Mobile Commerce Application Development: Implementing Location-aware Information Services

Christos K. Georgiadis
Dept. of Applied Informatics
University of Macedonia
Thessaloniki, GREECE
e-mail: geor@uom.gr

Abstract—Mobile setting’s particularities and limitations are serious. In fact, enhancing the mobile browsing user experience is feasible only if perceptual and contextual considerations are employed. A case study of a context-aware location-based application is designed and implemented in this paper. It is capable to identify and to depict on the map user’s current location, to search and detect routes (and to mark on the map the user’s preferred route while he is in move), to display various user personal points of interest/attractions, along user’s current route, and to edit user’s personal selections regarding attractions. It is a customized application, based on Microsoft MapPoint Web Service technology, since each user receives information which is strictly related to his identity. Users are certainly may benefit from such personalized services and have access only to information of specific context without being annoyed with needless operations.

Keywords—location-based information; context-awareness; Web Services; mobile setting.

I. INTRODUCTION

Increased sophistication of mobile technology makes itself an ideal channel for offering context-aware personalized services to mobile users. Such services actually, give pace to the rapid development of e-commerce conducted with portable devices, more commonly referred to as wireless or mobile commerce, as they are context-specific to each individual and thus, are capable to attract customer’s attention [1]. The interface usability of mobile applications is a critical factor for the acceptance of m-commerce, as a good interface design allows users achieve high performance [2]. Moreover, as 3G/UMTS services roll out, m-commerce is increasingly used to enable content delivery and payment for personalized and location-based services such as image content (maps, photos, etc.), as well as video and audio content, including full length music tracks [3].

In this work, we discuss issues about mobile setting and its relation with personalized and context-specific services. Then, we exploit Microsoft MapPoint Web Service technology to design search processes for addresses, attractions, routes, and creating processes of their respective maps. We implement a mobile commerce information-oriented application scenario to demonstrate some of the major implementation concerns that must be taken under consideration to offer such location-based services.

II. ANALYZING MOBILE SETTING

Typical m-commerce client devices such as cell phones or PDAs are certainly limited in terms of computational power resources, input/output facilities or memory [4]. These constraints elevate the importance of complementary technologies, capable to get the most of the available limited resources. Personalization technology is one of the most prominent ones. The efficiency of m-commerce applications in supporting users with the content they need in the most optimized approach, is considered as the way to holding them and to cultivate their loyalty [5].

Furthermore, the introduction of location-based services will allow users to obtain context-specific information to react upon. This makes mobile devices especially fit for providing time-critical and location-sensitive services. Several authors have stressed this functional value of m-commerce, thereby neglecting social value that can be derived from mobile services. The mobile channel is indeed an extremely personal medium that users carry with them at all times. Consequently, it has become part of the user’s social context and everyday life, influencing all areas of society, such as how employees work and how businesses run, advertising opportunities, personal communications, consumer purchases, location-aware services, information locating and retrieval [6].

On the other side, a key restraining factor concerning context-aware and personalized services is financial: cost issues are raised due to confusing billing schemes and their potential dependency on versatile network connections [7].

The characteristics of the mobile Commerce applications can be appreciated from three different viewpoints: system, environment, and user [8]. From the system’s viewpoint, mobile applications present disadvantages, because they provide a lower level of available system resources. Mobile devices, especially cellular phones, have lower multimedia processing capabilities, inconvenient input/output facilities (smaller screens/keyboards), and lower network connection speeds than desktop computers. However, from the environmental viewpoint there is an uncontested benefit: they enable users to access mobile Internet content anywhere and anytime.

User’s perspective characteristics must be regarded rather differently, because they are to a certain degree consequences of the system and of the environment. In addition, the multi-dimensioned concept of “mobility” influences on them in many ways. Mobile users perform their tasks in terms of place, time and context.
Different terms are used by the research community to describe user’s mobile setting, and their interactions within it, but these converge at the ones described below [9, 10, 11]:

- Spatial mobility denotes mainly the most immediate dimension of mobility, the extensive geographical movement of users.
- Temporal mobility refers to the ability of users for mobile browsing while engaged in a peripheral task.
- Contextual mobility signifies the character of the dynamic conditions in which users employ mobile devices. Users’ actions are intrinsically situated in a particular context that frames and it is framed by the performance of their actions recursively.

