Simple Mobile Services, a Service Creation Architecture for Mobile Services

G. Bartolomeo, F. Martire, S. Salsano, N. Blefari-Melazzi
DIE, University of Rome "Tor Vergata", Italy

***Abstract*— To date, mobile services have failed to match the explosive growth of the Web. This, we argue, is because current mobile services are difficult to find, to use, to trust, to design and deploy. The IST SMS (Simple Mobile Services) project takes up the challenge of creating innovative tools addressing the specific needs of mobile users and making it easier for individuals and small businesses to become providers of Simple Mobile Services. We hope that, thanks to SMS, a large base of people can create Simple Mobile Services, like it happened for the Web, where creating a web page is within the reach of non-expert users.**

In this paper, we first focus on one of the facets of SMS: service authoring tools. We highlight the importance of powerful authoring tools, to speed up and reduce the cost of service development. Then we focus on the location based services that are an important component of SMS.

***Index Terms*— simple mobile services, service creation, location based services**

INTRODUCTION

The primary objective of the SMS project [1]is producing tools that enable users with limited technical expertise to build, deploy and offer services based on mobile devices (mainly smartphones and PDAs).

SMS does not aim to build a single portal or a single service offered by a single organization, rather SMS will propose a "concept" and a set of tools/technologies. An "Open garden" approach is considered, there will not be a predominant or even exclusive role of an operator, but different players will be able to offer services based on SMS concepts and technology.

As for the underlying connections, depending on its capability the mobile terminal will be able to exploit and combine the largest possible set of wireless communication technology. The list will include, but is not limited to: circuit switched GSM /UMTS, cellular packet access on GPRS/UMTS, wi-fi, Bluetooth, near field communications (e.g. RFID).

As for the target end-user mobile devices, SMS mainly focus on devices that will be wide spread in a one to two years time frame. Therefore mobile phones that are in the "high-end" market today are OK, a typical example can be a Java enabled phone with Bluetooth, UMTS and a WLAN interface. Given this requirements, also PDAs are likely able to run SMS services.

It is important for SMS to integrate with existing services. For this purpose, we can classify the existing services in two classes: a) services directly offered to a human user trough a web interface; b) services that are offered to other machines through a SOA (Service Oriented Architecture) "web services" interface.

As for services directly offered to a human user trough a web interface, SMS will support using and augmenting existing services (e.g. all web sites existing in the internet in this moment) with easy accessibility from mobile devices and context awareness features. As for Service Oriented Architecture, we assume that the SMS architecture can both exploit existing "web services" and can offer SMS features as "web services" to the external world.

This work was partially supported by the EU under the project IST2006-034620 "Simple Mobile Services"

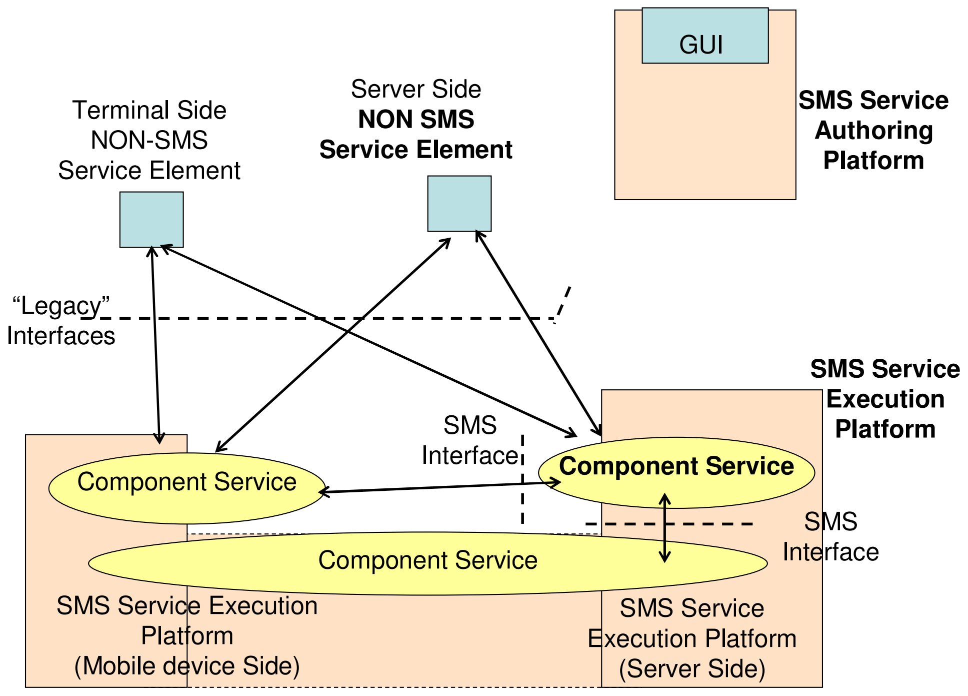
GUI
SMS Service Authoring Platform
Terminal Side NON-SMS Service Element
Server Side NON SMS Service Element
"Legacy" Interfaces
SMS Service Execution Platform
SMS Interface
Component Service
Component Service
Component Service
SMS Interface
SMS Service Execution Platform (Mobile device Side)
SMS Service Execution Platform (Server Side)

