In Proc. of TOOLS Pacific. Australian Computer Society Press, February 2002.

MUSA-Shadow: Concepts, Implementation, and Sample Applications
A Location-Based Service Supporting Multiple Devices

Sebastian Fischmeister, Guido Menkhaus, Wolfgang Pree
{*firstname.lastname*}*@uni-konstanz.de*
Software Research Lab
Constance University
D-78457 Constance, Germany

Abstract

The proliferation of mobile devices has refocused interest on location-information exploiting applications. The most prominent problems are the support of devices with widely varying computing and user interface capabilities and a programming model, which promotes the development of location-based services. The article presents the concepts and implementation of MUSA-Shadow, which addresses both issues through a service infrastructure that provides high-level programming abstraction for the development of location-based applications. Furthermore the system introduces novel concepts for user/service interaction via adaptable event graphs and implicit location information via mobile agents.

Keywords: *mobile computing, multi-platform support, location-awareness, user interfaces, mobile agents, location-based systems*

1. Introduction

The active badge project at the beginning of the 90's at Xeroc Parc marked the starting point of location-aware computing. Technological advances in component miniaturization and the emergence of wireless communication channels have engendered the new paradigm of mobile computing. This paradigm has fundamentally changed the way humans work with computational devices [2]. It enables users to access information and interact with services through portable devices regardless of environment properties such as physical position, logical position, time, etc. The proliferation of mobile devices and the commercial success of the telecommunication industry require the convergence of two research areas: location-awareness and mobile computing. Until now research has concentrated on the one hand on the development of the sensor infrastructure for location-aware computing systems and, on the other hand, on general architectures and toolkits. Several different location-sensing systems have been proposed and evaluated, sensor fusion systems have been discussed in respect of cost, accuracy, precision, and scalability [6, 7]. Also much effort has been put in researching general architectures and toolkits [4, 8]. The emergence of commercial-off-the-shelf (COTS) hardware that can easily be deployed and maintained shifts the focus from the sensor infrastructure and single-applications architectures to service infrastructures. That facilitates the development of applications on a diverse set of sensors and a constantly changing set of devices. The MUSA-Shadow project aims at researching and providing such a service infrastructure.

The concept of location is treated differently throughout the literature. There are basically two different notions in location-based systems: (1) a location represents the exact position of the person in terms of longitude and latitude with minimal error rate and (2) a location is the area covered by one transceiver. The location of the person is represented by the transceiver that currently serves the person's communication device. MUSA-Shadow uses the latter notion.

The article presents the MUSA-Shadow system that concentrates on the service infrastructure for location-aware applications. MUSA-Shadow allows application providers to exploit its service infrastructure to develop location-aware applications. The system presents a solution that tries to solve the following key issues involved in location-aware application development:

1. Multi-platform user interface support provides access to services using different access mechanisms.
2. A programming model for location-aware service platforms that gives specific support to location aware application development by relieving the developers from handling detailed location data.

MUSA-Shadow integrates two orthogonal concepts—MUSA and the Shadow infrastructure—forming a software infrastructure that addresses the two issues mentioned above. MUSA offers high-level programming abstractions that address the problem of multi-platform service access while Shadow helps integrating location-system information and offers a new user-service interaction mechanism.

The remainder of the article is structured as follows: The motivation of the work is presented in Section 2. Section 3 describes the architecture of the service infrastructure of MUSA-Shadow and its design objectives. The MUSA and the Shadow systems are discussed in Section 4 and 5, respectively. Section 6 presents implementation issues of the MUSA-Shadow system. Section 7 provides a short overview of related work and Section 8 concludes the article.

2. Motivation

With the advent of the mobile-computing paradigm it became evident that location information will be especially valuable for users with mobile computing devices. Applications are now built to cover a wide range of devices. However, implementing a specific solution for each platform is not feasible because of the difficulty of synchronizing all solutions on application's lifetime. Current practice is to implement a single solution targeting a specific platform even for applications that can be adequately accessed by various devices with different user interfaces. Systems are demanded that scale up well to environments that include a wide variety of different devices and that can easily and flexibly connect to the application logic. Due to the high diversity of the new class of mobile computing devices and their display limitations, the MUSA project concentrates on multi-platform support. We argue that the introduction of an abstract user interface (UI) language is an essential component that eases the development of UIs for mobile computing devices. Traditional UI languages were most often designed with a specific environment in mind. The design of an UI in a generic language and the explicit mapping to a concrete UI language is a more flexible and more future proof solution. Such flexible UIs will scale up well to environments that include a wide variety of different devices.

