GenMod: A Generic Architecture for Mobile Location Dependent Services

Sara Cristina Oropeza Hernández, Raul V. Ramirez-Velarde and Raul Perez-Cazares[1]

[1] Instituto Tecnológico y de Estudios Superiores de Monterrey- Campus Monterrey, Eugenio Garza Sada 2501 Sur, Monterrey, N.L., 64849. Mexico
sara.oropeza@gmail.com , {rramirez, raul.perez}@itesm.mx

Abstract. The design and development of a generic architecture to provide Mobile Location Dependent Services (MLDS) is challenging since most of location-sensing methods uses non-standard features. In addition, it is necessary to unify the concept of the logical place in which a service executes. Moreover, the applicability of MLDS hasn't been exploited yet, and all the possible functionalities they might provide haven't been taken into consideration when designing models and systems for provide MLDS. In this paper we propose ModGen, a generic architecture to provide MLDS. ModGen separates the implementation of the service, from the communication mechanisms used to discover and provide MLDS to users. Then we propose a system that incorporates two different services provided in a hospital. In this system we prove that our general model is capable to provide different types of MLDS to users using two different types of positioning technologies.

Keywords: mobile services, location dependent, e-health, distributed architecture

1 Introduction

Due to the proliferation of mobile devices in the market, and the new schemas of networks for mobile devices, mobile users have access to a vast variety of resources and services limited to desktop computers in the past. This kind of services is known as Mobile Location Services (MLS). They integrate the geographical position of the user with additional information to provide an added value to the user [7, 11]. We can distinguish two different types of MLS. Location based and location dependent services. We mainly focus on location dependent services defined as services that are available only on specific geographical areas. They are commonly known as Mobile Location Dependent Services (MLDS).

The existence of several positioning technologies from in-door and out-door environments [1, 6, 8, 13, 14, 16] facilitated the creation of MLDS leaving behind the issue of calculate user's current location.

However, there's still an issue related to correlate information sources and physical location. Another important aspect consists on propose a model capable to provide any kind of service implementation to the users.

In this paper we present the GenMod model. It is a distributed system that allows and supports the creation of MLDS. In this work we assume that a network infrastructure and robust positioning technologies exist. The plan of this paper is as follows. Section 2 describes the previews work done in the area of MLDS. Section 3 describes the concept of context of a service. In section 4 we describe the components of the proposed architecture. A prototype developed to evaluate the GenMod architecture is described on section 5. Finally section 6 conclusions and future work.

2 Related Work

Currently there exist several implementations that support MLDS. We distinguish the implementations that use any type of positioning technology from the ones that include it as part of their implementation.

The Agents2Go infrastructure [9] adapts the CDPD module of a mobile device providing information about the cell Id and other signal parameters in order to estimate the current location of the device. In the same fashion, M-mall [10] implements a complex system in which the position of the mobile device is obtained using Bluetooth (BT) network parameters such as packet delay calculation and BT access points location. M-mall implements an agent based approach composed of three agents; one agent is in charge of managing user-specific data such as user profile and preferences. Another agent obtains the mobile device's current location from the BT network. Finally there is a service manager agent that implements the service itself, and retrieves information from the other agents to deliver services to the user. On the other hand [9] proposes a platform to deploy any location dependent service on a CDPD network. This platform relies on a distributed architecture capable to communicate with an application residing in the mobile device. The main contribution of this work consists on provide an extensible architecture, and service contexts that maps regions with available services.

The work presented in [3] uses service contexts defined as “the logical abstraction of a place in which an agent executes” to provide “a variety of mobile agents accessing data and services on a fixed network infrastructure”. These mobile agents act on behalf of the users detecting context movement and discovering and providing services. An important contribution of this work is the use of logical contexts given by the identity of the user. This filter ensures the privacy of the data; therefore, users have limitations on the services they can access based on their identities. In this work, the implementation of the positioning technology is taken as a separated aspect enabling the use of different technologies to provide a physical location of the mobile device.

