MapBiquitous – An Approach for Integrated Indoor/Outdoor Location-Based Services

Thomas Springer

Technische Universitt Dresden, Computer Networks Group,
01062 Dresden, Germany
thomas.springer@tu-dresden.de

Abstract. Nowadays, location-based services based on GPS and map data are commonly available. Since GPS does not work in buildings and map data provide geographic information only, such services are limited to outdoor scenarios. Large research effort has been carried out to explore indoor positioning and navigation systems but up to now mostly proprietary and isolated solutions have been proposed. In this paper, we introduce the MapBiquitous system as an approach towards an integrated system for indoor and outdoor location-based services. It is based on a decentralized infrastructure of building servers providing information about the building geometry, positioning infrastructure and navigation. Building servers are dynamically discovered and provided information is seamlessly integrated into outdoor systems by the MapBiquitous client. We discuss the general approach and architecture of MapBiquitous and present experiences of implementing the MapBiquitous concepts and location-based services on top of it.

Keywords: location-based services, indoor positioning, building model, OpenGIS, Android, MapBiquitous.

1 Introduction

Finding an indian restaurant nearby, navigating to a selected target address, tagging information to a particular location or presenting information about the building the user is in front of are location-based services (LBS) commonly available for outdoor settings. OpenStreetMap, GoogleMaps and other services provide the map data necessary to implement such services while GPS or WiFi positioning is used to determine the user's current location.

Similar use cases exist for indoor scenarios. Museum guides providing information about the objects in close proximity to the user, navigation within a building from room to room or finding a printer nearby are such use cases. They have been already implemented as research prototypes and some even as productive systems. The major difference between outdoor and indoor LBS is the lack of a well established and broadly used technical foundation for indoor services. While GPS and map services operate in a global scale, indoor positioning systems are based on proprietary infrastructures usually limited to single buildings

J.Y. Zhang et al. (Eds.): MobiCASE 2011, LNICST 95, pp. 80–99, 2012.

and information about building geometry is hardly available to the public. Thus, services like navigating from home directly to the right terminal in the airport, tagging information to a shop in a shopping mall, and more general setting up indoor location-based services in an interoperable and integrated manner are hard to implement.

In this paper we present the MapBiquitous system following an integrated approach for indoor and outdoor location-based services. The research explores concepts and standard technologies to allow the provision of LBS for indoor and indoor/outdoor scenarios in an interoperable way. Based on the main idea to provide the necessary data about building geometry and annotating it with information about positioning infrastructure, navigation and semantic location information outdoor and indoor positioning, navigation and information tagging should be seamlessly coupled.

The main contributions of the presented work are the decentralized system architecture, the approach for modeling and providing building data, including information for indoor positioning and routing based on open standards (namely WfS, GML and OpenLS Directory Service) and a concept for the creation of client applications with efficient access to decentralized building information and seamless integration into outdoor systems.

The paper is organized as follows: in section 2 related work is discussed. After discussing the requirements in section 3 we present the MapBiquitous approach and main architecture in section 4. In section 5 we report on the implementation of the MapBiquitous system and the experiences obtained by implementing LBS using MapBiquitous. We close the paper with a conclusion and outlook to future work.

2 Related Work

Location-based services are a topic of research with different aspects. Fundamental technologies are required to determine the position of objects, devices and persons and to set them into relation with reference systems, geographical data or building data. On top of such an infrastructure, concepts for data access and service architectures are investigated to create LBS.

Positioning Technologies: GPS is well established for determining locations in outdoor settings but does not work properly indoor [5]. Different approaches try to provide Indoor GPS based on additional infrastructure in buildings like repeaters or to improve GPS receivers to deal with the decreased signal strength in buildings. Approaches based on cellular networks work also in buildings but are not accurate enough for indoor services.

This is the reason of the many efforts to create indoor positioning systems. Many research projects try to exploit the WiFi infrastructure widely available in buildings and mobile devices. Different approaches mainly based on lateration or fingerprinting [7] exist. The major issue is that strength of WiFi signals in buildings is influenced by manifold factors, like material and constellation of walls, used device, and device direction. Therefore, calculating the distance for

lateration based on signal strength does not yield accurate results. With fingerprinting, actual signal strength values of all visible access points are recorded for reference points in the intended coverage area. After that setup phase client locations can be determined based on matching a current measured fingerprint with the stored fingerprints. This concept can produce more accurate results with the cost of a large effort for setup and maintenance of the fingerprint database. Anyway, issues like influence of device direction and different scales for signal strength in different devices still remain unsolved [3].

Alternative approaches based on an infrastructure use ultrasonic [17] or infrared signals [16,9] or use RFID technology to determine the position of a device. Dependent on the density and arrangement of beacons these systems can achieve high accuracy. Their major drawback is the high effort for installing an additional infrastructure. Hence, such systems are not widely deployed.

Inertial systems work without a pre-installed infrastructure. Based on a known starting location relative movements are tracked with a sensor fixed on the body or placed freely in the pocket and used to calculate the new location [18,14]. In [19] an approach is described to even avoid the knowledge of the initial location. A common issue of inertial systems is the integration drift, measurement errors add over time and constantly increase inaccuracy of the determined position.

In summary, non of the mentioned approaches is in a state to complement the globally available GPS for indoor positioning. Our approach does not introduce any new positioning method but integrates multiple positioning technologies. In addition, we provide the necessary information for positioning like access point locations, fingerprint records, or beacon information as part of the building model. Thus, as long as the user's device is able to interoperate with the positioning technology in a building it can dynamically obtain the necessary information and use that technology in the building.

Geographic Data Provisioning: A second major building block for LBS is spatial data to visualize locations and to attach semantic information to them. Geographic Information Systems (GIS) serve as data providers for geographic information for outdoor services. Under the term web mapping applications like OpenStreetMap or Google Maps provide access to geographic data like street maps and satellite imagery. Via APIs functionality like searching and routing is made available and can be integrated into custom applications.

In this context the Open Geospatial Consortium (OGC) specifies interfaces and protocols to support interoperable solutions for accessing spatial information and providing LBS. The Web Map Service (WMS) [4] is a OGC standard for offering geo-referenced map data as raster images. The Web Feature Service (WFS) [15] is a service to provide geographic features encoded in XML. Such features might be meta data provided in addition to map data for spatial analysis but also vector data represented in XML. Both services can be accessed via HTTP.

