DESIGN AND IMPLEMENTATION ISSUES FOR EXPLORATIVE LOCATION-BASED APPLICATIONS: THE NEXUSRALLYE

Daniela Nicklas, Nicola Hönle, Michael Moltenbrey, Bernhard Mitschang
Universität Stuttgart, Institute of Parallel and Distributed System, Universitätsstr. 38, 70569 Stuttgart, <firstname.lastname>@hermes.informatik.uni-stuttgart.de

Abstract: Explorative Location-based Applications (eLBA), define a new class of applications that rely on both positioning (i.e. location information) and georeferenced information in addition to a flexible and efficient system infrastructure that supports a mobile and ubiquitous usage.

In this paper we define first a modeling framework to design eLBAs that builds on the concept of tasks as a very valuable system/user interaction and application structuring concept. In addition, we report on a system framework, the Nexus platform, that efficiently provides access to georeferenced information and positioning information. Our sample application, the NexusRallye, is used to exemplify important aspects of our solution platform and to show its advantages as compared to other approaches.

Key words: location-based services, context-awareness, mobile applications

1. INTRODUCTION

How can georeferenced information help people to find their way in an unknown environment? If you have a new job or are enrolled to an university, you have to learn about the locations and the facilities on the campus. Navigation applications only help on short notice: you find a specific way, but you do not get extensive spatial knowledge about the surroundings that you urgently need to act independently.

Normally, you learn this by try and error. Some organisations offer introductional guided tours for freshmen. But the information you need is often already there: on web pages, in databases, maybe already georeferenced for other applications (e.g. navigation). With this, location-based services in general become feasible. But sustainable learning can be achieved much easier by action and not just by information presented. Hence, there is a need for applications that provide georeferenced information and support the learning process by an explorative user-interaction paradigm: We call these *explorative location-based applications* (eLBA).

What makes an application explorative? Surely, a user can use any information system in an explorative way. An explorative application demands this from the user. But then, why should someone want to use such a demanding application? There has to be a proper motivation: first, the user must have the time for exploring. Secondly, there must be a visible advantage for the user (e.g. usable knowledge). And thirdly, a reward for using the application should be given.

An easy way to get people to do or learn something is to let them have fun. Hence, an important subclass of eLBAs are games, especially location-based quiz games, also called *rallies* (see Figure 1). Here, one or more users (called players) take a certain amount of time (e.g. “two hours on orientation day”) to solve a number of tasks, that are related to the location they are playing in. The rally could be cooperative (the players work as team) or competitive (they try to be better and faster than the others). Normally, there is a jury (humans or software) that assesses the performance of the players and offers some kind of rewards (gifts, points, etc...).

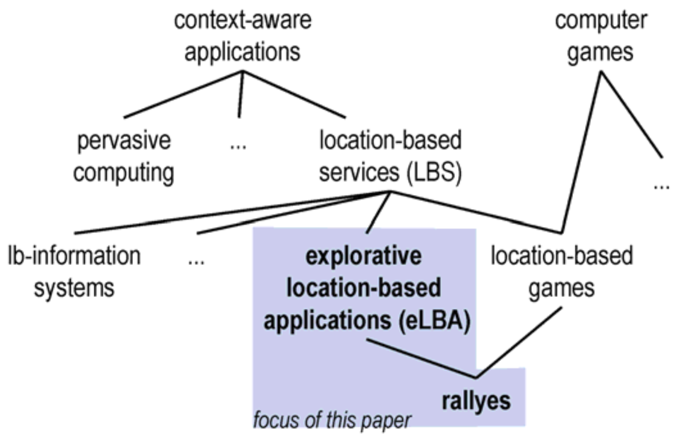

Figure 1. Related Application Classes

Our sample application, the NexusRallye, is targeted at freshmen who want to learn more about the university campus. This game is an electronic variant of a well established paper-based version which has been successfully used for several years, but it offers new possibilities like finding moving objects or dynamic changeable tasks.

We think that eLBAs, especially the subclass of computer games, are an interesting and inspiring application domain for context-aware systems research. It is even likely that the “killer application” everybody in this research area is looking for will be from the entertainment sector.