The AROUND architecture proposed in [5] provides a distributed service infrastructure, a contextualization process and a service name resolver. To map physical location with services, AROUND uses contexts. These contexts provide a different functionality from the ones proposed in [3]. This allows one context to be contained into another context. Moreover, the relationship between contexts only exists in one direction allowing query propagation from general to particular contexts. The name resolver is responsible of translating physical location to a specific context. Finally the work developed by [4] proposes portable software architecture for indoor-outdoor location sensing. As in the previous works, [4] uses the concept of zoning that in turn reduces the efforts required to locate the services. The hybrid indoor/outdoor system proposed is based on Bluetooth and GPS positioning technologies. To provide services, they model a zone as a set of physical coordinates including a description about the same. An application server stores all the available services while a topology server manages the topology of the zones as well as the topology maps used by the application server and the mobile device. To allow service discovery, the mobile device has is equipped with several agents that provide different functions such as user's current position and deliver to the user information related to the zone including services.

As can be seen, all the cited works provide an abstraction of a physical location in terms of available services. In this case, the user's current location is only used to find the zones or contexts that users can access. Another important factor is to separate the position technology from the architecture that provides services to mobile users. In addition, no one of the cited works takes into consideration the possibility to allow data addition or modification as a service functionality.

3 Context of a Service

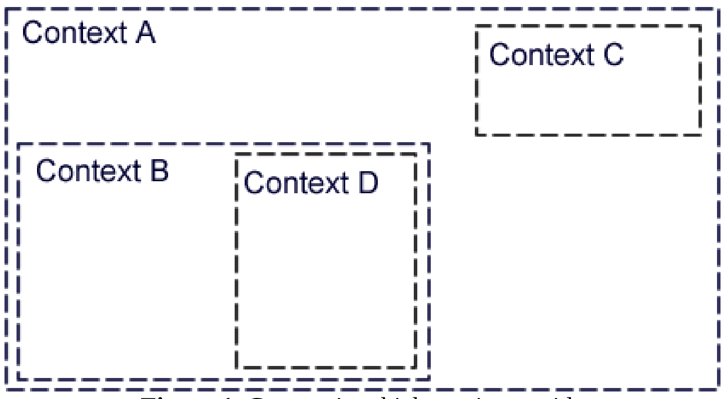


Figure 1. Context in which services reside

In this work we define an abstraction of a place in which services exist. This abstraction is called context of a service and maps services to a physical location. Each context associates a set of services that can be accessed by a mobile user. It is possible that a geographic place is mapped to more than one context. In this way, service duplication is avoided since services with bigger coverage areas can be defined into different contexts than those of services that need to be available in smaller areas. Fig. 1 illustrates this.

As stated in section 2, a context is mapped to a location. Nevertheless, in our architecture a context can be, albeit temporarily, set manually in order to access data and services that are no longer available. In other works, the location-context architecture can switch to client-server when it needs to.

4 Architecture Overview

We took into consideration the following requirements to define our model:

- Due to the variety of positioning technologies [1, 8, 13-16], the model shall be independent of those. Also it should be able to work when the user switches from one positioning technology to another.
- The discovery scope of the services shall work only inside of the current context in which the mobile device resides [12].
- The service discovery shall be performed by the mobile user. The service registration will be performed by an entity of our model.
- It shall be scalable in order to support a raise in the amount of users and services provided. Also needs to be extensible to add new capabilities and functions.
- The service not only delivers information to the user, it also allows the user to modify the information that it provides. This property can be a valuable factor for some applications.
- It shall add a low load to the network