While WMS usually responses data encoded in GIF, PNG or JPEG, the default payload of WFS is Geographic Markup Language (GML) [10]. GML is also specified by OGC as an XML-based language for geographic features including

representations of vector objects based on elements like point, line and polygon. Thus, GML is one option to model building geometry. CityGML, a particular schema definition for GML enables the modeling of 3D city models including buildings and their environment. In contrast to generic 3D vector formats like SVG or VRML, CityGML provides elements with the necessary semantics to represent and analyze city models including navigation functions. KML is used in Google Earth to model geographic features like geo-tags and routes, but it is a proprietary format.

In our approach we adopt WFS as a service for the provisioning of building information in GML format. Since both are OGC standards we ensure high interoperability. In addition, WFS can be based on various data sources as backend. Available implementations usually support various data formats and databases. In this way, building data providers are not restricted to a particular format but can choose any format as long as it can be transformed into GML.

Service Engineering: Systems like MagicMap[1] or PlaceLab[2] integrate various positioning technologies for indoor and outdoor. Both systems use WiFi lateration as main positioning method which require the availability of information about access point locations. Visualization of current location is based on floor plans which are provided as images. Thus, building geometry is not explicitly modeled in a standard format and the current semantic location is not known to the system. Locations are available based on a local reference system only.

In [13] a simple approach for indoor positioning and navigation based on dead reckoning and 2D barcodes is introduced. Users can download floor plans by reading a 2D barcode attached to publicly available maps in the building. With the download also the current position is provided. Due to the simplicity of the approach it could be adopted in any building. Anyway, building data is not explicitly modeled and positioning is limited to dead reckoning.

The REAL system [1] is a pedestrian navigation system which combines to sub-systems. The IRREAL (Infrared REAL) sub-system supports indoor positioning and navigation while ARREAL (Augmented Reality REAL) offers the same functionality outdoors. The system combines infrared-based positioning and GPS. The system uses map data for outdoor and 3D data in a proprietary format for indoor locations. The system is based on dedicated hardware which is restricted to infrared and GPS. Thus, the approach is limited to buildings which are equipped with the infrared infrastructure. In addition, building models have to be provided in the special format.

In [3] a concept for campus wide positioning based on WiFi fingerprinting is explored. The user's position is visualized on a map or a floor plan. Both are based on proprietary data sources, the fingerprinting data is separately provided in a database. At all, the system is limited to the campus with all necessary geographic, floor and fingerprinting data available. Positioning is restricted to WiFi fingerprinting.

[1] `http://www2.informatik.hu-berlin.de/rok/MagicMap/`
[2] `http://sourceforge.net/projects/placelab/`

As a conclusion, to the best knowledge of the authors there is currently no single system providing a technological foundation for integrated indoor and outdoor LBS. Even if some of the technological building blocks are available and partial solutions like standard services and modeling language or integrated support for multiple positioning technologies have been investigated large research work has to be carried out. The introduced MapBiquitous system represents a first step towards that envisioned goal of seamlessly integrated indoor/outdoor location based-services.

3 Requirements

The goal of the MapBiquitous system is to provided indoor and outdoor location-based services in an interoperable and seamlessly integrated manner. In the following we analyze the requirements for such a system.

As discussed in the previous section, there is no single positioning technology which can be globally used. GPS is available outdoors and should be complemented with indoor technologies. Since there is no standard for indoor positioning available, *multiple positioning technologies* should be supported by the system (A1). To achieve a seamless integration, all positioning technologies should be accessible via a *uniform interface* (A2). In addition, during movement the user might enter different buildings and later on leave them to move on. Therefore, the system should support a *seamless handover between different positioning technologies* if the user enters a new area (A3).

By explicitly providing data about the building geometry, positioning infrastructure and navigation, geographic data available for outdoors should be complemented with appropriate indoor information. To achieve a wide adoption, the data modeling and provisioning has to be based on *open standards* (A4). For high interoperability, all data should share a *common reference system* to represent geometric coordinates (A5). To allow the representation of meaningful location information the provided building data should contain *geometric and semantic information* (A6). The creation of building models should be possible with *minimal effort* (A7). The *model should be generic* to allow the modeling of any necessary aspect not limiting its applicability to specific domains (A8).

Access to building models should be based on a *standard format and protocol* (A9). Even if creation and maintenance of building models requires effort, data should be provided with *high accuracy and timeliness* (A10).

The overall system should operate in the same manner like todays outdoor systems. To support a large number of users the system has to be *highly scalable* (A11). It has to be *highly available* with minimum down time (A12). Building data, supported positioning technologies and system architecture should be *highly extensible* to support the integration of new data and technologies (A13). Last but not least, the system should be *easily to be adopted and integrated into applications* (A14).

4 MapBiquitous Approach

The major objective for creating MapBiquitous was to provide a system for the seamless integration of building data and positioning information with existing outdoor technologies in a way that location-based services could be created on top of it. In the following we introduce the architecture and main components of MapBiquitous and discuss our design decisions.

The architecture of MapBiquitous is shown in figure 1. The system consists of a client and a server part. On server side we foresee a decentralized infrastructure of *building servers* which can be discovered based on a *directory service*. MapBiquitous clients basically consist of the three components *Loader*, *Renderer* and *Locator*. In addition, an internal representation of building data is stored in the *building data storage* on the client. The Loader is responsible for accessing the building servers. It loads and processes building data to fill the building data storage.

The Locator contains several modules to support different indoor positioning technologies. It accesses the information provided for positioning in the building. This can be for instance locations of the WiFi access points but also for infrared beacons or RFID tags in the building. The information is used to configure the associated positioning module which is than started to provide periodic location updates.

The location updates are received by the Renderer which is responsible for the visualization of location information. It adopts the concept of overlays to present map and building data in an integrated manner. The Renderer accesses the building data storage for rendering the building geometry and triggers the Loader in case of location updates for dynamic access to building data. It is notified by the Loader if new building data is available in the storage.

The building servers are responsible for providing building data for public access based on a standard format and protocol. They maintain information about the building geometry, positioning infrastructure and navigation for one or a set of buildings.

