Location Based Services architecture
for Simple Mobile Services

Version 1.1, August 2008

F. Martire\(^{(1)}\), G. Bartolomeo\(^{(1)}\), C. Trubiani\(^{(1)}\), S. Salsano\(^{(1)}\)

\(^{(1)}\)University of Rome “Tor Vergata”

Available at: http://netgroup.uniroma2.it/twiki/bin/view.cgi/SMS/TechnicalReports

Contact: stefano.salsano@uniroma2.it

DISCLAIMER

This work has been performed in the context of the EU funded IST Simple Mobile Service project (http://www.ist-sms.org), but it does not necessarily represent the official position of the project itself and of its partners. Authors are solely responsible for the views, results and conclusions contained in this work.

COPYRIGHT

"Copyright and all rights therein are retained by authors. This work may not be modified or reposted without the explicit permission of the copyright holder"
# Table of Contents

1  SIMPLE MOBILE SERVICES AND LOCATION BASED SERVICES (LBS) ........................................ 4

1.1  THE SMS LOCALIZATION ARCHITECTURE OBJECTIVES .................................................... 4

1.2  OVERVIEW OF THE COMPONENTS OF THE SMS LOCALIZATION ARCHITECTURE .......... 8

2  SMS LOCALIZATION AND NAVIGATION PROCEDURES ................................................. 12

2.1  LOCALIZATION IN INDOOR PLACES .................................................................................. 12

   2.1.1  UML Sequence diagrams for indoor localization ......................................................... 17

2.2  SWITCHING OUTDOOR/INDOOR CONTEXTS ................................................................. 22

2.3  VIRTUAL LOCALIZATION SYSTEM ................................................................................. 26

2.4  VIRTUAL AND REAL LOCATION CONTEXT .................................................................. 27

2.5  REALIZING OUTDOOR FEATURES .................................................................................. 28

3  SMS LOCALIZATION AND NAVIGATION COMPONENTS ............................................... 29

3.1  CLIENT-SIDE COMPONENTS .......................................................................................... 30

   3.1.1  SMSCore .................................................................................................................... 30

   3.1.2  NavigationDispatcher .............................................................................................. 32

   3.1.3  Navigator .................................................................................................................. 36

   3.1.4  LocalizationProvider ............................................................................................... 40

   3.1.5  GPSController ......................................................................................................... 43

   3.1.6  GSM_CellIDController ............................................................................................ 45

   3.1.7  ZigbeeController ..................................................................................................... 46

   3.1.8  Wi-FiController ....................................................................................................... 47

   3.1.9  MapManager ........................................................................................................... 48

   3.1.10 SearchesManager ...................................................................................................... 50

3.2  SERVER-SIDE COMPONENTS ............................................................................................ 52

   3.2.1  SMSSSearchesServerProxy ...................................................................................... 52

   3.2.2  SMSMapServerProxy ............................................................................................. 53

   3.2.3  SMSCommunityServer .......................................................................................... 54
1 Simple Mobile Services and Location Based Services (LBS)

Simple Mobile Services [1][2][3] are typically targeted to specific environments. In many cases this means that services are associated with specific locations. Any user with an interest in a given location will be able to access services associated with the location.

Location Based Services use information about the user’s location to select the information they provide. Most users who access a service associated with a specific location will be actually visiting the location. In this sense SMS can integrate Location Based Services (LBS) being able to answer questions like: Where am I? What’s around me? How do I get there?.

Following we illustrate concrete examples and provide case studies of how our SMS Localization Architecture and the defined location information that come in the so called SMS “Position object” are used to rapidly develop new applications and services or to integrate existing ones that rely on positioning concepts and technologies.

1.1 The SMS Localization Architecture objectives

Location based Services are an hot topic in research. The contribution of this work has been the development of an original architecture, integrated in the Simple Mobile Service architecture, including the definition of a “Position object” and providing a reference implementation of location-based portable application.

According to the philosophy of SMS project, our aim is to provide simple to use interfaces relieving the developers and the users from the complexities related to the acquisition of a position information as well as hiding the particulars of the positioning technologies employed. What we have in mind is that for users, developers and service providers it will be simple not to interface with the jargon of map data or network elements. In fact, user and developers are typically interested in using the location awareness rather than dealing with underlying mechanisms used to achieve such awareness.

The Simple Mobile Service localization architecture includes mobile device applications, a middleware and a set of server side elements. Its structure is extremely modular and is decomposed in a set of components that can be composed to create services. In fact, during service execution, the architectural components residing in the user terminal will interact with localization technologies, smart spaces elements and with server side service entities to provide SMS value added services to the SMS user.

Moreover the overall architecture is conceived to support situations in which determining the exact position of a mobile terminal is not a strict requirement, but it is enough to identify the terminal position within a radius or inside an area (e.g. rooms indoor, or zones outdoor).

In this section we introduce our software architecture that makes a combined use of indoor and outdoor location-sensing technologies. Our vision moves from the observation that during recent
years many research activities have focused on developing a wide range of location based applications but they are specifically designed for either indoor or outdoor scenarios. Instead we are interested in designing a so called hybrid system aimed at supporting the development of applications that are able to solve localization problems independently from the environment and the location technology in use. Among these works the ActiveCampus [25] project provides a location-based reminders system using only the 802.11 radios in a university campus, ComMotion [24] using GPS sensing technology have prototyped a location-aware application that is specifically tailored to outdoor scenarios. Bluesic is a context-aware information system for tourism, based on Bluetooth technology [27].

The work describes our efforts to integrate and combine an indoor and an outdoor localization software modules into a complete navigation system providing a piece of software that will run on mobile devices. This way we will allow a user to seamlessly exploit, properly combining the use of indoor and outdoor location sensing technologies, guidance and navigation functionalities either in outdoor and indoor environments. Our approach takes also in account that recently, the terminals that are on the market, ranging from smart phones to simple mobile phones are equipped with several technologies exploitable for location sensing purposes. In outdoor scenarios we may utilize at the same time and in parallel not only the GPS as outdoor positioning system but also for example the 802.11 networks or the GSM antenna identifier (CellID). Thus our API is also meant to support situations in which the result of the implemented automatic mechanisms to locate people or devices can be refined using in parallel multiple position determination technologies. Therefore an expected outcome of this work is also the interoperability of positioning systems.

In a combined indoor/outdoor application, we can say that the information about the position of users or devices is provided by means of at least two different positioning systems. We designed and implemented our architecture also proposing a mechanism for automating the management and the switching between multiple location sensors in an application-transparent manner.

The very aim is to build an architecture offering a library that can be used to help implement location aware applications on top of an open programming interfaces (APIs) and protocols minimizing LBS development time and costs. As a goal we want to provide a standardized set of tools to facilitate, increase and encourage the proliferation of the research in this topic hoping that they will be widely used in the future.

For our project purposes moreover, we have designed and implemented a so called Position object to hold both geodetic (physical position e.g. lat and long coordinates) and symbolic (e.g. civic address) data. Different positioning systems generate different forms of position information, hence in order to manipulate homogeneously the location information we defined the Position object as a generic presentation format. Position objects identify a location in terms of parameters like: coordinates, textual address (e.g. Via del Politecnico 1, Roma), data accuracy, time stamp, and information about the positioning method used for the location determination, plus an optional textual description. We underline that people don’t think in terms of latitude and longitude so is useful to incorporate semantic in location data (e.g.“Mum’s Office”, “Park” or “5 minutes walk from home”). Moreover designing our architecture we envisioned that making the user free to author his current position would allow our system to also support scenarios in which is not possible to retrieve the exact position of a mobile terminal using a suitable positioning techniques. Hence all the Position object fields can be manually edited by the user itself.
Looking at location determination techniques in outdoor environments, Global Positioning System (GPS) [6] is the technology that can be used to acquire the position. In indoor environments where the GPS is unavailable, Wi-Fi [7], Zigbee[8], mobile phone infrastructure (cellID), Bluetooth, RFID, Visual Tags[5] placed in key spots such as entrances represent examples of promising technologies to pinpoint each user’s location. From the above Table you can see the technology we are currently able to support in our implementation of the Localization architecture.

We claim that we want to provide a localization and navigation architecture that can, in many ways, enable the simple creation and development of new context-sensitive services or supply existing applications with location awareness. Hence we have built our piece of software having in mind that these services can be local modules running on the same Virtual Machine that contains the SMS localization components or remote entities coming from third party developers.

Our SMS localization interface will provide, services like mapping, routing, spatial searching functions, geocoding capabilities, as well as location-based triggers and alerts under a single, open interface. We can have different implementations of the navigation and localization services on the terminal, the SMS Navigation/Localization Component should be able to adapt to these different implementations. Whatever the implementation will be, the idea is that the same interface will be offered to the other components.

<table>
<thead>
<tr>
<th>Outdoor Localization Systems</th>
<th>Indoor Localization Systems</th>
<th>Virtual Localization Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td>Wi-Fi</td>
<td>Visual Tag</td>
</tr>
<tr>
<td>CellID</td>
<td>Zigbee</td>
<td>RFID</td>
</tr>
<tr>
<td></td>
<td>Bluetooth</td>
<td></td>
</tr>
</tbody>
</table>

*Table 1-1 Localization Systems*
SMS Localization Interface

Other SMS Components

SMS Localization Interface

J2ME Application

Web services

Commercial navigation software offering an API

Figure 1-1 different implementations of the navigation and localization services

Figure 1-1 shows a set of possible underlying realizations of the functionality exposed by the SMS Navigation/Localization component. For example we could have a commercial navigation application (e.g. TomTom) which offers an API. In this case the SMS Navigation/Localization Component will drive the API of the commercial application. As second option we can use a mashup of a web based system in which the geocoding operations and the maps are retrieved from remote servers. In this case the SMS Navigation/Localization component will turn all its operations into requests to the remote servers. A third solution could exploit a navigation application explicitly developed to play the role of SMS Navigation/Localization component. In the rest of the document, we assume the third case and we will describe the J2ME applications that we have developed. Our application is capable to handle both outdoor and indoor navigation. On the mobile device side, in fact, the Simple Mobile Service architecture is based on an application named MOVE (Mobile Open and Very Easy).

Summing up the original contribution of this work is three-fold:

- we have designed an architecture on top of which there is a generic and high level software API, for technology-independent location sensing. This way we are able to decouple the user position provisioning from the underlying positioning technology
- we propose a software architecture that can in parallel handle and combine multiple indoor and outdoor location-sensing technologies. Moreover we have built our architectural components in such a way that they realize a transparent and automatic switch mechanism from indoor to outdoor (and vice versa) situations
• we defined a Position object as a generic container of positional information able either to describe indoor and outdoor spaces. Inside this Position object we also defined suitable additional data fields to properly and consistently combine, choose and handle the several available methods to determine a mobile user’s location.

1.2 Overview of the components of the SMS Localization Architecture

This section provides a more detailed description of the overall system and we will give details on the key role and functionality of the main components of the proposed software architecture. The illustration in Figure 1-2 depicts all the entities that take part in our framework.

First of all the reader can notice that we split the architecture in two main parts: a Server-side part and a Client-side one. A typical applicative scenario in fact requires the interaction between mobile devices and the proposed server side architecture. However we designed the system making users also able to exploit some location based services even when no external communication with a server counter part is available. This is possible if the location information is stored or cached in the client side.
To describe the entities that constitutes the whole system we list the main functionalities we are able to support for both outdoor and indoor scenarios:

1. Transparent switch Indoor/Outdoor
2. Determine people or mobile devices position relying on a physical localization system (indoor or outdoor location - sensing technology) or on a “virtual” one.
3. Visualize user’s position on the screen of the mobile device on 2D maps
4. Realize geo-coding functionalities
5. Ability to provide driving directions and guidance features.
6. Tracking
7. Dispatch the positional information to external infrastructural components and see the position of other peers.

Figure 1-2 The SMS Localization Architecture
Each one of these items is realized by a specific Component.

<table>
<thead>
<tr>
<th>Mapping between functionalities and Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transparent switch Indoor/Outdoor</td>
</tr>
<tr>
<td>Determine people or mobile devices position relying on a physical localization system (indoor or outdoor location sensing technology) or on a “virtual” one.</td>
</tr>
<tr>
<td>Visualize user’s position on the screen of the mobile device on 2D maps</td>
</tr>
<tr>
<td>Realize geo-coding functionalities</td>
</tr>
<tr>
<td>Ability to provide driving directions and guidance features.</td>
</tr>
<tr>
<td>Tracking</td>
</tr>
<tr>
<td>Dispatch the positional information to external infrastructural components and see the position of other peers</td>
</tr>
</tbody>
</table>

The **NavigationDispatcher** is responsible for managing the transparent and seamless switch between indoor and outdoor environments when a mobile device is moving from inside to outside a certain closed space and vice versa. This means that it is able to orchestrate the start and the stop of the navigation/localization applications specifically tailored to manage only one type of environment. The decision to instantiate a switching mechanism or to run a specific navigation/localization module relies on policies that take into account several parameters. For example, these are constituted by the current active localization technology (GPS, Wi-Fi, Zigbee) coupled with information that marks the user current position as a special place called “Gateway” point. This place identifies a so called boundary location that divides an “indoor” space from an “outdoor” one (e.g. the entrance door of a building). Think at a situation in which a user, turning on his device, has for example selected from the menu option a “followMe” operation that enables tracking functionalities. The aforementioned component is able to transparently make the system try to discover GPS signals automatically if the Wi-Fi indoor link gets lost near a Gateway point and vice versa.

The determination of a user position is delegated to the **Localization Provider**. This component is responsible for the retrieval of current location data and for the provision of the estimated position to other components filling in the fields of the Position objects. In order to decouple the user position provisioning from the underlying positioning technology it hides the specific location sensing methods used to retrieve position information to external entities. It therefore offers an high
level API that is location sensing technology-independent. For the sake of clarity we can imagine that the interface exposed is a simple and generic “getPosition” method.

The set of interfaces of the Localization Provider are intended to:

- Provide the user’s current position
- Provide a tracking service which can be exploited to follow user’s movement
- Provide a subscription service which informs the subscriber whenever the user has reached a given position

Regarding the exploitation of distinct positioning systems we need a sort of adapters to interface with the real location sensors. The Localization Provider determine the location data trough suitable controllers implemented as PlugIn modules to enable the simple extension of supported location methods, in Figure 1-2 shown as Localization System Controller. As an example we have a GPS_Controller that is in charge to connect to a Bluetooth GPS receiver, to retrieve the data coming out from the BT enabled devices, parse the NMEA sentences and extract the coordinates.

As for Outdoor environment we can rely on the GPS but in Indoor places we need a suitable deployed localization system. For this reason on the Server side of our architecture we defined the SMSLocalizationServer and the IndoorPositionServer. The first one communicates with the LocalizationProvider returning indoor positioning information (e.g. cartesian coordinates) localizing also a user on a specific map identified by a MapID. The IndoorPositionServer maintain the topology of the sensors used to determine the user position. A Wi-Fi_PositionServer has a database where each Access Point is identified by his MAC address and its indoor coordinates.