In Section 2, we relate our paper to other work. Section 3 describes the task-oriented approach and the requirements for our sample application. The tasks, positions of players and other context information are held in a spatial world model managed by the Nexus platform, which is described in Section 4. The functions and the design of NexusRallye itself is proposed in Section 5, before we conclude the paper in Section 6.

2. RELATED WORK

In Figure 1, we show how the application class of eLBAs relates to other application classes. Context-aware applications adapt their behaviour to the user’s context, i.e. her environment or situation (Dey and Abowd, 1999). The field of pervasive computing strives for disappearing or ubiquitous technologies, which requires context-information (Weiser, 1991). Location-based services (Voisard and Schiller, 2004) are applications that focus on the location context. There are already commercial products in this class, e.g. georeferenced mobile phone applications (location-based information system), which are often based on commercial platforms for location-based systems (Webraska, IntelliWhere, Openwave, and Esri, to name a few). As already mentioned, the focus of this paper are eLBAs, and, more specific, rallies, which fell also into the application class of computer games.

There are several research projects that developed context-aware games (Björk et al., 2001, 2002; Piekarski and Thomas, 2002; Starner et al., 2000). Most of them are focused on special aspects like user interaction, that can be shown in a game scenario.

The first commercial location-aware game was launched in 2004 by SingTel (2004) in Singapore. *Gunslingers* is a multi-player network game where players move around, and eventually track and engage other players (“enemies”) within their physical vicinity. They can earn virtual money and real prizes (like cell phones) by “shooting” other players or even take assignments for extra bonuses. However, this game shows only a small range of the opportunities context-aware gaming can offer.

3. TASK-ORIENTED APPROACH

To implement a rally we chose a task-oriented approach as user-interaction paradigm. Tasks ask the user to do something, e.g. answer a question or go to some location. In the following section, we classify different categories of tasks, and depict the advantages a task-oriented approach offers.

After this, we detail on how we can build the NexusRallye from tasks and what requirements to a system framework result from this.

3.1 Task Classification

Tasks can be put into two categories: *question tasks* and *action tasks* (Figure 2). The first category contains tasks like *multiple choice*, *value* and *free text*. Here, the participant’s answer typically depends on his presence, i.e. the participant has to be present at a certain place or at least has to have it visited before. A possible question might be: “How many cashiers are in the cafeteria?”. Additionally, tasks that belong to this category may also be independent of any context information and the participant’s location (e.g. “What is the birthday of Konrad Zuse?”). Value tasks require a simple textual reply (like a name) or a number (like a certain room number or telephone number). Free text tasks offer the participant to give a creative answer (e.g. “Write a short poem”).

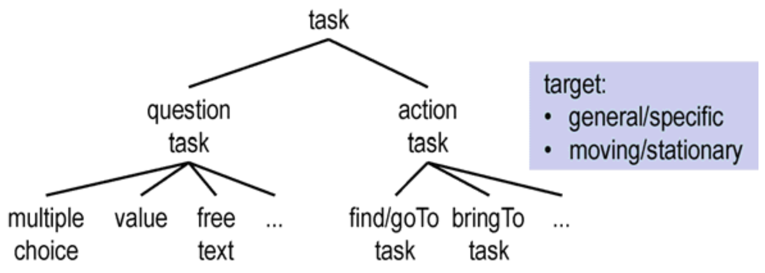

Figure 2. Task Classification

The second category, action tasks, comprises tasks where the player’s position is used to detect when the task is solved. This kind of tasks is only hard to use in a paper-based version of a game, since they require knowledge about the participant's current position, his environment and sometimes the position of other players. Shortly spoken, they require context information and location awareness.

First, find-tasks with stationary targets like "find a copy machine on the campus!" require context information about the environment where the participant resides. Secondly, find-tasks with mobile targets like "Find another player!" require accurate information about the participant's own position and those of the other players, since the target is mobile. Further types of tasks are well conceivable, e.g. bringTo-tasks, where certain items have to be carried to certain places. In addition you can distinguish action tasks with specified targets and with general targets. A specified target is unique, like a certain person or building, while a general target is one out of a set of equal classified targets, like copy machines. So the required context information has to support a classification mechanism.

