Leveraging GIS Technologies for Web-Based Smart Places Services

Cristiano di Flora and Christian Prehofer

Office of the CTO, Nokia, Finland
{cristiano.di-flora,christian.prehofer}@nokia.com
http://www.nokia.com

Abstract. This paper describes our experiences and lessons learnt in building a Geographic Information System (GIS) specifically designed for indoor location-based services within our smart places infrastructure. The proposed system was built by mashing-up commodity Open Source software and novel research prototypes realized within our labs. The overall approach relies on intense usage of standard web technologies and REpresentational State Transfer (REST) APIs as a way of enabling easy mashup of off-the-shelf and proprietary components. The key design and implementation aspects of our solution are described in detail, including a discussion on how we represented and augmented the concept of indoor location within our services. Further, we show how we integrated them with commodity GIS services originally designed for outdoor scenarios.

Keywords: Indoor, GIS, Smart Spaces.

1 Introduction

A smart space is a multi-user, multi-device, dynamic interaction environment that enhances a physical space by virtual services [3]. These services enable the participants to interact with each other as well as with other objects in the smart space. Indoor smart spaces are of particular interest in this context because people spend most of their time indoor rather than outdoor, which in turn makes the potential impact of indoor Location-Based Services (LBS) much bigger than that of outdoor LBS.

There has been considerable research and commercial success on outdoor GPS-based LBS, which motivated us to focus on services for indoor smart places. This paper describes our experiences and lessons learnt in building a Geographic Information System (GIS) specifically designed for indoor location-based services within our smart places infrastructure. While many research prototypes are built from scratch, our focus here is to understand how existing GIS technologies can be used for our mobile, indoor solution. In this way, we aim both to extend the scope of GIS systems and to use them as standards for indoor smart place applications. The proposed system was built by mashing-up commodity Open Source software and novel research prototypes realized within our labs. The overall approach relies on intense usage of standard web technologies and REpresentational

U. Brinkschulte, T. Givargis, and S. Russo (Eds.): SEUS 2008, LNCS 5287, pp. 256–267, 2008.

State Transfer (REST) APIs as a way of enabling easy mashup of off-the-shelf and proprietary components.

The paper is organized as follows. Section 2 discusses the rationale behind our work and related research. The key design and implementation aspects of our solution are described in detail in Sections 3 and 4, respectively, including a discussion on how we represented and augmented the concept of indoor location within our services, as well as on how we integrated them with commodity GIS services originally designed for outdoor scenarios. Section 5 concludes the paper by outlining the main lessons learnt and future research directions.

2 Motivation and Background

There is considerable research on ubiquitous and pervasive computing, also recently focusing more on internet of things or ambient computing. Many research projects have developed and trialed ubiquitous services. Also, several projects have developed software platforms and frameworks for this purpose. There is extensive literature on this, for instance [4] surveys 29 approaches up to 2004. This survey covers many systems for distributed mobile computing and most of them consider location information, even though positioning technology was not widely spread at this time. There is also more recent work on indoor positioning services, such as [5,13], and also on symbolic location models, going beyond plain coordinates [6].

GPS based LBS have been widely successful and have created an enormous ecosystem of applications using such services. It could even be argued that the easy access of web-based map tools has fueled the integration of applications by using the mashup approach. For instance, it is now very common for web-based services to show geo-tagged items on a map application. These interactive and integrated web-based applications have gained enormous momentum and are also labeled Web 2.0 technologies. Web 2.0 basically means that Web-sites are interactive, that users are actually creating content, and that the content and applications can be combined by application mashup and tagging of content.

This issue has been noted and research on user generated content for ubiquitous systems has been developed, e.g. in [7], focusing on annotating and user generated content. We think that there are a number of more essential research issues towards a wide spread eco-system of indoor positioning services. In our view, the key issues for the widespread usage of indoor positioning services are as follows:

- establishing common standards and practices for indoor positioning and map services which are interoperable and easily available;
- integrating indoor positioning services into existing web-based services;
- enabling mobile, context aware applications which are easy to deploy and use;
- ensuring privacy and security regarding personal data such as location information.

