LoL@ - A Prototype of a Network Independent Wireless Internet Service

Harald Kunczier[1], Elke Michlmayr[1], Günther Pospischil[2], Hermann Anegg[1]

[1]ftw. Forschungszentrum Telekommunikation Wien, Donau-City-Str. 1, A-1220 Vienna
Email: {kunczier, michlmayr, anegg}@ftw.at,
Phone: +43-1-5052830-0, Fax: +43-1-5052830-99

[2]Mobilkom Austria AG & Co KG, Obere Donaustrasse 29, A-1020 Vienna
Email: g.pospischil@mobilkom.at, Phone: +43-1-33161-0

ABSTRACT

In this paper we introduce LoL@, the Local Location Assistant, a mobile tourist guide for the inner city of Vienna. The project was carried out by several industry and research partners including Vienna University of Technology and the Telecommunications Research Center Vienna (ftw). Since upcoming mobile services are likely to utilize Internet technology in addition to existing telecommunication infrastructure, our vision was to create such a wireless Internet service prototype. We designed and implemented a proof-of-concept demonstrator of a flexible service architecture for network-independent wireless Internet services. LoL@ offers the look and feel known from the Internet combined with additional network functionality like location determination and call control. Different IP-bearers like LAN, WLAN, GPRS, or UMTS are supported. LoL@ features interactive map-based navigation by guiding a visitor along a tour through Vienna's inner city, offers terminal positioning and the provision of multimedia information related to the sights. This information, as well as an online tour diary with personal entries and digital photos, is also accessible in advance for tour planning or after the tour for later use.

I. INTRODUCTION

The increasing data rates in mobile communications as well as the convergence between telecommunications and data communications like the Internet has amplified the creation of mobile applications. The WAP-based [1] application environment, already state-of-the-art, with its binary HTML like language has been a first step towards a more flexible application creation. MExE classmark 2 and 3 [2] introducing Java in the mobile world has paved the way for further improvements of mobile applications and allows the usage of familiar Internet technology. However, there are still numerous challenges for the design of mobile applications. In this paper we will present LoL@, a wireless Internet service, implemented as a proof-of-concept demonstrator to point out possibilities, challenges and problems in the intersection between the Internet and the telecommunications world. The issues described are divided into three sections. After this introduction the second section will introduce the service LoL@ including a short use case. Section III will then identify several challenges like to access different network capabilities and some security issues related to the interworking between network operator and service provider. Possible solutions are presented on the basis of LoL@. Section IV will emphasize LoL@'s application design concepts trying to overcome challenges arising from limited device capabilities like small display sizes and challenges resulting from the characteristics of wireless networks like low bandwidth or frequent disconnections. We will conclude in the last section with ideas for future work and a short summary.

II. LOL@ - A PROTOTYPE MOBILE INTERNET SERVICE

Developing applications for the mobile Internet is a challenging task because of technical and user interaction constraints. We will describe this in more detail in Sections III and IV. However, these constraints should be considered already during the design phase of a mobile service. Figure 1 gives an overview of which capabilities and limitations had been most decisive to the design process of LoL@. The result is a prototype of a mobile tourist guide offering

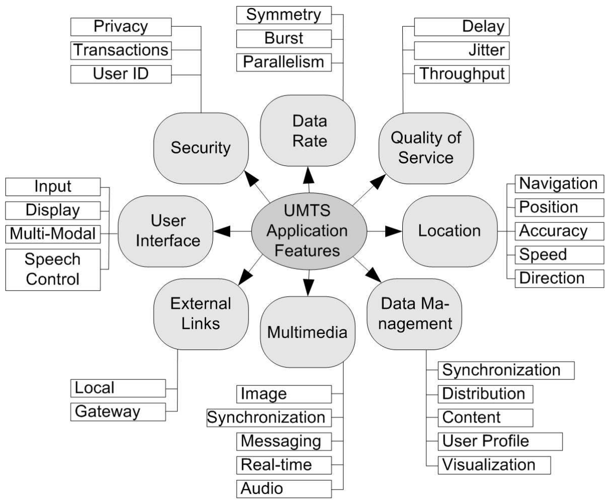
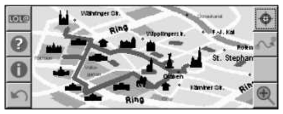
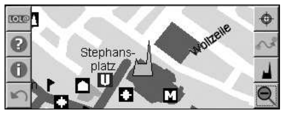


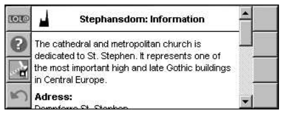
Fig. 1. UMTS Service Features



(a) Overview map



(b) Detailed map

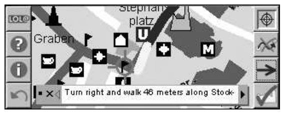


Stephansdom: Information

The cathedral and metropolitan church is dedicated to St. Stephen. It represents one of the most important high and late Gothic buildings in Central Europe.