Fig. 1 SMS Architectural elements.

SMS includes multimodal/multidevice and pervasive services which are typically implemented using various devices, peripherals, sensors/actuators, ad-hoc and unlicensed networks (bluetooth/zigbee). A multimodal user interaction should be used wherever convenient. For example a user could receive a phone call when this is simpler in order to confirm the booking of a ticket rather then interacting with limited screen capability. The context of use of SMS services (on the move while engaged in some other task) requires that each interaction is fast and short lived, therefore each interaction is not sufficient to complete more complex high level tasks such as 'shopping', 'wayfinding', 'travel' etc. Within SMS, context support can be used to develop services with reduced interactional complexity for the end-user.
Coming to the service authoring, in order to support the "non expert" user that wants to build a mobile service, an SMS "Service Authoring Wizard" is being developed. The 'Service Authoring Wizard' offers a wizard-like interface which the (non-expert) users use to make the necessary parameterization of an existing SMS template and thus finally deploy a service based on it. On the other hand, an "expert" user with programming skills makes use of the SMS "Advanced Service Authoring and Modeling Tool". It offers a richer functionality with which the advanced (expert) user will be able to create complex SMS services from existing service components (that have been developed separately by programmers). UML modeling approach will be used at this level, by defining an SMS UML profile. Existing UML/MDA tools are used to the maximum possible extent, so that the SMS "Advanced Service Authoring and Modeling Tool" is build upon the composition of existing UML/MDA tools with proper customization and enhancements.

Architecture

As shown in Fig. 1, we have four main elements in the SMS architecture: the SMS Component Services, the "NON-SMS Service Elements", the SMS Service Execution Platform, the SMS Service Authoring Platform.
An **SMS component service** is a portion of a service or a whole service that can be used to create more complex services or component services. An SMS component service offers an SMS interface so that it can be composed in a more complex service or component services. Example categories of component services are location based services (which will be dealt with later on in this paper), services related to retrieving user profile information, services relate to logging into services / authentication / authorization, services related to payments. **NON-SMS Service Elements** are for example Web sites or server nodes providing Web Services interfaces. The interfaces between the NON-SMS Service Elements and the SMS Service Execution Platform is denoted as