The action tasks can be used to guide the participant to certain places where he then discovers new tasks to solve.

We want our electronic rally to provide several advantages over a paper-based-version of similar games. First, it allows *dependencies* among tasks, i.e. some tasks only get visible after a participant solved requisite ones. This helps guiding the participant and prevents cheating. Dependencies in paper-based games are hard to implement if possible at all, since all tasks are shown on the sheet from the beginning.

Secondly, not only several players and groups of players, respectively, can play the same rally simultaneously, but also a rally designer can create tasks which consider several rally participants, e.g. "Find another player!".

Thirdly, a computer-based game allows automatic scoring. All tasks except free text tasks are scored automatically during the rally's runtime. This allows the participant to get immediate feedback on his performance, allows tasks to depend on the correct solution of previous tasks and allows competition between participants and reduces the final evaluation work, that has to be done by a jury.

Fourthly, it allows the creation of so-called *Easter eggs*. An Easter egg is any task that is not explicitly referenced by other tasks. A participant discovers an Easter egg by approaching the position where it has been virtually hidden. This might happen by chance or by following a hint from another task that was previously found. Additionally, whole hidden rallies (like hidden bonus levels in classical computer games) can be implemented, where the participant finds several new tasks which may result in bonus points.

The foundation of every rally is a pool of tasks. Everyone may provide new tasks of varying subjects and make the pool steadily growing. A rally designer picks a bunch of tasks from the pool and assembles them to a new rally, so existing tasks can be reused. In the sample scenario, student associations of different faculties could share a pool of general common tasks and add specialized ones from their subject.

3.2 NexusRallye

In order to understand the requirements to a system framework that supports eLBAs, we want to describe the NexusRallye as a sample eLBA.

While creating a certain rally for the NexusRallye system, a rally designer has to tie this new rally to a physical location defining the extent of the rally. Also the tasks he picks from the pool or those that are newly created have to be tied to physical locations which lie inside the rally's extent.

To help the players to find the tasks the designer can use action tasks like "Go to the library!". Tasks without an explicit reference like this are called Easter eggs, as we mentioned above.

Tasks may have predecessors on which they depend, i.e., certain tasks are not allowed to be executed until other tasks are fulfilled. Defining task dependencies by defining predecessors, one will get some informal order of tasks. Since we want the player to show explorative behaviour, the tasks must not be strictly ordered like, e.g., tasks in workflow systems.

Now, our rally is ready for play. What's the player's view? Because rallies are tied to locations, a participant may ask for a list with all close-by rallies available at his location. He starts one of them by registering for it. Then, the participant receives the first tasks for this rally. These tasks can serve as an introduction on how the rally works. Then, he has to walk around to find new tasks at certain places where the rally designer has hidden them. Tasks may guide the participant to places where other tasks are hidden, thus exploring the area that has been determinated by the task designer.

3.3 Architectural Requirements

In our vision of a rally-like eLBA we see the need for some sort of context management. This can be done in three different ways: local (i.e. on the client), server-side (i.e. accessible on a server, but specific for a single application) or shared (i.e. accessible by many different applications).

Local context management is certainly not feasible for the NexusRallye since there are possibly many players using it at the same time that need a common view on their context.

Server-side context management could be an option: a NexusRallye server could hold all context information that is needed by the application. But modelling context is expensive. As stated in the introduction, great amount of the information would be already there on databases or web servers. Only the tasks themselves are specific to the rally applications.

Hence, we consider the usage of a shared context management for the best solution. This can be done by a context management platform that has to fulfil the following requirements:

- Data servers of the platform store environment models and context information, e.g. plans of and information about buildings, as well as rallies and tasks, which can be linked to locations. Also they store information about mobile objects and their position (like the rally participants).
- Applications are able to query this stored information in a pull mode (direct queries) as well as in a push mode (defining events).
- Furthermore, geographic communication (sending a message in a certain geographic area) will be useful in a rally game for group collaboration or for spreading new tasks spontaneously.

