on these requirements, 6 technical problems have been identified.

1. How can the control request the same information from different GIS providers that use different technologies?
2. Is a Pocket-pc powerful enough to consume Web services?
3. Should we consider continuous map browsing? If yes, how can it be done?
4. How big should be the map and how far (in terms of geographical distances) should the user be able to navigate on the map?
5. Should we download one big map or several little maps?
6. How can we deal with low connectivity areas?

2.2. Existing Solutions

As this work has been done in SAP Labs Canada, the first investigations have been conducted within SAP internal solutions. Some previous projects such as Mobile Asset Management (MAM) already provided GIS services to end users through a third party software. It is triggered by a command line receiving longitudes and latitudes as parameters. Unfortunately, this solution is unconceivable in our case since we need a Java control we can control. This means that we cannot use any tierce application like Google Maps for Pocket-pc or Microsoft Street Companion. Further researches were conducted on open Source API but there is no such thing as an API capable of dealing with more than one GIS provider. Few resources have been found on forums explaining for example how to access the Google Map API from a Java application or how to create a Midlet[1] giving your instant location.

Lattanzi and Bogliolo [1] conducted a research on the continuous browsing of remote maps from a PDA[2]. They designed a Java application which uses RMI to request specific portions of a map from a unique server. Their study includes multithreading based on process priorities and intelligent algorithms that automatically download the next map based on border detection and minimum threshold. They ran several tests on performance so that the frame rate of the map displacement remains equal to 24 frames per second. Several parameters were involved: image resizing, image compression, parameters of the method that draws the image, minimum threshold, and job scheduling. Although many points of their study crosscut our requirements, their application does not deal with LBS. Nor does it adapt to different GIS providers and protocols.

[1] Java application running on mobile devices equipped with J2ME virtual machine.
[2] The difference between a PDA and a Pocket-pc mostly comes from the fact that a Pocket-pc only runs Microsoft Mobile operating system while a PDA can be shipped with different ones. Design and hardware can be identical.

Chen, et al. [2] proposed an application offering real time GPS navigation using spatial data[3] instead of digital maps. The architecture of their application consists of a mobile device and a service center. Both communicate through a wireless network. The service center (servers) is mainly used to execute complex calculations and other resource consuming tasks. Tasks separation is the key of their solution where each server can be dedicated to a different task such as spatial information service, spatial data transmission, spatial data analysis, spatial data download, and update while mobile device performs visualization, GPS, and projection. The main difference with our project is the fact that they use vector maps. Because they propose real time GPS navigation, they need to retrieve and download maps as fast as possible. However, even if vector maps are much smaller (in terms of kilobytes), they are not as easy to read as digital maps. Furthermore, their application depends on a single GIS provider and does not deliver functionalities such as routing, address retrieval, directions, or proximity business finding.

As none of the previous work offers similar or comparable features to our requirements, a new solution has been created.

[3] Spatial data: vector maps made of polygons, points, and lines.

Chapter 3

Solution: Flexible Location-based Control

The architecture of our control is made of two parts: a tap and drag component responsible for user interactions and rendering; and a GIS side made of one or more GIS solutions. Both communicate through an abstract interface that can be compared to a black box (see figure 3.1).

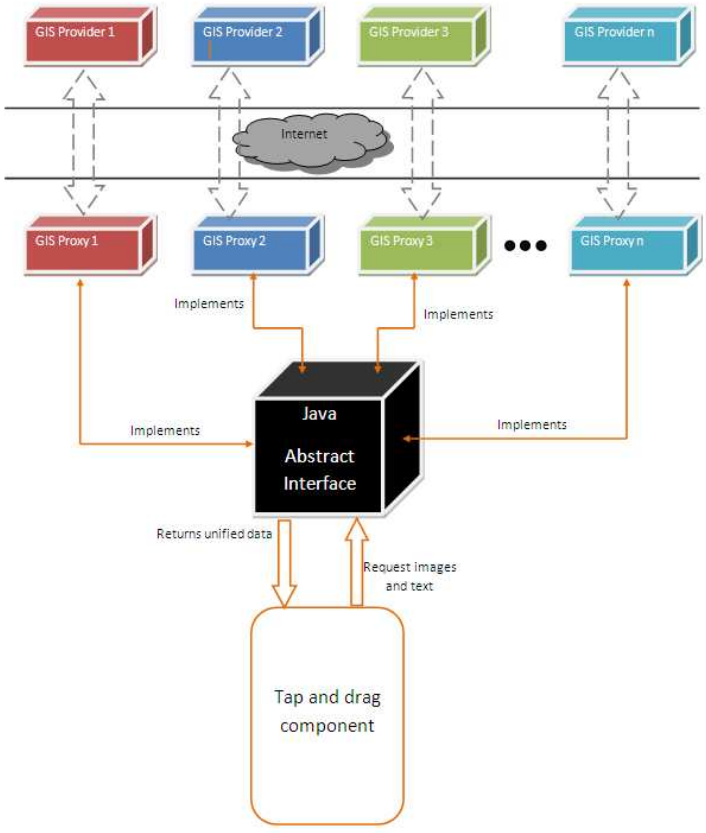

Figure 3. 1. Abstract interface as a black box.