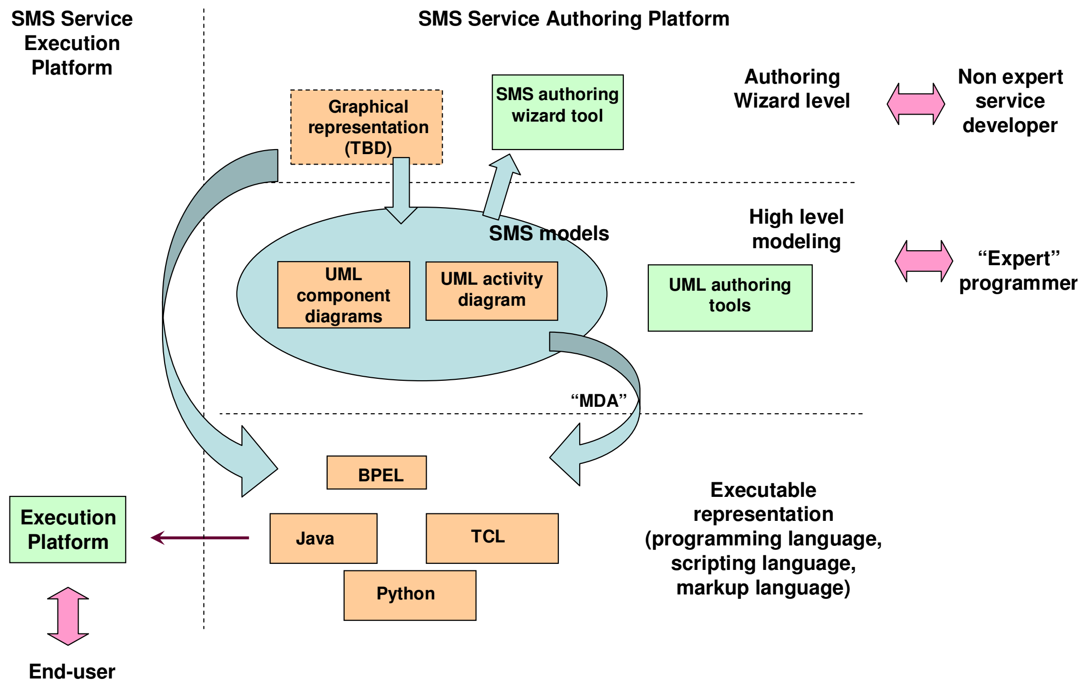
SMS Service Execution Platform
SMS Service Authoring Platform
Graphical representation (TBD)
SMS authoring wizard tool
Authoring Wizard level
Non expert service developer
SMS models
High level modeling
UML component diagrams
UML activity diagram
UML authoring tools
"Expert" programmer
"MDA"
BPEL
Java
TCL
Python
Execution Platform
Executable representation (programming language, scripting language, markup language)
End-user

Fig. 2. Models, translations and tools for Service Authoring

"Legacy Interface" in Fig. 1. This interface should support as much as different ways to interact with services as possible. One typical example is the SoA interaction model, based on Web Services: in this case the interfaces will be described with WSDL and the execution will be made using SOAP. Another example will be "plain" http/html for classical Web sites with human interaction.

The **SMS Service Execution Platform** is the set of nodes where the SMS services are executed. The Service Execution Platform can be seen as distributed entity with several entities that inter-work, both on server side and on terminal side. As shown in Fig.1 both terminal side and server side Service Execution Platforms that can interact with NON-SMS service element residing on Terminal Side or Server Side.

Note that the SMS Service Execution Platform is not meant as a given set of hardware nodes that will be provided by the SMS project. Rather, the SMS Service Execution Platform represents the abstraction of the set hardware nodes and software components that execute SMS services. This can include combination of applications and components developed by SMS and existing application/components/platforms. There can also be separate instances of the SMS Service Execution platform which are run by different providers. From the technical stand point, these separate instances should of course be able to inter-operate. The level of interoperation between different instances of the SMS service execution platform will likely be driven by business model issues.

Note that the notion of "Mobile device/Terminal Side" and "Server Side" is not really related to the client-server dichotomy in a classical client-server paradigm. We do support a flat "peer-to-peer" approach as much as possible. We want to differentiate between the mobile devices that typically have limited capacity and intermittent communication and a fixed host in the Internet (typically reachable with a public IP address) which can play the server role. In particular, both Terminal Side and Server Side can be "provider" and "consumer" of information as needed by the specific services.

The **SMS Service Authoring Platform** is used to create SMS services starting from Component Services. The SMS Service Authoring Platform provides support both for a "non-expert" service author and for an "expert" developer.

We close this short presentation of SMS architecture (please refer to [3] for a deeper understanding) by introducing the notion of Workflow. An SMS workflow represents the execution logic of an SMS service or component service, expressed as composition of other component services. The workflow may include conditions, loops and invocation of remote components. A workflow can be composed of different threads of control that can interact. The threads of control can be run on one or more different machines (i.e. mobile devices or fixed hosts).