The MapManager component has the task to visualize the user current position on a 2D map or a generic position as a red dot in the centre of the map. It orchestrates the downloading and the correct visualization of the images on the display. MapManager fetches the map of the requested area and dynamically crops or grabs the needed images with regards at the dimensions of user terminal’s screen. After receiving the request to load a specific map, the component first checks if the user has this resource already in the terminal memory storage and only if the resource is not available it sends the request to a remote Map server. As for Map servers for outdoor environment we currently use online Map providers, such as Google Map, MSN Virtual earth, Yahoo. Loading a Map requires sending the MapManager a longitude and latitude pair for outdoor spaces (OutdoorMapServer) or a Map URL in case of indoor areas (IndoorMapServer). We developed a so called SMSMapServerProxy that receives the request from the user equipment and depending on the chosen map provider routes them to one of the specific MapServers after having properly formatted the query. This way we can add with a minimum effort any Map Provider we want to include in the real implementation. When available, the received result is forwarded to the user trough the SMSMapServerProxy. If the user is in a closed space the MapManager will contact an IndoorMapManager after a quite complex interaction with other components. Details on this procedure can be found in the subsequent section.

Another core function of the system is the ability to provide geocoding functionalities. For this purpose we utilize the interaction between the SearchManager, the remote SMSSerchesServerProxy and the Geocoders. Considering outdoor scenarios in the implementation we actually use a “mashup” of the Google Maps Local searches, also used to retrieve driving directions. In case of
indoor localization the geocoding resolution is replaced by a search among the points of interest defined for the local area. This is possible because the indoor maps are actually Map objects. They have a layer structure, the bottom layer is the 2D map image, the upper layer is a graph that represent the topology of the space. The Map objects are complex data structures that are aggregates of an image (e.g. .jpg, .png) and a Property file in the form of a .txt file that contains the space properties with respect to the given area a building or a floor. This latter is hidden to the user, but is mandatory to help the system providing mapping and guidance support. All the geocoding requests return the result inside Position objects as position information container.

The system enables the user to find out his own location as well as the location of other peers. To this purpose a central Server, namely the “SMSCommunityServer”, is needed to collect, store and aggregate the location information about registered users. Depending on the utilized positioning method, if the positioning is terminal based the client application periodically updates on the Server the user position otherwise in case of network based systems we configure the infrastructure to report the position to the CommunityServer.

This way positional information becomes available to be exploited for social community services such as “Friend Finder”. Moreover using a publish/subscribe mechanisms the user could be alerted about content related to their current geographical position or for example prompted with a TODO list when he reaches a specified location (a sort of Geocoded Post-It).

2 SMS Localization and Navigation procedures

In this section we will provide a more detailed description of the typical interaction between the architectural components residing on the user equipment and the proposed server side architecture, trying to explain how we realize some of the functionalities described in the previous paragraph.

2.1 Localization in Indoor Places

When a user enters an “Indoor Place” like a building, an Airport or a University, where there is an SMS Indoor localization System he can use the Indoor Navigator module running on his mobile phone to discover, search and exploit local services

In the Indoor scenarios the user can be in two different status:

1) Connected to the network but not localized because there is not an SMS localization service framework available in the indoor place.

2) Connected to the network and localized.

The overall localization architecture is shown in figure in Figure 1-2.

2.1.1 Short intro to the SMILE middleware layer

In order to understand the procedures for Indoor Localization, we need to introduce the SMILE layer of the SMS architecture. SMILE is the communication middleware used in the SMS architec-
ture. Its main purpose is to facilitate the development of distributed applications [20]. The SMILE middleware is based on a layered approach which foresees a common abstraction layer and a “binding” into a specific communication mechanism, such as RMI [30], Web Services and SIP [28].

An SMS component that wants to offer a service will register itself to the “yellow page” service logically provided by the SMILE layer. The components publish their services with the name of the interface they offer, i.e. this is equivalent to declaring a “Port type” in WSDL [29]). The SMILE Yellow Page service search operation is used to retrieve the “contact point” of other processes exposing a given service. The contact point is a “Process ID” in terms of the SMILE middleware. The default binding mechanism that is considered in the SMS architecture is based on JSON [31] for the serialization of messages and on SIP for the transport of messages. Therefore, we can refer to SMILE-JS (SMILE-JS/JSON/SIP) to characterize the SMS communication middleware. Within SMILE-JS, a Process ID takes the form of a SIP URI: sip:user@domain;ptype=process-type;pname=process-name. Within SMILE-JS, the Yellow Page service is realized by a server called Yellow Pages Server.

### 2.1.2 Indoor localization procedures

When a user arrives in an Indoor place there are several preliminary “back stage” operations that the SMS components have to do interacting each other to be able to offer to the end user localization based services. The Navigation Dispatcher component has to detect which is the actual positioning technology in use sending a request to the Localization Provider Component invoking the “CurrentActiveTechnology” operation. The Localization Provider forwards this request to the Positioning Technology Controller components. Assuming that the localization technology in use inside the building is a Wi-Fi connection, the Localization Provider returns this information to the Navigation Dispatcher so it can start and activate the Indoor Navigator component.
Figure 2-1 Indoor Places - Find the SMSIndoorLocalizationServer
The Indoor Navigation component contacts a *Yellow Pages Server* using a SMILE Connection (see Figure 2-1). The Yellow Pages server address is already known by the device (i.e. it is stored into the configuration properties file of the Indoor Application). The Yellow Pages server provides the *IndoorLocalizationBroker* address to the Indoor Navigator component. (Currently we consider that there is only one Yellow Pages server and only one Localization Broker). Once the Indoor Navigator has obtained the Broker address, he can contact it by means of a SMILE-JS connection giving, as input a proper set of information to filter the response coming back. To let server know which AP is the origin of the request, a parameter associated to it is sent to the server. In our case, the physical address of the AP (MAC Address) is transmitted coupled with its GPS Coordinates if the latter data is available. Note that we are considering the case in which the user gets connected to the system across a wifi AP. Of course, the location of this access point has to be known by the system. One way of implementing this is to maintain a simple database, typically per hotspot provider, containing the location of every access point.

In principle, in order to communicate the AP location to the client device, one option would be to store in the access point itself this data, and sent it to the mobile terminal, for example by extending the DHCP server that is already commonly embedded in access points. This was not the approach that we have followed. Actually in our implementation we use the Access Point MAC address and an *IndoorLocalizationBroker* that maintains association between the AP and Coordinates. The Indoor Localization Broker will identify the SMILE address of the specific *SMSLocalizationServer* which is in charge to handle the localization and navigation functionalities on the basis of the AP originating the request and its location. As we already said the server has a database mapping
the APs physical addresses to their location and assigns an SMSServer to each zone represented by an AP and its coordinates.

Another way to achieve the GPS coordinates identifying the place in which the user is, can be obtained scanning a visual tag.

We assume that the mobile client application supports the Physical Mobile Interaction techniques: Touching (using NFC)[4], Pointing (using the recognition of visual markers) and Direct Input of simple identifiers[5][4] (see Figure 2-2). Ongoing work include the realization of several additional components like: “Semacode” to scan visual tags. The Localization Broker could use this information to select the correct SMSServer who can give a Navigation Service coverage to the user. If in the building there is not an available Indoor Navigation service the Localization Broker will notify this to the Indoor Navigator so the user will be in the state “connected but not localized”. In this case the user can anyway obtain a map of the building and use in a “static” fashion the Indoor Application. In fact he can fill in a form with its approximate present location as simple symbolic representation (the building name) or use a visual tag, to download the proper map from a Map Server.

The Indoor Navigator uses one of those information to request the map to the Map Manager component. This component verifies if the map of the building is already stored into the local storage space of the device (RMS memory or external memory card) and if he cannot find the map asks the SMS Map Server Broker for the address of the Map Server Proxy. This operation is useful also if a user wants to visualize a building map from a remote position.
Figure 2-3 Indoor Places – Find an indoor Map using Manual input information

If the Navigation Service is available (“connected and localized” condition) the Localization Broker sends the SMS Localization Server address to the Indoor Navigator. This component contacts the Localization Provider component asking it the user position. The Localization Provider asks this information to the SMS Localization Server by means of the Wi-Fi Controller or the Zigbee Controller component including the Client identifier that depending on the positioning system we are using could be the NIC MAC address (Wi-Fi localization method) or the Zigbee tag identifier. The SMS Localization Server contacts the right positioning server, asking for the user position. The Position Server returns the relative x and y coordinates of the user localized on a specific map and the Map Url. The SMS Localization Server sends the coordinates and the URL back to the Indoor Navigator going through the Wi-Fi or the Zigbee Controller and the Localization Provider component.

The Indoor Navigator sends the message to the Map Manager component and the component verifies if the map is already stored in the device, using the Map Url. If the map is not already stored, the Map Manager contacts the Indoor Map Server sending him the Map Url so he can download the map in a jpeg, png format and the map properties text file.
2.1.3 UML Sequence diagrams for indoor localization

Focusing on the indoor localization mechanisms, the basic steps towards a localization service discovery are now discussed.

First and foremost, it is necessary to know whether the current wireless technology (e.g. WiFi) is active or not in the building (e.g. if the user associated to a WiFi access point); as we can see in Figure 2-5, in order to accomplish this task the IndoorNavigator makes an appropriate request to the LocalizationProvider, which will respond with a confirmation/denial. This answer contains some important information, e.g. the WiFi AccessPoint MAC address and, in case, its GPS coordinates. This information may be later used as a filter, to let the IndoorNavigator discover and later exploit the most appropriate SMSLocalizationServer, responsible for the SMS services in the building; that is to say that generally many SMSLocalizationServers are present, since a specific one is only responsible for a specific domain (e.g. this domain may be a campus, a building, or even a floor), although the functional architecture picture presents only one SMSLocalizationServer entity.
We may also say that the LocalizationProvider could be able to select the current active technology between multiple available current positioning methods basing his decision on parameters like power requirements, the resolution in terms of time and space accuracy.

**Figure 2-5 Discover Active Localization Technology**

The next step concerns the discovery of the localization service. The IndoorNavigator has to obtain a SMSLocalizationServer URL with all the information contained in the previous LocalizationProvider response. According to SMILE procedures, the IndoorNavigator has to consult a yellow pages server, here named YPServer, to obtain the address of the IndoorLocalizationBroker; it is assumed that the YPServer URL is initially known by the MoveClient (see Figure 2-6).
Generally there could be many brokers, or many localization servers, or many yellow pages servers, etc: these entities are distributed, and this is a general statement regarding all the functional entities in the localization architecture.

It is now time for the IndoorNavigator to retrieve an appropriate map, which has to be compliant with the current user position. As far as this task is concerned, the MapManager is the staple entity, since it encompasses almost the entire maps handling logic, including their visualization on the device display.

A map is identified by an ID, that is an absolute reference to the effective location of the map, e.g. http://160.80.80.47/Engineer/Elec_Eng/3fl/Bsect/map.png, where http://160.80.80.47 is the URL of the MapServer that stores the needed map.

**Figure 2-6 Retrieve the SMSLocalizationServerUrl**
In figure 2-7 we can see that the SMSLocalizationServer sends the map ID back to the IndoorNavigator; then the IndoorNavigator forwards this message to the MapManager, which verifies if the map is already stored in the device, by means of the Map ID. If the map is not already stored, then the MapManager contacts the IndoorMapServer thanks to the retrieved URL, so that it
can download the map object, that is the image itself in a jpeg or png format and the map properties text file, which is important for the map visualization and navigation logic. The MapManager has to manage the map visualization on the device display, with the proper zoom level and with the user represented as a point centered on the screen.
2.2 Switching Outdoor/Indoor contexts

To let the reader better understand the details of this paragraph we have to make a little digression on a new and original concept introduced by the SMS: the Mobile Electronic Memo (MEM). A MEM is an electronic “note”, containing a structured set of attributes associated with a specific class of information (e.g. information describing a location, a person, a service, or a Web site). From the end-user point of view, a MEM appears as an icon which can be expanded into a “card” showing a certain number of fields associated with a specific context. MEMs will be readable both by humans and by computer applications. This means that users can browse their MEMs for useful information (e.g. the address of the restaurant) or pass them to MEM-enabled applications which use the information in the MEM to provide a special service navigation services which guide the user to a location mentioned in the MEM (“Drive Me To” functionality); and mapping services which show the location mentioned in the MEM on a map.

Considering a “Drive Me To” destination functionality in general we will have to deal with the seamless switching between indoor and outdoor contexts while the user is on the move. Exploit the integrated indoor and outdoor localization software modules, integrated into a complete navigation
system is possible thanks to the NavigationDispatcher and the Position objects. Following we present a suitable scenario involving this switching mechanism.

*Mr Jones has booked a medical consult on Policlinico Tor Vergata booking page; then the booking system has sent a MEM to his mobile device. This MEM contains all the details of his booking, such as:*

<table>
<thead>
<tr>
<th>Prenotazione per: Sig. Jones, Michael</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visita cardiologica</td>
</tr>
<tr>
<td>Dott. Rossi Mario</td>
</tr>
<tr>
<td>Reparto cardiologia, Piano 3, Stanza n. 127</td>
</tr>
<tr>
<td>Edificio C, Policlinico Tor Vergata</td>
</tr>
<tr>
<td>Via del Politecnico 1</td>
</tr>
<tr>
<td>Roma, RM 00133</td>
</tr>
<tr>
<td>Italia</td>
</tr>
</tbody>
</table>

Visualizing this MEM on his mobile device, Mr Jones wants to reach the doctor’s studio, so he clicks on the “driveMeTo” functionality from the options menu, having previously turned on his Bluetooth GPS.

Generally, there could be a macroscopic Indoor→Outdoor→Indoor sequence, but in this case we reduce the complexity only to an OUT→IN scenario, having in mind that a similar procedure may be applied to an IN→OUT scenario and generalized, by subsequent repetitions, to the IN→OUT→IN scenario.

Coming to the decision of seamlessly switching from an indoor (ZigBee/WiFi) to an outdoor (GPS) localization technology, or viceversa, is a NavigationDispatcher task; this transition is realized by passing through the so called gateways. If we think to a building we can identify a gateway with the door through which a person goes from the building internals to the open space. The NavigationDispatcher has to detect the current active technology through the LocalizationProvider; in this case it finds that a GPS technology is available. The NavigationDispatcher keeps memory of the target position, that is an indoor relative position expressed as (x,y) cartesian coordinates inside an indoor map. The MEM that was received actually hides two Position objects: the first has the civic part that contains a human readable description of the target, formatted according to the CivicAddress part of the Position object, and the indoor part that contains the cartesian coordinates of the target referred to the indoor map they have to be mapped on. The second one contains the outdoor gateway position expressed as (lat,long) values plus the indoor gateway position expressed as (x,y) values, plus a radius that gives the extension of the gateway area (e.g. 3 mt). Having found that the current active technology is an outdoor one, the NavigationDispatcher starts the OutdoorNavigator giving as parameter of the driveMeTo operation the gateway outdoor position. The OutdoorNavigator retrieves the current user position from the GPS; having the start position and the gateway position, it is able to lead the user to the building entrance. As the gateway outdoor position has been reached, the OutdoorNavigator sends a message to the NavigationDispatcher to notify it that the task has been completed.
The NavigationDispatcher now has to compare the currently reached position and the target position, to see whether the current (lat,long) values belong to the gateway area. If this comparison yields an affirmative result, then the NavigationDispatcher knows that an OUT→IN switch has to be executed. At this point the NavigationDispatcher has to detect again the current active technology through the LocalizationProvider; if there is an active indoor localization service, then the NavigationDispatcher starts the IndoorNavigator. Having the indoor gateway cartesian coordinates and the indoor target cartesian coordinates, the IndoorNavigator is able to lead the user to the desired destination, exploiting the indoor localization service and the retrieved map along with its properties.
**Figure 2-9 DriveMeTo - Outdoor/Indoor switching**
2.3 Virtual localization system

There is a further important consideration about the absence of an indoor localization system, since it can be assumed that nowadays the main localization system that generally may be available is a GPS system, that is an outdoor technology. For now the spread of indoor localization services and technologies is not wide, therefore when a user is in a building and wishes to exploit an indoor localization service, in 90% of cases that indoor localization service might be indeed not available; it is the outdoor localization system that has to provide, somehow, all the information needed to roam inside that building.