Regarding the first item, we see several missing pieces. First, indoor maps are currently not widely available in well-known standard formats, protocols, and positioning infrastructures. While GPS is nowadays widely available outdoors, there is no such established and deployed indoor positioning technology. Moreover, most of them need dedicated device-side or infrastructure-side hardware. Indoor positioning has been using very different technologies such as bluetooth, RFID or WLAN based. Even further, there are no established standards for handling maps and geo-spatial data, whereas such standards are available for outdoor LBS [12]. Our approach here is to build on WLAN based positioning, which we see as a widely available technology and further use such open standards for geospacial information systems for indoor settings.

Regarding the second issue above, integrating with existing services is a must, as such services not only provide considerable technology but also toolkits and substantial amount of already available geospatial data as, for example, in outdoor maps applications. This may also include personal data, such as private contacts or pictures. We also observe here that most of these application mashups use technologies which are simple and integrate easily into web applications, such as REpresentational State Transfer (REST) APIs and RSS feeds.

Another problem is that deploying mobile services is difficult, as there is a large variety of devices and different operating systems (in different version) on the market. Furthermore, these devices can be quite different in terms of resources like memory and connectivity. This problem has hampered the deployment of mobile services in general, and it is more severe in our case as we also want to integrate positioning information with other context data. We are focusing on web based applications as they are easy to access, to deploy, and to manage. However, they do not have access to local context data of the user. Different options can be chosen in this context, based on downloadable client software for context collection as well also using upcoming standards for browser-based access to context data such as the W3C Delivery Context Client Interface (DCCI) initiative [16].

We focus in this paper on the above challenges. In summary, we show how to use GIS solutions for indoor settings and show how the web can be used as a platform for these services. This covers connecting to services in other web based applications. We also discuss how to integrate context information into our service. A similar architecture for mobile devices is presented in [1], where the location context data is sent separately to a server, while the service is hosted from a web server which obtains the context data from this server. This paper focuses on visibility models and does not cover indoor positioning aspects as done here. Other works on applying mobile GIS solutions are for instance [8], which focuses on tour guide applications. [5] focuses on indoor GIS for mobile devices, but does not address application mashup and platform aspects nor interactive JavaScript frontends as we do here.

Another challenge, which is closely tied with the above ones, is to connect with (other) applications and enable application mashups while preserving privacy. The issue is that ubiquitous context information is typically privacy sensitive

and existing application mashup techniques do not support this suffciently. We do not cover in full detail more secure ways for application mashup. Here, we see a few key ingredients emerging, such as OpenID and OpenAuth [19], which rely on novel and more flexible, decentralized approaches to authentication and authorization and thus can provide a mashup-friendly security infrastructure.

3 The Proposed Web-Based Indoor GIS

3.1 The Web as a Platform for Smart Places

Before discussing how we designed and implemented our indoor GIS, it is worth shedding some light on the architecture of the overall web based smart spaces platform that the GIS is part of. A detailed discussion of this architecture goes beyond the scope of this article. The interested reader may refer to [11] for further details about it. In this sub-section we will focus on the key design decisions underlying this architecture, and on their effects and implications on the proposed GIS solution. Compared to related work in this area, the decisions that characterize our approach are as follows:

HTTP-based communication: HTTP and web services are used as the primary means for integrating software across devices in the smart space, similarly to what currently happens on the Internet, where HTTP forms the cross platform glue that allows mashups across such an extremely heterogeneous network infrastructure like the Internet.

Reuse of existing web technology: a key problem with existing solutions in the ubiquitous and pervasive computing research community is that they are rarely reusable. By relying on existing web technology, the infrastructure opens up to a large number of devices already available in the current market.

Multiple runtimes for application execution: High end mobile devices are now offering a much wider range of runtimes for implementing and running applications and services, including traditional Java and C++ runtimes as well as support for Python and other scripting languages. In this way, platform developers can use several run times. This means that many existing components used on the web can be used in a smart space context as well. Moreover, having the basic location enablers available through HTTP based communication, makes it possible and easy to mashup local (both situated and on device) and remote location-based web services all together.