As we can see on the figure 3.1, the interface enables the decoupling between the rendering and the GIS sides. On the lower part of the box, the graphical display initiates requests and renders what comes out from the black box. It does not need to know how data is being retrieved and from which GIS provider they are coming from. On the upper part, there must be at least one local GIS local proxy satisfying the functionalities defined by the interface. This proxy acts as an interface to a specific GIS provider. It is to note that the GIS local proxies, if more than one, cannot be mixed or interact with each others. However, the user can easily switch from one to another through a selection menu. Once the data has been downloaded, the control displays digital images arranged in a grid architecture made of 9 sub-squares acting as a single block when the user drags the map. The user can interact with the map by zooming in and out, investigating further locations thanks to a map expansion mechanism, and look for closest restaurants and hotels. This control relies on a non persistent caching system storing maps (and their related information) identified by a unique set of matrix coordinates. The caching system requests maps and details from a single GIS provider through a unique abstract interface that each GIS local proxy must implement. The validation has been made using two different GIS providers to demonstrate the flexibility and adaptability capacities of our control: Microsoft Mappoint that uses Web services (fully implemented) and Google Maps that uses AJAX (partially implemented).

Chapter 4

Conclusion

This work proposes a reusable Java control for mobile devices capable of adapting to any GIS provider (to the best of our knowledge). Even if results are very encouraging, further investigations and coding would be required.

First, it would be necessary to complete the Google implementation and add more GIS providers to guarantee total flexibility. Along with the development, more functionality could be added to the interface.

Overall performances could be improved by adding a dedicated server to the architecture as Chen et al. [2] did in their study. This server could be responsible to generate the calls to the servers and would communicate with the Pocket-pc device through a simple protocol that we could create. Indeed, test results showed that the major overhead for the application on the Pocket-pc is not the map downloading process but the Web service request that is quite a heavy task for such small amount of resources. Thus, by delegating the hard work to a more powerful machine, the application would run a lot faster (see figure 7.1).

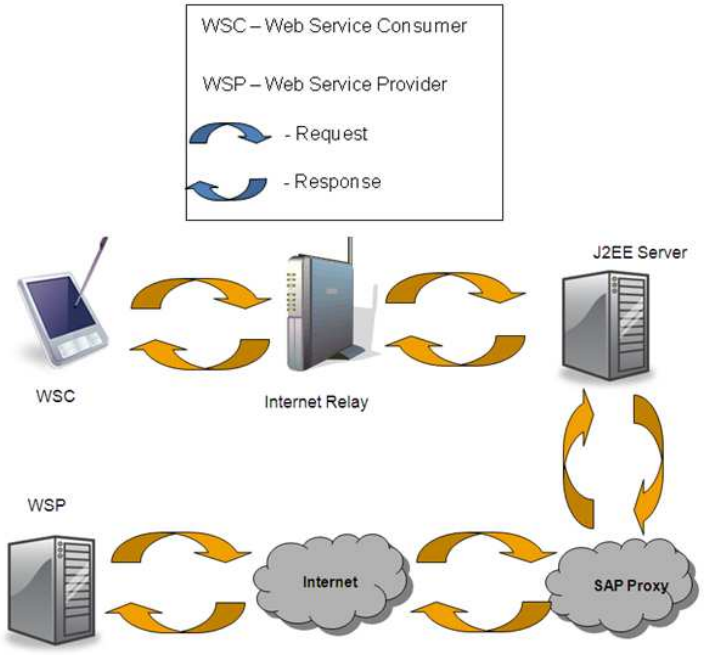

Figure 4. 1. Future architecture.

Regarding the tap and drag component, the performances have been proven satisfactory. However, with the release of future JVM for Pocket-pc, we could implement a continuous drag with no interruption during the map expansion. Indeed, the JVM would integrate non blocking I/O libraries (“nio” for instance). These features could be used to design an intelligent multithreaded algorithm for map retrieval that would give the sensation to the end user that the map is infinite.