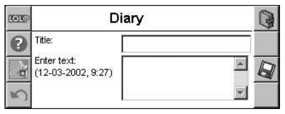
Adress:

(c) PoI Information



Turn right and walk 46 meters along Stock

(d) Navigation



Diary

Title:

Enter text:
(12-03-2002, 9:27)

(e) Tour Diary

Fig. 2. LoL@ GUI

- guidance to user selectable points of interests (PoIs)
- interactive multi-media information (pictures, video, text) about the selected PoI and
- a flexible tour diary with automatic and manual entries, allowing to substitute currently used textbooks.

The application offers a map-centric graphical user interface with additional hypertext information screens and is optimized for the limited screen size of mobile devices. A sample usage scenario looks like this: When the user starts LoL@ she/he is automatically authenticated (see Section III for further details) and logged on. Thus, the user does not have to enter an additional password. After choosing a tour she/he is presented with the overview map as shown in Fig. 2(a). A region of interest can be selected, the user is able to zoom into the detail map (Fig. 2(b)) and then select a PoI for which to get information (Fig. 2(c)). If the user decides to visit the PoI, LoL@ can determine her/his current position and navigate the user through the city of Vienna to the desired PoI. The navigation is assisted visually (Fig. 2(d)) as well as acoustically. When the PoI is found (or whenever the user wants to) textual notes or photos can be stored in the tour diary (Fig. 2(e)). Additional information like the admission fee of an exhibition or some photos to get a first impression can be requested to support the decision whether to visit or rather find an even more interesting PoI.

III. Architecture

Portable devices and mobile computers bring up a new set of problems the entire value chain has to cope with. Point-to-point and end-to-end solutions have been developed to become more independent of the traditionally slower development of the network domain. The open system model of the Internet has enabled everybody who has access to the network to act as an application service provider and has thus speeded up the service development process. However, in the Internet this has also paralyzed business by providing more opportunities for Denial of Service (DoS) attacks or spreading of viruses. Telecommunications in contrast has chosen the contrary way of standardizing the network as well as services in detail by the International Telecommunication Union (ITU) and other dedicated standardization bodies. The network is not constructed to just provide data bearer capabilities, but also contains intelligent elements, providing a set of well-defined telephony services, like call forwarding or barring. Consequently ”dumb” terminals can be used allowing easier interoperability between different telephony networks. The disadvantage of this network model is the cost and delay that occurs upon introducing new services [3]. Our approach to design LoL@ was thus to find an architecture-based solution as a compromise between the open Internet world and the strongly standardized telecommunications domain.

Figure 3 shows the architecture we use for LoL@. LoL@ may use GSM networks (utilizing a Network Access Server (NAS) and circuit switched communication) or packet networks (GPRS, UMTS). A CORBA-based OSA/Parlay API [4], as defined for UMTS, was implemented for localization and call control, allowing a simple connection to other OSA enabled networks (including 3G) without using network specific protocols. Thus, we can easily include features like click-to-dial and user localization into browser based third party Internet applications like LoL@.

