LoL@: Designing a location based UMTS application

Hermann Anegg, Harald Kunczier, Elke Michlmayr, Günther Pospischil, Martina Umlauft
anegg@ftw.at, kunczier@ftw.at, michlmayr@ftw.at, pospischil@ftw.at, umlauft@ftw.at
Forschungszentrum Telekommunikation Wien, Donau-City-Straße 1, 1220 Wien/Austria

Abstract

UMTS services integrate mobile communications and the Internet, based on a powerful and standardized service architecture. To achieve a positive user experience, careful resource planning, service and protocol design are necessary. After dealing with some fundamental issues of UMTS services, this paper presents LoL@, the Local Location Assistant. It is a prototype of a location based UMTS service, combining localization/navigation and multimedia. These components can be seen as "killer components", they can be combined to provide various services over various networks.

Keywords: mobile Internet, location based services, SIP, mobile multimedia.

1. Introduction

UMTS will only be successful if it provides a portfolio of attractive services. There will not be a single *killer application*, people's needs vary too much to be satisfied by a single application. Therefore, UMTS standardization has defined a framework to develop a rich set of services. It consists of a concept for *service access* (Virtual Home Environment [1][2]), a *network architecture* (Open Service Access [3][4]), and a *terminal architecture* (Mobile Execution Environment [5]).

LoL@, the Local Location Assistant, is a prototype of a UMTS location based service. It is designed for tourists who walk on a sight-seeing tour through the first district of Vienna. LoL@ guides them along the tour and provides multimedia information related to the sights.

In Section 2, the UMTS service architecture is shown, followed by the LoL@ architecture components in Section 3. The paper ends with summary and conclusions in Section 4.

2. UMTS Service Architecture

In GSM, the *available services*, e.g. voice, fax, data, are *standardized*. This ensures compatibility between different networks and terminals, but it impedes development of new services. The introduction of SAT and WAP was the first step towards an open service environment. However, both concepts are not sufficient for complex UMTS services because they are designed for very limited GSM phones and do not provide access to all relevant network elements, like the user profile server (HSS) or multimedia subsystem.

To overcome the current limitations, a flexible service environment for UMTS has been standardized. The main goals are to facilitate *quick service development* and *convenient service access*. Figure 1 shows how these goals can be achieved via the virtual home environment (VHE), open service access (OSA), and mobile execution environment (MExE).

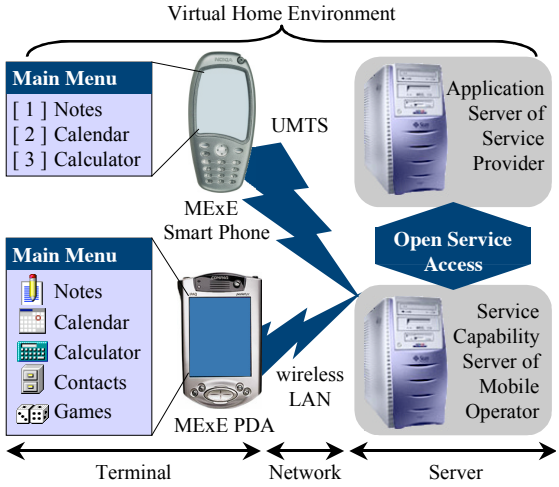

Figure 1: Virtual Home Environment (VHE), Mobile Execution Environment (MExE) and Open Service Access (OSA)

VHE is a concept for service access with different devices (e.g. Smart Phone, PDA or Laptop) over different networks (GSM/GPRS, UMTS, wireless LAN). In all situations, the functionality and look & feel shall remain the same as far as possible. To realize VHE in mobile networks, MExE and OSA are used.