For our NexusRallye we want the players to use small portable devices like PDAs with wireless LAN interfaces to get new tasks and submit their solutions. The devices have to locate themselves, e.g. using GPS or infrared emitters. Consequently, putting the content management onto the client would result into client overload.

4. THE NEXUS PLATFORM

As will be shown in Section 5, the Nexus platform represents a very valuable basis to host eLBAs. Before doing this, we will give in this section a short overview of the mission and the architecture of the Nexus platform as shown in Figure 4. For more details see Nicklas et al. (2001).

4.1 The Mission

The goal of the Nexus platform is to support all kinds of context-aware applications with a shared, global world model (Figure 3). To achieve this, the platform federates local models from so-called context servers. The local models contain different types of context information: representations of real world objects like streets, rooms or persons and virtual objects that link to digital information spaces. Sensors keep local models up to date (e.g. the position of a person).

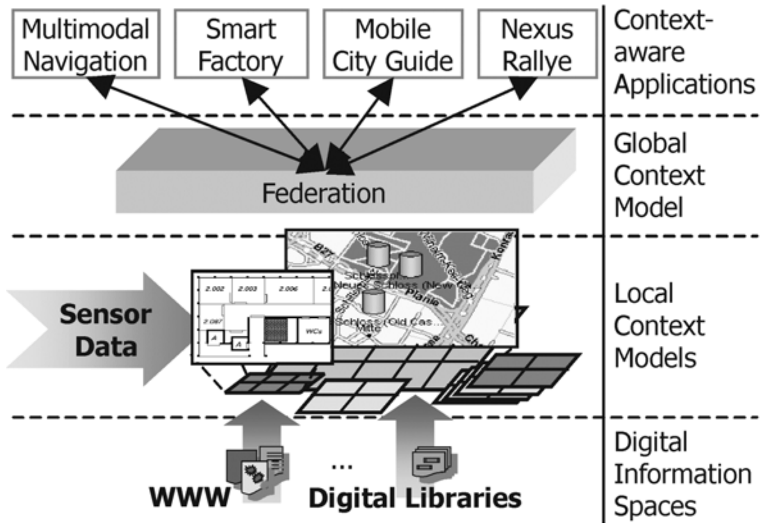

Figure 3. Vision of the Nexus Platform

4.2 Context Server

A context server stores a local context model. It is comparable to a web server in the WWW. To be part of the Nexus platform, the server has to fulfil two requirements: it has to implement a certain interface (simple spatial queries and results in a specified XML language) and it is registered with its service area and object types to the Area Service Register (comparable to a spatially enhanced DNS).

There can be many different implementations of a context server. For providing large scale geographical models, we used a spatially enhanced database. We cope with the high update rates of the positions of mobile users using a distributed main memory system (Leonhardi and Rothermel, 2002). For a Aware Home we adopted a lightweight spatial server as Context Server (Lehmann et al., 2004). Even small scaled sensor platforms like the ContextCube (Bauer et al., 2003) can be used as context server.

4.3 Federation

A federation node mediates between applications and context servers. It has the same interface as a context server, but does not store models (except caching). Instead, it analyses an application query, determines the context servers that could fulfil the query and distributes the query to these servers. Then it combines the incoming result sets to a consistent view and returns it to the application.

For query distribution and service discovery, a Nexus node uses the Area Service Register (ASR). This service is a directory of the available local context models and stores the address of their context server, their object types and their extent. More details about the federation tier can be found in Nicklas et al. (2001).