Modeling Services and Service Composition in SMS

Modeling of services and service composition in SMS can be performed at two different levels:

- High-level modeling (for expert user)
- Authoring wizard level (for non-expert user)

In the high level modeling most of the implementation and deployment details and of the interactions needed to realize a service can be hidden, making the development process of mobile services more effective. UML language is the chosen representation level here. UML authoring tools should support the modeling of SMS services, entities, component services and workflows. One of the main objectives of providing the high-level modeling representation is to have automatic derivation of executable components.
It will be possible to automatically derive a "low-level" representation (i.e. with the greatest level of detail) of SMS services and component services. This corresponds to an "executable" representation that can be run on a machine (maybe interacting with other machines). In principle, this "low-level" representation may use programming or scripting language (JAVA, python, tcl), or a business process languages (for example a BPEL). The ongoing work of the SMS project is focusing on JAVA programming language (in particular J2ME CLDC for the terminal side components that needs to be run over the smart phones). As UML/MDA tools the project is using the combination of MagigDraw and AndroMDA.
As the high level workflow representation may be still too complex for the target audience (non expert service developers), we introduce an additional level of abstraction in the form of the authoring wizard.
Fig. 2 tries to provide a graphical representation of the SMS modeling approach. The core part is a UML representation of the SML component services, which enable a service authoring process based on UML diagrams. Existing UML authoring tools should be used to the maximum possible extent. The "user" of this level is the "expert programmer" or "expert service developer".
The UML diagrams produced by the UML authoring tools needs to be translated into executable code with minimal "manual" intervention. As much as possible the translation process should be executed at "run" time rather than at "service creation" time, allowing for a context dependent translation. Existing tools for code generation from UML diagrams will be reused. The executable representation languages that are represented in Fig. 2 are only meant as possible examples.
The Authoring Wizard level is offered to the "Non-expert service developer". The representation of services and of their combination should be made in terms that needs to be understood by users with no expertise in software engineering nor in programming. A graphical representation could be defined and used to this purpose. Ideally it should be possible from the UML high level modeling level to interact with the Authoring Wizard by extending the catalogue of "component services" that can be glued together by the Authoring wizard (this is represented by the small upward arrow in Fig. 2). The Authoring Wizard could provide a representation of the composed service in term of UML modeling (this is represented by the small downward arrow in Fig. 2). This is not mandatory, other directions can be explored like for example direct mapping from Authoring wizard level to an executable representation.
In the down left side of Fig. 2 the SMS execution platform is represented, which includes terminal side and "server" side computing elements. The services that are developed using the SMS service authoring platform will be deployed and run over the execution platform. The "End-user" will be using the provided services.

Methodology and scenarios

In order to derive the specification of the architecture and the design of the SMS system, the SMS project is following the classical approach of starting from user scenarios and user requirements. This process is documented in [2]. Scenarios play a fundamental role, they act as a source of requirements for technology, applications, and business cases.
Two types of scenarios are considered, one type deals with the final end-user of mobile services, another type deals with the developer of mobile services.
Considering the end-user, the scenarios refer to situations in which the user is on the move and has an immediate need for relatively simple information: e.g. how to reach a given location, information on a flight, information on a painting in an exhibition. In general, SMS

TABLE I – LOCATION BASED COMPONENTS

Component service	Functionality
"I'm here"	It updates user location information on a recipient (typically a server side element)
"Where are you"	It is used to query the user position to the user terminal itself.
"Where is he?/ Where is it"	It queries the position of an entity (a user or a place) to a server.
Position provider	It provides the position of an entity on request. It may be the object itself or a third entity that knows the position.
User presence trigger	It detects the fact that a user has entered an area

infers the services best adapted to user needs from users' general characteristics (stored in the user profile) and their current context (location, time, activity etc.). A detailed description of the User Scenarios can be found in [2], we just list their names here to get an impression of their content: Leaving for London, Keeping the airport running, A Good Restaurant, Looking for an ice-cream.
From the developer point of view, SMS makes it possible to provide single services, or sets of services to well-defined populations of mobile users, engaged in specific kinds of activity, at a given time in a given location. Examples include sets of services offered by an airport, or a shopping mall but also integrated sets of services offered by smaller actors. Some scenarios included in names of scenarios included in [2] are: The Newspaper Shop Owner, Managing Airport equipment, Setting up a museum service.
The identified scenarios have been "de-structured" to identify the key elements, such that other scenarios could be created combining basic building blocks that provide simple functionalities into more complex services. The further process of identification of architectural components that will be reflected in the system implementation is dealt with in [3]. For example Table 1 shows the identified location based component services and their functionalities. This is a subset of the more than 30 different component services (not limited to location based) identified in [3].

SMS and Location Based Services

SMS services are 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 three questions: Where am I? What's around me? How do I get there?.
During service execution components in the terminal will interact with localization technologies, smart spaces elements and with server side service elements to provide SMS services to the user.