The system is organized in multiple layers, which are depicted in Figure 1 and briefly described in the following. It is worth noting that, as far as indoor GIS components are concerned, in this sub-section we will only describe their very high-level role within the overall Smart Places platform picture. Please refer to Section 4.2 for further details about how we designed the GIS components and what commodity components we included in them.

The **Base Platform & Communication** layer contains several commodity commercial and/or open source components that we see as necessary to realize a

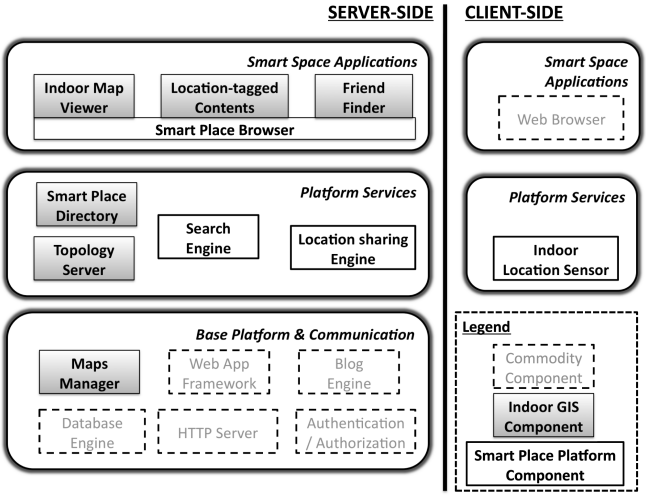

Fig. 1. Bird's eye view of our Smart Places platform, including server-side (left side) and client-side (right side) components. Greyed boxes represent Indoor GIS components. White boxes represent either commodity open source components (dashed border boxes) or smart places platform specific components (solid border boxes).

full web platform in the smart space. Using commodity web components allows us to bring many features to the smart space such as, for example, easy creation and deployment of services using well known authoring/distribution tools and framework for web applications, user management and security solutions, database and content management systems. As far as location-awareness support is concerned, the platform currently includes a **Maps Manager** component, which is in charge of providing Create Read Update Delete (CRUD) primitives for indoor maps data, i.e., raster layers representing building and floor plans, as well as several features (vector) layers representing floor topology elements (rooms, corridors, halls) and static Points-of-Interests (e.g., in an airport smart place, they could include restaurants, shops, check-in desks, terminals, gates, and other categories of interest to people visiting that space).

Platform Services can be created by using the technologies in the **Base Platform & Communication** layer. A few typical platform services are shown in Figure 1. They include two actual indoor GIS components, namely **Topology Server** and **Smart Place Directory**. The **Topology Server** is in charge of providing low-level information about the physical structure of the available smart places, such as for example details about how a given building is structured in floors, wings, and rooms. Places are modeled by using a hybrid hierarchical location model [6], and each physical location is assigned a URI that other services and applications can use as tags to associate items (e.g. media contents, blog posts, user location) to a certain physical location. The **Smart Place Directory** acts as a mediator between **Smart Space Applications** and the other indoor GIS components of the platform (i.e., components belonging

to either the **Platform Services** or **Base Platform & Communication** layers). It provides a set of REST APIs that allow **Smart Space Applications** to access other **Platform Services** and **Base Platform & Communication** functionality through a consistent and common API. The API allows CRUD access to most of the location-dependent data available in the smart place, such as people location, location-specific contents, and POIs, in a web-application friendly way. In fact, it supports several widely adopted data-interchange formats, including XML-based formats like ATOM and RSS feeds, as well as more lightweight formats such as the JavaScript Object Notation (JSON). All the mentioned components so far are intended to be deployed on the server-side.

Smart Space Applications re-use and combine the **Platform Services** functionality with other on-device features in order to provide meaningful and helpful functionality to end-users. These may be web applications, which can be accessed using a browser and which can be hosted on the mobile web application server, or alternatively they can be implemented as stand alone applications written using any of the existing device specific development kits and that access the deployed **Platform Services** through http-based communication. In Figure 1 we show a few key examples of such applications, which are described in the following. The **Smart Place Browser** represents a very generic entry point to available applications, providing an AJAX API for **Smart Space Applications** developers to create new end-user applications. The API exposes and leverages the key abstractions implemented by **Platform Services** to a developer-friendly interface. New applications, such as the **Indoor Map Viewer**, **Location-tagged Contents**, or **Friend Finder**, can be easily implemented on top of this API, as we will show in Section 4.2.