As described in the above paragraph, the system tries to discover a localization service, resulting this time unavailable. Then the IndoorNavigator prompts the user that none localization system is available and asks him if he wishes to make the application try to find the indoor map he needs. In other words, the user has to exploit a so called virtual indoor localization system. There are now different possible mechanisms to retrieve the needed map, e.g.:

1. The IndoorNavigator asks the YPServer for a MapServerBroker URL, which is able to obtain a list of likely IndoorMapServers once given loose data by the user, such as “Athens Airport” or “St. James Hospital”. The MapServerBroker has to semantically interpret the data given by the user to return, in a best-effort fashion, a list of likely IndoorMapServers that may contain the requested map. The client side will visualize a list of maps “friendly names”, which actually hide the effective absolute map ID (e.g http://160.80.80.64/Europe/Greece/Athens/Athens_Airport.png). This approach gives the user the responsibility to choose the right map, or even none map if he finds out that the map he is demanding is not present in that list. If the user selects a map, then the MapManager will download it and display it on the client screen.
2. Let the user be associated to a Wi-Fi Access Point belonging to the Wi-Fi network of the current indoor place. Having the Access Point MAC address, the IndoorNavigator, through the YPServer, finds the SMS_ServicesServerProxy URL, that is able to geocode the MAC address and returns a Position object containing the MAC address itself along with its related geographic coordinates. The IndoorNavigator then looks through the YPServer for the MapServerBroker URL, which will return a list of IndoorMapServers suitable for the given geographic coordinates. The user will be prompted with this list, from which he can select the right IndoorMapServer URL (hidden behind a friendly map name). The MapManager will contact this IndoorMapServer to download the map, being even able to set and visualize the current user position, considered coincident with the position of the Access Point, using the geocoded Access Point MAC that the user is connected to. This identical procedure may be executed exploiting a GSM Cell ID.

3. Let the user use a Visual Tag which embeds the map URL. The MapManager is directly able to download the map from the proper IndoorMapServer. The Visual Tag could even give a relative position, which can be mapped into the downloaded image by the MapManager. However, the Visual Tag could only contain the map reference, and in this case the MapManager will display the map centered on the center of the map itself.

This virtual localization system can be applied as well to an outdoor scenario, in which the user cannot rely on a suitable outdoor localization technology, e.g., a GPS system. The user manually inserts the information describing his current position (e.g., “Rome” or “Via delle gondole”, etc.), which are then geocoded by the outdoor application exploiting, for example, the “findAPlace” operation; having now the geocoded information, the outdoor application is able to set and visualize the current user position on a proper map.

2.4 Virtual and real location context

What particularly marks out our Localization API is a new use of the local storage providing a persistent database of Position objects. This shifts the emphasis very much onto the mobile client in terms of location aware applications, enabling a local mapping from physical positions to symbolic locations [10] and the handling of so called real and virtual location context. Position objects (see following paragraphs for details on the definition and implementation of the Position object) are aggregates of AddressInfo and geographic Coordinates objects. Position objects can have different roles in the architecture. They are transitory, so might be updated over time, reflecting the dynamic and real movement of a mobile device when are used to store position information coming from a positioning system in tracking scenarios. In contrast they can be intended to be persisted in the mobile data store and used by the user to provide coordinates or merely places (University of “Tor Vergata” in Rome) for applications to exploit LBS services setting them as virtual current position without relying on any actual positioning technology. We can think to a user which is in Rome but running the application he sets as his current position a place elsewhere using Position objects saved yet in a bookmarks fashion (they help users find the same place again..) or creating them on the fly.
2.5 Realizing Outdoor features

We now consider the realization of outdoor location based services. The target was to realize services optimized for being used on mobile browsers. Just to give few examples in this category: “Google Mobile”, “Google Maps Mobile”, Yahoo Mobile. A first market trend considers the usage of “mobile browsers”, trying to replicate the very successful paradigm of the “fixed” web on the mobile devices. The advantage of this solution is that in principle the services would be accessible to every mobile device with a browser. The problems are that the mobile browser market is much more fragmented than the PC browser market and that mobile browsers still lack the capability (like Javascript support) that makes the PC browser a powerful engine to support very advanced services. Therefore the second trend, which is currently gaining momentum is to develop dedicated applications on the mobile device for the mobile services, examples being the “Yahoo Go” application or the location based applications within Nokia recent high end phones (e.g. N95). Downsides of this approach are that there will be a plethora of different applications that will interoperate and that the service creation process will be “proprietary” i.e. each application will be under the control of its developer.
With reference to the above discussion on “browser” vs. “application” approaches, the project has chosen the latter one, but it is proposing an open-specification and open source approach in which the specification of the component interfaces is given and the implementation of components is released under an open source license.

Locating, mapping, routing and navigating functionalities as well as business and amenity searching involve interaction between our client application running on the user equipment and a server side counterpart exploiting an HTTP client-server communication. This latter we developed in the form of gateways toward legacy web based services. In fact we have realized a “mashup” of a web based system: the MOVE application is connected to a “proxy” that in turns is able to access exiting web location based services. These services are adapted in terms of page content to the characteristics of the mobile devices.

3 SMS Localization and Navigation Components

This section identifies and describes the SMS Localization and Navigation Components and their operations. Following there is a more detailed explanation in which we explode Component’s functionality, their interfaces and operations.

We stress here that the Localization components defines and offer generic interfaces for positioning that is intended to work with a wide range of positioning technologies that can bee also simply added, though extensions are allowed for specific purposes. The SMS navigation and localization architecture has been defined in a modular way.

The role of the SMS Navigation/Localization Components are to allow a generic SMS Component to retrieve information about the user position and to provide guidance information to the terminal. The SMS Nav./Loc. components are the interface towards navigation and localization services and reside on the Mobile Terminal and on the Network. They implements the operation of the “Interact with navigation/Localization SW” component described in [3]. The SMS Navigation/Localization Components expose an API to the “SMS Generic Components” tailored to build location based applications.

We underline to the reader the difference between interfaces exposed to Components, and the different nature of what we call the “EndUserInterfaces”. This interfaces describe the operations that are used to allow the end user to requests functionalities choosing them from a graphic and menu based interface. Actually this way to represent these functionalities offered to the end user is not a proper way of doing it because we give to the functionalities that the user can exploit pressing softkey buttons and menu options on his mobile phone the same meaning we gave to the functionalities that components in the architecture can exploit.

From Figure 3-1 you can gain a complete vision of this powerful architecture we have designed and concretely implemented. From the picture you also may visualize which are the interaction between Components trough the exposed interfaces.
3.1 **Client-Side Components**

3.1.1 **SMSCore**

This component is a specific example of a generic SMSCore component.

3.1.1.1 **EndUserCoreInterface**

This interface is used to allow the user to request operations choosing them from a graphic and menu-based interface.

**Figure 3-1 UML model of the architectural Components**
1. **startNavigatorOp (OneWay)**

This operation is used to Start the Navigator Application specifying which one to use.

**IN startNavigatorReq**

| navigatorType | String the specific navigator that has to be used |

**NOTES:**

2. **displayPathOp (OneWay)**

This operation is used to display, with a proper zoom level fitted to contain the wall path, the route from a beginning point to an end point.

**IN displayPathOp**

| startPosition | Position the path start position |
| endPosition   | Position the path end position |

**NOTES:**


This operation is used to display (centered) on a proper outdoor/indoor map the user current position retrieved from the positioning technology in use. If the user is not localized the operation can return a fault or the last saved position specifying that this information is not up to date.

**IN displayUserCurrentPosition**
NOTES:

4. driveMeToDestinationOp (OneWay)
This operation is used to save the GPS antenna from which we will retrieve data from the BT GPS connected with the device. This mechanism requires a search of the BT visible devices, the pairing, the selection of the device to use. All this steps are under the user direct control.

IN configureGPS

NOTES:

5. activateGPS Op (OneWay)
This operation is used to activate the GPS connected with the mobile devices. If the configuration procedure has not yet be done this operation will ask the user to previous configure the antenna to use.

IN activateGPS

3.1.2 NavigationDispatcher
This component is used to select, in a transparent access fashion, the proper Navigation Application (Outdoor application, Indoor application) to start in order to handle the task that the generic SMScomponent is requesting taking into account the goal the component is trying to achieve.
3.1.2.1 SMSNavigationDispatcherInterface

This interface is used to expose the functionalities that other Components can exploit when they need to communicate with the localization Applications and Positioning systems and to forward the requested operation to the proper Localization Component.

1. displayGenericPositionOp (OneWay)

IN displayGenericPosition

| position | Position [1..*] the set of positions that have to be shown on the map with the proper zoom level to display them all |

NOTES:

2. SendMem (Notification)

This operation is used to allow the requester to see on a display the position referred to a map of a generic thing. The operation can also show on the map multiple objects taking into account that the display have to be done taking into account the screen size in order to be able to put an the screen all the requested objects.

NOTES:

3. displayPathOp (OneWay)

This operation is used to display the route from a beginning point to an end point, with a proper zoom level fitted to contain the wall path.

IN displayPathOp
Location Based Services architecture
for Simple Mobile Services

<table>
<thead>
<tr>
<th>startPosition</th>
<th>Position the path start position</th>
</tr>
</thead>
<tbody>
<tr>
<td>endPosition</td>
<td>Position the path end position</td>
</tr>
</tbody>
</table>

NOTES:


This operation is used to display (centered) on a proper outdoor/indoor map the current user position retrieved from the positioning technology in use. If the user is not localized the operation can return a fault or the last saved position specifying that this information is not up to date.

**IN displayUserCurrentPosition**

NOTES:

5. **driveMeToDestinationOp (OneWay)**

This operation is used to guide the service requester from his current position to the chosen destination by asking it to the technology in use.

**IN driveMeToDestination**

| position | Position the position that the user has to be led to |

NOTES:

6. **targetPositionReachedOp (Notification)**
This operation is used to notify the requester of an operation that is using localization and navigation functionalities like “Drive me to” that the localization component has completed his task.

**OUT targetPositionReached**

| msg | String the notification message |

**NOTES:**

7. **startNavigatorOp (OneWay)**

This operation is used to Start the Navigator Application specifying which one to use.

**IN startNavigatorReq**

| navigatorType | String the specific navigator to use |

**NOTES:**

8. **stopNavigatorOp (OneWay)**

This operation is used to Stop a specific Navigator Application.

**IN stopNavigatorReq**

| navigatorType | String the specific navigator to stop |

**NOTES:**
9. navigatorClosedOp (Notification)
This operation is used to allow the notification to a component that a navigator application was closed. This is useful to handle situations in which the component has to do some other operations.

OUT navigatorClosed

| msg | String the notification message |

3.1.3 Navigator
This service provides information about places.

3.1.3.1 SearchForAddress
This interface is used to look for the address of a given place or activity. It uses a simple Filter object which keeps two parameters:

Filter {
    String name
    Enumeration kind: PLACE, ACTIVITY
}

The response contains a number of matching activities or places, expressed as bookmarks (which can be stored or shown on a map)

TDB describe the Bookmark object here

1. LookFor (RequestResponse)
returns an array of Bookmarks matching the filter criteria

IN LookForRequestMessage
Location Based Services architecture
for Simple Mobile Services

what Filter

OUT LookForResponseMessage

pin[1..*] Bookmark

2. StoreBookmark (RequestResponse)
Allow to store a set of Bookmarks

IN StoreBookmarkRequestMessage

bks[1..*] Bookmark The set of Bookmarks which have to be stored provided as an array
token String An authorization token

OUT StoreBookmarkResponseMessage

3. RetrieveBookmark (RequestResponse)
Allow to retrieve a set of Bookmarks

In order to select which Bookmarks are to be returned, a filter is used; the filter criteria can be specified as a parameter of the request. **TDB describe the filter object**

IN RetrieveBookmarkRequestMessage

filter BookmarkFilter A filter uses to retrieve a list of Bookmarks
4. DeleteBookmark (RequestResponse)
Allow to delete a set of Bookmarks

In order to select which Bookmarks are to be returned, a filter is used; the filter criteria can be specified as a parameter of the request. *TDB describe the filter object*

**IN DeleteBookmarkRequestMessage**

<table>
<thead>
<tr>
<th>filter</th>
<th>BookmarkFilter</th>
<th>A filter uses to identify which Bookmarks are to be deleted</th>
</tr>
</thead>
<tbody>
<tr>
<td>token</td>
<td>String</td>
<td>An authorization token</td>
</tr>
</tbody>
</table>

**OUT DeleteBookmarkResponseMessage**

| deleted | boolean        |

### 3.1.3.2 DirectionsService

*An interface providing support for getting direction; typically it is used to navigate from one place to another.*

*TDB define the 'step' object*

**1. GetDirections (RequestResponse)**
Get driving, walking or cycling directions from one point to another
the origina and destination parameter are passed as strings, e.g. 'via del Politecnico, 1 Turin' or 'Piazza Navona Roma'; anyway they can also contain explicit coordinates like '40.938000, 14.288230', but always in form of string

**IN GetDirectionsRequestMessage**

| fromString | &nbsp |
| to | String &nbsp |
| kind | ENUMERATION could be WALKING, CYCLING, DRIVING |

**OUT GetDirectionsResponseMessage**

| step[1..*] | Step a list of steps driving from the origin to the destination |

### 3.1.3.3 Display

An interface providing support for display objects related to a given location using maps.

1. **DisplayDirections (OneWay)**
   
   Allow to display directions on a map

   **IN DisplayDirectionsMessage**

   | step[1..*] | Step a list of steps driving from the origin to the destination |

2. **DisplayBookmark (OneWay)**

   Allow to display bookmarks on a map
IN DisplayDirectionsMessage

| bks[1..*] | Bookmark | The list of Bookmarks which are to be displayed |

3.1.4 LocalizationProvider

The Location Provider component handles queries for the location of a device and orchestrates the business logic that manage the communication with different localization technologies hidden to the requester. Actually generic interfaces allow to exploit different sources of location information at the same time being able to choose between multiple methods of determining the location of a single device.

3.1.4.1 TargetPositionNotification

This interface is used to subscribe, unsubscribe and receive notifications when the user reaches a given position.