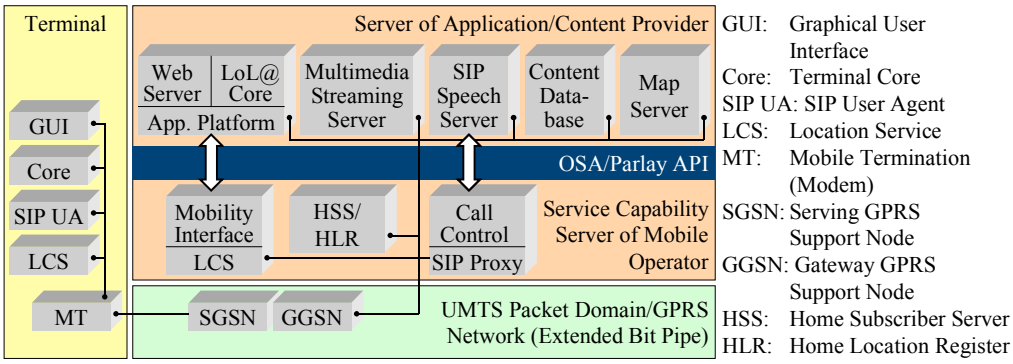

Figure 2: LoL@ Service Architecture

Internet service development is based on terminals with browsers for html, JAVAScript, and JAVA applets. A similar approach is taken with MExE. *MExE is a JAVA-based execution environment* for mobile devices. Internet-like applications may be extended with mobile specific features, like call control or access to the phone book. These extensions are defined in the JAVA Phone API [6] and Java 2 Micro Edition (J2ME) Connected Limited Device Configuration (CLDC [7]) with the Mobile Information Device Profile (MIDP [8]).

The business logic of Internet services is usually implemented in an *application server*. The application server provides *platform services*, like data conversion, subscriber and content management, or access to content databases. Parts of the business logic are often implemented as *servlets* on top of a Web-Server.

In the Internet, the transport network is usually seen only as a bit-pipe. However, mobile networks offer additional functionality, e.g. call control, localization, user/device profile access. An abstraction of such a network functionality is called *Service Capability Feature* (SCF), provided by a *Service Capability Server* (SCS). OSA consists of a framework for authorization, authentication and discovery of network features and provides a standardized interface to SCFs. With OSA, network services can be included into an application in the same manner as conventional platform services by using a set of CORBA functions.

3. LoL@ Service Architecture

Figure 2 shows the components of LoL@. A typical service request starts in the *service user interface* (GUI component of the terminal, see Section 3.1). After an interaction with the *client part of the business logic* (Terminal Core), the request traverses the UMTS/GPRS network. Finally it reaches the destination server (usually the LoL@ Core), which processes the request (Section 3.2). It may contact other servers of an application/content provider or the mobile network operator. Finally, a response is sent to the terminal.

LoL@ is designed for a PDA-like MExE phone with a 320x120 pixel color LCD display and pen input. Currently, the device is simulated on a laptop with a Web-Browser and JAVA 1.1 virtual machine, extended by a mobile termination for network access.

The terminal provides the following features:

1. *Service user interface (GUI)*, including a JAVA applet for displaying maps, a html window for hypertext information, and JAVA AWT buttons for service control.
2. The *client part of the business logic (Core)*, implemented as a (hidden) JAVA applet. In traditional Web-Applications, the business logic resides completely in the server. We decided to put a part of it into the terminal to improve response time and minimize air interface traffic.
3. *SIP User Agent* for packet switched voice (Voice over IP) sessions, either for human-to-human communication or human-to-machine communication (speech control).
4. *Terminal LCS*, providing connectivity to external localization devices, like a GPS receiver or Bluetooth transceiver, and the location server (LCS, see Section 3.3).

The server domain hosts the application plattform and the supporting servers. Our application platform is a Web-Server with several LoL@ servlets; we do not use a specific application server. The following

network related functions are implemented as Service Capability Features (SCF):

1. *Call Control SCF*, using a SIP Proxy for application level call control.
2. *Localization SCF*, used to provide Localization Services (LCS). It connects to the terminal counter part (to retrieve e.g. GPS coordinates). This data is combined with network related information, e.g. cell site database or network based location information. Applications may use the LCS infrastructure via the OSA/Parlay Mobility Interface.
3. *User Profile SCF*, used for authorization of location requests and user identification. We use an LDAP database for this purpose; in an operational network this SCF is the HSS (Home Subscriber Server).