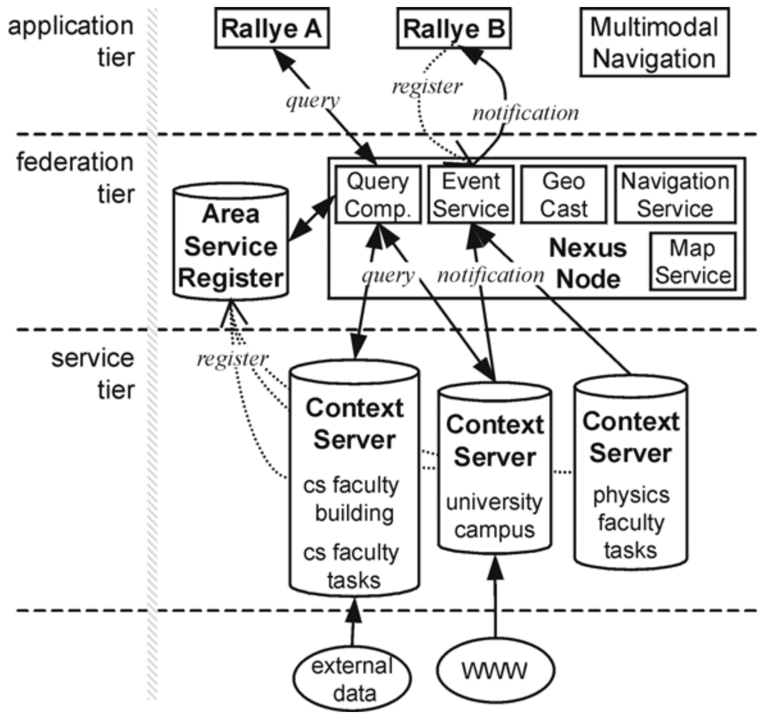

Figure 4. Architecture of the Nexus Platform

4.4 Additional Services

In addition to the query functionality, every Nexus node supports value-added services. They use the federated context model to implement advanced services having their own interface. In Figure 4, you can see three different value-added services of the Nexus platform: The map service renders maps based on a selected area. The navigation service computes multimodal navigation routes across the borders of local context models. With GeoCast, you can address a message to a geographic addressed area, to send it to every mobile application that is currently in this region.

4.5 Information Access

A context-aware application can use the Nexus platform in three different ways.

Pull Mode. An application can send queries to the federation to get all kind of information about its surrounding including infrastructure, points of interest (POIs), mobile objects (friend finder) an so on.

Push Mode. An application can register to the Event Service (Bauer and Rothermel, 2004) to get a notification when a certain state of the world occurs, e.g. the user has entered a building, the temperature in a room exceeds a certain value, or two persons meet.

Value Added Services. An application can use value added services like the map service or the navigation service to shift basic functions into the infrastructure.

4.6 Assessment

As stated in Section 3.3, we want to use a context management platform for our rally application. The Nexus platform fulfils our requirements of such a platform:

- Context servers are able to store environment models and context information, e.g. rallies and tasks, as well as information about mobile objects and their position.
- Applications are able to query this information in pull mode as well as in push mode, using the Event Service.
- Geographic communication (GeoCast) and other useful value-added services are supported.

5. DESIGN AND IMPLEMENTATION OF THE NEXUSRALLYE

In this Section, we describe the data model and the implementation of the NexusRallye. Since we used the Nexus platform as a basis for our application, we were disburdened from the details of context management. Instead, we define the data model of the application with respect to the Nexus data model and just have to specify the new data objects like tasks and rallies.