Location-based services are targeted at a wide range of consumer devices. As a consequence, architectures and concepts must be introduced that support this diversity and are still suitable for location-based solutions. Although approaches, which centralize location information and location-exploiting applications, seem the most prominent (e.g., the location-based system model used for cellular phones proposed in the GSM standards 101.726, 101.527, 101.529, 144.031 and 148.071), the MUSA-Shadow project uses a scalable distributed approach for location services. MUSA-Shadow uses a novel concept of implicit location information using mobile agents that migrate to the location where the location information originates; in contrast to this approach, the centralized approaches require that explicit requests to acquire location information be issued. Finally, the system minimizes complexity and provides high-level support of the location to the developers. This eases and speeds up development of new service.

3. High-Level Architecture and Design Goals

Figure 1 illustrates the high-level architecture of the MUSA-Shadow system. MUSA provides the infrastructure that mediates between the user and the service logic. Its interaction is described by an event graph. The Shadow system provides the infrastructure that allows applications to be plugged in, which then use location information.

The objective of the MUSA-Shadow system is to simplify the task of creating and maintaining location-aware systems for mobile users. The system relieves the application developer from dealing with the details of location-based information by reallocating location-aware computing to the service infrastructure. The fact that high-level abstractions and commonly used service in location-aware computing are shifted to the service infrastructure of the

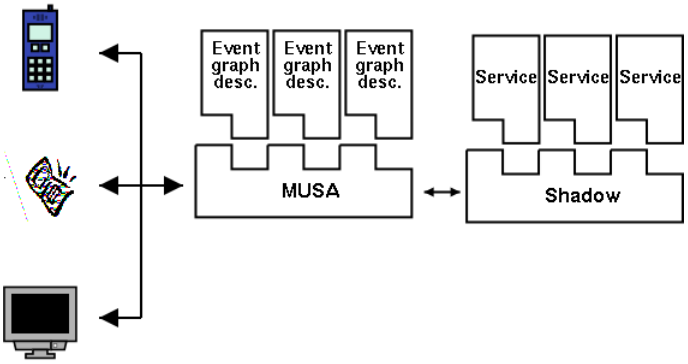


Figure 1. High-level Architecture of MUSA-Shadow System.

MUSA-Shadow system makes it easier and faster to develop location-aware applications on a diverse and constantly chaining set of devices. A number of design goals have been identified to build the MUSA-Shadow infrastructure:

Flexibility. In system architecture two opposing principal concerns are the minimization of collaboration between components and maximization of cohesion within a component. Collaboration must be minimized in order to avoid interference between different concerns. The decomposition of concerns eases the integration and change of new components. At the same time, it is desirable to maximize the cohesion of a system to effectively localize concerns. The MUSA architecture provides separation of concerns at two levels [10].

1. Between the service and the abstract UI. The separation aims at the mutual reduction of modifications in the case of a change of the service or UI.
2. Between the abstract UI and the concrete UI. The goal is to decouple the abstraction from its implementation, so that the two can vary independently. It facilitates the removal, addition and modification of UI concerns.

The separation of concerns directly impacts the flexibility of the service and promotes and eases the modification, extension and maintenance activity [3].

Extensibility. The Shadow infrastructure supports a plug and play mechanism, where service can come and go as they like. This allows to dynamically adapt the number of available applications in one location. Mobile agents intrinsically provide such extensibility mechanisms. Other agents can replace agents or agents can operate in parallel. So, given the case that a new feature must be available, a mobile agent is sent to the location that realizes this feature. In case of the MUSA-Shadow system, the mobile agent would deliver a service plugin and plug it into the running system.

Scalability. The concept of scalability mainly refers to whether the system can, for instance, cope with a high number of requests. Thus, the MUSA-Shadow system must be able to handle a large number of requests (i.e., location changes and inter-agent and agent-human communication). As MUSA-Shadow does not rely on a centralized approach, all locations can be split into smaller locations if necessary, or one location can be computed concurrently at two or more workstations.