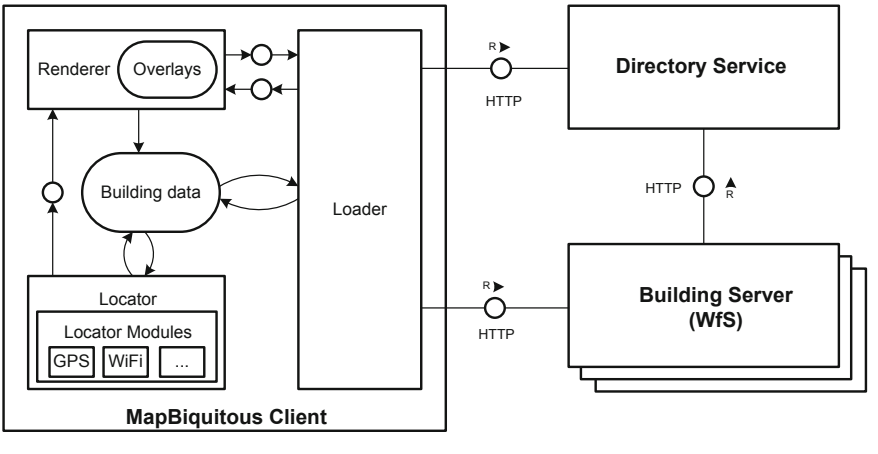

Fig. 1. Architecture of MapBiquitous

We decided to use a decentralized architecture to support high scalability (A11) and availability (A12) of the system and data. In contrast to a central server, with a decentralized architecture load is naturally balanced between the building servers and a single point of failure as well as a performance bottleneck are avoided. In case of failure of one building server only a single building or a small set of buildings is not available. All other data remains accessible.

Another reason for the partitioning of building data to decentralized servers is caused by the building data itself. Compared to map data, we assume a much higher change rate of building data. Even if the building geometry is stable, semantic information like usage of rooms and navigation information might change frequently. Furthermore, information about the positioning infrastructure will change over time, for instance if the WiFi infrastructure is updated. Moreover, building data is maintained by the owner or user of the building. Depending on the usage and type of the building we expect that the owner might want to decide if and what information about the building is provided to the public. We assume that owners want to keep control about the published data. A decentralized approach naturally supports these requirements since any building owner can decide by its own to provide a building server with the extend and level of detail he is willing to publish.

Issues of the decentralized approach might be the heterogeneity of technology, formats and quality of building data. Building data providers have to agree on common standards to offer their data in an interoperable way. It is up to the provider to keep the building data up to date and to provide accurate data. Even if these issues are challenging, we believe that they could be solved by establishing standard technologies. Because of the technological and administrative advantages we decided to follow a decentralized approach.

In the following sections we describe the introduced MapBiquitous components in detail.

4.1 Building Model

MapBiquitous should provide information about building geometry to complement the outdoor location information based on maps. Associated with that information data for indoor positioning and navigation should be provided. As previously discussed we propose a logic partitioning of building data at the granularity of buildings. Thus, each building is represented with its own self contained model. This model is layered as shown in figure 2. The concept foresees a layer for the building outline and one layer per floor. In the example we present the model of the computer science faculty building of TU Dresden which consists of the building outline, a basement and 4 floors. These layers cover the model of the building geometry per floor.

Rooms, floors and areas of stairs and elevators are represented as polygons. For representing the coordinates we use the World Geodetic System (WGS84). The WGS84 reference system has a global scale, thus allowing to use coordinates with a uniform reference system all around the globe (A5). WGS84 is also used by GPS and map services for representing locations. Thus, no mapping of coordinates

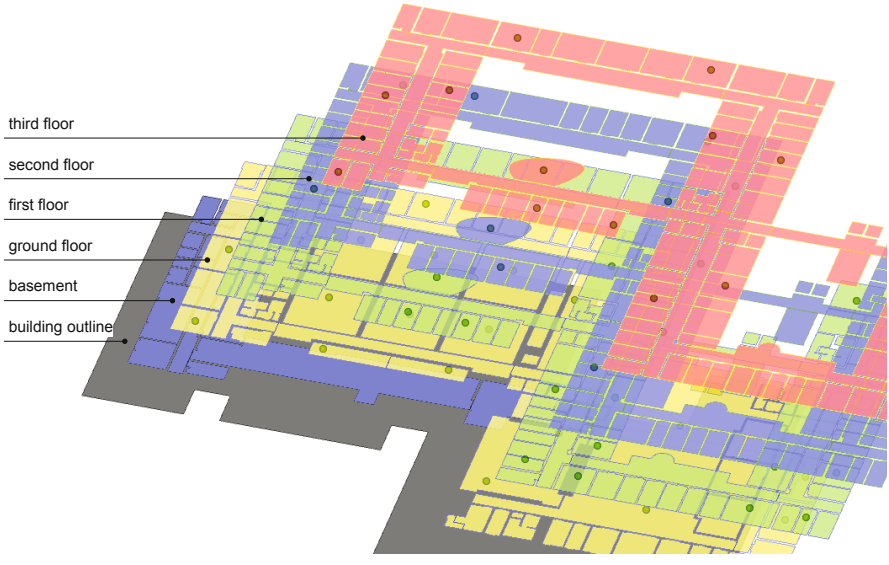

Fig. 2. Layered model of the computer science faculty building of TU Dresden

is required on client side. Especially, building geometry can be directly drawn on maps without complex pre-processing. In addition, semantic information like room number, type and usage of rooms is attached to the polygons (A6).

In addition to the floor plans additional layers are provided containing information for positioning and navigation. The idea is to add one layer per floor and positioning technology available in the building. Another layer is added per floor for navigation information. In this way, the building model can cover different positioning technologies in parallel and can be easily extended (A1).

For access building data is represented based on GML. It is a standard language specified by OGC for representing geographic features including vector objects based on elements like point, line and polygon (A8). For the client side, especially for mobile clients, GML is challenging because of the required XML processing. We decided to use GML for flexible and highly interoperable access to building models. In combination with WFS as service for access it is possible to support various internal representations of building data, which can be transformed into GML (A9).