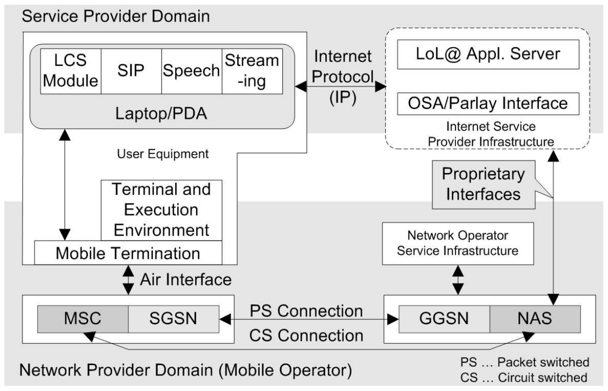


Fig. 3. LoL@ Architecture

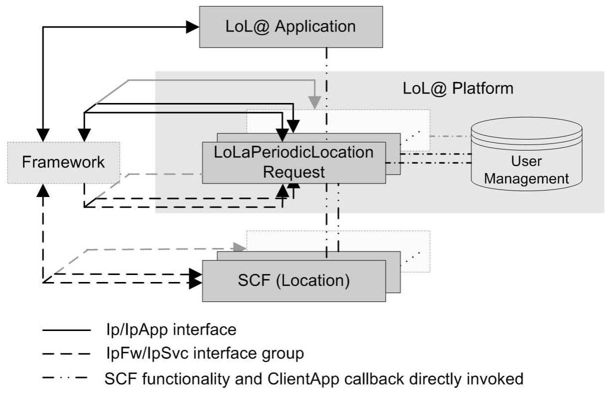

LoL@ Application
LoL@ Platform
Framework
LoLaPeriodicLocation Request
User Management
SCF (Location)
Ip/IpApp interface
IpFw/IpSvc interface group
SCF functionality and ClientApp callback directly invoked

Fig. 4. LoL@ Platform

The implemented architecture of LoL@ represents the entire OSA/Parlay architecture (except the framework) consisting of the Parlay services (service capability features) for localization and call control embedded in service capability servers hosted by the network operator. The main benefit of this type of architecture is the standardized API a third-party service provider can access. In the network layer we use SIP for the call control and a proprietary implementation for the localization capabilities. This was necessary since no localization technology, except cell ID, was available in the operator's network. The Parlay Application Programming Interfaces (APIs) are currently defined in a programming language independent Interface Definition Language (IDL) and will also be available in XML [5]. This allows the creation of a connection between the Parlay service and the Parlay service user via mutually calling distributed objects.

Additionally our architecture shows a platform between the OSA/Parlay API and the third party provider for two purposes. First, we offer simplified interfaces, similar to the currently running Parlay X efforts [6] since the majority of location based applications will require only a subset of the API's functionality to request the location information. Second, we enhance the offered service functionality while still keeping the interfaces compliant with the standard.

We have reduced the set of datatypes by avoiding 5 user defined datatypes and have also reduced the number of parameters (6 instead of 18) significantly. Web programmers, not familiar with accessing telecommunications services have reported less trouble during the implementation of LoL@ and thus the concept ensured a faster and more efficient implementation of the application.

The platform offers a simple third party user management, which coordinates the OSA/Parlay requests, and can compute the speed of a mobile station based on consecutive localization requests. This was implemented for demonstration purpose only, showing that this enhanced service can again provide APIs compliant to the Parlay standard and thus being offered to potential customers in a standardized way. This equals the ”VAS” (Value Added Service) concept described in [5]. The concept and a sample interface (for a simple periodic location request) is shown in Fig. 4. The advantages for a mobile service like LoL@ are (see also [5]):

- Network functionality can be extended, e.g. to offer velocity information. Even the role of a virtual network operator is imaginable.
- Interfaces can be adapted to meet specific constraints of service customers.
- Application users can be managed by the platform. In LoL@'s case this is restricted to the storage of a service access permission, but could be enhanced to a complete user management.

A. *Location Subsystem*

In the traditional world, where computers are not mobile, services do not have to be capable of delivering content based on the current position. In the mobile world however, the space domain can be used to prepare content individually according to the requirements of specific environments. Some can imagine location dependent information support like getting an on-line map showing the nearest restaurant or a push based advertisement showing special offers of miscellaneous stores. Requirement is a location enabled network infrastructure. Two important aspects we will discuss here: (1) the single user log in and (2) a Session Initiation Protocol (SIP [7]) based push architecture to provide location data efficiently.

1) LoL@ Single User Log In: The user has to be authenticated and the application has to be authorized by the network operator to prevent misuse. In the Telco world the SIM (Subscriber Identity Module) is used for