4. Multiple User Interfaces — Single Application (MUSA)

The MUSA system allows to describe a user interface in a highly device independent way. The abstract description is transformed to a concrete user interface language via a transformation, which is controlled by style sheets.

4.1. Construction of a UI

A basic concept of user-interface construction is the separation of content and style. Style refers to information that is used to construct the final presentation of the user interface. Content information refers to non-interaction and interaction content.

Style. Style information comes in form of style objects. The designer submits this information separately from the content of a user interface. Style objects carry presentation information, which come in one of the following forms.

1. **Encoding presentation information.** Encoding presentation represents low-level information of how to map content objects to concrete user-interface objects. Concrete user-interface objects determine the type of a content object and its appearance on a concrete user interface. E.g., the style determines whether a content object that allows the navigation from one dialog to another dialog is presented as a link or a button.
2. **Presentation model information.** Presentation model information considers high-level aspects of layout management. It declares the layout of a user interface and determines the position of the encoded content objects in relation to other content objects.

Content. Content objects are defined separately from style object information. The basic distinction is between non-interaction and interaction content objects.

1. **Non-interaction content objects.** Non-interaction content objects are objects that are presented to the user, however, the user cannot interact with these objects. Typically, text is a content object that the user can view, but she cannot interact with. Non-interaction content object have a declarative character.
2. **Interaction content objects.** This category of content objects comprises two subcategories: (1) Interaction content objects that interact with the API of the application. This type of content objects offers the user to submit an event, which is transformed into a call of the API of the application logic. The application processes the event and returns data, which are interpreted and the results are presented to the user. (2) Interaction content objects that do not interact with the API of the application. This type of content objects offer to the user the possibility to submit an event. However, the events do not involve interaction with the application logic. The target of these events is solely the user interface and processed exclusively by it.

4.2. High-level Architecture of MUSA

Figure 2 illustrates the high-level architecture of MUSA. The system is conceptually split into four tiers and employs an event-driven design.

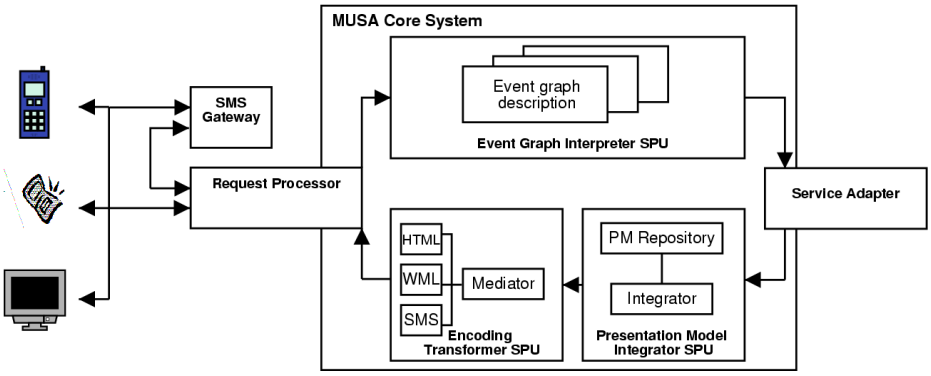


Figure 2. High-level architecture of MUSA system.

Client. The client environment represents the first tier. No service data is installed on the client side and the client communicates via wireline or wireless Internet with the service. Typically the client is some sort of browser, but could also be some device with no visualization capacity like a telephone.

Request Processor and SMS Gateway. The request processor deals with the client's request. The communication between the services and the client UI passes through the request processor. It forwards the communication stream to the MUSA core system. The request processor is the link between the MUSA core system and the client and converts the client requests into events that are used throughout the MUSA architecture.

The short message service (SMS) gateway is a module that is logically located between the client and the request processor. It receives short messages from the client. The short messages have a predefined syntax and semantic and the gateway converts them into events that are forwarded to the request processor. The request processor receives the request and returns the answer to the SMS gateway, which in turn transfers the result to the client.

MUSA Core System. The MUSA core system consists of specialized processing units (SPU), which reflect the separation of content and style.