4.2 Building Data Access

As described in the previous section building data is provided as vector data on distributed building servers. To access building data from a client, the building server providing the appropriate data has to be discovered. After that, building data has to be downloaded to be locally available at the client.

There are three methods for including building information into services and applications: embedded, initial URL and dynamic discovery. With the *embedded*

method, building data is delivered and installed together with the service or application package. This is useful if LBS should be provided for a fixed set of buildings. In this way transfer and processing time is avoided.

A more dynamic way of accessing building data is to start with a *known building server URL*. As described, each building can contain links to neighboring buildings. Thus, starting with an initial URL close-by building data can be discovered avoiding the necessity of providing a directory service. This method can be applied to provide LBS for a larger set of buildings without the effort to provide a directory service.

The most flexible method is the *dynamic discovery* of building services using a directory service. As shown in figure 1 a directory service or a federation of directory services has to be provided. Clients can lookup building servers by providing their current location. The directory service responds with a list of URLs for building servers in proximity to the given location. Of course, building servers have to be registered before with their URL and location.

OGC specified the OpenGIS Location Service Specification (OpenLS) [8] to define interfaces and protocols for an interoperable service framework. One of the core services is the Directory Service which allows to register and lookup object or resources with location and further attributes. The OpenLS Directory Service uses WGS84 coordinates as well. Thus, coordinate transformations between different reference systems can be avoided (A5). We decided to use OpenLS Directory Service to support the dynamic discovery method in MapBiquitous because of these features and its availability as a standard (A4).

The WFS is the service of choice for accessing building data in MapBiquitous. WFS provides a simple protocol and interface for accessing geographical features based on HTTP. Its main operations are `getCapabilities()`, `describeFeatureType()`, and `getFeature()`.

Based on a given or gathered URL access to a WFS server starts with a `getCapabilities()` request to get information about the WFS server and a list of layeres offered for the associated building. For each layer name, title, projection and its extend are provided with the response. From the response the available layers can be extracted. Information about the detailed layer type and content can be accessed using the `describeFeatureType()` operation. One request has to be send per layer. In this way, building geometry layers can be distinguished from positioning information and navigation layers without loading the whole content of the layers.

To access the complete layer data, a layer can be requested using the `getFeature()` operation. Thus, download of layer data can be accomplished stepwise, starting with the layer containing the building outline and the layer representing the current user location. In addition the layers providing information about positioning technologies supported by the client device can be loaded.

Hence, WFS allows the exploration and download of the provided features, i.e. building model layers, in a dynamic and selective way ensuring extensibility of and flexible access to the provided data based on a standard format and protocol (A9).

4.3 Client Architecture

As mentioned during the introduction of the architecture, the access, processing and visualization of building data is performed on the client side following the concept of a fat client. An alternative would be to process building information on the server side and to provide image data of maps with building data already projected on the map to thin clients. This would ease the implementation of clients, especially web clients. Ideally, an integration into existing map services could be achieved enabling the use of existing clients for map data.

Anyway, beside the visualization of building geometry information about positioning technologies is typically required at the client side to perform positioning. Therefore, existing clients would have to be extended or the functionality for positioning has to be separated from the visualization of the map and building data. Processing of building data at the server would also limit the flexibility of how building data could be used.

We decided to follow the approach of a fat client to support high flexibility for service creation with MapBiquitous. The main decision is to support layered vector data instead of pre-processed image data. If the provided building servers based on WFS are combined with map servers based on WMS the creation of web clients could be supported as well using the building data of MapBiquitous.

In the following sections we describe the client-side components in detail.

Building Data Storage. The core of the MapBiquitous client is the storage for building data on client-side. It is based on the location model depicted in figure 3. The location model is a hybrid model [2] which supports the representation of geometric as well as semantic location information. Especially, relations

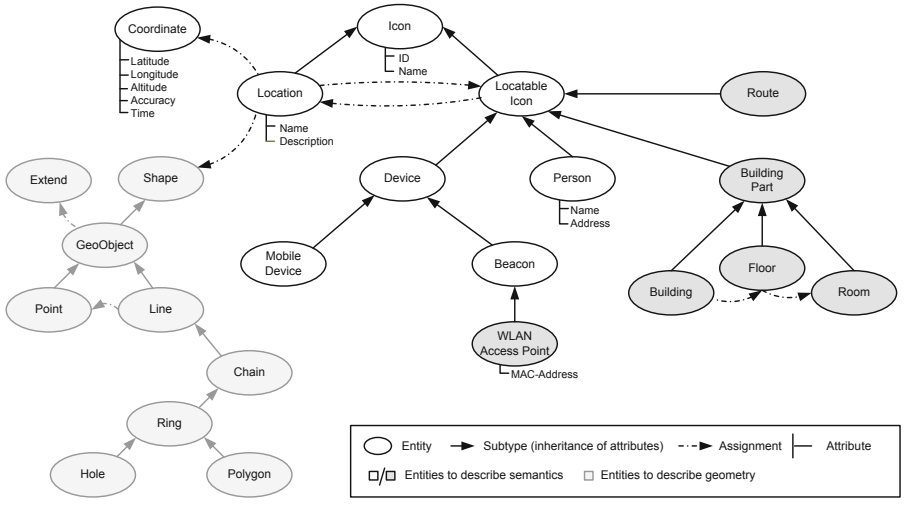

Fig. 3. Location model used as internal data representation in the client-side building data storage

between entities containing geometric and semantic information can be explicitly represented (A6).

The root element is `Item`. Every other entity directly or indirectly inherits from that element. Thus, each element has an id and name. The direct subtypes of Item are `Location` and `LocateableItem`. LocateableItem can be used to represent any object which can have a location like `Person`, `Device`, `BuildingPart`, or `Route`. A `BuildingPart` can for instance be a `Floor` or `Room`.

All named elements contain semantic information. The geometric information is represented by the elements derived from `Location`. The element `Coordinate` assigned to `Location` represents the geometric position of an `Item` with a latitude, longitude and altitude based on WGS84. Moreover, each `Coordinate` has a accuracy value and a time stamp.