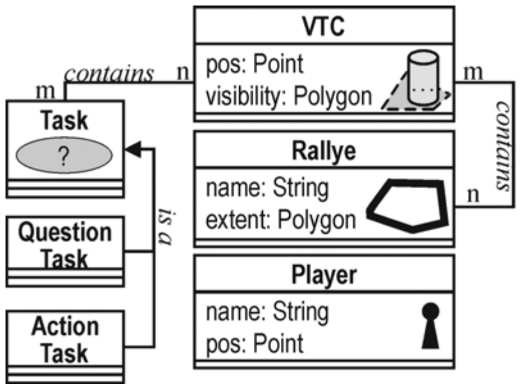

Figure 5. Data Model

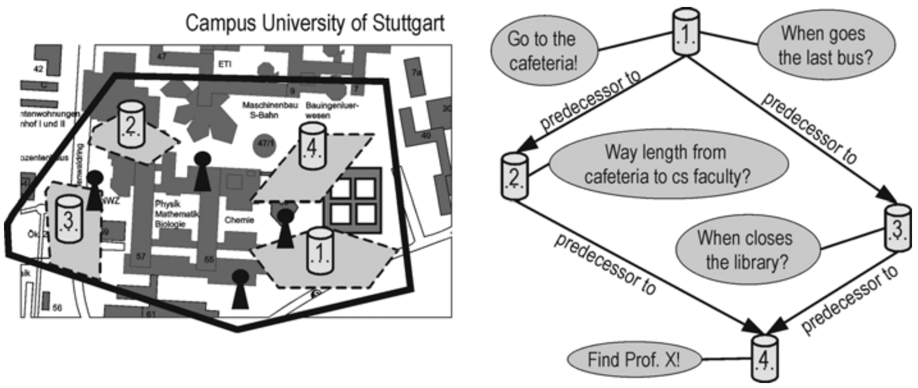

Figure 6. Sample Rally Scenario

5.1 Data Model

The NexusRallye needs context information. We refer to that information as context model. As seen in Figure 5, the NexusRallye's context model contains at least five different object types: *Question Tasks* and *Action Tasks*, *Virtual Task Containers* (VTC), the *Rallye* itself and the *Players*. Depending on the actions tasks, additional objects types are needed to model the real world, like special buildings, copy machines, or whatever must be found or reached in an action task. A sample rally scenario can be seen in Figure 6.

- The *Task* objects (*Question Tasks* and *Action Tasks*, respectively) do not have a spatial attribute. They contain the question or action description and the right answer (if applicable) for automatic scoring.
- *Virtual Task Containers* (VTCs) arrange tasks to groups and virtually link them to a physical location (similar to a Virtual Information Tower, see Leonhardi et al. (1999)). Every VTC is visible in a certain area in the real world. If a participant enters such an area, the VTC gets visible and the associated tasks are loaded to the client application. They are now ready to be solved.

  A VTC may have predecessors on which it depends, i.e. this VTC only gets visible when all its predecessors have been solved.

  Note that one or more tasks can be tied to one VTC as well as one task can be tied to one or more VTCs, the latter facilitates the reuse of tasks in different rallies.
- A *Rallye* object contains an arbitrary number of VTCs. It has a name and description and an area describing its extent in addition to references to these VTCs. The dependencies between VTCs are modeled in the rally and not in the VTCs itself. Therefore, VTCs may belong to several rallies. This reduces the overhead for designing new rallies, since VTCs can be reused.
- To support action tasks, the *Players* and their positions have to be part of the context-model to determine whether a player has reached a certain location or met another player.

5.2 Implementation

We implemented the NexusRallye as client application for the Nexus platform, which is described in Section 4. The Nexus platform can easily cope with the NexusRallye's data model. As mentioned in Section 4.5, the client has two ways for accessing the data: First, it can query the model with a simple, spatial query language and get the resulting objects in an XML format. Secondly, it can register for spatial events like *OnEnterArea* or *OnMeeting* and gets notified by the platform when this event occurs (e.g. the player enters the visibility area of a VTC or meets another player).

The NexusRallye was developed for an iPAQ using Java Micro Edition Platform. In that environment, we experienced several technical problems: incompatibilities between Java and the hardware platform or a seemingly non-deterministic behaviour of the Virtual Machine. The client runs well on a standard laptop, but this device is not suitable for field trial. But we are optimistic that future improvements of hardware and software development environments for mobile devices will solve these problems.

We did no user studies so far because we did not want to burden students with laptops. In the next section, we focus on the experiences during the design and implementation periods.

5.3 Technical Experiences

The implementation of our sample rally was significantly disburdend by the usage of the Nexus platform. All the management of the spatial context data was handled by the platform. We just had to design the schema extensions for task and VTC objects, store them in the model and register for “OnEnterArea”-events with the visibility areas of the VTCs.

The concept of Virtual Task Containers facilitates reuse. Instead of defining tasks for certain locations, they are just referenced by the VTCs. With this, tasks can be bound to different places and also be used by different rallies.