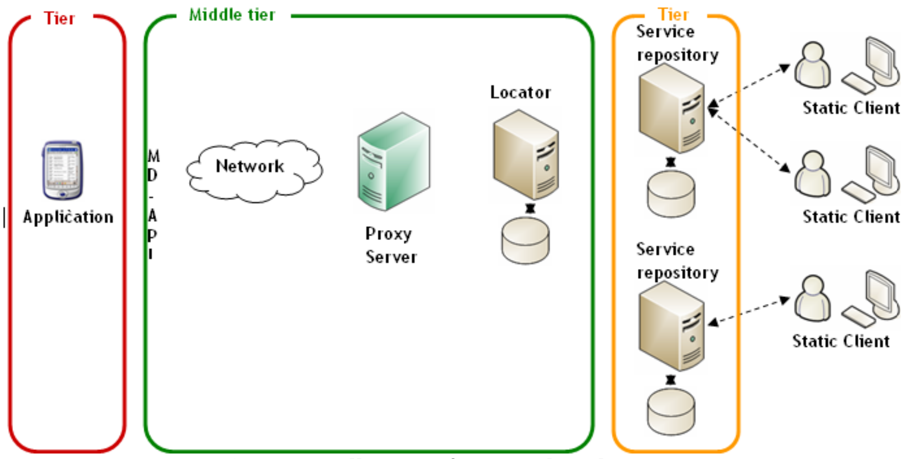


Figure 2: Full View of GenMod Architecture

GenMod is a three tier distributed architecture. The first layer comprises the application user end. The intermediate layer is more complex. It is composed by several components of the GenMod architecture. We'll cover these elements later. The third layer consists on the data repositories that allocate and implement the services. Fig. 2 maps the elements that conform the GenMod architecture to the three tier model. As can be seen a static client can also access a data repository (this depends on the nature of the service).

4. 1 MD-API

The application that resides in the mobile device is the entity that interacts directly with the user and offers functions such as service display and data gathering from the user. However, the application doesn't interact with the rest of the components in the architecture. The MD-API is a communications API that mediates between the application and the architecture components. It performs the operations of service discovery on behalf of users providing well defined primitives to access a service in the form of messages. Also, it provides a set of standard functions to manipulate the data delivered by the service in an easy way. The MD-API is considered a part of the middle tier but it resides in the mobile device. Hence, the MD-API hides implementation complexity and communication process to the programmer.

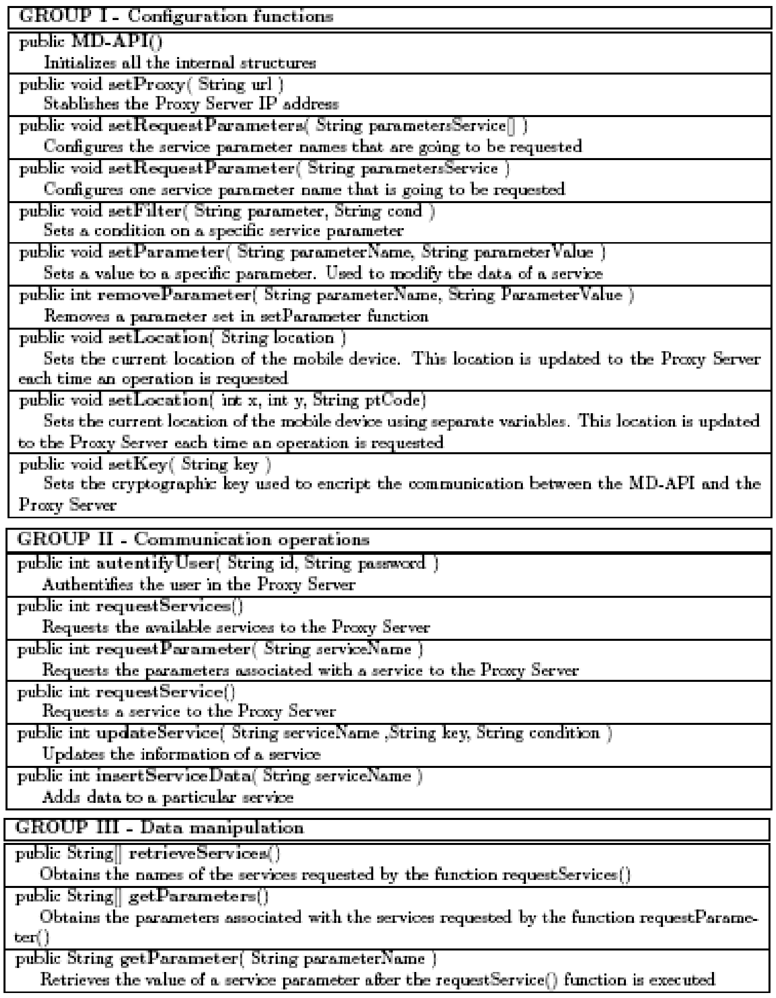
The MD-API is composed by three different groups of functions. The first group allows the configuration of the operation that is going to be performed. In the second group we find the operation itself as communications functions. The MD-API stores the responses from the service requests, creating a data cache that can be used in disconnected environments. Also, since the user had access to previous contexts and the data used in those contexts is stored on the mobile device, contexts can be manually switched (usually by short amounts of time) in which case data can be accessed locally or remotely as in a regular client-server environment. The third group of functions is used to manipulate service responses. Table 1 shows brief descriptions of the MD-API.