It is worth noting that, as Figure 1 clearly shows, our approach is based on a very thin-client model, in which no particular **Base Platform & Communication** or **Smart Space Applications** components are assumed to be deployed and pre-installed on the client-side. In fact, in order to use the services, the client-side just needs to have a web browser capable of rendering rich web applications. Additional components, such as for example the **Indoor Location Sensor** in Figure 1, might be required in order to enable usage of some applications (like the **Friend Finder**) or to improve user experience with other applications (e.g., to automatically adapt or initialize the UI of the **Indoor Map Viewer** and **Location-tagged Contents** applications based on the actual indoor location of the end-user).

3.2 The Adopted Indoor-Positioning Technique

The proposed solution relies on an experimental WLAN indoor positioning technique under research and development at Nokia Research Center. The technique is based on WLAN scanning and on further processing of the scanning results, including measurement of the received signal strength from all reachable access points, from which the current location of the mobile device is calculated. All steps are performed on the terminal side, e.g. on end-users smart phone or PDA. In the rest of this sub-section we will just shed some light on the key aspects of

this technique that are required in order to understand the herewith described indoor GIS solution. The interested reader can refer to [9] for further details on its design and implementation.

One interesting aspect of the adopted indoor positioning technique lies in its quick and easy deployment in out-of-the-lab real-world settings. In fact, the technique only requires a-priori knowledge of a list of known WLAN APs along with information about their physical location in the target building (which is typically a well-known piece of information for, but it does not require any off-line measurements of the received signal strength nevertheless. In other words, no radio maps of the target environment and related calibration of the algorithm are required, which in turn makes the proposed smart places infrastructure more easy to set-up than other state-of-the-art solutions [10].

The outcome of the proposed algorithm is a symbolic location information structured according to the location model mentioned in Section 3.1. It is worth noting that the concepts of building, floor, and section could be eventually replaced by other concepts and semantics if needed. In other words, different symbolic location models with different granularities could be adopted as far as the technique is accurate enough to support the required granularity. For example, in an environment with very large sections and rooms, such as a shopping mall or an airport, the model could also take into account the concept of rooms within a single section.

4 Implementation and Prototyping

4.1 Implementing the Indoor GIS Prototype

In Section 3.1 we introduced the main indoor GIS components at a very high-level of detail. Since we wanted to create a practical GIS solution for indoor smart spaces that could work also from mobile devices, we needed to combine the web-based smart space platform and the indoor positioning technique, described in Section 3.1 and 3.2, respectively, with additional components providing traditional GIS functionality, such as maps, navigation, and geo-spatial queries support to commercial mobile devices.

In compliance with our overall smart spaces approach described in Section 3, we decided to rely on open APIs and protocols suitable for integration with web-based applications and services. In the following we discuss a few key implementation decisions that we needed to take when prototyping the indoor GIS part of our smart places platform. In addition, we also provide further details about how indoor GIS components interact one with each other in order to fulfill their responsibilities. More specifically, we describe how we implemented and prototyped our first example of an indoor GIS system based on the design guidelines and concepts described in Section 3. The overall architecture of the implemented prototype is depicted in Figure 2.

When implementing the first prototype we had to satisfy the key requirement of providing simple REST APIs for other services to create composite functionality (mashups) out of the basic building boxes provided by our platform. To