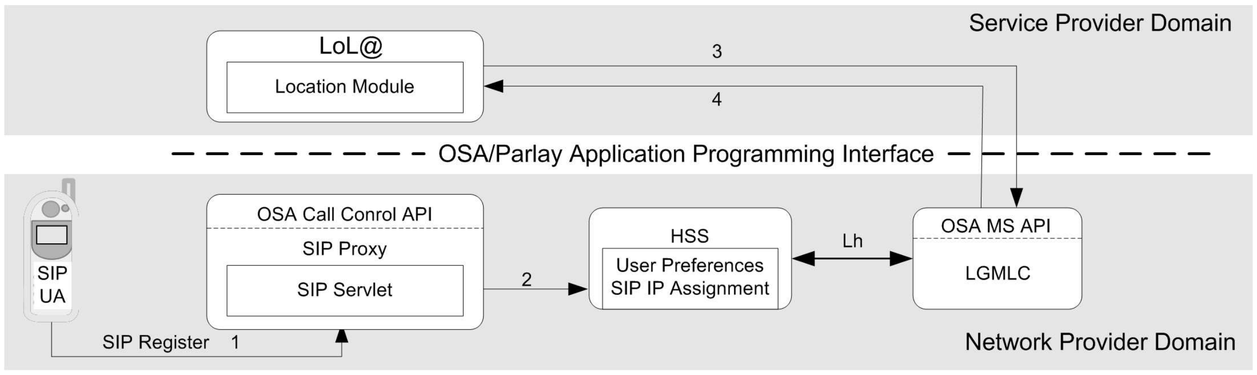


Fig. 5. LoL@ single log in concept

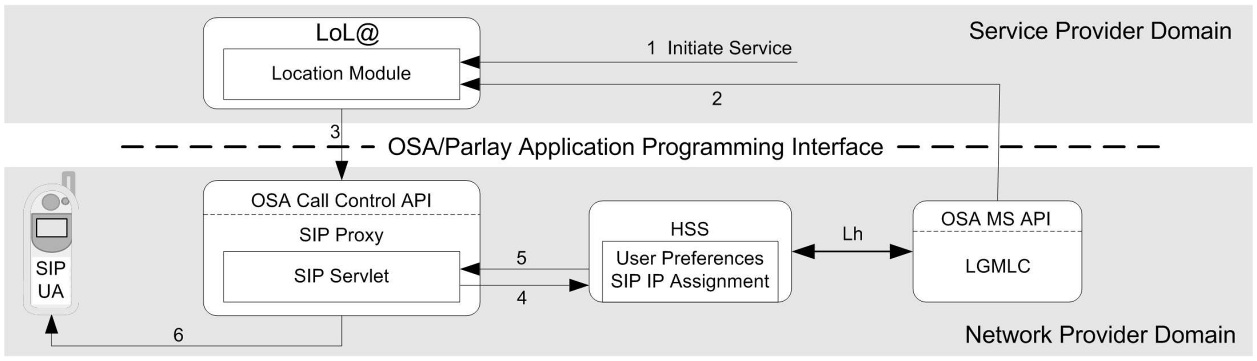


Fig. 6. LoL@ push

authentication and the authorization is stored in a central server, the HLR (Home Location Register). The core network itself does not provide any security mechanisms [3]. Instead it is simply protected by shielding it from the outside world, i.e. only a telephony operator can offer services or connectivity to the network. To allow access to the LoL@ service the user must thus be authenticated and authorized. In contrast to the Internet with repeated username-password log-ins we have implemented a SIP-based sign-on mechanism allowing a user to access all subscribed services after a single log-in when turning on the mobile device. The mechanism is shown in Fig. 5.

A SIP register message including the current network address (IP address) is automatically sent to a SIP proxy (step 1) after the mobile SIP UA (user agent) is turned on. This ensures that the UA is reachable in the corresponding network. Furthermore the currently assigned IP address is stored in the HSS (Home Subscriber Server) (step 2). The HSS thus contains the current SIP-IP assignment and personal settings of the specific user. Whenever a mobile service tries to access network functionality, e.g. for user localization, the HSS is checked and determines whether the service is authorized. Thus the user does not need to log on separately for LoL@. This task is performed automatically when the SIP UA is launched. The user can then access the entire service portfolio of LoL@.

2) LoL@'s SIP based push architecture: Push services use the limited bandwidth of mobile networks efficiently because communication only occurs if there is information available [8]. Thus, push services are especially suitable for the notification of asynchronous events. LoL@ uses this concept for the delivery of location data. Whenever new location data becomes available to the server (via the OSA API) the information is ”pushed” to the mobile device. The control flow (Fig. 6) is as follows: The user starts LoL@ and turns on the location detection, which activates the periodic location request (1). If a new location estimate is available, the GMLC notifies the LoL@ application (2). The application determines whether the movement of the user was big enough to send a notification to the LoL@ client running in the user terminal. In this case a SIP push is issued (3). The SIP proxy identifies the request and activates the push servlet which checks the push settings in the HSS (4,5). If push is allowed, the push request is sent to the SIP UA running in the terminal (6). The UA is responsible to forward the request to the local LoL@ applet which updates the display. It is important to note that this push request itself is not location based because the user location is the content of the request. Even though this situation might look like a standard SIP session activation at the first glance, there are three important differences:

1) Push preferences are checked
2) The SIP session is automatically accepted and closed after data delivery.
3) Session data is directly sent to the appropriate application.