The element `Shape` it the main element to represent the geometry of a `Location`. Based on the different subtypes arbitrary geometric structures can be represented using `Points`, `Lines`, `Chains`, etc. In this way building structure can be internally represented. The element `Extend`, assigned to each `GeoObject` represents the minimal and maximal values of coordinates of a `GeoObject`.

The presented location model integrates geometric and semantic infomation by assigning a `Location` to a `LocateableItem`. The location model is extensible and allows the representation of arbitrary geometric and semantic information.

Loader. The Loader component is responsible for discovering building servers and accessing building data. It is triggered by the Renderer to load data from building servers. Two cases are handled by different notifications. In case of a location change, the Renderer triggers the Loader to retrieve building data for a defined geo-window. This geo-window defines a geographic area by the minimum and maximum values for latitude and longitude. The loader starts with an lookup at the directory service to discover all building servers in the given area. Based on the lookup results the Loader than checks the building data storage for cached data. Data not available locally is requested from the building servers. For newly discovered buildings the loader requests the building outline and the description of available positioning information.

Further floor data and the layers for positioning technologies supported by the device are downloaded on follow-up requests by the Renderer. They are triggered when the device is in close proximity to a building or by user interactions.

After the download of building data XML processing is performed and new building data is stored in the building data storage. After that the Renderer is notified by the Loader about the availability of new building data.

Renderer. The Renderer is responsible for the integrated visualization of map and building data. The basic graphical layer presents map data. For the integration of building data the Renderer maintains a set of overlays. Based on WGS84 which are the common base for map and building data building geometry can be directly drawn on the map. The conversion of WGS84 coordinates to screen coordinates is usually provided by map-based views directly.

The Renderer accesses the building data storage to obtain the data about the building geometry. Based on the current view building data is represented in different levels of detail. For each level of detail a separate overlay is maintained. Based on the distance between the user position or focus point of the view buildings are represented in three different levels. For the lowest level of detail an icon is presented for each discovered building server (LOD1). For the next higher level of detail the building outline is drawn (LOD2). At the highest level of detail the complete floor plan is presented (LOD3).

In case of location changes and user interactions like changing the zoom level of the view or scrolling on the map the Renderer triggers the Loader to load building data. It gets notified by the Loader if new data is available in the building data storage.

Locator. The Locator is responsible for the positioning of the user device. It manages a set of locator modules which access to device hardware for positioning. They share a common interface for accessing location information (A1, A2). The Locator configures, starts and stops the Locator modules dependent on the proximity to buildings supporting the positioning technologies (A3). For instance, to save energy its useful to switch on WiFi and GPS only if needed. Indoors GPS can be usually deactivated while WiFi should be activated to get location updates.

The Locator accesses the building server storage to obtain the information required to configure the locator modules. For instance the WiFi access points and their locations are stored in the location model. In case of location changes the Locator sends notifications to the Renderer.

5 Evaluation

For evaluating our concepts we implemented MapBiquitous for desktop and mobile devices. To test the MapBiquitous system two applications were implemented, namely a WiFi analysis tool for the university campus and a location-based game. We report on the experiences made during the implementation of the MapBiquitous system and the applications.

5.1 MapBiquitous Implementation

MapBiquitous Server. MapBiquitous has been implemented using standard technologies on the server side. The directory service is based on the OpenLS Directory Service standard and provides an interface for registering and looking up resources with WGS84 coordinates and further attributes. The functionality has been implemented based on Apache Tomcat, MySQL and JSP. HTTP requests are processed by a JSP servlet, which parses the XML requests, extracts the geo-window, queries the database to get all available building servers in that area and generates the response messages.

For the provision of building servers the WFS implementation MapServer[3], particularly the package MapServer for Windows[4] is used. MapServer is an open source project running on different platforms including Microsoft Windows, Linux and Mac OS X. It supports a rich set of raster and vector data formats, geographic data sources and OGC standards (among them WMS, WFS, and GML). MapServer for Windows runs with Apache HTTP Server and PHP.

We currently provide building data for the campus of TU Dresden. Building geometry and semantic information about room numbers and usage of rooms is available from an internal database for building management maintained by TU Dresden. This data has to be modified to enable provision with WFS. For modification we use the open source tool Quantum GIS[5]. Information for positioning and navigation is currently added manually using Quantum GIS. We currently model the location of access points for WiFi positioning. According to the concept separate layers are introduced for access point locations for each floor. In addition we modeled fingerprint data and path information for navigation in separate layers. With Quantum GIS the internal representation of building data is transformed to ESRI shapefiles [6].

ESRI shapefiles consist of a main file (.shp) containing a sequence of records of geometric elements, a dBASE table (.dbf) containing the attributes of all geometric elements in the main file and an index file (.shx) which contains the information to assign the attributes to the geometric elements. Coordinates of geometric elements are represented based on WGS84.

The shapefiles can be directly used as data sources for MapServer. The configuration of MapService is performed based on a map file (.map). The map file contains all information about the data to be provided by WFS. It contains a link to the local shapefiles, a definition of the URL for accessing the building server, and information about the provides layers.

Currently all logic building servers are physically implemented on the same server. We provide building data for four buildings at the university campus, namely the computer science faculty, the mensa, the lecture hall and the "'Barkhausenbau"', a complex building used by different faculties.

MapBiquitous Client. Two clients have been implemented for MapBiquitous: a desktop client and a client for mobile devices.

Desktop Client. The implementation of the desktop client (see figure 4) is based on Java Standard Edition. Visualization in the *Renderer* adopts the JXMapViewer component of the SwingX-WS-Framework by SwingLabs[6]. The component supports the visualization of maps from different providers, namely OpenStreetMaps, Google Maps and Yahoo Maps. On top of the map layer overlays can be created. The user interface allows interaction with the map and building data like zooming and dragging, searching for building servers, selection of layers for presentation (floor plans and WiFi access point locations).