Because of their mobility and in correspondence with its dimensions, we distinguish three attributes worth noticing regarding mobile device usage [12]:

- Users have a tendency to treat their mobile device in a quite personal and emotional way [8]. They prefer to access more personalized services when are involved in mobile browsing. Spatial mobility must be considered as the major reason behind this behavior.
- Users have limited attention as they manage their mobile devices [11]. This is because they usually are involved at the same time in other tasks (e.g., walking). Temporal mobility is the reason of this phenomenon.
- Users manage their mobile devices in broadly mixed environments that are relatively unsteady from one moment to the next. Contextual mobility requires context-sensitivity on mobile device operations. So, mobile device is able to detect the user’s setting (such as location and resources nearby) and subsequently to propose this information to the mobile application.

III. ANALYZING TYPICAL LOCATION-BASED USER REQUIREMENTS

A case study of a context-aware location-based application is designed and implemented. It is capable to identify the current location of the user and to provide the following information:

- depiction on the map of user’s current location,
- route search/detection and marking on the map of user’s preferred route while he is in move,
- displaying various user personal points of interest/attractions (such as his home, his office, his favorite restaurants, etc.), along user’s current route,
- depiction on the map of general interest type attractions (e.g. museums, subway stations and hospitals),
- editing of user personal attractions (adding new ones, deleting existing, etc.),
- editing of general interest attractions’ categories.

It is undoubtedly a customized application, since each user receives information which is strictly related to his identity. The personalization of services is accomplished by exploiting information available from the mobile device’s SIM card. In our case, obviously, there is no card SIM, as we use the .NET mobile device programming environment for simulating the mobile application’s behavior.

Thus, in our case, users to enter the system should give some personal codes. That is, there is a form in which a registered user will present his credentials, while an additional form, can be used by a new user to create an account. As a result, a data base in the portable device carries user’s data.

The existence of the device’ database creates some additional problems. One has to do with the capacity of devices, which affects the size of the database. Since it can not be very large, data of multiple users is difficult to be saved on the same device. In other words it is not feasible different users to use the same device, and their personal settings (passwords and attractions) to remain stored on the device. But this is obviously not so important, since the mobile device is used mainly as a personal (single user) client machine.

Another similar problem has to do with users’ ability to connect to the system from different devices. They should create their account in any device from the beginning, and then adjust the parameters they want.

A viable solution is to maintain a database on a server. The device will have to link to this server, and request and receive data whenever it wants (merge replication process).

Location-based applications may base all their operations not only on the user’s current location, but also on various other information about the status of the user, such as time, weather, etc. (context-aware wider character of location-based applications). Finding user’s location is possible by various technologies, but as this application was developed and tested locally, finding position and movement-on-a-path operations are made by simulation. That is, a virtual movement of a certain user and its current position changes are defined through the application environment. Certainly, finding the virtual user’s current location is accomplished through our application design.

The application consists of three separate units, which interact with each other:

1. User interface.
2. Mobile database on the device, which deletes its data when session ends.
3. Server database, which stores and maintains all mobile users’ data, and additionally sends the appropriate data to the proper mobile database (based on user identity information).

IV. OUR CASE STUDY

A. MapPoint Web Service Technology

MapPoint Web Service (MWS) technology [13] is used to design search processes for addresses, attractions, routes, and creating processes of their respective maps. The MWS is actually a Microsoft’s Internet service, which is designed to work using the Graphical Information System (GIS) and responds to various scenarios mapping, which involve different types of applications, such as portals, web pages, but mainly mobile applications. In conjunction with
MapPoint Location Server (server which detects the location of a mobile device) they create a "package" particularly effective for implementing an integrated location-based application.

MWS’ typical capabilities is to display maps (render), to find the coordinates of an address (geo-coding), to find the address through given coordinates (reverse geo-coding), to search for addresses within walking distance from user's current location, and to present route instructions.

MWS has four basic services:
- Find Service: finding addresses, attractions, coordinates of points, and points within a distance from the position of the user, depending on the type and how to do the search.
- Render Service: creating the image of the map, based on the data we want.
- Route Service: creates a path between two locations we have stated. The type of the route and the points from which the path will pass, depends on the data we provide.
- Common Service: contains classes, which are common in all three previous services, such as the definition of DataSource.

The general categories for the attractions available from MWS are:
- Airport
- AncientSite (Archaeological sites)
- Hospital
- MetroStation
- Museum
- ShoppingCenter (Convention Centers)
- Stadium (Stages-Sports Facilities)

The MapPoint Location Server (MLS) detects the current user’s location, identifying the mobile device (including simple mobile devices, Pocket PCs, PDAs, SmartPhones and all registered devices). To find the position, many methods are available, such as Cell-ID, A-GPS, triangular methods, etc.