1. subscribeTargetPosition (RequestResponse)

Allow to subscribe to the event "the user has reached a target position" and receive a consequent notification

IN SubscribeTargetPositionRequest

<table>
<thead>
<tr>
<th>targetPosition</th>
<th>Position</th>
<th>The position the user has to reach</th>
</tr>
</thead>
<tbody>
<tr>
<td>expiration</td>
<td>DateTime</td>
<td>An optional parameter useful to limit the subscription within a given timeframe</td>
</tr>
</tbody>
</table>
Location Based Services architecture for Simple Mobile Services

**OUT SubscribeTargetPositionResponse**

| result | Boolean | A boolean signalling whether the subscription have been successfully |

**FAULT SubscribeTargetPositionFault**

| reason | String | The reason why this operation couldn't have been carried out |

2. **unsubscribeTargetPosition (RequestResponse)**

Cancel the subscription

*A void message is sent back to confirm the unsubscription.*

**IN UnsubscribeTargetPositionRequest**

| targetPosition | Position | The position the user has to reach |

**OUT UnsubscribeTargetPositionResponse**

3. **notifyReachedTargetPosition (Notification)**

This operation is used to notify subscriber that the user has reached the target position

**OUT NotifyReachedTargetPositionMessage**
Location Based Services architecture
for Simple Mobile Services

| targetPosition | Position | The position the user has reached |

3.1.4.2 UserTracking

This interface is used to subscribe, unsubscribe and receive notifications related to the tracking service, which informs about the user’s current position

1. subscribeTracking (RequestResponse)

Allow to subscribe to the user tracking

IN SubscribeTrackingRequest

| expiration | DateTime | An optional parameter useful to limit the subscription within a given timeframe |

OUT SubscribeTrackingResponse

| result | Boolean | A boolean signalling whether the subscription have been successfully |

FAULT SubscribeTrackingFault

| reason | String | The reason why this operation couldn't have been carried out |

2. unsubscribeTracking (RequestResponse)

Cancel the subscription

A void message is sent back to confirm the unsubscription.

IN UnsubscribeTrackingRequest
OUT UnsubscribeTrackingResponse

3. NotifyUserPosition (Notification)
This operation is used to notify the current user position to the subscriber

OUT NotifyUserPositionMessage

<table>
<thead>
<tr>
<th>currentPosition</th>
<th>Position</th>
<th>The current user position</th>
</tr>
</thead>
</table>

4. getUserPosition (RequestResponse)
Get the current user position

IN getUserPositionRequest

OUT getUserPositionResponse

<table>
<thead>
<tr>
<th>currentPosition</th>
<th>Position</th>
<th>The current user position</th>
</tr>
</thead>
</table>

3.1.5 GPSController

This service is used to manage all the operations that require a communication between the user device and the BT GPS that a user can use to exploit a GPS positioning method in Outdoor spaces.
3.1.5.1 GPSInterface

TDB describe the Bookmark object here

1. ConfigureGPSOp (OneWay)
This operation is used to start and complete the BT devices pairing and discovering procedure at the end of which if a BT GPS antenna has been found the user can save it and use it calling the activateGPS operation.

IN configureGPS

NOTES:

2. ActivateGPSOp (OneWay)
This operation is used to handle the connection to the BT GPS attached to the user device to retrieve NMEA sentences.

IN activateGPS

NOTES:

3. GetPositionOp (RequestResponse)
This operation is used to ask for position information fill in the Position objects data fields with raw data retrieved from the NMEA sentences
Position can be combined with other information to provide personalized applications and services.

**IN GetPositionReq**

<table>
<thead>
<tr>
<th>requesterType</th>
<th>String</th>
</tr>
</thead>
</table>

**OUT GetPosition**

<table>
<thead>
<tr>
<th>position</th>
<th>Position[1]</th>
</tr>
</thead>
</table>

**FAULT GetPositionFault**

<table>
<thead>
<tr>
<th>faultReason</th>
<th>String</th>
</tr>
</thead>
</table>

### 3.1.6 GSM_CellIDController

This service is used to handle all the operations required to retrieve positioning information using GSM CellID.

#### 3.1.6.1 GSM_CellIDInterface

TDB describe the Bookmark object here

1. **GetPositionOp** *(RequestResponse)*

This operation is used to ask for position information in return we obtain a Position object with data fields filled with raw data retrieved from the specific technology
IN GetPositionReq

requesterType | String

OUT GetPositionRes

position | Position[1]

FAULT GetPositionFault

faultReason | String

3.1.7 ZigbeeController

This service is used to handle all the operations required to retrieve positioning information from a Zigbee positioning System.

3.1.7.1 ZigbeeInterface

1. GetPositionOp (RequestResponse)

This operation is used to ask for position information in return we obtain a Position object with data fields filled with raw data retrieved from the specific technology

IN GetPositionReq
Location Based Services architecture
for Simple Mobile Services

requesterType | String

OUT GetPosition
Res

position | Position[1]

FAULT GetPositionFault

faultReason | String

3.1.8 Wi-FiController

This service has the task to handle a user localization query done with a Wi-Fi positioning system.

3.1.8.1 Wi-FiInterface

1. GetPositionOp (RequestResponse)
This operation is used to ask for position information in return we obtain a Position object with data fields filled with raw data retrieved from the specific technology

IN GetPositionReq

requesterType | String

OUT GetPositionRes
3.1.9 MapManager

This service is used to retrieve the maps needed to accomplish tasks related with display functionalities of a single position, list of positions, routes etc. It is aimed to manage images that can be retrieved from a remote MapServer or from a local Storage space (RMS memory, external memory card).

3.1.9.1 MapManagerInterface

1. MapManagerOp (RequestResponse)

The component receives a request directly from a Navigation application (Outdoor/Indoor).

This request contains the parameters that allow this component to return the proper map or a list of maps. If the request is coming from the Outdoor application it will contain Lat and Long values. These data will be translated in a suitable format that is compatible with the specific remote MapServer we will contact to download the tile set (e.g. if we use Google Map we have to transform Lat and Long in a TileID that contains this point) or to find the images if we have the tiles in the local storage saved with the specific Map Server format corresponding name. If we are using the Indoor Localization System we will have relative coordinates and the ID of the Map representing the place we are in. Moreover in the response we obtain even the distance related to the top-left of the screen on which we want to display the position.
### Position

<table>
<thead>
<tr>
<th>position</th>
<th>Position [1..*]</th>
</tr>
</thead>
<tbody>
<tr>
<td>screenSize</td>
<td>int[2]</td>
</tr>
<tr>
<td>requesterType</td>
<td>String</td>
</tr>
</tbody>
</table>

### OUT MapManagerRes

<table>
<thead>
<tr>
<th>map</th>
<th>Map [1..*]</th>
</tr>
</thead>
<tbody>
<tr>
<td>top_left_offset</td>
<td>int[2]</td>
</tr>
</tbody>
</table>

### FAULT MapManagerFault

| faultReason | String |

### NOTES:

#### 3.1.9.2 SMSMapManagerInterface

1. **MapManagerOp (RequestResponse)**

The component receives a request from a generic SMSComponent.

*This request contains the parameters that allow this component to return the proper map or a list of maps. If the request is coming from the Outdoor application it will contain Lat and Long values. These data will be translated in a suitable format that is compatible with the specific remote MapServer we will contact to download the tile set (e.g. if we use Google Map we have to transform Lat and Long in a TileID that contains this point) or to find the images if we have the tiles in the local storage saved with the specific Map Server format corresponding name. If we are using the Indoor Localization System we will have relative coordinates and the ID of the Map representing the place we are in. Moreover in the response we obtain even the distance related to the top-left of the screen on which we want to display the position.*
3.1.10 SearchesManager

With this service we realize searches of places, category of businesses and driving directions relying on the interaction between this entity and a remote Geocoder. The end user or the generic SMSComponent provide to this entity the parameter of the search (e.g. a specific address or merely a place).

3.1.10.1 SearchesManagerInterface

1. SearchesManagerOp (RequestResponse)
The component receives a request directly from the Outdoor Navigation application.

*Dependent on the type of search we are requesting (1-a place, 2-a business category near somewhere, 3-driving directions to go from a start point to an end point) the “search” array will be filled differently. In case 1) the array will encompass only a single address, in case 2) the address concatenated with the business category and in the latter case the start and end position. The searchType indicates which operation is requested and so enable the proper formatting of the request String to send to the remote SMSSearchesProxy.*

**IN SearchesManagerReq**

<table>
<thead>
<tr>
<th>search</th>
<th>String[2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchType</td>
<td>String</td>
</tr>
</tbody>
</table>

**OUT SearchesManagerRes**

<table>
<thead>
<tr>
<th>position</th>
<th>Position [1..*]</th>
</tr>
</thead>
</table>

**FAULT SearchesManagerFault**

<table>
<thead>
<tr>
<th>faultReason</th>
<th>String</th>
</tr>
</thead>
</table>

**NOTES:**

3.1.10.2 SMSSearchesManagerInterface

**1. SearchesManagerOp (RequestResponse)**

The component receives a request from a generic SMSCOMPONENT.
Depending on the type of search we are requesting (1-a place, 2-a business category near somewhere, 3-driving directions to go from a start point to an end point) the “search” array will be filled differently. In case 1) the array will encompass only a single address, in case 2) the address concatenated with the business category and in the latter case the start and end position. The searchType indicates which operation is requested and so enable the proper formatting of the request String to send to the remote SMSSSearchesProxy.

**IN SearchesManagerReq**

<table>
<thead>
<tr>
<th>Search</th>
<th>String[2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchType</td>
<td>String</td>
</tr>
</tbody>
</table>

**OUT SearchesManagerRes**

<table>
<thead>
<tr>
<th>Position</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>[1..*]</td>
</tr>
</tbody>
</table>

**FAULT SearchesManagerFault**

<table>
<thead>
<tr>
<th>faultReason</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>faultReason</td>
<td>String</td>
</tr>
</tbody>
</table>

### 3.2 Server-Side Components

#### 3.2.1 SMSSSearchesServerProxy

This service is used to manage Geocoding requests contacting a Specific Geocoder service.
3.2.1.1 SMSSearchesServerProxyInterface

This interface receives geocoding requests from remote entities and return position objects whose fields are properly filled with the result of the queries that this component forward to specific Geocoders (e.g. Google Maps, Yahoo local searches…)

1. GetGeocodedSearchesOp (RequestResponse)

This operation is used to

IN GetGeocodedSearchesReq

| search | String [0..*] |

OUT GetGeocodedSearchesRes

| position | Position [0..*] |

FAULT GetGeocodedSearchesFault

| faultReason | String &nbsp |

3.2.2 SMSMapServerProxy

This service realizes the mashup interacting with the web based map server. Infact it is responsible for forwarding the request to the specific map server after having translated it according to a suitable format that depends on which server we rely on. Moreover it can support a caching features that avoids to request the maps each time in real-time from the map server. As for indoor maps this entity uses the MapID coming from the MapManager request to find the map or the mapset needed. This operation can involve forwarding the request to another mapserver or return directly to the client the requested image/s because the SMSServerProxy has it/them in his own map database.
3.2.2.1 SMSMapServerProxyInterface

1. GetMapOp (RequestResponse)
This operation is used to obtain a map from a specific Map Server.

The MapDetails are the proper parameters that depend on the server we rely on and we have to specify when formatting the query to select the needed image.

IN GetMapReq

| mapDetails | MapDetails [1..*] |

OUT GetMapRes

| map | Map [0..*] |

FAULT GetMapFault

| faultReason | String |

3.2.3 SMSCommunityServer

When SMS Nav./Loc. application has acquired the localization information from whatever localization technology, it can make this information available to other SMS components (always respecting the user privacy policies). In particular there can be servers that collect the user localization information and make it available for services like finding buddies in the neighbours, or to associate advertising information to users’ current location.
3.2.3.1 SMSCommunityServerInterface

1. GetPositionOp (RequestResponse)
This operation is used to ask for position information fill in the Position objects data fields with raw data retrieved from the NMEA sentences

Position can be combined with other information to provide personalized applications and services.

IN GetPositionReq

requesterType String

OUT GetPositionRes

position Position[1..*]

FAULT GetPositionFault

faultReason String

NOTES:

2. UpdatePositionOp (OneWay)
This operation is used to ask for position information fill in the Position objects data fields with raw data retrieved from the NMEA sentences

IN UpdatePosition
3.2.4 **MapServerBroker**

*This service is used to obtain the Indoor Map Server Address URL*

3.2.4.1 **MapServerBrokerInterface**

1. *GetIndoorMapServerAddressOp (RequestResponse)*

This operation is used to instantiate a request to the Map Server Broker that allows to obtain the Indoor Map Server Address.

**IN GetIndoorMapServerAddressReq**

<table>
<thead>
<tr>
<th>position</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1..*]</td>
<td></td>
</tr>
</tbody>
</table>

**OUT GetIndoorMapServerAddressRes**

| IndoorMapServerAddress | String [1..*] |

**FAULT GetIndoorMapServerAddressFault**

<table>
<thead>
<tr>
<th>faultReason</th>
<th>String</th>
</tr>
</thead>
</table>
3.2.5 IndoorMapServer

This service uses the MapID (e.g. the URL of the image) coming from the MapManager request to find the map or the mapset needed. This operation can involve forwarding the request to another mapserver or return directly to the client the requested image/s because the SMSServerGateway has it/them in its own map database.

3.2.5.1 IndoorMapServerInterface

1. GetMapOp (RequestResponse)

This operation is used to obtain a map from a server.

*If we are using the Indoor Localization System we will have the mapID and the URL of the specific Map Server. This URL can be the SMS Map Server Gateway address or a generic map server URL. In this case the server will use the URL to contact the right Map Server. The operation returns the map with its properties file.*

IN GetMapReq

```

| mapDetails | MapDetails [1..*] |
```

OUT GetMapRes

```

| map | Map [0..*] |
```

FAULT GetMapFault

```

| faultReason | String |
```
3.2.6 IndoorLocalizationBroker

This service is used to obtain the SMS Localization Server Address when the Indoor Navigation component needs to know if a localization service is offered in a specific location such as a building.

3.2.6.1 IndoorLocalizationBrokerInterface

This interface is used for the search of a specific SMS Server.

TDB describe the Bookmark object here

1. GetLocalizationServerAddressOp (RequestResponse)

This operation is used to instantiate a request to the Indoor Localization Broker that allows to obtain the SMS Localization Server Address. The Localization Broker makes a search using for example the Access Point Address as a filter.

The Localization Broker makes a search using some localization device or building geographical position information as a filter.

IN GetLocalizationServerAddressReq

| filters       | String[1..*] The localization device or building geographical position information |

OUT GetLocalizationServerAddressRes

| ServerDispURL | String The SMS Localization Server URL |

FAULT GetLocalizationServerAddressFault
3.2.6.2 IndoorLocalizationBrokerPublishInterface

This interface is used to register SMS Servers and devices on the database.

1. LocalizationServerDispUrlOp (OneWay)
Allows to register a SMS Localization Server and the localization devices.

IN LocalizationServerPublishUrl

| ServerDispUrl | String The SMS Localization Server Url |
| DevicesParameters | String[1..*] Information about the localization infrastructure devices |

3.2.7 SMSLocalizationServer

The Indoor modules need an external “Nav/Loc Coordinator Server” which cooperates with the plugin running in the terminal to track the position of the user. Thanks to the interaction between the local Plugin and the external SMSLocalizationServer, the SMS system is able to evaluate the spatial information about an SMS user (e.g. User A has a position described with x,y coordinates on the .jpg Map whose ID is z). It can be provided with multiple interface active at a particular time, for instance GPRS and WLAN interfaces, then a device can be connected to both networks and it can switch between them according to parameters, such as the best signal quality or better area coverage. Moreover it contains the association between a MapID and the URL of the server to contact to retrieve the specific map. This URL can be obtained using the lookup mechanism offered by the YellowPages service on which IndoorMapServers can publish their service.