[3] `http://mapserver.org/`
[4] `http://www.maptools.org/ms4w/`
[5] `http://www.qgis.org/`
[6] `http://www.swinglabs.org`

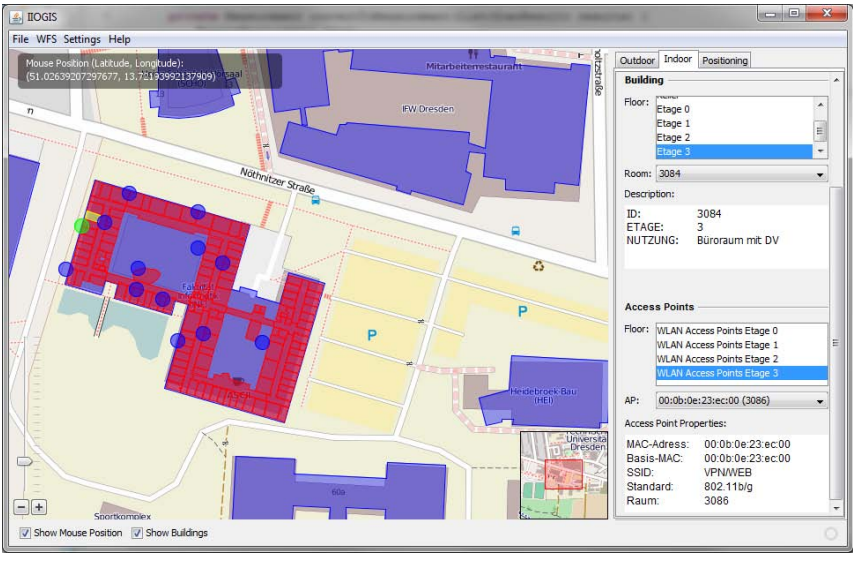

Fig. 4. Desktop client for MapBiquitous

The *Locator* supports GPS and WiFi positioning based on lateration. GPS Positioning has been tested with the Holux GPSlim236 device, an external GPS device which can be connected via USB or Bluetooth. The Bluetooth connection is mapped to a serial port. After defining a port for the connection to the GPS device, GPS location information can be accessed via a CommPort. WiFi positioning is based on PlaceLab. PlaceLab consists of the three components *Spotter* for accessing the WiFi adapter, *Mapper* for the provision of access point locations and *Tracker* for calculating the current position. The Mapper is connected with the building data storage of the client to allow access of the access point locations. With that information, the Tracker can be started to periodically calculate the current position of the device.

The *Loader* uses HTTP connections based on the java.net library for accessing WFS. DOM4J and XPathQueries is used for XML processing. All available data is loaded from building servers immediately after discovery. In this way after a short waiting time all building data is available.

Android Client. The mobile client was implemented using Android 2.2. The *Renderer* uses the `MapView` and `MapViewActivity` provided by Android to integrate Google Maps into applications. The MapView provides the possibility to create overlays which is used by the Renderer to integrate building data and map data. The user interface provides zooming and dragging for the MapView (see left part of figure 5).

During the implementation we noticed a limitation of the MapView regarding zooming. MapView only supports zooming up to level 19 (21 in satellite mode). Higher zoom levels can be set but zooming is not performed. This zoom level was not appropriate for presenting floor plan details. Thus, as a workaround

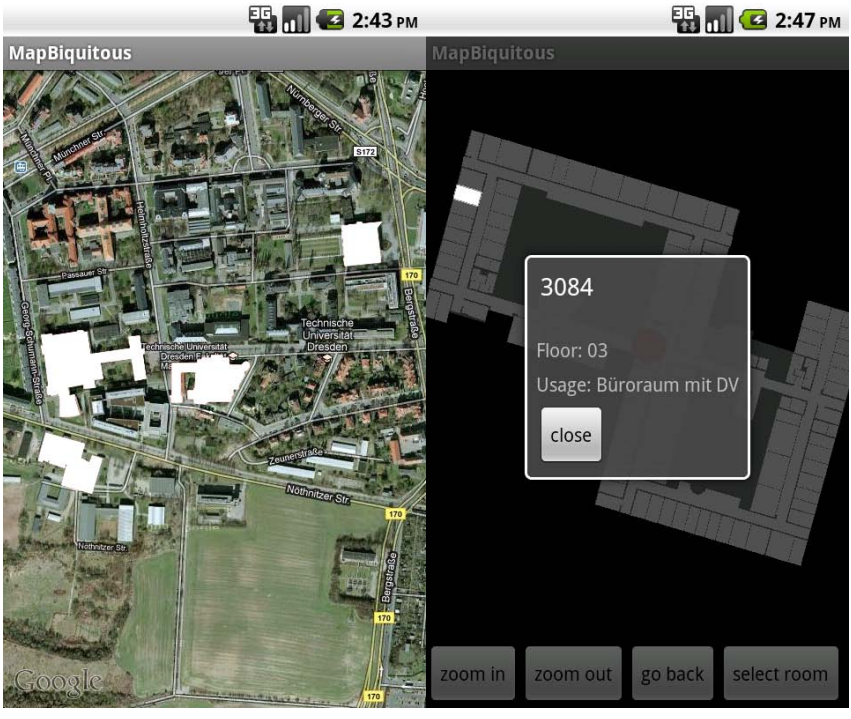

Fig. 5. Android client for MapBiquitous with MapView and overlays (left) and OpenGL view with floor details (right)

we implemented for the interaction with detailed floor plans a separate view based on OpenGL. The view allows the selection of building levels and rooms for visualization. Selected rooms are highlighted. If a room is selected, semantic information for the room is presented (see right part of figure 5).

The *Loader* of the mobile client has been optimized for the access of building data using wireless connections. Instead of loading the complete building geometry data at once, it is loaded stepwise and only if necessary for visualization. After discovering a new building server, the layer containing the building outline is loaded and processed. Only if the zoom level is high and the building is still visible floor plans are loaded. This is done dynamically based on the calculated current position. If the user is outdoors, the ground floor is presented. If the user is indoors, the current floor is determined as part of the user's location and only that floor is loaded. Anyway, to allow continuous positioning, all layers for positioning with the building are loaded at once. The implementation of XML processing is improved as well by using a SAX parser.

The *Locator* also supports WiFi positioning and GPS. For both technologies Android `Services` are implemented which can be shared between different apps. The services periodically calculate the current position of the device and

propagate location updates via intents. These intents can be received by registering `BroadcastReceivers` for these intent types. GPS positioning is implemented using the `LocationManager` API provided by Android. WiFi positioning is implemented using PlaceLab in a version ported to Android. For porting PlaceLab to Android a new Spotter was implemented which accesses the `WiFiManager` API provided by Android.