There are several design aspects that were taking into consideration while designing the MD-API. First we defined a set of operation status values for each function that turned out to be usually paired. In this way, we ensure that users will always get feedback from the operation performed. Secondly, we assumed that all the data that is exchanged between the MD-API and the service is only text. Hence, the interface of the MD-API is given by strings and arrays containing strings. The third design consideration is split functionality. In this way, it is possible to combine functions to allow different types of service requests.

4.2 Proxy server

This component interacts directly with the application through the MD-API. As mentioned before, communications between MD-API and the Proxy Server are implemented by a series of messages that describe the operation and related data that is taking action. The Proxy Server obtains the available services from the service repositories based on the user's current location. Then, it filters the services based on the user profile. In this way, it restricts access to services avoiding the user having to discover them. As can be seen, the Proxy Server models the association between physical location and specific criteria (i.e. security, business models, hierarchy of the user inside of the organization, etc.) to determine the available services for a mobile user. The basic functions that it performs on behalf of the user are the following.

- Provides the necessary parameters to access a service
- Keeps track of the available services through an establishment of a temporal session. This session ends when the user doesn't contact the Proxy Server for a long period of time determined by factors such as application nature, user's movement, etc.
- Acts as a service directory



GROUP I - Configuration functions	
public MD-API()	Initializes all the internal structures
public void setProxy(String url)	Stablishes the Proxy Server IP address
public void setRequestParameters(String parametersService[])	Configures the service parameter names that are going to be requested
public void setRequestParameter(String parametersService)	Configures one service parameter name that is going to be requested
public void setFilter(String parameter, String cond)	Sets a condition on a specific service parameter
public void setParameter(String parameterName, String parameterValue)	Sets a value to a specific parameter. Used to modify the data of a service
public int removeParameter(String parameterName, String ParameterValue)	Removes a parameter set in setParameter function
public void setLocation(String location)	Sets the current location of the mobile device. This location is updated to the Proxy Server each time an operation is requested
public void setLocation(int x, int y, String ptCode)	Sets the current location of the mobile device using separate variables. This location is updated to the Proxy Server each time an operation is requested
public void setKey(String key)	Sets the cryptographic key used to encript the communication between the MD-API and the Proxy Server
GROUP II - Communication operations	
public int autentifyUser(String id, String password)	Authentifies the user in the Proxy Server
public int requestServices()	Requests the available services to the Proxy Server
public int requestParameter(String serviceName)	Requests the parameters associated with a service to the Proxy Server
public int requestService()	Requests a service to the Proxy Server
public int updateService(String serviceName ,String key, String condition)	Updates the information of a service
public int insertServiceData(String serviceName)	Adds data to a particular service
GROUP III - Data manipulation	
public String[] retrieveServices()	Obtains the names of the services requested by the function requestServices()
public String[] getParameters()	Obtains the parameters associated with the services requested by the function requestParameter()
public String getParameter(String parameterName)	Retrieves the value of a service parameter after the requestService() function is executed

Table 1. MD-API description

To ensure security in all the steps of the communication the Proxy Server encrypts all the information that is sent across the network. Also, we provide a challenge-response authentication mechanism between the Proxy Server and the mobile application.

4. 3 Locator

This component finds the repositories that might contain available services to mobile users based on their current location. It can be considered as a router that determines to which place the request should be forwarded. To determine this, it uses a set of tables that maps contexts to physical coordinates. To reduce the search time, the location divides a geographical location into zones. Then the zones are divided into sub zones. Finally, the sub zones will match one or more contexts.

4.4 Service Repositories

The context of a service can be physically mapped to one or more service repositories. In this way each repository may contain several services that exist in the same context. In addition the service repositories contain a service descriptor that includes service name, service parameters and some additional information about the service itself.

As can be seen from the Fig. 2 service repositories include computers that process the queries coming from the mobile device. No matter the implementation of the service, the information retrieved by any service should fit into the message format.