B. Mobile and Server Database

The mobile device’s database contains three tables: ID, POI and CAT_POIS. The first one maintains user’s personal credentials, while the last ones user’s personal sights and categories, that is those points of interest and categories that user has added on his profile. Server database contains tables with corresponding names. They exchange data with each other, using Merge Replication process of the Internet Information Services (ISS) web server. The exchange of data is accomplished in both directions, meaning that mobile database tables can not only send data to server, but also may receive data from it.

Microsoft SQL Server Management Studio is used to create server database. And mobile database tables are automatically generated and updated through synchronization.

Table ID has two columns, the username and password. Table POI contains the name, the coordinates and a brief description for locations that a certain user has personally selected. Table CAT_POIS consists of two columns, the username and poi-catname. The second column stores the name of user’s selected general category. Obviously, a certain username may appear in several records of the table, to represent user’s multiple selections on the attractions’ general categories.

V. MOBILE APPLICATION FUNCTIONALITY

The application consists of seven forms and seven code classes. Six of the classes used to connect to the MWS, and perform a different type of work. The seventh class is used for synchronizing server and mobile databases.

A. GetAddress Application Class

This class finds the coordinates of one or more addresses, based on user provided information, such as the address of point (street name and number), or the zip code. Without providing zip code, then most likely to be found more of one addresses, otherwise is found only one. The results are stored on a FindResults variable (MWS variable type).

The class is used during the storage of user’s personal attractions. And its purpose is twofold. Firstly to enable users to choose the desired location-point among others, although they have not entered the zip code and thus several addresses are found. Secondly, to store the coordinates of a point finally chosen by the user, in case that he asks to illustrate the point on the map. In such a case, location’s coordinates are ready to be used, and there is no need to search them through the MWS.

B. Generalpois1 Application Class

This class finds all general attractions of a user selected category. Its parameters are:
- entityname: the category of attractions that user has selected,
- distance: the distance over which should be the results desired by the user,
- mylatlongs: holds the coordinates of the user's current location, which will be used if the user wants attractions within a specified distance.

Not setting the distance parameter means that user wants to see all the sights in that category, without distance limits. Another important issue is that the MWS SearchContext parameter cannot limit the search to specific region (e.g. Attica), but only to specific country (e.g. Greece). A viable solution to this issue is by using code statement like the following:

\[
\text{If fr.FoundLocation.Entity.DisplayName.IndexOf("Attiki") <> -1 Then ...}
\]

This line of code checks over whether found locations are located in Attica. The check is made on whether the found address contains the word Attica.

The search process must be carried out twice: in order to be able to save the results (second step) that user desires, the size of foundEntities MWS variable must be defined (first step).

The other case is when user wants the attractions at a specific distance from its current location to be displayed.
The parameter distance is expressed in meter units. It should be converted into degrees, so that it can take part in calculations with the attractions’ and current location’s coordinates.

C. Generalpois2 Application Class

This class accepts the results found by the generalPois1 class and creates the map, with the representation of all the attractions found, plus the user’s current location. Depending on the number of results (MWS variable entitiesnum) that have been already found, the size of both mylocations MWS variable (items that will appear) and pushpins MWS variable (the "pins" to be placed on the map and will represent the respective point of mylocations) is defined. Once given the coordinates, then the map is created by using the ViewByBoundingLocations class.

The label parameter of pushpins variable stores a number, which represents the series that the attraction has to the list of results. So, a ListBox user interface control, which stores the name of the attractions, along with the pushpins' label parameter is capable to inform users what exactly represents any numbers displayed on the map.

D. Viewpois Application Class

This class returns an image with the user’s current location and the potential various attractions that he personally selected to be displayed. It works just as the generalpois2 class. An important differentiation is the zoom parameter’s setting of the views MWS variable.

E. MakeRoute Class

This class creates a path between two points. Depending on the user’s selections, various personal attractions can also be presented along this path.

Each time the MakeRoute class is called, it creates a new path, which is part of the journey up the first time that the class was called for this route. Each time the route is divided into segments (sections). In every move, the user’s current location is moved to the next segment, to show that the user has moved on set route. In order to have a proper representation of the move, we should have kept the number of parts of the route that was created when the class was invited for the first time on this route.

The MWS variable kindofRoute sets the type of route to be used for the creation of the route. The SegmentPreference.Shortest value creates the shortest route (based on distance), between points, while the SegmentPreference.Quickest value creates the fastest route, based on the time of transition from the beginning to the end.