3.1. LoL@ User Interface

LoL@ will be used outdoors on a PDA-like device. Therefore, environmental problems, device constraints and special user needs pose additional challenges. Consider, for example, low LCD contrast in direct sunlight or the uncomfortable and small keyboard. Additionally, people are on the road and might be nervous because they are lost in a foreign city, or pressed for time (e.g. they want to find a tourist attraction before it closes).

We consider the following four interaction scenarios to be the most important ones:

1. A user is in the hotel and wants to preview the tour or get information about the sights.
2. A user is on the tour and wants information for the current point of interest (PoI).
3. A user wants to be guided to the next PoI.
4. A user is lost and needs guidance (routing).

Our application is *map-centric*; i.e. we use a map-metaphor with additional textual information screens. In the map we use "select, then apply command" interactions: a user clicks on a PoI, then on the "info" button to get information for this PoI. A *browser-metaphor* is used to navigate through hypertext information. To avoid user distraction, the maximum depth of the menu structure is restricted to 5. In addition to pen and limited keyboard input we use server-side speech recognition, based on SIP-controlled Voice-over-IP sessions. LoL@'s speech control provides a set of commands which are used as "short-cuts" to access often used menu items.

A sample usage scenario might look like this: When the user starts LoL@, user information is retrieved from the HSS and the user is logged on automatically (the user does not have to enter an additional password). After choosing a tour he is presented with the overview map as shown in Figure 3.a. From there, he can select a region of interest and zoom into the detail map (Fig. 3.b), then select a PoI for which to get information (Fig. 3.c).

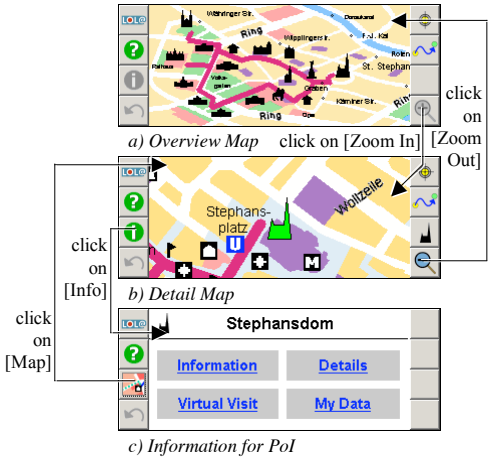

Figure 3: Screenshots and User Interaction

In contrast to the GUIDE project [9] we decided to use context awareness only in the sense that the current location is used as the default selected PoI, and do not change or restrict the displayed information or possible actions based on user location.

Other important design issues (see [10], [11]) considered in LoL@ include *stability* (buttons stay in the same place, user selections do not get lost when the map is zoomed) and the *avoidance of "modes"* as much as possible. The map screen and textual screens, for example, can be seen as two differing "views" rather than application modi; i.e. when a new PoI is selected in the textual screen, this selection is also relevant for the map screen.

3.2. LoL@ Content Preparation

We use flexible templates for content preparation. This allows to model human-computer interaction flow in *machine-readable and human-understandable* form. Integration of input data from various data sources with different access methods and data formats is an essential factor. Another key issue is support for multiple output formats.

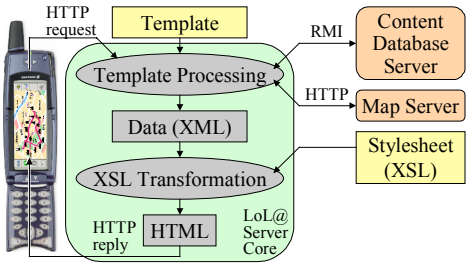

Figure 4: Template-based Content Preparation

A template contains commands, links, parameters and data (Fig. 5). Commands (Line 6) are used to define which data is needed and from which source it has to be fetched. Links (Line 5) describe the hypertextual structure of Lol@. The HTTP parameters of the request are needed for processing (Line 12). Placeholders in the template are replaced with the actual values of the parameters at runtime. Necessary data which is not available from any data source is stored in the template.