3.2.7.1 SMSLocalizationServerInterface
1. GetPositionOp (RequestResponse)

This operation is used to ask for position information in return we obtain a Position object with data fields filled with raw data retrieved from the specific technology.

**IN GetPositionReq**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>technologyType</td>
<td>String</td>
</tr>
<tr>
<td>technologyEntityID</td>
<td>String</td>
</tr>
</tbody>
</table>

**OUT GetPositionRes**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>Position</td>
</tr>
<tr>
<td></td>
<td>[1..*]</td>
</tr>
</tbody>
</table>

**FAULT GetPositionDispFault**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>faultReason</td>
<td>String</td>
</tr>
</tbody>
</table>

3.2.8 Wi-FiPositionServer

This service act as a coordinator for the Wi-Fi positioning system. It maintains the association between a user and his relative position in the indoor space. This means that using a suitable method to calculate the position of a user (position triangulation using at least three APs, or the association between the AP that the user is attached to and its coordinates) it can return on request the x,y coordinates that identify the real position of the requester and the mapID that contains these coordinates.

3.2.8.1 Wi-FiPosServerInterface
**1. GetEntityPositionOp (RequestResponse)**

This operation is used to ask for position information in return we obtain a Position object with data fields filled with raw data retrieved from the specific technology

### IN GetEntityPositionReq

<table>
<thead>
<tr>
<th>technologyEntityID</th>
<th>String</th>
</tr>
</thead>
</table>

### OUT GetEntityPositionRes

<table>
<thead>
<tr>
<th>position</th>
<th>Position</th>
</tr>
</thead>
</table>

### FAULT GetEntityPositionFault

<table>
<thead>
<tr>
<th>faultReason</th>
<th>String</th>
</tr>
</thead>
</table>

**3.2.9 ZigbeePositionServer**

A typical Zigbee localization system is based on a set of reference Zigbee nodes, a Zigbee coordinator node and of course by the set of Zigbee devices to be localized (we can use a Zigbee chip on the SIM, on the terminal or on an external small portable device). These set of elements cooperate to evaluate the user position. Interestingly, the final user localization can be done on the Zigbee device to be localized, yielding very scalable solutions. The Zigbee device could be able to communicate directly with the terminal (e.g. it could have Bluetooth connection to the terminal) or it could communicate its position to the Zigbee coordinator. The Zigbee coordinator will be linked to an SMS server side entity that acts a location provider and will be able to distribute this information to the SMS terminal.

**3.2.9.1 ZigbeePosServerInterface**
1. GetEntityPositionOp (RequestResponse)
This operation is used to ask for position information in return we obtain a Position object with data fields filled with raw data retrieved from the specific technology

IN GetEntityPositionReq

| technologyEntityID | String |

OUT GetEntityPositionRes

| position | Position |

FAULT GetEntityPositionFault

| faultReason | String |

4. The SMS “Position Object”

A PositionObject is a Location information container. The Position Object should be able to include both address information expressed in “human readable form” and geographic coordinates within a reference system like GPS coordinates. A location may also contain details such as the location’s postal address, country, URL. We can say that our notion of Position objects is something like aggregates of data we can simply mark as addresses and geographic coordinates. Through this Position objects we create records of information that we can split into two main parts: the first one we can call the “AddressInfo” part and the other one the “Geographic Coordinates” part.

It is possible to assign a label to a position object, something like a meaningful position name description. The label can help us to determine the correspondence between latitude and longitude (e.g. Lat:41.54332; Long:12,1234) coordinates the real point on the earth of them in a simple readable fashion. This way we avoid to think that this point is mapped in the middle of a virtual land because we have marked it with the label e.g. “Grocery”. Moreover supposing that we may want to
share with somebody this location information having assigned a label the point let these data become a meaningful information even for others.

Position Objects can have different roles in the architecture. They can be transitory, in the sense that they might be updated at time intervals, reflecting the real movement of a mobile device when are used to store position information coming from a positioning system in tracking scenarios. In contrast they can be intended to be persistent data in the mobile memory storage and used by the user to provide coordinates or just places (University of Tor Vergata, Rome) for applications in order to exploit LBS services, setting them as virtual current position without relying on any actual positioning technology. For our Localization Architecture we needed an object that could contain a semantic meaning coupled with geographic data like the place’s name, its friendly description, etc.

4.1 SMS Position object tags

As a first step we identified some SMS position object tags that appeared to be mandatory in an initial proposed definition of this location information container. The first list enumerates the tags we identified as address tags.

<table>
<thead>
<tr>
<th>SMS AddressInfo Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>NATION</td>
</tr>
<tr>
<td>CITY</td>
</tr>
<tr>
<td>ZIP CODE</td>
</tr>
<tr>
<td>STREET ADDRESS</td>
</tr>
<tr>
<td>FRIENDLY-NAME</td>
</tr>
<tr>
<td>AREA TYPE</td>
</tr>
<tr>
<td>ZONE</td>
</tr>
<tr>
<td>BUILDING</td>
</tr>
<tr>
<td>POSTAL CODE</td>
</tr>
</tbody>
</table>

Table 4-1 SMS AddressInfo Position objects tags

As for the common notion of street or nation we can skip the explanation of the meaning of these fields and offer a simple overview of the tags that we may consider not so familiar.

Following a more precise explanation of some proposed tags.

AreaType : examples of AreaType include shopping centre, University, transport station…

Friendly name : used to define an address with a free text (e.g. Mum’s office, 5 Minutes walk from home…)

As we said above our Position object are aggregates of addresses and geodetic fields so we needed tag to maintain the geodetic part of the information.
Since the location information is connected to the accuracy concept, such as the extension of the area covered by the address we saved in the “AddressInfo” records and time and temporal references to the instant on which we filled in our information container we added some supplementary tags to the aforementioned SMS tags that we have listed yet. Time attributes are useful in situations in which mobile terminals cannot rely on actual available positioning systems. A solution should allow access to the last known location but having in mind that it can be outdated. The Age tag is a reference to the instant in which we filled the Position object information. Moreover, we can consider even the temporal validity of an address and in this sense, Temporal Validity we mean the start and end date of validity of a geocoded address.

Following the list of these supplementary tags

<table>
<thead>
<tr>
<th>SMS Supplementary tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELIABILITY ON POSITION (E.G. RADIUS)</td>
</tr>
<tr>
<td>TEMPORAL VALIDITY</td>
</tr>
<tr>
<td>AGE</td>
</tr>
</tbody>
</table>

4.2 Overview of related standards

Our second step in the Position object definition was to find possible mappings of our records to existing fields defined in other location definition formats. Following we list the above identified tags and their binding with other Standards that handle the location information. In the table below you will find SMS tags and the other location information container standards we examined: OASIS Customer Information Quality (CIQ) xAL(extensible Address Language)[11] an international standard for address formatting, vCard [17] standard and the Civic address part of the SIP-PIDF LO (Location Object)[15], the extension to the SIP-IETF PIDF(Presence Information Data Format).

AddressInfo tags

<table>
<thead>
<tr>
<th>SMS</th>
<th>OASIS xAL</th>
<th>vCard ADR</th>
<th>SIP-PIDF LO – civicAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>NATION</td>
<td>COUNTRY</td>
<td>COUNTRY</td>
<td>COUNTRY</td>
</tr>
<tr>
<td>CITY</td>
<td>LOCALITY</td>
<td>CITY/LOCALITY</td>
<td>A3</td>
</tr>
<tr>
<td>ZIP CODE</td>
<td>POSTALCODE</td>
<td>POSTAL CODE</td>
<td>PC</td>
</tr>
</tbody>
</table>
Location Based Services architecture
for Simple Mobile Services

<table>
<thead>
<tr>
<th>STREET ADDRESS</th>
<th>THOROUGHFARE</th>
<th>STREET</th>
<th>A6.PRD,POD,STS,HNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRIENDLY-NAME</td>
<td>ADDRESSLINE (PROPOSED MAPPINGS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AREA TYPE</td>
<td>FIRM / PREMISE (PROPOSED MAPPINGS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZONE</td>
<td>DEPENDENTLOCALITY</td>
<td>EXTENDED INFORMATION (?)</td>
<td>A4</td>
</tr>
<tr>
<td>BUILDING</td>
<td>PREMISE/SUBPREMISE (PROPOSED MAPPING)</td>
<td></td>
<td>BLD</td>
</tr>
<tr>
<td>POSTAL CODE</td>
<td>PostBox</td>
<td>P.O. box</td>
<td>POBOX</td>
</tr>
</tbody>
</table>

Table 4-4 SMS AddressInfo Position objects tags and their relation with other standards

In the subsequent paragraphs we will deepen in the details of examining the relation of our defined tags and the relation of them with the Standards we considered.

Note that in italic there are our proposed mapping between some xAL tags and the identified SMS tags with respect to the fact that Countries have very different address formats and the xAL format offers not univoque possible bindings. In fact address formatting is outside the scope of xAL so is up to the specific application that uses this address language to decide how to combine the xAL tags to form a Country specific address.

**Geospatial tags**

<table>
<thead>
<tr>
<th>SMS</th>
<th>GML</th>
<th>KML</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS coordinates</td>
<td>GML Point</td>
<td>KML Point</td>
</tr>
</tbody>
</table>

Table 4-5 SMS geospatial Position objects tags and their binding with other standards

We will go deepen in GML (Geographic Markup Language)[19] and KML (Keyhole Markup Language)[14] details in the following paragraphs but we anticipate that this two markup languages are used to describe geographic coordinates using geometric basic types (i.e. Point, Lines, Polygons).

**Supplementary tags**

<table>
<thead>
<tr>
<th>SMS</th>
<th>xAL</th>
<th>vCard</th>
<th>SIPF-PIDF LO</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELIABILITY ON POSITION (E.G. RADIUS)</td>
<td>ADDRESSDETAILS ACCURACY</td>
<td>ADDRESSDETAILS ACCURACY</td>
<td>ADDRESSDETAILS ACCURACY LEVE</td>
</tr>
</tbody>
</table>
4.2.1 xAL (Extensible extensible Address Language)

4.2.1.1 Existing xAL specification

Investigating the xAL standard and his real usages we found that the Google Map on-line Geocoder Services uses the xAL language to express the geocoded locations he returns to users requests. So to have an idea of how much the xAL tags covered our defined records determining as a possible solution to use xAL Standard for our project purposes we tried to geocode some addresses with the Google Map Geocoder and see the result of the geocoding request.

Following we show two different geocoding examples obtained requesting a geocoding service to the Google Map Geocoder that as we said above formats the given address to xAL. These example will also show the different way how different Country Addresses are modelled using xAL. We stress that as is stated in[18] the order in which the contents of xAL is combined to form a legal address is country specific.

The geocoding service offered By Google Maps returns as a possible output an xml format.

LEGENDA: The underlined tags show xAL (extensible Address Location) OASIS[11] not used by Google geocoder. The italic ones address xAL elements with country specific format.

| 23 ARCHER STREET | VIA DEL POLITECNICO 1 |
| CHATSWOOD, NSW 2067 | ROMA, RM 00133 |
| AUSTRALIA | ITALIA |

```xml
<Placemark id="p1">
  <AddressDetails Accuracy="8">
    <Address>
      23 ARCHER ST, CHATSWOOD, NSW 2067, AUSTRALIA</Address>
    <Country>
      <CountryName>AU</CountryName>
      <CountryNameCode>AU</CountryNameCode>
    </Country>
    <CountryName>ITALY</CountryName>
    <CountryNameCode>IT</CountryNameCode>
  </AddressDetails>
</Placemark>
```

Table 4-6 SMS Supplementary Position objects tags and their binding with other standards

<table>
<thead>
<tr>
<th>TEMPORAL VALIDITY</th>
<th>VALIDFROMDATE/VALIDDATE</th>
<th>GML : TIMEPERIOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGE</td>
<td></td>
<td>GML : TIMEINSTANT</td>
</tr>
</tbody>
</table>
Following the output obtained specifying as a parameter in the Google geocoding service request URI: `output = json`[16].

We show also this output format because in our system architecture we use JSON (JAVA Serialization Object Notation) in inter Components Communication, so have an output in this format let us have quickly something well suited to fit in our messages structure. We skip the overview about json to present it in a subsequent paragraph.
Table 4-8 Two geocoding examples using Google Maps geocoder service: output format in JSON

<table>
<thead>
<tr>
<th>Address 1</th>
<th>Address 2</th>
</tr>
</thead>
</table>
| **23 ARCHER STREET**  
**CHATSWOOD, NSW 2067**  
**AUSTRALIA**  

```
{
  "NAME":"23 ARCHER STREET, CHATSWOOD, NSW 2067, AUSTRALIA",
  "STATUS":{
    "code":200,
    "REQUEST":"GEOCODE"
  },
  "PLACEMARK":{
    "ID":"p1",
    "ADDRESS":"23 ARCHER ST, CHATSWOOD, NSW 2067, AUSTRALIA",
    "ADDRESSDETAILS":{
      "COUNTRY":{
        "COUNTRYNAMECODE":"AU",
        "ADMINISTRATIVEAREA":{
          "ADMINISTRATIVENAMENEW":"NSW",
          "LOCALITY":{
            "LOCALITYNAME":"CHATSWOOD",
            "THOROUGHFARE":{
              "THOROUGHFARENAMENEW":"23 ARCHER ST"
            },
            "POSTALCODE":{
              "POSTALCODENUMBER":"2067"
            }
          }
        }
      },
      "ACCURACY":8
    },
    "POINT":{
      "COORDINATES":{
        151.187245,
        -33.799442,
        0
      }
    }
  }
```

<table>
<thead>
<tr>
<th>Address 2</th>
<th>Address 2</th>
</tr>
</thead>
</table>
| **VIA DEL POLITECNICO 1**  
**ROMA, RM 00133**  
**ITALIA**  

```
{
  "NAME":"VIA DEL POLITECNICO 1, ROMA, RM 00133 ITALIA",
  "STATUS":{
    "code":200,
    "REQUEST":"GEOCODE"
  },
  "PLACEMARK":{
    "ID":"p1",
    "ADDRESS":"VIA DEL POLITECNICO, 00133 ROMA, RM (LAZIO), ITALY",
    "ADDRESSDETAILS":{
      "COUNTRY":{
        "COUNTRYNAMECODE":"IT",
        "ADMINISTRATIVEAREA":{
          "ADMINISTRATIVENAMENEW":"LAZIO",
          "SUBADMINISTRATIVEAREA":{
            "ADMINISTRATIVENAMENEW":"RM",
            "LOCALITY":{
              "LOCALITYNAME":"ROMA",
              "THOROUGHFARE":{
                "THOROUGHFARENAMENEW":"VIA DEL POLITECNICO"
              },
              "POSTALCODE":{
                "POSTALCODENUMBER":"00133"
              }
            }
          }
        }
      },
      "ACCURACY":6
    },
    "POINT":{
      "COORDINATES":{
        12.618403,
        41.854170,
        0
      }
    }
  }
```

*Table 4-8 Two geocoding examples using Google Maps geocoder service: output format in JSON*
We can notice that the Address tag is defined in xAL as a sub-element of “AddressDetails” element that is used to define a general address. The Google geocoder use the Address tag as a tag with the same hierarchy as addressDetails.