Definition of the SMS Navigation/Localization Architecture.

The user location system will be one of the service building blocks in the SMS toolkit. The options offered by the toolkit will depend on the infrastructure available in the service delivery environment (both terminal side and server side). This means that the architecture should be independent of any specific localization technology. The service will be able to use the same localization functions even when underlying localization technologies are different. The platform will automatically adapt to the functionality available in target locations, so that the specifics of the underlying localization system can be hidden to the service author.
Integrating different location systems as service building blocks in the SMS toolkit will require clear interfaces and adaptation layers between specific implementations and the other components of SMS.
The SMS navigation and localization architecture has been defined in a modular way, as shown in Fig. 3. The role of the SMS Navigation/ Localization Component is 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. component is the interface towards navigation and localization services and it resides on the Mobile Terminal. It implements the operation of the "Interact with navigation/Localization SW" component described in [3]. The SMS Navigation/ Localization Component exposes an API to the "SMS Generic Components" tailored to build location based applications. The main functionality of this API are:

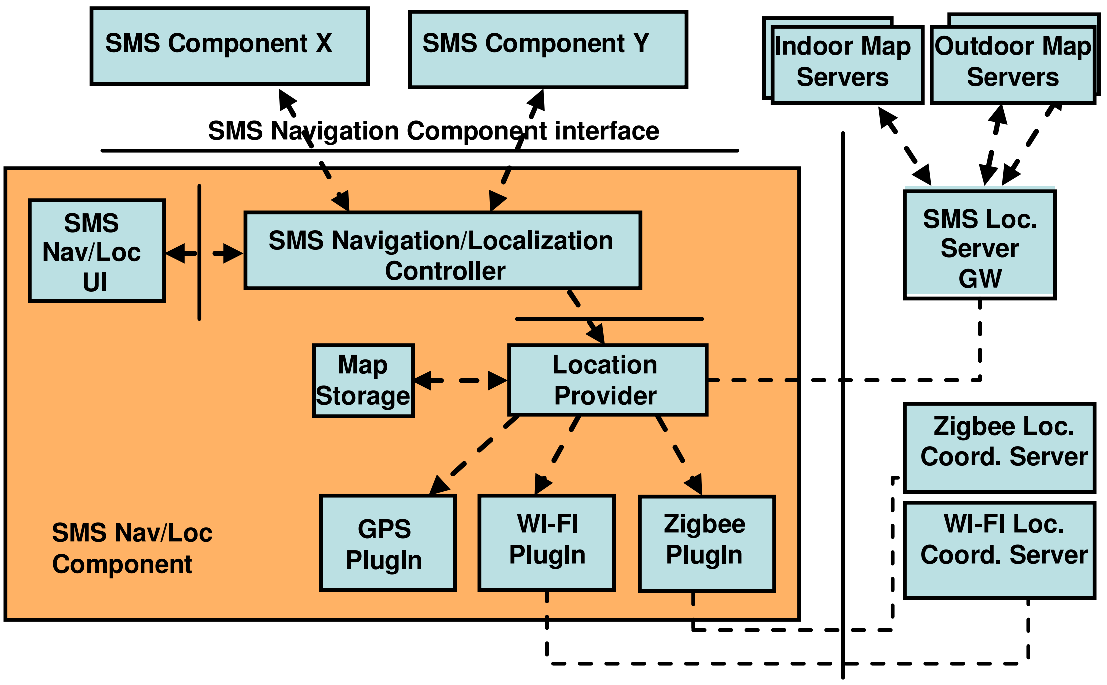

Fig. 3. Navigation and Localization Components for the trials.

- Display User Current Position
- Display Generic Position
- Display Path
- Drive Me To Destination
- Bookmarks&Markers

The request and the display of maps can be parameterized with different scales.
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. 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 another option we can use a navigation application explicitly developed to play the role of SMS Navigation/Localization component. Note that there can be more than one navigation application that needs to interoperate (e.g. one outdoor navigation application and one indoor navigation application).
In the rest of the document, we assume the latter case and we will describe the J2ME application that we have developed. We split this application into a "SMS Nav./Loc. Controller" sub-component and a "SMS Nav./Loc. User Interface" sub-component. Our application is capable to handle both outdoor and indoor navigation.