Depending on the implementation, a service can also be provided to static users. It is possible that a static client wants to access a service. Although static clients don't belong to the three tier architecture to provide MLDS to users, it is possible for a mobile client to switch to static mode in order to access a now unavailable service or to manually switch context and even access data locally functioning as a disconnected device data would be later updated as soon as communications a reestablished)

Having taken that into consideration, the basic operations defined for a service turn out to be very simple consisting on request, modify and update information functions.

4.5 Interactions between components

Fig. 3 shows the interaction between components and mobile application via MD-API. To start the process the user tries to get authenticated. The application uses MD-API functions to establish a session with the Proxy Server. At this point, all the sides have been authenticated and the communication between mobile application and proxy server is encrypted using a private key schema. After that, the user triggers a service request including his current location and the valid session identifier. The Proxy Server forwards this request to the locator that in turn determines the available services for that location. The Proxy Server receives this information, filters the services using user's profile, updates his session with the available services, and finally sends the service names and identifiers to the user. At this point the application shows the available services to the user. When a user requests a service for the first time, the Proxy Server requests the service parameters through the locator to the service repositories. This information is sent to the mobile application. The user selects the parameters that she wants to know from the service and requests it. The Proxy Server will validate if the requested service exists for the current user session. If it does, the Proxy Server forwards the service request to the locator that determines the service repository that can fulfill the request. The request is processed and the results are sent to the Proxy Agent which in turn forwards the results via the MD-API.

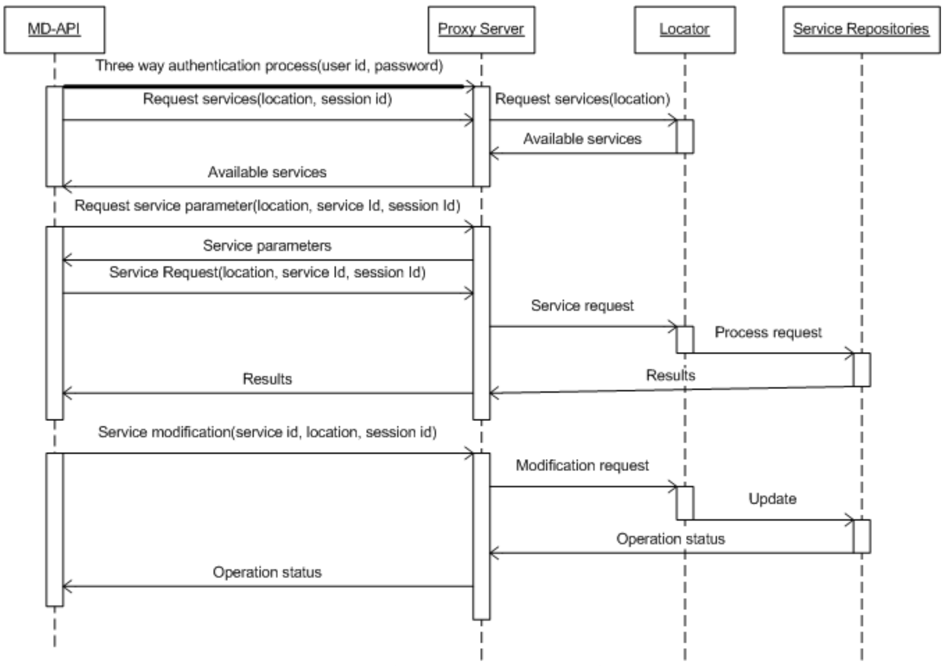

MD-API
Proxy Server
Locator
Service Repositories
Three way authentication process(user id, password)
Request services(location, session id)
Request services(location)
Available services
Available services
Request service parameter(location, service Id, session Id)
Service parameters
Service Request(location, service Id, session Id)
Service request
Process request
Results
Results
Service modification(service id, location, session id)
Modification request
Update
Operation status
Operation status

Figure 3: Activity Diagram of the Components of the GenMod Architecture

5 Evaluation Prototype

To evaluate the effectiveness of the proposed architecture we developed a prototype and tested it in a simulated hospital environment. Among the advantages of using this technology in hospitals we can name error reduction in prescription and ordering of medication, cost reduction and improved efficiency of medication and other supplies administration and improve clinical decision making by providing physicians and clinicians with access to electronic health records for their patients.