IV. Application design concepts

For the LoL@ application itself, we rely on standard Internet protocols and technology, like HTTP, HTML and Java. The network is treated as a bit pipe extended by call control, single sign-on, and location functionality accessed via the OSA API.

Compared to the fixed Internet and its comfortable interfaces, formidable constraints are imposed by mobile devices and mobile networks.

The displays of mobile devices are small compared to desktop computers' displays. The input possibilities are limited. Application design for mobile devices is influenced by these characteristics. The use of the limited input and output media must be optimized to ensure a positive user experience. For the LoL@ demonstrator we assume a screen size of 320x120 pixels and pen input. A multimodal user interface was designed. A detailed discussion of usability issues concerning LoL@'s user interface as well as a documentation of the user interface design process can be found in [9] and [10].

Even if the network link is slow, the application's response time must correlate with the response times and attention spans of human beings. Users are impatient and do not want to wait for the information they have requested. In contrast to Internet applications, where the business logic resides completely at the server, we decided to split the application's business logic between terminal and server. Simple interactions (e.g. the pressing of a button) that cause only small changes in the state of the application can be handled by the terminal part. This minimizes network connections and hence improves the response time of the application. In addition, putting logic at the terminal makes it possible to optimize utilization of network resources. Whenever possible in LoL@'s data flow, we do not rely on the standard HTTP request/reply communication scheme, which sets up a network connection each time the user requests information, but transfer larger information items. Presentation of these data items to the user is then controlled by the terminal part of the business logic. This turned out to be especially useful for the interactive routing process. Given the user's source and his/her desired destination position as parameters of the request, all required information is gathered from the server in one chunk of data, which is subdivided and presented subsequently in the vector map and in text boxes at the terminal. The terminal part of LoL@'s business logic is implemented as Java applets on top of a Web browser to minimize the memory footprint of the application. The map viewer component as well as the application's service control facilities (service control buttons, see Fig. 2(a)), which are stored locally at the terminal, can either be downloaded at first use or installed permanently. In the latter case, version management of local components must be considered. This is subject for future work.

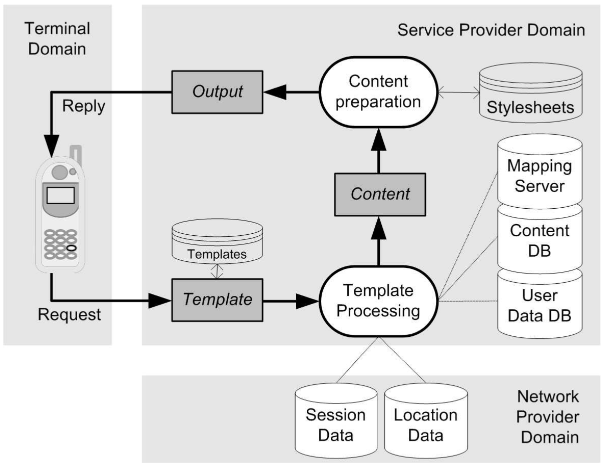


Fig. 7. Content preparation and content delivery

To overcome problems with network disconnection (caused either by the user or by the network) we implemented a suspend/resume functionality that allows a user to continue using LoL@ in the same state as before the interruption. Relevant application state information is stored at the LoL@ server, allowing a resynchronization of client and server states upon a client restart. Using SIP addresses for user tracking is secure compared to the unreliable mechanisms available in Internet applications (hidden HTML form fields, URL rewriting or cookies) where data about the client's state is transferred between client and server in a way that can be manipulated by the client. Since state synchronization may not happen permanently because of the limited bandwidth, only major changes in user state completely done at the client (e.g. switch between map and text view) are reported to the server for resume. Minor changes are not reported and therefore lost when resume occurs.

For the server part of LoL@'s business logic, it is necessary to build a scalable architecture that is able to adapt and deliver content to a wide variety of mobile devices. A content delivery system to support different Java-based terminal types was designed (see Fig. 7). Device independence is achieved by uncoupling application data and presentation of that data. The design strictly separates

- content retrieval (business logic) and
- content presentation (presentation logic).

The *business logic* component is responsible for collecting content from different data sources within the service provider domain, e.g. a database with tourist attractions, a routing server, or a video streaming server. The LoL@ user interface including its data flow is completely defined using templates in XML [11] format. The template-based approach results in an extensible and easy-to-maintain system that supports adding new data sources and subsequently defining a user interface for these new features. Changing the application's user interface by editing the templates is a straightforward task.