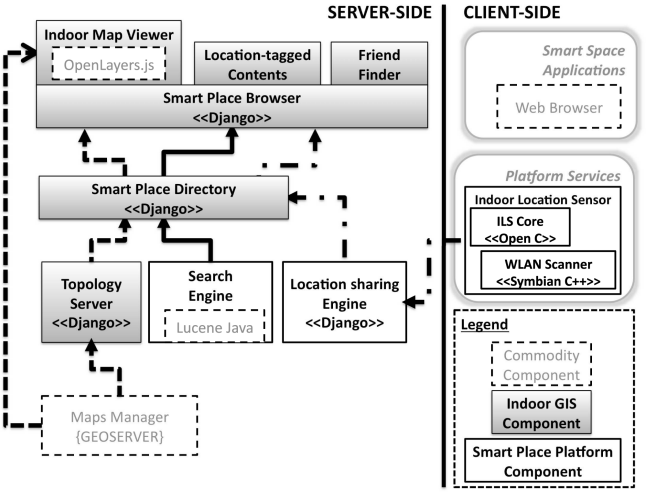

Fig. 2. The architecture of our indoor GIS implementation prototype. Greyed boxes represent Indoor GIS components. White boxes represent either commodity open source components (dashed border boxes) or smart places platform specific components (solid border boxes). Dashed lines, solid lines, and dash-dotted lines represent interaction related to Indoor Map Viewer, Location-tagged Contents, and Friend Finder case study applications, respectively.

this aim, we implemented the **Smart Place Directory** in the Python programming language by using the Django web application framework. Using the Django framework enabled us to quickly implement different facades for the same back-end data, in order to support CRUD interfaces to manage location-dependent smart place resources based on all the data interchange formats (JSON, ATOM, RSS) mentioned in Section 3.1. Similarly, most of the realized web services were implemented as Django applications. This gave us a lot of flexibility in designing the actual web service interfaces we wanted to use to expose the indoor location-dependent data stored in our back-end.

As far as the **Maps Manager** component is concerned, we wanted to rely as much as possible on established standards for representing indoor GIS data, such as maps, points-of-interests, and related meta-data: to this aim, we decided to adopt the Open Geospatial Consortium (OGC) Web Map Service (WMS) and Web Feature Service (WFS) [12] to represent maps and POIs in the **Maps Manager** component. This decision was motivated not only by technical requirements, but also by our higher level goal of evaluating how easy and feasible was the idea of using commodity GIS solutions to support indoor LBS. Several commodity implementations of WMS/WFS servers were available in both the commercial and open source community. After evaluating the different available options, we decided to adopt the open source Geoserver platform [14], which provided quite a complete implementation of WMS and WFS specifications. Geoserver was also very well integrated with other GIS tools (such as, for example, uDig [17] or GRASS GIS [18]

standalone tools, as well as with JavaScript GIS APIs such as OpenLayers [15]). However, since we wanted to rely just on WMS and WFS services, Geoserver could be in principle be replaced by any other working implementation of the WMS and WFS specifications (like the Mapserver software for example), without affecting the rest of the platform. It is worth noting that, since Geoserver satisfied all the requirements we had for the **Maps Manager**, it is indicated in Figure 2 as a commodity open source component, and not as a novel indoor GIS component as previously indicated in the conceptual design diagram of Figure 1.

When designing our solution, we wanted to provide very open APIs for accessing the available Indoor GIS data (not only from WMS/WFS sources but also from other existing solutions) from both mobile devices and fixed/desktop devices. To this aim we had to consider that, although the OGC web services already provided a comprehensive set of APIs, their interfaces were not very friendly to JavaScript / Ajax developers. Moreover, as already mentioned, we wanted to keep the APIs as open as possible, so as to not preclude interoperability of our solution with other protocols different from the OGC WMS/WFS standards. To satisfy such requirements we decided to adopt the open source OpenLayers JavaScript API [15], which provided quite a comprehensive and developer-friendly set of JavaScript abstractions to deal with most common GIS functionality (e.g., dynamic creation of maps by mashing up map data coming from different sources, DOM interface to most of the supported GIS data representation formats) while solving already a lot of common cross-browser issues related to differences in the low-level JavaScript APIs of different browser engines.