### 4.2.1.2 Using xAL tags as a possible format for our Position object

The lesson we learned from the geocoding of simple addresses like the ones we presented in the previous paragraph was that xAL well suits to cover our SMS defined tags. A further step now is to do the same thing with a more complex address that has a more precise and detailed structure. The result we found was that we have different possible choices to map some specific address parts into xAL fields due to the flexibility of the xAL address formatting alternatives. Moreover we note that xAL lacks of what is concerned with the “Age” tag we defined as a mandatory tag for our Position object. We will now see where xAL can help us to express Position objects and where we have to use other means to find a complete Position object formal definition.

#### 4.2.1.2.1 Premise, SubPremise and Firm tags

Now we see a more detailed example taking into account how we may use xAL to format addresses that have a more complex structure than the previous ones we examined.

Edificio dell’informazione, PIANO 4, Stanza n 7452

Facoltà di ingegneria, Università di Roma “Tor Vergata”

Via del Politecnico 1

Roma, RM 00133

Italia

We identified three xAL elements that can cover the address parts that the elements we found in the geocoding of the simple address request don’t cover: **Premise, SubPremise and Firm**.

As stated in[18] the Premise Element is used by the following xAL elements:
- Locality
- DependentLocality
- Thoroughfare
- Premise (recursive).

This element is a container and has sub-elements to define the Premise in an address. It can occur multiple times and is optional (0 or more). Inside a Premise tag we can have the SubPremise Element that can keep the other address parts that identify low level hierarchies references. In the table below we show how we can format the examined address using the “Premise” tag and his nested SubPremise element we can use to express the “Department” and the “room” concepts. The Type attribute we can add to the Premise and SubPremise elements help us to differentiate the type of these records.

Examples of premise include house, building, shopping centre, transport station, etc
Summing up a possible mapping is:

- UNIVERSITY = PREMISE.
- DIPARTIMENTO = SUBPREMISE
- ROOM = SUBPREMISE

An alternative way of formatting the same address identify the University as an AreaType that we can map in the “Firm” xAL tag keeping the Premise tag only to maintain the building name.

As stated in [18] Firm element is used by the following elements:
- POBox
- Thoroughfare
- Premise
- SubPremise.

Possible values include company, university, shop, etc.

The proposed mapping between our tags and the OASIS xAL tags are:

- BUILDING = PREMISE.
- UNIVERSITY = FIRM
- ROOM = SUBPREMISE

So two different proposals of different way of handling “nested elements” and as example the above complex addresses using the Premise tag and the Firm tag is shown below.

```
<PREMISE TYPE="Building">
    <BUILDINGNAME>EDIFICIO
dell’INFORMAZIONE</BUILDINGNAME>
    <SupPREMISENUMBER>4</SupPREMISENUMBER>
</PREMISE>
<SubPREMISETYPE="PIANO"/>

<FIRM TYPE="Università">
    <FIRMNAME>UNIVERSITÀ DI ROMA “TOR VERGATA</FIRMNAME>
    <DEPARTMENT>
        <DEPARTMENTNAME>FACOLTÀ DI INGEGNERIA</DEPARTMENTNAME>
    </DEPARTMENT>
</FIRM>
</PREMISE>

<PREMISE TYPE="Università ">
    <PREMISENAME>UNIVERSITÀ DI ROMA “TOR VERGATA</PREMISENAME>
    <SubPREMISETYPE="DEPARTMENT">
        <SubPREMISENAME>FACOLTÀ DI INGEGNERIA</SubPREMISENAME>
    </SubPREMISETYPE>
    <SubPREMISETYPE="PIANO">
        <SubPREMISENUMBER>4</SubPREMISENUMBER>
    </SubPREMISETYPE>
</PREMISE>
<SubPREMISENUMBER>7452</SubPREMISENUMBER>
</FIRM>
</PREMISE>
```

Table 4-9 Proposed alternative mappings between xal tags and a complex structure address
4.2.1.2.2 SMS Friendly-name tag and the xAL AddressLine

For what concerns the SMS Friendly Name tag we propose a mapping to the xAL tag: **AddressLine** that is used to define addresses as a free format text.

Hence we can add to our Position object fields a specific tag called `<AddressLine>` that can be intended as a generic name that the User can use to identify a specific position object. So for an example location such as

Edificio dell’informazione, PIANO 4, Stanza n 7452, Facoltà di ingegneria, Università di Roma “Tor Vergata”, Via del Politecnico 1, Roma, RM 00133, Italia

The proposed mapping let us define this address with the help of the AddressLine tag:

```xml
<AddressLine>Prof. Nicola Blefari’s Office</AddressLine>
```

4.2.1.2.3 GPS Coordinates and the xAL Geospatial tag

For what is concerned with the geodetic part of our Position object investigating the xAL Standard we found the Geospatial tag.

SMS GPS Coordinates tag should be specified within the Geospatial tag. In xAL standard this element has as descendant geospatial features described by the GML language.

Geospatial tag is a container that references the entire GML namespace leaving it to the adopters to decide how they apply GML.

```xml
<Geospatial>
  <gml:Point xmlns:gml="http://www.opengis.net/gml">
    <coordinates>12.618403,41.854170,0</coordinates>
  </gml:Point>
</Geospatial>
```

4.2.1.2.4 xAL and the Temporal concepts

Now we will examine the time and temporal aspects:

- **SMS Temporal validity**

In xAL Standard we have the tags `ValidFromDate` and `ValiToDate` that define the start and the end date of the validity of an address. In our opinion these elements well fit the SMS Temporal validity meaning.

- **Age**

Unfortunately there aren’t mappings between SMS Age tag and xAL elements infact in the xAL Documentation we found: "*xAL only defines the XML vocabulary to represent addresses*".

Lacking xAL with this information we though to a solution using the temporal values we can retrieve if we have a GPS device connected to our mobile phone and a suitable parser for NMEA sen-
tences that the GPS device transmits. In fact, we the temporal information are embedded in NMEA sentences. The reader that is unfamiliar with NMEA sentences can get more information at [12]. If we take a look at the $GPGGA and $GPRMC sentences shown below, we can see specific fields which carry the UTC (Coordinated Universal Time). Our implemented navigation application will describe in the implementation part of this work deals in fact with these two sentences. This information compared with the current time retrieved from the client System Properties make us able to define a so-called “Age” field with Application-specific characteristics.

Below there is an example of a $GPGGA and of a $GPRMC sentences that come out from a GPS device from which we can retrieve temporal parameters.

$GPGGA,011016,3240.1041,N,09727.3700,W,1,06,0.84,207.5,M,-23.5,M,,*4A

Field 1: GPGGA
Field 2: Time, in form HHMMSS, in UTC.

$GPRMC,011018,A,3240.1042,N,09727.3699,W,0.114,77.1,230803,5.6,E*5D

Field 1: GPRMC
Field 2: Time, in form HHMMSS, in UTC
Field 10: Date, in form DDMMYY
(Note that the date goes with the time in UTC). 230803 = Aug 23, 2003

4.2.1.3 Conclusions about xAL

After a detailed analysis the OASIS xAL Standard seemed enough flexible to allow a customized XML mapping of a specific address format. This flexibility provided by xAL allows us to choose our own representation format but after having a look to his so much nested and hierarchical structure we decided to make our representation simpler and to adopt a Flat structure. Therefore we put our interest in SIP –PIDF LO that allows to define addresses in simple flat format.

4.2.2 GML (Geographic Markup Language) and KML (Keyhole Markup Language)

As we stated in the initial part in which we listed all the SMS Position object mandatory tags and also in the xAL paragraph we identified in the GML language the standard that we can use to embed our geodetic data. In this section we will give an overview of this language and also we will present the related KML language which is a GML Application schema used by Google Map in his geocoding service.

4.2.2.1 GML

The GML[13] (Geographic Markup Language) language is the de facto open standard to exchange data between GIS (Geographic Information System) systems and for the representation of a geo-
graphic information data. GML is an XML geospatial vocabulary defined by Geospatial Consortium (OGC) which can be shared across a community of interest.

4.2.2.2 GML Profiles: Point geometry

Profiles are logic restrictions to GML vocabulary. These Profiles are defined to simplify the GML usage and are specified by GML specifications. For our project purposes the basic geometry structure we will use is the Point geometry.

- **Point Profile** for applications with a point type dataset

Profiles are different from application schemas (see subsequent section) and GML namespaces, they define a GML subset.

GML Point Profile has:

```xml
<PhotoCollection xmlns="http://www.myphotos.org"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.myphotos.org
MyGoodPhotos.xsd">
  <items>
    <Item>
      <name>Lynn Valley</name>
      <description>A shot of the falls from the suspension bridge</description>
      <where>North Vancouver</where>
      <position>
        <gml:Point srsDimension="2" srsName="urn:ogc:def:crs:EPSG:6.6:4326">
          <gml:pos>49.40 -123.26</gml:pos>
        </gml:Point>
      </position>
    </Item>
  </items>
</PhotoCollection>
```

We stress that using **Profile Point**, the unique geometry is `<gml:Point>`.

4.2.2.3 GML Application schemas

As in [19] in order to expose an application's geographic data with GML, a community or organization creates an XML schema specific to the application domain of interest (the application schema). This schema describes the object types whose data the community is interested in and which community applications must expose. For example, an application for tourism may define in its application schema object types like: monuments, places of interest, museums and sightseeing. Those object types reference the object types defined in the GML standard. This is the way KML language does.
4.2.2.4 KML (Keyhole Markup Language)

KML [14] (Keyhole Markup Language) is an XML based language created to handle geospatial data in 3D dimension with application programs like Google Earth, Google Maps e Google Mobile. The kml.xsd schema file specify a set of elements (geographic bookmarks, images, polygons, 3D models, label texts and descriptions...) to be visualized with Google Earth, Map and Mobile. KML instead defines geographic objects and their styling and their graphical representation.

Each location has a mandatory field that specify a longitude and a latitude. Google Maps merges some KML elements that are copied from GML and for the civic part uses a small group of tags taken from xAL (extensible Address Location) OASIS standard.

4.2.2.5 KML and Google Maps

Google Maps supports the subsequent KML elements:

- Placemarks
- Icons
- Folders
- Descriptive HTML
- KMZ (compressed KML, including attached images)
- Polylines and polygons
- Styles for polylines and polygons, including color, fill, and opacity
- Network links to import data dynamically
- Ground overlays and screen overlays

A list of the KML Elements is presented in the Figure below
We noticed that a fair number of tags can be used to define our Position Object schema. These tags that we will describe hereafter are:

- **Point**
- **Placemark**
- **TimeSpan**
- **TimeInstant**

**Point**

A Point is a geographic position with a related longitude, latitude and altitude (optional). When a Point is inside a Placemark the Point itself determines the location of the Placemark. Hereafter an example of a KML Point inside a Placemark tag.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.1">
```
<Placemark>
  <name>Simple placemark</name>
  <description>Attached to the ground. Intelligently places itself at the height of the underlying terrain.</description>
  <Point>
    <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
  </Point>
</Placemark>

4.2.2.6 KML and the xAL namespace

The way KML include the xAL Standard to describe the address information part of a position is the Placemark element. A Placemark infact is a Feature with an associated geometry. Inside the Placemark tag we find the xAL Address tag that keep nested in it all the xAL elements we described in the previous paragraphs.

Following the schema of a Placemark that shows the relation between this KML element and the xAL Standard.

<TimeSpan id="ID">
  <!-- inherited from Feature element -->
  <name>...<name> <!-- string -->
  <visibility>1</visibility> <!-- boolean -->
  <open>1</open> <!-- boolean -->
  <address>...</address> <!-- string -->
  <AddressDetails xmlns="urn:oasis:names:tc:ciq:xsd:xsdschema:xAL:2.0">...</AddressDetails> <!-- string -->
  <phoneNumber>...</phoneNumber> <!-- string -->
  <Snippet maxLines="2">...</Snippet> <!-- string -->
  <description>...</description> <!-- string -->
  <LookAt>...</LookAt>
  <TimePrimitive>...</TimePrimitive>
  <styleUrl>...</styleUrl> <!-- anyURI -->
  <StyleSelector>...</StyleSelector>
  <Region>...</Region>
  <Metadata>...</Metadata>
  <!-- specific to Placemark element -->
  <Geometry>...</Geometry>
</Placemark>

TimeSpan

We found that the kml.xsd contains a set of useful definitions for time related objects and properties.

Following we show the kml:TimeSpan schema syntax:

<TimeSpan id="ID">
  <begin>...</begin> <!-- kml:dateTime -->
  <end>...</end> <!-- kml:dateTime -->
</TimeSpan>
TimeSpan represents an extent in time bounded by begin and end dateTimes.

The dateTime is defined according to XML Schema time. The value can be expressed as yyyy-mm-ddTh:mm:sszzzzzz, where T is the separator between the date and the time, and the time zone is either Z (for UTC) or zzzzzz, which represents ±hh:mm in relation to UTC. Additionally, the value can be expressed as a date only. The following example shows the timeSpan representing Colorado's statehood. It contains only a <begin> tag because Colorado became a state on August 1, 1876, and continues to be a state:

```xml
<Placemark>
  <name>Colorado</name>
  ...
  <TimeSpan>
    <begin>1876-08-01</begin>
  </TimeSpan>
</Placemark>
```

TimeStamp

A TimeStamp is an instant in time. It has a single child (property) called when/ timePosition. A TimeStamp is thus written:

```xml
<TimeStamp id=ID>
  <when>...</when>      <!-- kml:dateTime -->
</TimeStamp>
```

- **dateTime** second resolution
- **date** day resolution
- **gYearMonth** month
- **gYear** year

Hereafter some instances of the TimeStamp element:

- **gYear (YYYY)**
  ```xml
  <TimeStamp>
    <when>1997</when>
  </TimeStamp>
  ```

- **gYearMonth (YYYY-MM)**
  ```xml
  <TimeStamp>
    <when>1997-07</when>
  </TimeStamp>
  ```

- **date (YYYY-MM-DD)**
  ```xml
  <TimeStamp>
    <when>1997-07-16</when>
  </TimeStamp>
  ```

- **dateTime (YYYY-MM-DDTh:mm:ssZ)**
4.2.2.7 Comparison between KML and GML

GML is not a presentation language - a language for geographic data presentation. It is not designed as mapping language or intended for graphical display. It leaves this to other grammars like SVG. It is designed to represent geographic objects. KML instead defines geographic objects and their styling and their graphical representation. This observation represent the main difference between GML and KML languages to which we have to add that the KML language also, by the mean of including the xAI namespace can describe even the human readable part of a location. Regarding the overlap between the GML and KML elements following we put a table that lets the reader note this aspect.

<table>
<thead>
<tr>
<th>KML_Elements</th>
<th>GML_Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;address&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;altitudeMode&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;begin&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;color&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;coordinates&gt;</td>
<td>&lt;coordinates&gt;</td>
</tr>
<tr>
<td>&lt;description&gt;</td>
<td>&lt;description&gt;</td>
</tr>
<tr>
<td>&lt;Document&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;drawOrder&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;east&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;end&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;extrude&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;fill&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;Folder&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;geomColor&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;GeometryCollection&gt;</td>
<td>&lt;GeometryCollection&gt;</td>
</tr>
<tr>
<td>&lt;geomScale&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;GroundOverlay&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;h&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;heading&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;href&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;Icon&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;IconStyle&gt;</td>
<td></td>
</tr>
<tr>
<td>&lt;innerBoundaryIs&gt;</td>
<td>&lt;innerBoundaryIs&gt;</td>
</tr>
<tr>
<td>Tag</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>&lt;latitude&gt;</td>
<td>Latitude coordinate</td>
</tr>
<tr>
<td>&lt;longitude&gt;</td>
<td>Longitude coordinate</td>
</tr>
<tr>
<td>&lt;labelColor&gt;</td>
<td>Label color</td>
</tr>
<tr>
<td>&lt;NetworkLink&gt;</td>
<td>Network link</td>
</tr>
<tr>
<td>&lt;north&gt;</td>
<td>North coordinate</td>
</tr>
<tr>
<td>&lt;ObjArrayField&gt;</td>
<td>Object array field</td>
</tr>
<tr>
<td>&lt;ObjField&gt;</td>
<td>Object field</td>
</tr>
<tr>
<td>&lt;open&gt;</td>
<td>Open</td>
</tr>
<tr>
<td>&lt;overlayXY&gt;</td>
<td>Overlay XY</td>
</tr>
<tr>
<td>&lt;Pair&gt;</td>
<td>Pair</td>
</tr>
<tr>
<td>&lt;parent&gt;</td>
<td>Parent</td>
</tr>
<tr>
<td>&lt;Placemark&gt;</td>
<td>Placemark</td>
</tr>
<tr>
<td>&lt;Point&gt;</td>
<td>Point</td>
</tr>
<tr>
<td>&lt;Polygon&gt;</td>
<td>Polygon</td>
</tr>
<tr>
<td>&lt;PolyStyle&gt;</td>
<td>Poly-style</td>
</tr>
<tr>
<td>&lt;range&gt;</td>
<td>Range</td>
</tr>
<tr>
<td>&lt;refreshInterval&gt;</td>
<td>Refresh interval</td>
</tr>
<tr>
<td>&lt;refreshVisibility&gt;</td>
<td>Refresh visibility</td>
</tr>
<tr>
<td>&lt;rotation&gt;</td>
<td>Rotation</td>
</tr>
<tr>
<td>&lt;Schema&gt;</td>
<td>Schema</td>
</tr>
<tr>
<td>&lt;screenXY&gt;</td>
<td>Screen XY</td>
</tr>
<tr>
<td>&lt;SimpleArrayField&gt;</td>
<td>Simple array field</td>
</tr>
<tr>
<td>&lt;SimpleField&gt;</td>
<td>Simple field</td>
</tr>
<tr>
<td>&lt;size&gt;</td>
<td>Size</td>
</tr>
<tr>
<td>&lt;south&gt;</td>
<td>South</td>
</tr>
<tr>
<td>&lt;snippet&gt;</td>
<td>Snippet</td>
</tr>
<tr>
<td>&lt;style&gt;</td>
<td>Style</td>
</tr>
<tr>
<td>&lt;StyleMap&gt;</td>
<td>Style map</td>
</tr>
<tr>
<td>&lt;styleUrl&gt;</td>
<td>Style URL</td>
</tr>
<tr>
<td>&lt;tessellate&gt;</td>
<td>Tessellate</td>
</tr>
<tr>
<td>&lt;tilt&gt;</td>
<td>Tilt</td>
</tr>
<tr>
<td>&lt;TimeSpan&gt;</td>
<td>Time span</td>
</tr>
<tr>
<td>&lt;TimeStamp&gt;</td>
<td>Time stamp</td>
</tr>
<tr>
<td>&lt;type&gt;</td>
<td>Type</td>
</tr>
</tbody>
</table>
From the table is clear that the KML TimeSpan and TimeInstant are the same as GML TimeInterval and TimeInstant. This latter can be found in the GML temporal.xsd schema (see http://schemas.opengis.net/gml/3.1.1/base/temporal.xsd).

### 4.2.2.8 Using GML/KML to support our Position Object

Our first proposal was to use GML for what is concerned with geodetic related objects and properties and to exploit xAL fields to express Addresses. An another alternative is to use the KML tags in a similar way as Google Maps does. This solution was after discarded because we decided to adopt for the “AddressInfo” tags a flat structure that simplify the too nested and complex structure offered by the xAL lStandard. Hence as a new solution we came out with an other choice taking into account this observation. A possible alternative in fact is represented by the SIP IETF Location Object.

### 4.2.3 SIP PIDF(Presence Information Data Format) +LO(Location Object)

The IETF Location Object (LO) is an object recently added to SIP Presence mechanisms by extending the PIDF (Presence Information Data Format) object. The reader interested in SIP Presence mechanisms topic can have more information having a look to [15] RFC. For our project purposes the use of the LO with his fields integrated with new tags defined to have a complete mapping for all the possible address formats.

### 4.2.3.1 Overview or SIP PIDF Location Object

As you can read in the specific RFC [15] “the document describes an object format for carrying geographical information on the Internet. This location object extends the XML-based Presence Information Data Format (PIDF [2]) which was designed for communicating privacy-sensitive presence information and which has similar properties to allow the encapsulation of location information within a presence document”. With respect to the Presence information the location information is kept inside the so called Location Object (LO) [15] that encompasses the data related with the geografic user position and the rules to process them with respect to privacy aspects. The LO is embedded in the PIDF without modifications, this choice simplify the integration of the LO itself and allows its transportation with other presence protocols (able to use the XML language). The Location Object is a complex Datatype that realizes an extension of the <status> parameter.
The LO parameters are included in the `<geopriv>...</geopriv>` tags whose role is to manage privacy mechanisms on information marked as sensible data. This object is splitted into two different parts:

the LOCATION INFORMATION, that represents the payload of location object

and the

USAGE RULES.

Hence the fields contained in the LO are: the `<location-info>` and the `<usage-rules>` tags.

### 4.2.3.2 The LO "compound location"

To use a standard format for the location data it is assumed to use the Geographic Markup Language (GML 3.0) notation. To express the civic Address a more suitable format, the so called “civicLoc” is used. A compound location is a term that inherits to a Location that has a geodetic part and a civic one. Following there is an Example that exploits both the GML notation and the civicLoc. The target is in San Francisco.

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <tuple id="sg89ae">
    <status>
      <gp:geopriv>
        <gp:location-info>
          <gml:location>
            <gml:Point gml:id="point1" srsName="epsg:4326">
            </gml:Point>
          </gml:location>
          <gp:usage-rules>
            <gp:retransmission-allowed>no</gp:retransmission-allowed>
          </gp:usage-rules>
        </gp:location-info>
      </gp:geopriv>
    </status>
    <timestamp>2003-06-22T20:57:29Z</timestamp>
  </tuple>
</pres>
```

An example of the PIDF-LO with the civicLoc notation.

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <tuple id="sg89ae">
    <status>
      <gp:geopriv>
        <gp:location-info>
          <cl:civicLoc>
            <cl:locName>
              San Francisco
            </cl:locName>
          </cl:civicLoc>
        </gp:location-info>
      </gp:geopriv>
    </status>
    <timestamp>2003-06-22T20:57:29Z</timestamp>
  </tuple>
</pres>
```
4.2.3.3 The CivicLoc elements

The civicLoc encompasses the following elements:

<table>
<thead>
<tr>
<th>Label</th>
<th>CAtype</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>language</td>
<td>0</td>
<td>The country is identified by the</td>
<td>en</td>
</tr>
<tr>
<td></td>
<td></td>
<td>two-letter ISO 3166 code.</td>
<td></td>
</tr>
<tr>
<td>country</td>
<td></td>
<td>national subdivisions (state,</td>
<td>US</td>
</tr>
<tr>
<td></td>
<td></td>
<td>region, province, prefecture)</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>1</td>
<td>city, township, shi (JP)</td>
<td>New York</td>
</tr>
<tr>
<td></td>
<td></td>
<td>city division, borough, city</td>
<td>Manhattan</td>
</tr>
<tr>
<td></td>
<td></td>
<td>district, ward, chou (JP)</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>2</td>
<td>county, parish, gun (JP), district (IN)</td>
<td>King's County</td>
</tr>
<tr>
<td>A3</td>
<td>3</td>
<td>neighborhood, block</td>
<td>Morningside Heights</td>
</tr>
<tr>
<td>A4</td>
<td>4</td>
<td>street</td>
<td>Broadway</td>
</tr>
<tr>
<td>PRD</td>
<td>16</td>
<td>Leading street direction</td>
<td>N, W</td>
</tr>
<tr>
<td>POD</td>
<td>17</td>
<td>Trailing street suffix</td>
<td>SW</td>
</tr>
</tbody>
</table>
For each LocationObject we can also specify a level of detail that takes into account that describing a user location we may use precise and detailed information or simply a reference to the area or the Country he is in. Hence we can have different accuracy levels that we list hereafter.

<table>
<thead>
<tr>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>{&lt;Country&gt;, &lt;A1&gt;, &lt;A2&gt;, &lt;A3&gt;, &lt;A4&gt;, &lt;A5&gt;, &lt;A6&gt;, &lt;PRD&gt;, &lt;POD&gt;, &lt;STS&gt;, &lt;HNO&gt;, &lt;HNS&gt;, &lt;LMK&gt;, &lt;LOC&gt;, &lt;PC&gt;, &lt;NAM&gt;, &lt;FLR&gt;, &lt;ZIP&gt;}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Building</th>
</tr>
</thead>
<tbody>
<tr>
<td>{&lt;Country&gt;, &lt;A1&gt;, &lt;A2&gt;, &lt;A3&gt;, &lt;A4&gt;, &lt;A5&gt;, &lt;A6&gt;, &lt;PRD&gt;, &lt;POD&gt;, &lt;STS&gt;, &lt;HNO&gt;, &lt;HNS&gt;, &lt;LMK&gt;, &lt;PC&gt;, &lt;ZIP&gt; }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>{&lt;Country&gt;, &lt;A1&gt;, &lt;A2&gt;}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>{&lt;Country&gt;, &lt;A1&gt;}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>{&lt;Country&gt;}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>'NULL'</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
</tr>
</tbody>
</table>

Now we can show the civic address Schema.

Civic Address Schema

```xml
<?xml version="1.0"?>
<xs:schema
  targetNamespace="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
```
Location Based Services architecture
for Simple Mobile Services

<xs:element name="civicAddress" type="ca:civicAddress"/>
<xs:complexType name="civicAddress">
<xs:sequence>
  <xs:element name="country" type="ca:iso3166a2" minOccurs="0"/>
  <xs:element name="A1" type="ca:caType" minOccurs="0"/>
  <xs:element name="A2" type="ca:caType" minOccurs="0"/>
  <xs:element name="A3" type="ca:caType" minOccurs="0"/>
  <xs:element name="A4" type="ca:caType" minOccurs="0"/>
  <xs:element name="A5" type="ca:caType" minOccurs="0"/>
  <xs:element name="A6" type="ca:caType" minOccurs="0"/>
  <xs:element name="PRM" type="ca:caType" minOccurs="0"/>
  <xs:element name="PRD" type="ca:caType" minOccurs="0"/>
  <xs:element name="RD3" type="ca:caType" minOccurs="0"/>
  <xs:element name="STS" type="ca:caType" minOccurs="0"/>
  <xs:element name="POD" type="ca:caType" minOccurs="0"/>
  <xs:element name="POM" type="ca:caType" minOccurs="0"/>
  <xs:element name="RDSEC" type="ca:caType" minOccurs="0"/>
  <xs:element name="RDBR" type="ca:caType" minOccurs="0"/>
  <xs:element name="RDSUBBR" type="ca:caType" minOccurs="0"/>
  <xs:element name="HNO" type="ca:caType" minOccurs="0"/>
  <xs:element name="HNS" type="ca:caType" minOccurs="0"/>
  <xs:element name="LMK" type="ca:caType" minOccurs="0"/>
  <xs:element name="LOC" type="ca:caType" minOccurs="0"/>
  <xs:element name="FLR" type="ca:caType" minOccurs="0"/>
  <xs:element name="NAM" type="ca:caType" minOccurs="0"/>
  <xs:element name="PC" type="ca:caType" minOccurs="0"/>
  <xs:element name="BLD" type="ca:caType" minOccurs="0"/>
  <xs:element name="UNIT" type="ca:caType" minOccurs="0"/>
  <xs:element name="ROOM" type="ca:caType" minOccurs="0"/>
  <xs:element name="SEAT" type="ca:caType" minOccurs="0"/>
  <xs:element name="PLC" type="ca:caType" minOccurs="0"/>
  <xs:element name="PCN" type="ca:caType" minOccurs="0"/>
  <xs:element name="POBOX" type="ca:caType" minOccurs="0"/>
  <xs:element name="ADDCODE" type="ca:caType" minOccurs="0"/>
  <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
<xs:attribute ref="xml:lang" use="optional"/>
</xs:complexType>
### 4.2.3.4 The CivicLoc extensions

From the first specification of the civicLoc mandatory fields new ones were added to be able to manage more refined Address formats.

<table>
<thead>
<tr>
<th>New Civic Field</th>
<th>CAtype</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLD</td>
<td>25</td>
<td>Building (structure)</td>
<td>Hope Theatre</td>
</tr>
<tr>
<td>UNIT</td>
<td>26</td>
<td>Unit (apartment, suite)</td>
<td>12a</td>
</tr>
<tr>
<td>ROOM</td>
<td>28</td>
<td>Room</td>
<td>450F</td>
</tr>
<tr>
<td>PLC</td>
<td>29</td>
<td>Place-type</td>
<td>office</td>
</tr>
<tr>
<td>PCN</td>
<td>30</td>
<td>Postal community name</td>
<td>Leonia</td>
</tr>
<tr>
<td>POBOX</td>
<td>31</td>
<td>Post office box (P.O. box)</td>
<td>U40</td>
</tr>
<tr>
<td>ADDCODE</td>
<td>32</td>
<td>Additional Code</td>
<td>13203000003</td>
</tr>
<tr>
<td>SEAT</td>
<td>33</td>
<td>Seat (desk, cubicle, workstation)</td>
<td>WS 181</td>
</tr>
<tr>
<td>RD</td>
<td>34</td>
<td>Primary road or street</td>
<td>Broadway</td>
</tr>
<tr>
<td>RDSEC</td>
<td>35</td>
<td>Road section</td>
<td>14</td>
</tr>
<tr>
<td>RDBR</td>
<td>36</td>
<td>Road branch</td>
<td>Lane 7</td>
</tr>
<tr>
<td>RDSUBBR</td>
<td>37</td>
<td>Road sub-branch</td>
<td>Alley 8</td>
</tr>
<tr>
<td>PRM</td>
<td>38</td>
<td>Road pre-modifier</td>
<td>Old</td>
</tr>
<tr>
<td>POM</td>
<td>39</td>
<td>Road post-modifier</td>
<td>Extended</td>
</tr>
</tbody>
</table>

*Table 4-12 New fields added to manage more refined Address formats*

### 4.2.3.5 The Relative Location SIP IETF Location Object values

In our architecture we deal also with indoor spaces so we need a way to represent even this kind of information within the Position object. As for this aspect we have to add to the fields that describes civic Address Locations even fields that identify a point in a closed space that can be projected to an Indoor Map in terms of Cartesian coordinates. A relative Location is defined with two new parame-
ters, an anchor point and a relative position. The anchor point like a geodetic marker defines a point used to measure the relative position parameter. The measure is expressed on a 3D space (X, Y and Z). The est-ovest dimension is identified by the X parameter and the nord-sud one by the Y tag. A positive value of the Y parameter expresses a location which lies north from the anchor point, a positive value of the X parameter identifies a position situated est from the anchor. An optional is the measure Unit in use (UoM). The table below lists the relative position fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Civic Field</th>
<th>CAtype</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>REFPT</td>
<td>REFERENCE POINT</td>
<td>40</td>
<td>Reference Point</td>
<td>Elevator</td>
</tr>
<tr>
<td>RELPOS-X</td>
<td>RELATIVE POSITION X</td>
<td>41</td>
<td>Relative Position X</td>
<td>&lt;X&gt;-12&lt;/X&gt;</td>
</tr>
<tr>
<td>RELPOS-Y</td>
<td>RELATIVE POSITION Y</td>
<td>42</td>
<td>Relative Position Y</td>
<td>&lt;Y&gt;35&lt;/Y&gt;</td>
</tr>
<tr>
<td>RELPOS-Z</td>
<td>RELATIVE POSITION Z</td>
<td>43</td>
<td>Relative Position Z</td>
<td>&lt;Z&gt;60&lt;/Z&gt; optional</td>
</tr>
<tr>
<td>RELPOS-UOW</td>
<td>UNITS OF MEASUREMENT FOR RELATIVE POSITION</td>
<td>44</td>
<td>Units of measurement for relative position</td>
<td>&lt;UoW&gt;feet&lt;/UoW&gt; Optional, default value meters</td>
</tr>
</tbody>
</table>

Table 4-13 Parameters describing relative positions

With respect to the xsd Schema