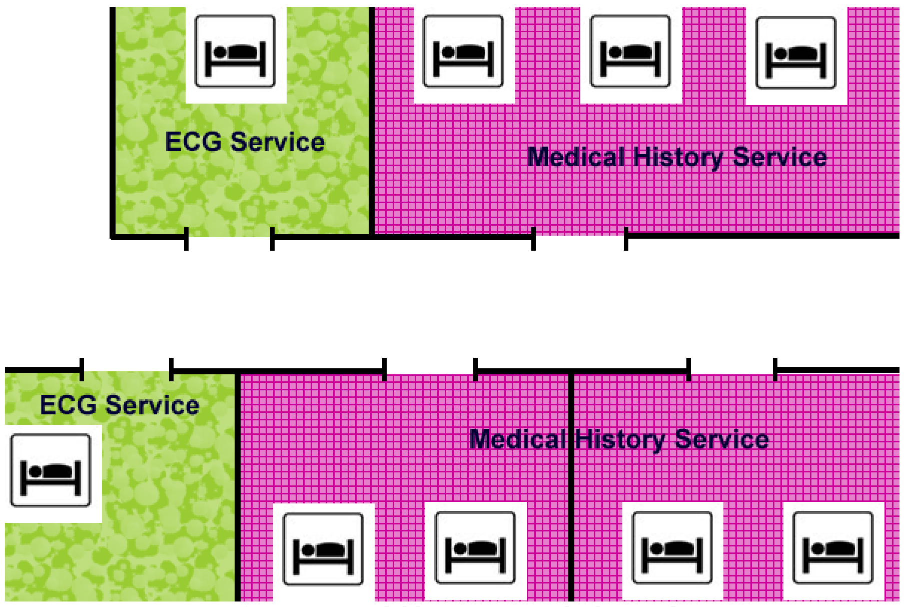


Figure 4: Available services in the hospital

We based our simulation environment on the work developed by [2] which allows a Wi-Fi connected laptop to access different types of services while it moves. However, offered services are centralized and don't depend on location which means the hospital administration is more rapid but just as inclined to human error.

Our prototype consists on two MLDS offered in a hospital. Fig. 4 shows service distribution in the hospital. The offered services are patient medical history (modify and check) and Electrocardiogram orders (create and check). To simulate the position technology we developed an application using J2SE that notifies the mobile device its current location each time the mobile application requests it. The application of the mobile device was implemented with J2ME and we use Sun Java Wireless Toolkit to run it. We also implement the MD-API to allow connectivity with the rest of the components of the architecture. The prototype is able to work using Bluetooth and Wi-Fi based positioning technologies.

Let's assume the following scenario to test our prototype. Our user Helen is staring her hospital guard. When she enters into a patient's room she accesses the application and authenticates herself. The application looks for the available services and shows to Helen the medical history of the patients of that room. She selects the option “View history” from Jane Smith and it appears on her device. Helen decides that the medication prescription needs to be changed so she selects the “Modify” option. After that Helen saves the current modifications. While doing all of these operations Helen didn't move to a different place. Since she was inside of the patient history service context, she was able to perform all the operations successfully. Now she moves to a different room. While she is walking she realizes that she’s forgotten to check the dosage of Jane Smith’s medication. At this point she is no longer in the context that provides the service so she is not able to have access to it. She continues to the Electrocardiogram room. Helen needs to create an Electrocardiogram order to some patient. Since she is in a hurry when she receives the parameter list that needs to be filled, Helen leaves the room. When she tries to save her order, the system refuses to update this information in the server because she's not inside of the electrocardiogram service context.

Nevertheless, because Helen has three options to complete her duties:

- With a special access code, Helen could request a manual context switch
- She can switch to client-server mode with another access code and access any necessary data just like a static client would
- If the action is not urgent, Helen can switch to local access using the device's local cache. As soon as service access is reestablished the information is updated in the correct repository.

The results on functionality and usability are very satisfactory. As result, the prototype is in the process of being deployed for evaluation by medical staff of the Metropolitan Hospital in Monterrey, Mexico.