The intents created by both services are received by the Renderer. The Renderer itself creates intents to trigger the Loader. Similarly, the Loader creates intents received by the Renderer if new layers are loaded, processed and integrated into the building data storage as described in section 4.3.

5.2 Applications Based on MapBiquitous

For testing the MapBiquitous system, we implemented two location-based applications, namely a WiFi analyzer and a location-based game called LocPairs.

WiFi Analyzer. The purpose of the WiFi analyzer is to allow the testing of the WiFi infrastructure at the campus of TU Dresden. A network administrator should be able to perform measurements at reference points, access former measurements and analyze the infrastructure on the move to detect areas with low communication quality or unavailable access points. As shown in the left part of figure 6, the visualization of the app is based on MapBiquitous.

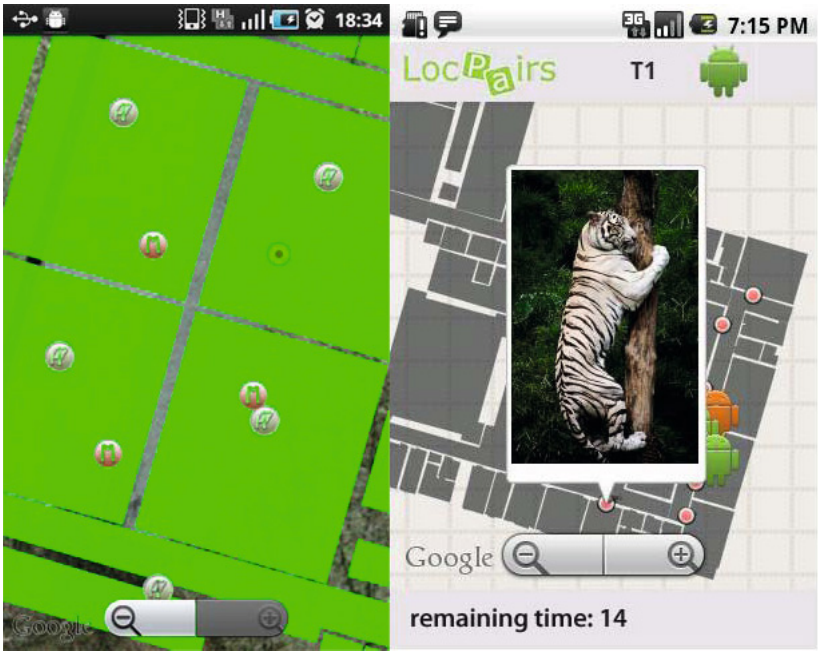

Fig. 6. MapBiquitous applications: WiFi-Analyzer (left) and LocPairs (right)

Because the WiFi infrastructure should be tested WiFi positioning has been switched off. We assume that the administrator can visually detect its location. Outdoor and indoor reference points can be defined directly by pointing on the floor plan or map. Similarly, former measurements are represented by icons directly on the map and floor plan. Details about measurements can be obtained by pointing to the assigned icon.

The WiFi-Analyzer is implemented by adopting the Android client of MapBiquitous. The Renderer has been extended with new functionality for visualizing the reference points and former measurements as well as for the interaction with the MapView to perform measurements at newly defined reference points. The Loader, Locator and the MapBiquitous server-side was adopted without any changes. For persistent storage of measurement data on a server, a separate web service was implemented.

LocPairs. LocPairs is a location-based game adopting the principle of the well-known pairs game. The playing field for LocPairs is an area inside a building. Pictures are represented by attaching 2D barcodes to doors of a floor. Scanning a barcode means uncovering a picture.

The game is played by two teams with two players each. Alternately, each team is in turn to find a pair of pictures. Players have to move to the right door within a limited time. If both players have reached the doors, they can scan the barcode and uncover the pictures. If they found a pair the can continue, otherwise it is the other teams turn.

The purpose of the game is to simplify the maintenance of a fingerprint database. Each time a barcode is scanned the fingerprint of that location is measured. Since the location of barcodes is known, the fingerprints can be assigned with that location and sent to the fingerprint database.

LocPairs was implemented based on the Mobilis plattform [12], a service-based platform for implementing mobile social applications. The Android client of MapBiquitous was integrated with the client-side services of Mobilis. To create the application, the Renderer was extended to create the map-based view for the game. The other views were added using separate activities. The MapBiquitous server-side was adopted as is and provided in parallel to the Mobilis server.

5.3 Discussion

The experiences and practical tests carried out with the implementation of the MapBiquitous system have demonstrated the feasibility of major design decisions. The decentralized architecture of building servers enables a natural balancing of the load, high availability of building data and robustness of the system (A11, A12). Furthermore, it allows building owners to decide if and which building data is published and which technologies are used to represent and process building data. The only technological requirement is the usage of WFS and GML. Anyway, by extending the Loader to support different modules for different formats and protocols even this requirement could be relaxed.

The decision to follow a fat client approach requires higher effort for creating clients. In addition processing of building data is performed mainly on the client side. Using WFS and GML as standard technologies requires XML processing which is challenging for mobile clients. In our first implementation of the Android client on a Nexus One Android device it took about 3 seconds to load and process the building outline and 40 seconds for the current floor plan. Loading and processing has been done in a sequential way. After the creation of parallel threads for loading and optimization of XML-processing using a SAX parser, times could be reduced to 1,5 seconds to load and process the building outline and about 8 seconds for the current floor plan. To further reduce waiting times for the user we implemented caching and prefetching of building data. In combination with the three introduced levels of details for presenting building data loading and processing is now transparent to the user. Advantages of the fat client approach are the higher flexibility for processing building data offered as vector data instead of raster images. In addition, positioning has to be performed on the client side anyway. With a fat client positioning and visualization can be implemented in a flexible way.

The consequent adoption of WGS84 simplifies the integration of building and map data because transformation of coordinates between different reference systems is avoided (A5).

The effort for creating and maintaining building models and information for positioning is high. For modern buildings digital building data might already exist. For public buildings data often exists for building management which is often already constantly updated as it is the case for our university campus. Anyway, updates usually have to be performed manually. Even worse, for WiFi infrastructures access point locations or fingerprinting data might not be available. Crowd-sourcing approaches as presented with OpenRoomMap [11] and the LocPairs application could help but need deeper exploration (A7, A10).