Templates are processed in 3 steps (Fig. 4):

1. Replace parameter placeholders with the actual values of the parameters.
2. Execute command: contact the various data sources and fetch the data.
3. Check links: It is checked if content is available for a link, otherwise the link is not shown in the generated HTML page.

```
01: <s101>
02:  <header>List of Tours</header>
03:  <content>
04:   <list>
05:    <linktext>
06:     <command source="db">
07:      <text>select TourId AS ID, Title
08:       AS title, Duration, Length from Tour
09:      </text>
10:     </command>
11:    </linktext>
12:    <linkpara check="no">
13:     <name>sid</name>
14:     <var>102</var>
15:    </linkpara>
16:   </list>
17:  </content>
18: </s101>
```

Figure 5: XML templates

A dynamic XML data structure (see [12], [13]) is used. Based on the intermediate processing results, the data items in the template are dynamically changed, extended or removed. The results of the template processing are filled-out templates which contain all the data necessary to create the requested LoL@ screen.

An XSLT processor transforms the XML data according to the rendering rules defined in an XSL stylesheet [14]. In our case, the stylesheet defines HTML code for a Web-Browser. Based on the layout-definition in the stylesheet, other output media (e.g. WAP-Browsers) can be supported easily.

3.3. Localization and Navigation Concept

LoL@'s localization concept is designed to provide an accurate and reliable determination of the terminal position. This objective is divided into two tasks: *(a) transportation and management of the localization data* and *(b) efficient localization*.

The first task is achieved by LoL@'s localization architecture (Fig. 6). It extends the 3GPP standard [15] with an end-to-end solution for terminal to GMLC (Gateway Mobile Location Center) communication allowing to provide location services on any legacy network. Still, network LCS methods can easily be included; in this case the GMLC selects the appropriate LCS method. The GMLC implements the OSA/Parlay Mobility Interface which provides standardized connectivity towards various location clients.

In our case the LoL@ servlet acts as location client via the LCS platform service. It issues a location request to the GMLC. After security checking (e.g. if the mobile user has agreed to be localized) and determining the actual position of the terminal, a response is returned to the location client.

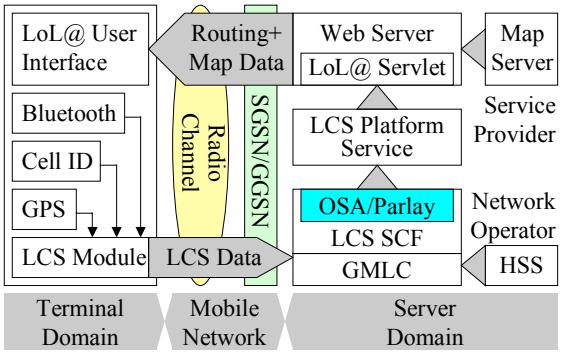

Figure 6: Localization Architecture

The second task, efficient and accurate localization, can be achieved via a hybrid concept using different localization methods. Localization methods typically rely on communication between known reference points (e.g. base stations or satellites) and the target (e.g. MS) which should be localized. Based on the known geometry configuration of the reference points, the *target's position is calculated by intersecting spheres* (with centers at the

reference point's position and radius defined by the signal propagation time between reference point and target multiplied with the speed of light). GPS, which is one part of Lola@'s hybrid localization concept, uses this method. GPS provides accurate positioning if at least 4 satellites are visible. This condition is usually fulfilled in rural areas and open places, but GPS does not work well in narrow street canyons. In such situations, a *complementary technology* is required to keep localization accuracy high. We use a method which is based on *receive power measurements* (RX-levels). A database holds RF-patterns (RX-levels of the serving and neighboring cells) which characterize a position. Multipath propagation and signal characteristics are taken into account automatically since the database is constructed from measured data. The obvious disadvantage of a large measurement effort is compensated by the expected accuracy also in urban areas and the applicability for legacy networks. When the position of a MS is requested by authorized entities, one or several consecutive snapshots of the RF-patterns are taken and compared to the database (Fig. 7). Time variance of the environment is combated via least squares and hidden markov models as well as tracking. The selection of localization methods is performed in real time, based on availability and QoS aspects.