6 Conclusions

This work proposed a distributed architecture to provide general MLBS. To define a scalable and extensible architecture, we establish a set of requirements adapted from previous research that provide useful guidelines to define MLDS. Our main goals were the security, provide services independent from positioning technology, add new functionalities to the services such as data insertion and modification, scalability and extensibility. Also we define the context in which a service executes. A context may contain several services. In addition, we avoided the service duplication allowing one geographical zone to be mapped by more than one contexts.

The overall architecture has also been implemented, and a high level MD-API has been developed. Such API has been tested under a mobile application that resides in a cell phone. This application provided two different types of services to a hospital. In order to prove that GenMod is a general model for MLDS the nature of the services provided in the hospital were totally different. One service is based on several query operations and data creation, while the other requires less query operations and allows modification of the data. In both cases, our model was able to provide services, leaving the service implementation open, but having in mind a common communication method.

We are looking to enhance the security aspects provided in our model. Also, it is possible that services provide more functionality in the future. For this reason, the messages that are exchanged between the Proxy Server and the MD-API could de modified. On the other hand, an important factor is the standardization of positioning technologies. A standardization process for positioning technologies and service identifiers could ensure that MLDS can be more uniform and could be used as a fist step into MLDS standardization.

References

[1] Paramvir Bahal and Venkata Padmanabhan. Radar: an in-building rf-ased ser location and tracking system. In Computer, page 26, 2005.

[2] Jakob E. Bardram. Applications of context-aware computing in hospital work: Examples and design principles. In SAC '04: Proceedings of the 2004 ACM Symposium on Applied Computing, pages 1574-1579. ACM Press, 2004.

[3] Giacomo Cabri, Letizia Leonardi, and Franco Zambonelli. Location-dependent services for mobile users. In Systems, Man and Cybernetics, Part A, IEEE Transactions on, volume 33, pages 667-681, 2003.

[4] Cristiano di Flora, Massimo Ficco, Stefano Russo, and Vincenzo Vecchio. Indoor and outdoor location based services for portable wireless devices. In ICDCS Workshops, pages 244-250, 2005.

[5] Rui Jose, Adriano Moreira, Filipe Meneses, and Geo Coulson. An open architecture for developing mobile location-based applications over the internet. iscc, page 0500, 2001.

[6] Kamol Kaemarungsi and Prashant Krishnamurthy. Modeling of in door positioning systems based on location finger printing. In INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, pages 1012- 1022, 2004.

[7] Nokia. Mobile location services, white paper, 2001.

[8] John Pagonis and Jonathan Dixon. Location awareness and location based services part i, 2004. www.symbian.com.

[9] Olga Ratsimor, Vladimir Korolev, Anupam Joshi, and Timothy Finin. Agents2go: an infrastructure for location-dependent service discovery in the mobile electronic commerce environment. In WMC '01: Proceedings of the 1st International Workshop on Mobile Commerce, pages 31-37, New York, NY, USA, 2001. ACM Press.

[10] Jaime Garcia Reinoso, Javier Vales Alonso, Francisco J. Gonzalez Castaño, Luis E. Anido Rifin, and Pedro S. Rodriguez Hernandez. A new m-commerce concept: m-mall. In WELCOM '01: Proceedings of the Second International Workshop on Electronic Commerce, pages 14-25. Springer-Verlag, 2001.

[11] Jochen H. Schiller and Agnes Voisard. Location-Based Services. Morgan
Kaufmann, 2004.

[12] Alex C. Snoeren. Requirements for a general location service. http://cite-seer.ist.psu.edu/snoeren97requirements.html.

[13] IEEE Computer Society. System uses wi-fi to provide location services.Computer, 38(8):26, 2005.

[14] Martin Vossiek, Leif Wiebking, Peter Gulden, Jan Wieghardt, Clemens Homann, and Patric Heide. Wireless local positioning. Microwave Magazine, IEEE, 4(4):77-86, 2003.

[15] Yufei Wang, Xiaodong Jia, and Chris Rizos. Two new algorithms for indoor wireless positioning system (wps), 2004. www.gmat.unsw.edu.au/snap/publications/wangy etal2004a.pdf.

[16] Jason Yipin Ye. Atlantis: Location based services with bluetooth, white paper, 2005. http://whitepapers.zdnet.co.uk.