The next task could consist in making the caching system persistent. Indeed, knowing that field technicians generally operate in the same region, the maps could be stored and updated on a certain schedule. The persistency would make the already stored maps available for future requests speeding up the process and making it more reliable (less connections). However, this feature would only be feasible for location requests and not route requests as for most of the services, routes are painted directly on the maps meaning that we could not reuse the maps for other requests. On the other hand, networks will keep being improved and will cover more and more areas

delivering high speed rates. Therefore, taking into consideration future network improvements, we decided not to modify the caching system.

Finally, it would be interesting to expand our project to a wider set of devices such as Smartphones and Blackberries. It is believed that very soon, a unique Java platform will be released, common to all mobile devices, abstracting all kinds of hardware in the same environment. This release will enable to distribute applications at a much larger scale ensuring interoperability. As a consequence, our solution could be integrated within any system and within any Java mobile application.

Bibliography

[1] *Java-based continuous browsing of remote maps from a wireless PDA: a feasibility study.* **Lattanzi, Emanuele and Bogliolo, Alessandro.** Urbino, Italy : IEEE Computer Society Press, 29 Aug 2002, Vol. 1. 0-7803-7304-9.

[2] *Research on mobile GIS based on LBS.* **Chen, Feixiang, et al.** Inst. of Remote Sensing Applications, Chinese Acad. of Sci., Beijing, China : IEE Computer Society Press, 2005, Vol. 2. 0-7803-9050-4 .

[3] **Corporate, Sap.** SAP Development Standards And Tools. *SAP Corporate Portal.* [Online] SAP , 2008.

[4] Software development process. *Wikipedia.* [Online] Wikimedia Foundation, Inc., 2007. http://en.wikipedia.org/wiki/Software_development_cycle.

[5] Agile Software development. *wikipedia.* [Online] Wikimedia Foundation, Inc., 2007. http://en.wikipedia.org/wiki/Agile_software_development.

[6] **Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.** *Design Patterns: Elements of Reusable Object-Oriented.* s.l. : Addison-Wesley, 1994. ISBN 0-201-63361-2.

[7] **IBM.** Widget Class. *SWT Documentation.* [Online] IBM, 2005. http://help.eclipse.org/help31/nftopic/org.eclipse.platform.doc.isv/reference/api/org/eclipse/swt/widgets/Widget.html.

[8] —. Control Class. *SWT Documentation.* [Online] IBM, 2005. http://help.eclipse.org/help31/nftopic/org.eclipse.platform.doc.isv/reference/api/org/eclipse/swt/widgets/Control.html.

[9] —. GC Class. *SWT Documentation.* [Online] IBM, 2005. http://help.eclipse.org/help31/nftopic/org.eclipse.platform.doc.isv/reference/api/org/eclipse/swt/graphics/GC.html.

[10] Web Service. *Wikipedia.* [Online] Wikimedia Foundation, Inc. http://en.wikipedia.org/wiki/Web_service.

[11] **Microsoft.** Microsoft Mappoint Web Service. *Microsoft.* [Online] Microsoft. http://www.microsoft.com/mappoint/products/webservice/default.mspx.

[12] —. MSDN Mappoint. *MSDN.* [Online] Microsoft. http://msdn2.microsoft.com/en-us/library/aa286512.aspx.

[13] **Foundation, Apache Software.** Apache AXIS2. *Apache.* [Online] Apache Software Foundation.

[14] MapPoint Multi-platform Assistance Center. *Mappoint Web Service.* [Online] Infusiondev.com. http://go.mappoint.net/mappointmpac/default.aspx?main=article.aspx&id=J20085&anchor=getLocationInfo(net.mappoint.s.mappoint_30.LatLong,%20java.lang.String,%20net.mappoint.s.mappoint_30.GetInfoOptions.

[15] Method Stub . *Wikipedia.* [Online] Wikimedia Foundation, Inc. http://en.wikipedia.org/wiki/Method_stub.

[16] **Microsoft.** Working With Map Views. *msdn.* [Online] Microsoft, 2007. http://msdn2.microsoft.com/en-us/library/ms983235.aspx.

[17] **Google.** Google Map Api Documentation. *Google Map.* [Online] Google, 2007. http://www.google.com/apis/maps/documentation/index.html.

[18] Nucleus. Evaluating the productivity impact of mobile devices. *Nucleus.* [Online] March 2008. http://nucleusresearch.com/index.php?option=com_remository&Itemid=65&func=download&id=808&chk=1de2affb11cdb6813653d48c44a6cbe4&no_html=1. Report I30.