An example of line code to create a certain path follows:

```csharp
```

Three different cases exist: to not display any attraction along the route, to display all the user's personal sights, and to display those user’s personal sights, which are in a certain distance from the user. The choice of the case is based on MWS pois variable’s value. In general, the movement is divided, for all three previous cases into three sections:

1. the movement carried out from beginning to the last but one part,
2. the movement carried out in the penultimate part,
3. the movement as user has reached the end of the path.

F. MakeRoutePois Application Class

This class creates a path between the user’s current location and a certain attraction which is selected by the user. The class is differentiated from the previous MakeRoute class in its way to present the map. The image’s center is not the user’s current location: what is here important is to display the whole path. For this reason, the previously mentioned MWS ViewByBoundingLocations class is not used. Instead, the MWS variable mapSpec is used to hold the path.

G. Replication Application Class

This class is used to synchronize server and mobile databases. The parameter InternetUrl holds the virtual address, set up on our server, to allow for communication between the bases:

```csharp
replication.InternetUrl = "http://192.168.1.2/ReplSync/sqlcesa30.dll"
```

ReplSync is the folder that has been created for the exchange of data while sqlcesa30.dll is an essential dynamic link library .NET file for exchanging data.

VI. USING THE LOCATION-BASED SERVICES

In this paragraph we will present a typical application scenario to demonstrate specific contextual, personalized and user-friendly features of our approach.

Let us suppose that user Alice wishes to see on her mobile the route from her current location to the Airport. Fig.1 depicts the navigation chart of the application, so Alice has to follow the selections’ path:

1. ‘Map’ --> ‘Find Route’ --> ‘To General Attraction’
2. ‘Manage Attractions’ --> ‘General Attractions Categories’
3. ‘Eiste edo’ pin declares her current location of Alice’s route. But not only that: Alice may easily choose Airport destination because all categories of attractions that are not of interest for her, are not be provided as possible selections on her mobile device. This can be done, because Alice in the past has used the following option to declare her interests:

- ‘Manage Attractions’ --> ‘General Attractions Categories’

Alice, as shown in Fig.2, may see now a map which presents a route according to her preferences, created by MakeRoute class. ‘Eiste edo’ pin declares her current location.
location, while ‘Arxi’ and ‘Goneis’ pins are personal attractions that are close to this route.

As she moves, Alice decides to stop get informed about the closest personal attractions, or even to change the distance parameter that decides if a personal attraction is close to the route. She just has to select:

‘Route Options’ --> ‘Changing Attractions’

The map will reflect the changes regarding the presence of her personal attractions on the route. If Alice meets by change a friend with a car, willing to take her to the airport, she may select:

‘Route Options’ --> ‘Changing Route Type’

In this way, she receives a different route on the map, taking into account the potential one-way streets in this area.

She also may enrich her map, by selecting the ‘Find General Attractions’ submenu. In this way, she may choose which general attractions (based on their category or on their distance from the specific route) to be displayed on the map. All these attractions are depicted as numbered pins on the map, but there is a proper menu option to get the explanations about these numbers on the map.

If Alice decides to get more detailed information on route directions (see Fig. 3), she has to select:

‘Context-aware Options’ --> ‘Route Directions’

Also, Alice may ask to see the whole route on her mobile device, using the ‘Selected Route’ option of the Context-aware Options menu. In Fig.4, the route to the ‘Archeologiko Mousio’, an attraction of the museum...
category is presented. Note the difference between figures 2 and 4: user’s current location (‘Eiste Edo’) is no longer at the center of the display to depict the whole selected route.

The application has similar functionality when users requesting route maps with destinations being personal attractions. It must be also clarified, that the ‘Default Route’ is making use of a specific destination location, namely ‘Telos’, which has taken its value when user made the registration. In this way, every user has a quick way to ask a route map from his current location to a specific (frequent enough) destination.

If Alice has no interest to be informed about route maps, she may select:

‘Map’ --> ‘Present Current Location’

With this submenu has access to the most simple location-based services, that is to display on the map her current location with or without nearby attractions (general and/or personal).

Finally, it must be noticed that ‘Manage Attractions’ option provides powerful administrative functions to Alice, to add, edit, or delete personal attractions. In this way, Alice may enjoy personalized location-based services.

VII. CONCLUSION AND FUTURE WORK

This paper initially begins with introducing mobile setting issues, context-awareness, location-based services and its significance to personalize mobile users. Further, end user application is presented which make use of location and show how location is a powerful enabling factor for any mobile application.

Future effort in research and implementation will be focused on investigating a more detailed categorization of the users as well as in observing and storing the user’s behavior in a flexible context model.

REFERENCES