4.2 Evaluating the Indoor GIS Prototype

In order to validate and to refine our JavaScript GIS interfaces, we implemented a case study application, namely **Indoor Map Viewer**, which combined our indoor GIS back-end services and accessed them through an OpenLayers interface. Similarly, in order to test the indoor location sensing feature, we implemented a **Friend Finder** application which allowed end-users to check their own friends' location, to see it on a map, and to evaluate the distance between themselves and their friends in the smart place. In order to show indoor maps, the **Friend Finder** application re-used most of the **Indoor Map Viewer** functionality. Similarly, in order to validate and refine the location-based search functionality, we implemented a **Location-tagged Contents** case study application, which allowed end-users to generate, search, and retrieve location-based contents, such as pictures, videos, text documents, and blog posts and comments.

The implemented applications allowed us to evaluate and refine the web-based approach to indoor LBS service provisioning to commercial mobile devices. The implemented applications confirmed the feasibility of using web technologies for fast and easy deployment of smart places services. Most of the time was spent to implement and refine the business logic of our indoor GIS components, and we were able to easily deploy and run the services on a heterogeneous device base. As for server-side components, we were able to deploy them on Linux, Windows, and Apple Mac Os X devices. The implemented case study examples, realized

as Ajax applications, could be accessed from mobile devices, including Nokia S60 devices as well as Linux Internet Tablets (Nokia N800 and N810), and in general from any device running a web browser that included either the Mozilla or Safari web engines (including desktop / laptop clients).

As far as JavaScript clients for geospatial services are concerned, using OpenLayers raised a few performance issues for resource constrained devices like the Nokia S60 devices. These issues included at least the following problems. The first problem consisted in that dynamic memory footprint requirements of OpenLayers are still too large with respect to a not negligible set of commercial devices. OpenLayers maps initialization requires about 2.5 MB of RAM memory to be dynamically allocated, and further usage of some special OpenLayers controls can easily increase the amount of dynamically allocated RAM up to 5 or 6 MB. The dynamic memory footprint analysis also showed that dynamic memory allocation in OpenLayers components has not been optimized for memory constrained devices. A lot of RAM resources, allocated by OpenLayers code, were not released, thus leading to frequent and not negligible memory leaks, which in turn caused also more recent and powerful devices to return memory full errors when trying to visualize some of the implemented applications. While these dynamic memory issues do not cause any problems in desktop or Internet Tablet devices, on certain low-end or less recent devices (e.g. Nokia N80 or E70) it was impossible to run the implemented applications, due to the limited amount of RAM memory available on those devices. We believe that this problem will be solved in future devices, since the amount of RAM memory available on such devices is rapidly increasing. However, the memory leaking problems can only be solved with a different and more efficient memory allocation / deallocation approach in OpenLayers implementation. We believe that similar problems may arise when trying to reuse other commodity libraries, initially designed for desktop clients, for mobile rich internet applications development.

Another problem we experienced is related to OpenLayers dependency on DOM 2.0 APIs. OpenLayers assumes a complete DOM 2.0 or 3.0 model to be supported by the browser engine. Unfortunately, browser engines even on very recent devices (like the Nokia N95) do not fully support these specifications. This created problems when parsing some of the XML documents returned by our smart places API through the DOM API. The problem was solved by recoding the applications in such a way that they were relying on JSON-formatted interface rather than on ATOM/RSS formatted data. In this way we were able to guarantee that all applications could still work on all the mentioned types of devices. Such a modification had also the positive side-effect of improving the performance of the provided applications due to the more light-weight logic required for parsing and creating the data.

5 Lessons Learnt and Future Work

This paper discussed our experiences with building an indoor GIS based on commodity GIS standards and protocols and Web 2.0 application development

principles. We showed that the combination of scripting languages with web application frameworks, such as Python and Django, gave us a lot of flexibility in designing the actual web service interfaces we wanted to use to expose indoor GIS data. We implemented a few case study applications on top of the proposed GIS solution, which confirmed the feasibility of using web technologies for fast and easy deployment of smart places services on a heterogeneous set of commercial off-the-shelf devices. As far as JavaScript GIS clients are concerned, we pointed it our that using commodity libraries, such as OpenLayers, can create a few performance issues for resource constrained devices. Moreover, commodity libraries might have not been designed by taking into account the limited RAM capacity of mobile devices. Frequent and not negligible memory leaks created problems also on more recent and powerful devices. Overall, we believe there is a clear need in research and industry to agree on standards for indoor LBS, with respect to both geospatial data representations and related APIs to re-use them in a Web 2.0 environment. Existing outdoor-related standards might lay the groundwork for such activities, even though they should be extended in order to support not only the concept of location as physical position but also symbolic location concepts. Our future work will concern refinement of the positioning technique and related location model, as well as work on more thin clients better suited for mobile usage in terms of memory consumption and UI paradigms.