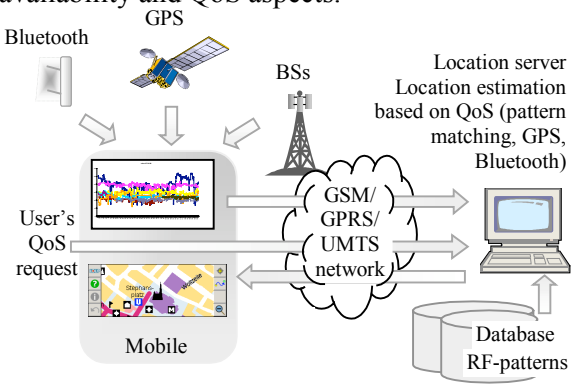

Figure 7: Lol@'s Hybrid Localization Concept

4. Summary and conclusions

UMTS will not depend on a single killer application, but it will use several killer components to implement various attractive services. Careful component design, data flow management and protocol optimizations are required to use the scarce resources efficiently. Localization/navigation, efficient content preparation and user interface considerations are the most important features, they are studied in LoL@, a UMTS application prototype.

5. Acknowledgements

This work is supported within the Austrian competence center program ***Kplus*** and by the member companies of ftw (Alcatel Austria, Austria Telecommunications, Connect Austria, Ericsson Austria, Mobilkom Austria AG, Nokia Austria, and Siemens Österreich).

References

[1] TS 22.121, "VHE", ftp://ftp.3gpp.org/Specs/2000-12/Rel-4/22_series/22121-400.zip, 3GPP, 12-2000.

[2] TS 23.127: "Virtual Home Environment (Release 4)", ftp://ftp.3gpp.org/Specs/2000-12/Rel-4/23_series/23127-400.zip, 3GPP, 12-2000.

[3] TS 22.127, "Stage 1 Service Requirement for OSA", ftp://ftp.3gpp.org/Specs/2000-12/Rel-4/23_series/22127-400.zip, 3GPP, 12-2000.

[4] TS 29.198, "OSA Application Programming Interface – Part 1" ftp://ftp.3gpp.org/Specs/2000-12/R1999/29_series/29198-320.zip, 3GPP, 12-2000.

[5] TS 23.057: "MS Application Execution Environment (MExE); Stage 2", ftp://ftp.3gpp.org/Specs/2000-12/Rel-4/23_series/23057-400.zip, 3GPP, 12-2000.

[6] Java Telephony API, JTAPI version 1.3, http://java.sun.com/products/jtapi/

[7] "Connected Limited Device config., Java 2ME version 1.0", http://java.sun.com/aboutJava/communityprocess/final/jsr030/

[8] "Mobile Information Device Profile, Java 2ME version 1.0", http://java.sun.com/aboutJava/communityprocess/final/jsr037/

[9] Keith Cheverst, et al, "Providing Tailored (Context-Aware) Inf. to City Visitors", 2000, Springer Verlag Berlin, Heidelberg.

[10] Donald Norman, "The Design of Everyday Things", Doubleday/Currency, 1990, ISBN 0385267746.

[11] Jenny Preece, "Human Computer Interaction", Addison-Wesley, 1994.

[12] W3C, "Extensible Markup Language (XML)", http://www.w3.org/XML

[13] W3C, "Extensible Stylesheet Language (XSL)", http://www.w3.org/Style/XSL

[14] Elliotte R. Harold, "The XML Bible, 2nd Ed.", Hungry Minds Inc., 2001, ISBN 0764547607.

[15] TS 23.271: "Funct. stage 2 descr. of LCS", ftp://ftp.3gpp.org/Specs/2000-12/Rel-4/23_series/23271-400.zip, 3GPP, 12-2000.