6. CONCLUSION

In this paper we have argued that eLBAs (explorative Location-based Applications), define a new and interesting class of applications that rely on both positioning information and georeferenced information. In order to support a flexible modelling, we developed a task-based approach that guarantees high reuse as well as system/user interaction. Furthermore, eLBAs run supported by the Nexus platform that provides for an integrated and efficient management of and access to position data and georeferenced data. All the modelling and execution issues are highlighted using our sample eLBA, the NexusRallye.

Preliminary experiences show that the task-oriented approach is very valuable. This is especially the case, when one wants to assemble new explorative applications based on existing and new tasks. In order to better support this, we can imagine that task orchestration could be supported by well-known workflow technology.

The work presented in this paper is just initial work that concentrated on the technical aspect and on looking for a proof-of-concept prototype. In a real usage environment our concept of explorativeness has to be matched with the well-known ones found in the literature.

Hence future work will concentrate on the flexible and efficient management of tasks and their orchestration, and on user studies with suitable devices to proof the acceptability of the task paradigm.

ACKNOWLEDGEMENTS

The Nexus project is funded by the German Research Association (DFG) within the Centre of Excellence (SFB) 627.

REFERENCES

Bauer, M., Becker, C., Hähner, J., and Schiele, G., 2003: ContextCube–providing context information ubiquitously. Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops (ICDCS 2003).

Bauer, M., and Rothermel, K., 2004: How to Observe Real-World Events through a Distributed World Model. Proc. of the 10th Int. Cong. on Parallel and Distributed Systems (ICPADS 2004), Newport Beach, California.

Björk, S., Falk, J., Hansson, R., and Ljungstrand, P., 2001: Pirates! using the physical world as a game board. Conference on Human-Computer Interaction, INTERACT 2001.

Björk, S., Holopainen, J., Ljungstrand, P., and Akesson, K.-P., 2002: Designing ubiquitous computer games - a report from a workshop exploring ubiquitous computing entertainment. Personal and Ubiquitous Computing. 6: 443–458.

Dey, A., and Abowd, G., 1999: Towards a better understanding of context and context-awareness. Georgia Tech GVU Technical Report, GIT-GVU-99-22, 1999.

Esri: Arcloation Solutions. http://www.esri.com

IntelliWhere: Location Server. http://www.intelliwhere.com

Lehmann, O., Bauer, M., Becker, C., and Nicklas, D., 2004: From Home to World - Supporting Context-aware Applications through World Models. Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications, 2004.

Leonhardi, A., Kubach, U., Rothermel, K., and Fritz, A., 1999: Virtual information towers - a metaphor for intuitive, location-aware information access in a mobile environment. Proc. of the Third International Symposium on Wearable Computers (ISCW 99).

Leonhardi, A., and Rothermel, K., 2002: Architecture of a Large scale Location Service. Proc. of the 22nd Int. Conf. on Distributed Computing Systems (ICDCS 2002), Vienna, Austria.

Nicklas, D., Grossmann, M., Schwarz, T., Volz, S., and Mitschang, B., 2001: A model-based, open architecture for mobile, spatially aware applications. Proceedings of the 7th International Symposium on Spatial and Temporal Databases, SSTD 2001.

Openwave: Mobile Services Platform. http://www.openwave.com

Piekarski, W. and Thomas, B., 2002: ARQuake: the outdoor augmented reality gaming system. Communications of the ACM 45(1):36–38.

SingTel., 2004: Gunslingers member page. http://guns.mikoishi.com/gunsSingTel/index.html. Visited April 2004.

Starner, T., Leibe, B., Singletary, B., Lyons, K., Gandy, M., and Pair, J., 2000: Towards augmented reality gaming. Proceedings of IMAGINA 2000 Conference, Monaco.

Voisard, A., and Schiller, J., 2004: Location-Based Services. Morgan Kaufmann, 2004.

Webraska Mobile Technologie: smartzone Geospatial Platform. Product Brochure, http://www.webraska.com/News/files/SZ_Geo_Platform_1203.pdf

Weiser, M, 1991: The Computer for the Twenty-First Century. Scientific American 265, September 1991.