1)SMS Navigation/Localization Controller:

The SMS Nav/Loc Controller is the heart of the Nav/Loc component. It controls the overall logic and interacts with the other SMS components, with the SMS Nav/Loc User Interface and with the "Location Provider".
The communication with the with the Location Provider hides the details of specific localization technologies that can be used by the terminal. At the moment we have three different Localization Technologies (i.e. GPS, Wi-Fi and Zigbee) managed with specific Plugins but with this architecture allows to add whatever technology we want. In order to show the user his own position on a suitable map the Controller has to sort it depending on the currently active localization technology (outdoor localization like GPS or Indoor localization technologies such as Wi-Fi, Zigbee). To accomplish this task, as a first step, it needs to be aware if the user is Outdoor or Indoor. This information can be retrieved through the use of the API exposed by the "Location Provider" component:

- EntityPosition: return the position of a specific Entity
- GetTechInUse: information about the technology currently in use

Currently we have three so called "Localization PlugIn": the GPS Plugin (for Outdoor localization) whose task is to manage the Bluetooth connection with a Bluetooth GPS, a Wi-Fi Plugin (mainly for Indoor localization) and a Zigbee PlugIn (for Indoor localization). 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 Loc/Nav Coordinator Server, 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).
The proposed architecture supports different solutions for retrieving the maps. The maps can be pre-stored on the Terminal itself (like it typically happens with navigation software like TomTom) or they can downloaded on demand from suitable Map Servers (like it happen with web based map services like Google Maps). The implemented application relies for Outdoor maps on the “mashup” of existing web based map servers. It is able to is able to show outdoor maps downloaded from the existing web based map services. The mashup is realized by an external element denoted SMS “Location Server Gateway” in Fig. 3. The SMS Nav./Loc. Controller interacts with the Location Server Gateway using an SMS specific dialogue, then the Location Server Gateway issues http requests to a web based map server according to its format.
For Indoor maps, the implemented navigation application relies on a simple map server implemented as a web server. This map server provides the maps when requested from the clients using http requests.
When SMS Nav./Loc. Controller 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.

2) SMS Navigation/Localization User Interface:

“SMS Nav./Loc. User Interface” can be used by the user to “manually” browse a map, to find a place in the map, to find a route to a destination. The user can exploit all the functionality of the API that the SMS Nav/Loc component exposes to the other SMS components, plus some additional functions like the above mentioned finding functions.

The current list of features of our navigation application is as follows:

– Choose the proper localization technology(GPS outdoor/Wi-fi indoor)
– Display different maps from a map server o a memory stick
– Display current position (gps position, indoor position)
– Read gps data from BT Gps (raw NMEA data)
– Uses NMEA sentences: (i.e. RMC, GGA)
– Download a map for a given position or an area
– Bookmarks&Markers (edit, load, delete bookmarks and markers) support.
– Search for a place and set the center of the map to a search result.
– Search for a category of interest.
– Search for a route and display the path on a map with the related textual description.
– Turn by turn Directions
– Navigate a Map
– Map Zoom In/Zoom out

Fig.4 shows some screenshots of the developed applications, in particular related to the indoor navigation.

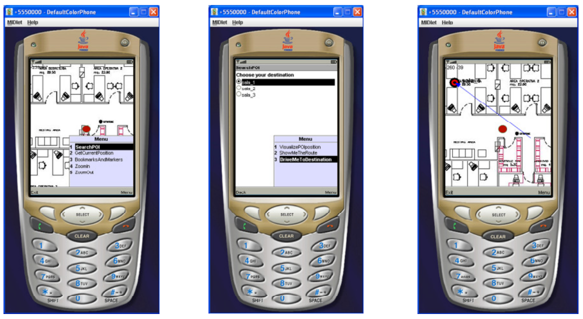

Fig. 4. Screenshots of our J2ME Application

ACKNOWLEDGEMENTS

The authors would like to acknowledge the contributions of their colleagues from the SMS consortium.

REFERENCES

[1] The SMS project web site: http://www.ist-sms.org
[2] R.Walker (ed.) “User, developer and service provider scenarios, human factors and business requirements for SMS”, Deliverable D2.1 of IST SMS project http://www.ist-sms.org/server/deliverables.php
[3] S. Salsano (ed.) “Initial system architecture specification”, Deliverable D3.1 or IST SMS project http://www.ist-sms.org/server/deliverables.php