The templates do not define the design of the screens, but rather their structure, i.e. the data items necessary. A template contains commands, links, and parameters. Commands are used to define which data is needed and from which data source it has to be fetched. Links describe LoL@'s data flow, i.e. hypertextual structure. The HTTP parameters of the request are needed for processing. Template processing consists of three steps:

1) Substitute the parameter placeholders in the template with their actual values.
2) Execute commands, e.g contact the various data sources to to fetch the data.
3) Check links: If no content is available for a link, it is removed.

The results of template processing are filled-out templates which contain all data to satisfy the user's request.

In the next step, the *presentation logic* component prepares the content data according to the viewing device's capabilities. Different presentations of the same data are delivered to clients depending on which presentation is suitable for being displayed on the terminal. XSL [12] stylesheets define which layout information is added to content data during the XSL transformation process. Different output formats, like WML [13] or HTML for different display sizes, can be supported by creating appropriate XSL stylesheets. Integration of a content negotiation system like CC/PP or UAProf is subject for future work. More information about the content delivery system can be found in [14].

V. Conclusions and future work

In this paper we have introduced LoL@, the Local Location Assistant, a mobile tourist guide. We have demonstrated that the combination of standardized internet technologies on top of wireless networks is a promising concept for future wireless services. Standardized interfaces to create a functional split between the network and service providers, but open enough to keep it flexible for new ideas, are necessary to allow fast and efficient service development. In the near future more powerful mobile devices will allow a wide spreading of multifunctional services exceeding common ones like simple data download, voice traffic or MMS. LoL@ demonstrates a promising concept, but further work will be necessary to allow a higher personalization regarding content as well as the presentation of the content. Network functionality must become easily accessible to service providers while security e.g. for push services must be ensured.

References

[1] Wapforum, "Wireless application environment specification," 1999, http://www.wapforum.org.

[2] 3GPP TSG Services and System Aspects, "TS 22.057: Mobile Execution Environment (MExE), Service description, Stage 1 (Rel. 4)," April 2001, http://www.3gpp.org/ftp/specs/2001-03/Rel-4/22 series/22057-400.zip.

[3] G. Pospischil, "Application development for UMTS," Ph.D. dissertation, Vienna University of Technology, Department of Communications and Radio-Frequency Engineering, October 2002.

[4] 3GPP TSG Services and System Aspects, "TS 22.127: Stage 1 Service Requirement for the Open Service Access (OSA) (Rel. 4)," April 2001, http://www.3gpp.org/ftp/specs/2001-03/Rel-4/22 series/22127-410.zip.

[5] S. Bessler and J. Zeiss, "Eine Offene Service Plattform für Telekomanwendungen," *Praxis der Informationsverarbeitung und Kommunikation*, vol. 4, pp. 202–207, 2002.

[6] Parlay X Working Group, "Parlay APIs 4.0, Parlay X Web Services White Paper - Version 1.0," December 2002, http://www.parlay.org.

[7] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol," *RFC 2543, IETF*, March 1999, http://www.ietf.org.

[8] G. Pospischil, J. Stadler, and I. Miladinovic, "Location based push architectures for the mobile internet," November 2002, to be published in R. Prasad ans S. Dixit (ed.): Wireless IP Evolution, Artech House, Norwood, USA.

[9] G. Pospischil, M. Umlauft, and E. Michlmayr, "Designing LoL@, a mobile tourist guide for UMTS," in *Proc. 4th Symp. Human-Computer Interaction with Mobile Devices*, September 2002.

[10] M. Umlauft, G. Pospischil, G. Niklfeld, and E. Michlmayr, "LoL@, a mobile tourist guide for UMTS," *Journal of Information Technology & Tourism*, 2003.

[11] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, *Extensible Markup Language (XML) 1.0*, 2nd ed. World Wide Web Consortium, October 2000, http://www.w3.org/TR/2000/REC-xml-20001006.

[12] J. Clark, *XSL Transformations (XSLT) Version 1.0*. World Wide Web Consortium, November 1999, http://www.w3.org/TR/1999/REC-xslt-19991116.

[13] *Wireless Markup Language version 1.3 Specification*, WAP Forum, http://www.wapforum.org/.

[14] E. Michlmayr, "Flexible Content Management for the LoL@ UMTS application," Master's thesis, Vienna University of Technology, May 2002.