As a result of our experiments with WiFi positioning, accuracy of lateration based on known locations of access points is often to low for indoor applications. Caused be influences of device type, device orientation, access point constellation and moving persons in practice, WiFi based positioning is far away from accuracy in the range of meters. Approaches based on WiFi need further improvements and should be combined with alternative approaches like 2D barcodes, inertial positioning, Bluetooth or NFC.

The publication of building data causes security risks. For instance the knowledge about the usage type of rooms could motivate burglars. Furthermore, the availability of precise positioning technologies and indoor information influences privacy. Such information should not be published without the knowledge and agreement of the involved people. To avoid the publication of security sensitive building data, access control could be established for a subset of that data. By offering only a subset of data to the public such risks can be avoided.

6 Conclusion

In this paper a novel approach is presented to integrate technologies for indoor and outdoor location-based services in a seamless manner. We introduced a decentralized infrastructure of building servers providing explicitly modeled data about the building geometry, positioning and navigation. Building data is offered by open standards, namely WFS and GML to achieve high interoperability of the system. At client side building data is combined with map data for visualizing indoor and outdoor locations in an integrated manner. In addition, information for positioning is exploited at client side for indoor positioning with different technologies. The evaluation has shown that major design decisions are feasible and location-based services can be created adopting the MapBiquitous system. In summary, the presented work is a first step towards the envisioned goal.

Future work will address the named challenges to improve the MapBiquitous approach. Working towards an application for navigating within the campus of TU Dresden, data of all buildings has to be integrated into MapBiquitous WFS servers. Moreover, the issue of providing accurate indoor positioning has to be solved by following a WiFi fingerprinting approach. The work for navigation across buildings and outdoor is currently in an experimental state and has to be completed. With the availability of the campus navigation application we plan to carry out a field trial with a larger user group. Further research goals are the exploration of crowd-sourcing approaches as presented with the LocPairs application to decrease the effort for maintaining building, positioning and navigation data and the combination of indoor positioning approaches for device independent usage.

Acknowledgment. The authors would like to thank the many contributors of MapBiquitous. Jan Scholze developed the concepts and prototype of the initial system and desktop client during his student and master thesis. The students of the practical course on development of mobile and distributed systems implemented and evaluated the current Android prototype and the two introduced location-based services.

References

1. Baus, J., Krüger, A., Wahlster, W.: A resource-adaptive mobile navigation system. In: Proceedings of the 7th International Conference on Intelligent User Interfaces, IUI 2002, pp. 15–22. ACM, New York (2002)
2. Becker, C., Dürr, F.: On location models for ubiquitous computing. Personal Ubiquitous Comput. 9, 20–31 (2005)
3. Ching, W., Teh, R.J., Li, B., Rizos, C.: Uniwide wifi based positioning system. In: IEEE International Symposium on Technology and Society (ISTAS), pp. 180–189 (June 2010)
4. de la Beaujardiere, J.: penGIS web map server implementation specification, Open Geospatial Consortium Inc., Tech. Rep. OGC® 06-042, Version 1.3.0 (2006)

5. Dedes, G., Dempster, A.G.: Indoor gps positioning - challenges and opportunities. In: IEEE 62nd Vehicular Technology Conference, VTC 2005 Fall, vol. 1, pp. 412–415 (September 2005)
6. I. E. Environmental Systems Research Institute, Esri shapefile technical description, an esri white paper july 1998, Environmental Systems Research Institute, Inc. (ESRI), Tech. Rep. (1998)
7. Leppräkoski, H., Tikkinen, S., Perttula, A., Takala, J.: Comparison of indoor positioning algorithms using wlan fingerprints. In: Proceedings of European Navigation Conference Global Navigation Satellite Systems (ENC-GNSS 2009) (2009)
8. Mabrouk, M.: OpenGIS location services (openls): Core services, Open Geospatial Consortium Inc., Tech. Rep. OGC 07-074, version 1.2 (2008)
9. McCarthy, M.R., Muller, H.L.: RF free ultrasonic positioning. In: IEEE International Symposium on Wearable Computers, p. 79 (2003)
10. Portele, C.: OpenGIS geography markup language (gml) encoding standard, Open Geospatial Consortium Inc., Tech. Rep. OGC 07-036, Version 3.2.1 (2007)
11. Rice, A., Woodman, O.: Crowd-sourcing world models with openroommap. In: 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), March 29-April 2, pp. 764–767 (2010)
12. Schuster, D., Springer, T., Schill, A.: Service-based development of mobile real-time collaboration applications for social networks, Mannheim, Germany (2010)
13. Serra, A., Carboni, D., Marotto, V.: Indoor pedestrian navigation system using a modern smartphone. In: Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services, Ser. MobileHCI 2010, pp. 397–398. ACM, New York (2010)
14. Steinhoff, U., Schiele, B.: Dead reckoning from the pocket - an experimental study. In: 2010 IEEE International Conference on Pervasive Computing and Communications (PerCom), March 29-April 2, pp. 162–170 (2010)
15. Vretanos, P.P.A.: OpenGIS web feature service 2.0 interface standard, Open Geospatial Consortium Inc., Tech. Rep. OGC 09-025r1 and ISO/DIS 19142, Version 2.0.0 (2010)
16. Want, R., Hopper, A., Falcão, V., Gibbons, J.: The active badge location system. ACM Trans. Inf. Syst. 10, 91–102 (1992)
17. Ward, A., Jones, A., Hopper, A.: A new location technique for the active office. IEEE Personal Communications 4(5), 42–47 (1997)
18. Woodman, O., Harle, R.: Pedestrian localisation for indoor environments. In: Proceedings of the 10th International Conference on Ubiquitous Computing, Ser. UbiComp 2008, pp. 114–123. ACM, New York (2008)
19. Woodman, O., Harle, R.: RF-Based Initialisation for Inertial Pedestrian Tracking. In: Tokuda, H., Beigl, M., Friday, A., Brush, A.J.B., Tobe, Y. (eds.) Pervasive 2009. LNCS, vol. 5538, pp. 238–255. Springer, Heidelberg (2009)