```xml
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
 schemaLocation="http://www.w3.org/2001/xml.xsd"/>
  <xs:complexType name="caType">
    <xs:simpleContent>
      <xs:extension base="xs:token">
        <xs:attribute ref="xml:lang" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="civicAddress">
    <xs:sequence>
      <!-- additions to civicAddress -->
      <xs:element name="refpt" type="ca:caType" minOccurs="0"/>
      <xs:element name="relpos-x" type="ca:caType" minOccurs="0"/>
      <xs:element name="relpos-y" type="ca:caType" minOccurs="0"/>
      <xs:element name="relpos-z" type="ca:caType" minOccurs="0"/>
      <xs:element name="relpos-uom" type="ca:UoM" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="UoM">
    <xs:restriction base="ca:caType">
      <xs:enumeration value="feet"/>
      <xs:enumeration value="meters"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```
Giving examples of the use of the notion of relative position we can have:

1.
<REFPT>elevator-1</REFPT>
<RELPOS-X>-20</RELPOS-X>
<RELPOS-Y>-31</RELPOS-Y>

2.
<REFPT>corner_office</REFPT>
<RELPOS-X>23</RELPOS-X>
<RELPOS-Y>51</RELPOS-Y>
<RELPOS-UOM>feet</RELPOS-UOM>

4.2.3.6 The “method” and “provided-by” fields

Investigating the LO RFC we found two elements that in our first definition of the SMS Position object tags we did not define. These new parameters are:

- method
- provided by

In [15] we read that the optional ‘method’ element describes the way that the location information was derived or discovered. An example of this element (for a geographical position system) is:

<method>gps</method>.

For our project purposes this field can be very useful because the location information sources can be of different types and with this field we can decide how much authoritative the location information is giving it a level of trust.

The ‘provided –by’ element identifies and describes the entity or organization that supplied the specific location information. Also this type of information is helpull for our project purposes because we can add to the location information a level of trustability.

An example is:

<provided-by> SMS Service </provided-by>
4.2.3.7 Our proposed extensions to PIDF Location Object

Dealing with the Location Object fields we noticed that some pieces of localization information cannot be covered with this standard. Hence we decided to use this format to maintain our data retrieved from for example a geocoding request or parsing the NMEA sentence coming out from a Bluetooth receiver connected with our mobile device but we added some new fields to this standard to manage more precise data and details. We stress that in our architecture we are not limited to only one method to acquire the Localization information and so we have to define tags that can keep this information.

The new fields we added and the description of their meanings are:

<table>
<thead>
<tr>
<th>New Civic Field</th>
<th>CAtype</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARA</td>
<td>Area type</td>
<td>University</td>
<td></td>
</tr>
<tr>
<td>ADAR</td>
<td>AddressDetails Accuracy Radius</td>
<td>50 meters</td>
<td></td>
</tr>
<tr>
<td>ADAL</td>
<td>AddressDetails Accuracy Level</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>FDNM</td>
<td>Friendly-name</td>
<td>Ricevimento Blefari</td>
<td></td>
</tr>
<tr>
<td>AreaUID</td>
<td>The URL to the University AreaType web site</td>
<td><a href="http://uniroma2.it">http://uniroma2.it</a></td>
<td></td>
</tr>
<tr>
<td>AreaGSMCellID</td>
<td>The GSM cell identification code</td>
<td>cellID(e.g. 639E), LAC(Location Area Code e.g. 1030), MCC (Mobile Country Code)</td>
<td></td>
</tr>
<tr>
<td>AreaGSMCellIDOperator</td>
<td>The operator that provides the GSM CellID</td>
<td>MNC(Mobile Code)</td>
<td></td>
</tr>
<tr>
<td>AreaWLANAPMAC</td>
<td>Mac wlan access Point we are attached</td>
<td>00-08-74-4C-7F-1D</td>
<td></td>
</tr>
<tr>
<td>AreaBTAPMAC</td>
<td>Mac BT access Point we are attached</td>
<td>11-08-74-4C-7F-1D</td>
<td></td>
</tr>
<tr>
<td>TSMP</td>
<td>Time Stamp</td>
<td>2003-06-2T20:57:29</td>
<td></td>
</tr>
</tbody>
</table>

*Table 4-14 Our proposed new tags to the LO CivicAddress*

AddressDetails Accuracy Level Constants are:

<table>
<thead>
<tr>
<th>Constants</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unknown location.</td>
</tr>
<tr>
<td>1</td>
<td>Country level accuracy.</td>
</tr>
<tr>
<td></td>
<td>Location Based Services architecture for Simple Mobile Services</td>
</tr>
<tr>
<td>---</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>2</td>
<td>Region (state, province, prefecture, etc.) level accuracy.</td>
</tr>
<tr>
<td>3</td>
<td>Sub-region (county, municipality, etc.) level accuracy.</td>
</tr>
<tr>
<td>4</td>
<td>Town (city, village) level accuracy.</td>
</tr>
<tr>
<td>5</td>
<td>Post code (zip code) level accuracy.</td>
</tr>
<tr>
<td>6</td>
<td>Street level accuracy.</td>
</tr>
<tr>
<td>7</td>
<td>Intersection level accuracy.</td>
</tr>
<tr>
<td>8</td>
<td>Address level accuracy.</td>
</tr>
</tbody>
</table>

Hereafter we present a Civic Address Schema with the proposed extensions in *italic*

```xml
<?xml version="1.0"?>
<xs:schema
    targetNamespace="urn:ietf:params:xml:ns:pidf:geopriv10:civicAddr"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
                 schemaLocation="http://www.w3.org/2001/xml.xsd"/>

    <xs:simpleType name="iso3166a2">
        <xs:restriction base="xs:token">
            <xs:pattern value="[A-Z]{2}"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="caType">
        <xs:simpleContent>
            <xs:extension base="xs:token">
                <xs:attribute ref="xml:lang" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:element name="civicAddress" type="ca:civicAddress"/>
    <xs:complexType name="civicAddress">
        <xs:sequence>
            <xs:element name="country" type="ca:iso3166a2" minOccurs="0"/>
            <xs:element name="A1" type="ca:caType" minOccurs="0"/>
            <xs:element name="A2" type="ca:caType" minOccurs="0"/>
            <xs:element name="A3" type="ca:caType" minOccurs="0"/>
            <xs:element name="A4" type="ca:caType" minOccurs="0"/>
            <xs:element name="A5" type="ca:caType" minOccurs="0"/>
            <xs:element name="A6" type="ca:caType" minOccurs="0"/>
            <xs:element name="PRM" type="ca:caType" minOccurs="0"/>
            <xs:element name="PRD" type="ca:caType" minOccurs="0"/>
            <xs:element name="RD" type="ca:caType" minOccurs="0"/>
            <xs:element name="STS" type="ca:caType" minOccurs="0"/>
            <xs:element name="POD" type="ca:caType" minOccurs="0"/>
            <xs:element name="POM" type="ca:caType" minOccurs="0"/>
            <xs:element name="RDSEC" type="ca:caType" minOccurs="0"/>
            <xs:element name="RDBR" type="ca:caType" minOccurs="0"/>
            <xs:element name="RDSUBBR" type="ca:caType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```
4.3 The proposed solution

After this long dissertation that showed our path through the definition and formalization of the SMS Position object fields we can now and finally offer to the reader the xsd Schema instance of a specific address. An XML instance that shows a “compound location” (civic part + geodetic part) exploiting the extended civicloc is listed below.

[Ed. Note: Additional explanatory text could be added]

Edificio dell’informazione, PIANO 4, Stanza n 7452
Facoltà di ingegneria, Università di Roma “Tor Vergata”
Via del Politecnico 1
The proposed bindings with the Location object elements are:

```xml
<cle:LMK>Università di roma “Tor Vergata”</cle:LMK>
<cle:Unit>Facoltà d’ingegneria</cle:Unit>
<cle:BLD>Edificio dell’informazione</cle:BLD>
```

### 4.3.1 JSON format

Starting from the point of view that the SMS architecture uses JavaScript Object Notation (JSON)[16] rather XML as the preferred format for coding the information we need a translation from an XML format to the JSON object format. The JSON protocol is lighter-weight and therefore especially suited for running on mobile devices and for being transmitted over wireless/cellular channels. 
JSON example of a “compound location” (civic part + geodetic part) exploiting the extended civicloc.

Edificio dell’informazione, PIANO 4, Stanza n 7452

Facoltà di ingegneria, Università di Roma “Tor Vergata”

Via del Politecnico 1

Roma, RM 00133 Italia

"Position":{
  "civicAddress":{
    "country":"IT",
    "A1":"Lazio",
    "A2":"Roma",
    "PC":"00133",
    "A6":"Via del Politecnico",
    "HNO":"1",
    "ARA":"University",
    "LMK":"Università di roma \"Tor Vergata\"",
    "Unit":"Facoltà d'ingegneria",
    "BLD":"Edificio dell'informazione",
    "FLR":"4",
    "Room":"7452"
  }
  "Point":{
    "lat":"12.618403",
    "long":"41.854170"
  }
  "TSMP":"2003-06-22T20:57:29"
  "METHOD":"geocode"
},

"Position Object JSON format:

{  "part":{"{  "object":{  "outdoorPositionObject":{  "gmlPoint":{  "lng":",  "lat":",  "method":",  "provided-by":",  "TSMP":"  },  "civicAddress":{  "ARA":",  "PCN":",  "ADDCODE":",  "STS":",  "POM":",  "UNIT":",  "PRM":",  "}  }
}  }
"POBOX":",
"A6":",
"A5":",
"FLR":",
"PLC":",
"POD":",
"A4":",
"A3":",
"PRD":",
"A2":"A2",
"A1":"a1",
"LMK":",
"HNS":",
"HNO":",
"RDBR":",
"RDSUBBR":",
"TMSP":",
"RD":",
"FDNM":",
"RDSEC":",
"METHOD":",
"PROVIDED-BY":",
"Country":",
"LOC":",
"BLD":",
"ADAL":",
"ADAR":",
"ROOM":",
"NAM":",
"AREAUID":",
"AREAGSMCELLID":",
"AREAGSMCELLIDOPERATOR":",
"AREALELANAPMAC":",
"AREABTAPMAC":,

"SEAT":"
}
"indoorPositionObject":{
  "METHOD":"inputMode",
  "PROVIDED-BY":"Generator",
  "TMSP":",
  "RelPosU":",
  "RelPosV":",
  "RelPosW":",
  "MapId":",
  "RefPt":",
  "RelPosUW":"
}
"name":"genericPosition",
"type":"it.uniroma2.sms.location.JSONSerializableGenericPosition"
]}
"name":"it.uniroma2.sms.location.MsgProva"

4.4 Advanced usages of the Position object

There could be several parallel active localization technologies, e.g. GPS and GSM, or Wi-Fi and
GSM, etc. At one time instant there could be different Position objects belonging to different
technologies, each carrying localization information coming from a specific localization system.
1. **FUSION**: One possible operation is to coherently fuse these objects to obtain another Position object that carries a more significant localization information, e.g. a fusion of GSM cell id (plus the GPS coordinates of that cell) and the user’s geocoded position. This fusion can be useful in order to give a more accurate user’s position inside the GSM cell.

![Figure 4-2 Position Objects Fusion](image-url)
2. **AGGREGATION**: We could even think of a Position object “history”:

![Diagram showing different Position objects over time](image)

*Figure 4-3 Different Position information filling in Position Objects during time*

The above picture shows a temporal generation of several Position objects, each possibly coming from different localization technologies. Our localization architecture can maintain a cache of the more recent objects that carry position information; this history of position information can be assembled as an array of Position objects, eventually sent to evolved external components able to handle it for location based purposes.
3. **ITERATION**: The last possible advanced handling of the Position object is the iterative refinement of the localization information.

"Paolo walks around the city and suddenly finds a nice pub and wants to send a MEM to his friend Francesca to tell her where he is in that moment and to reach him in order to have a drink"

In this MEM, Paolo wants to send to his friend Francesca a MEM which is most possibly human readable. At that moment, only the GSM cell id information is available on his device, so the application tries to refine this information by using the geocoding mechanisms we have described. In this way the application has now the GPS coordinates for that cell id, plus an ADAR (Address Detail Accuracy Radius) which specifies the cell extension. At this point, another possible transformation may be done, that is for example a reverse geocoding operation. The application now disposes of a human readable string, that is the full address of the BTS; having an ADAR, the application will chop the address to the level of accuracy corresponding to the previous localization operation: for example, if the result of the reverse geocoding is “Roma, Trastevere, Viale Trastevere, 170”, then the chopped string will be
“Roma, Trastevere”. This is the best-effort result that will be sent in the MEM, although Paolo wanted to tell Francesca his exact position.

5 Implementation of localization functions in the MOVE application

MOVE [22] is a J2ME application that runs in the user’s mobile device and provides access to the “Simple Mobile Services”, including the localization based services. It is an open platform which can be used to build further services. It uses a GUI framework to handle the user interaction, in particular in [22] the use of Thinlet [21] GUI framework (in its porting to J2ME) is described.

The technical details about the implementation of the localization functions in the MOVE application and a description of the offered APIs can be found in [22].
6 References

[27] BLUESIC: context-aware information system for tourism, based on bluetooth technology Juan Pec, Carlos Fernández, Carlos J. Escudero. Departamento de Electrónica y Sistemas, University of La Coruña, Campus de Elviña s/n, A Coruña, España
[29] WSDL : www.w3.org/TR/wsdl
Location Based Services architecture
for Simple Mobile Services

[31] JSON : http://json.org/