Acknowledgments. The authors would like to acknowledge the contributions from the Smart Place project at Nokia Research, in particular Jilles van Gurp and Heikki Mattila.

References

1. Simon, R., Fröhlich, P.: A mobile application framework for the geospatial web. In: WWW 2007: Proceedings of the 16th international conference on World Wide Web, pp. 381–390. ACM, New York (2007)
2. Griswold, W.G., Shanahan, P., Brown, S.W., Boyer, R., Ratto, M., Shapiro, R.B., Truong, T.M.: Activecampus: Experiments in community-oriented ubiquitous computing. Computer 37(10), 73–81 (2004)
3. Wang, X., Dong, J.S., Chin, C.Y., Hettiarachchi, S.R., Zhang, D.: Semantic Space: an infrastructure for smart spaces. IEEE Pervasive Computing 3(3), 32–39 (2004)
4. Endres, C., Butz, A., MacWilliams, A.: A survey of software infrastructures and frameworks for ubiquitous computing. Mob. Inf. Syst. 1(1), 41–80 (2005)
5. Candy, J.: A Mobile Indoor Location-based GIS Application. In: 5th International Symposium on Mobile Mapping Technologies (MMT 2007), Padua, Italy (2007) (last checked on 6.5.2008), `http://giswww1.bcit.ca/georanger/candy_jonathan.pdf`
6. Becker, C., Durr, F.: On location models for ubiquitous computing. In: Personal and Ubiquitous Computing, vol. 9(1), pp. 20–31. Springer, Heidelberg (2005)
7. Lopez-de Ipina, D., Vazquez, J., Abaitua, J.: A context-aware mobile mashup platform for ubiquitous web. In: 3rd IET International Conference on Intelligent Environments, pp. 116–123 (2007)

8. Kim, J.W., Kim, C.S., Gautam, A., Lee, Y.: Location-Based Tour Guide System Using Mobile GIS and Web Crawling. In: Kwon, Y.-J., Bouju, A., Claramunt, C. (eds.) W2GIS 2004. LNCS, vol. 3428, pp. 51–63. Springer, Heidelberg (2005)
9. Hermersdorf, M.: Indoor Positioning with a WLAN Access Point List on a Mobile Device. In: International Workshop on World-Sensor Web (WSW 2006), Boulder, CO, USA, October 31 (2006) (last checked on 6.5.2008), http://www.sensorplanet.org/wsw2006/9_Hermersdorf_indoor_pos_WSW2006_final.pdf
10. Haeberlen, A., Flannery, E., Ladd, A.M., Rudys, A., Wallach, D.S., Kavraki, L.E.: Practical robust localization over large-scale 802.11 wireless networks. In: Proceedings of the 10th Annual international Conference on Mobile Computing and Networking (MOBICOM 2004), pp. 70–84. ACM, New York (2004)
11. Van Gurp, J., Prehofer, C., di Flora, C.: Experiences with Realizing Smart Space Web Service Applications. In: 1st IEEE International Peer-to-Peer for Handheld Devices Workshop at the CCNC 2008 conference in Las Vegas (2008)
12. OpenGIS standards specifications, http://www.opengeospatial.org/standards
13. Ekahau Inc., http://www.ekahau.com/
14. Geoserver web site, http://geoserver.org
15. OpenLayers web site, http://www.openlayers.org
16. Delivery Context Client Interfaces (DCCI) 1.0, W3C Candidate Recommendatio (December 2007), http://www.w3.org/TR/DPF/
17. uDig - User-friendly Desktop Internet GIS, http://udig.refractions.net/
18. GRASS - Geographic Resources Analysis Support System, http://grass.itc.it/
19. OAuth Core 1.0 Protocol (December 2007), http://oauth.net/core/1.0/