- **Event graph interpreter SPU.** The event-graph interpreter mediates between the user interaction and the delivery of service data to the user UI through the service adapter. The event-graph interpreter contains the event-graph and handles the event processing. The event graph implements the interaction between user interfaces and service logic.
- **Encoding transformer SPU.** The transformer SPU assigns and maps the events of the event graph to concrete UI elements, which trigger the specified events. The SPU applies a transformation on the event graph dependent on the user's device. Figure 2 shows three transformer (HTML, WML and SMS).
- **Presentation model integrator SPU.** The integrator SPU models the overall presentation layout of the events, which are enabled within the current interaction between the user and the application. The integration of the events into a presentation model (PM) is particularly important for HTML, since the user is used to have nicely designed HTML pages, which is less important for WML pages for example. The UI designer creates the PM and submits them into the PM repository.

Service Adapter and Service Logic. The service logic is the body of code for which the MUSA system provides the multi-platform features. The service logic is accessible via service adapters, which connects the MUSA system to the service. In this case the Shadow system hosts a set of services. The service has no knowledge about the event-based interaction structure or the presentation issues of the service data. It is designed independently from these issues.

4.3. Event Graph

The event graph is the heart of the MUSA core system. The introduction of the event graph follows the idea of the reactive constraint graph in [3] and the abstract depiction hierarchy presented in [11]. The event graph implements an interaction-oriented UI description. The use of general or generic UI elements is replaced in support of the description of the interaction between interface and user by means of the event graph. The event graph is an abstract description of a service, which is presented to the user who interacts with it through a UI. Events in the event graph do not specifically define UI elements. However, events are assigned and eventually mapped to UI Elements that are able to trigger the events in the transformer SPU. The UI elements trigger the event either on display of the UI elements or in response to user interaction

The introduction of the event graph as a high-level abstraction of the service logic/user interaction allows rapid development of UIs for service. This concept eases the implementation and design of services. A set of hierarchical structures that help to divide a service into smaller parts further promote the development (Figure 3).

- **Simple Events.** An event is an abstract interaction object (AIO) that represents an abstraction of a concrete interaction object (CIO) from the presentational and the behavioral point of view.
- **Composite event.** An event is composite if it is composed of other events.
- **Dialoglet.** A dialoglet consists of a number of events, which belong logically to each other.
- **Dialog.** A dialog is designed to represent a task or a sub task of a specific web-based service. A dialog contains one or more dialog profiles. A dialog profile represents the dialog through the filter of a specific device profile. A dialog is composed of an initial window, from which other windows, concrete representations of dialog segments are chained.
- **Service.** A service is composed of a sequence of dialogs.

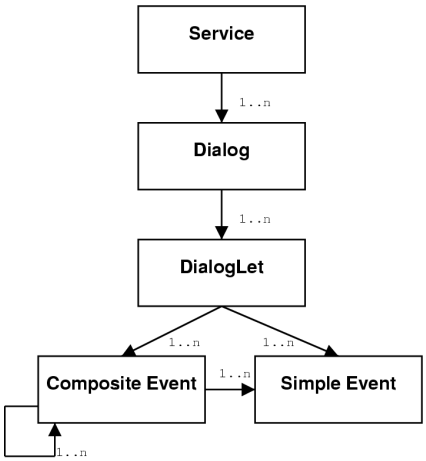


Figure 3. Structure of the event graph.

The objective of the concept of the event graph is to structure the service design by using the abstractions listed above. Each of these play an important role in service UI design in practice. By providing the event graph for describing these abstractions, the vocabulary of the designers informal design practices is matched. This makes it easy for the designer to map its vocabulary to the abstractions, both in terms of formalizing an informal specification and communicating the results to other stakeholders.

5. Shadow

Mobile agent technology is a new approach for designing software in distributed systems. It is part of the mobile code paradigm [5]. The main components of the mobile agent technology are mobile agents and mobile agent systems. The Shadow system uses mobile agents to realize its novel concept for location-based systems.

5.1. Concepts

Figure 4 shows two locations or cells[1], each hosting the communication infrastructure and the MUSA-Shadow system. The Shadow system is built on top of a mobile-agent system. Each human user owns its personal mobile agent, which is the "electronic shadow" of that user in the virtual space. The mobile agent interacts on behalf of its user with applications, local services, and other mobile agents to execute tasks defined by the user. As shown in Figure 4, for instance, user A and B reside physically in the left location and their mobile agents reside logically in the same location as their human counterparts. User D is moving from the left to the right location. As a result of the physical movement, the mobile agent of user D moves with him. User C is currently interacting with his user agent.

The communication between the user and its mobile agent is done via different devices such as a mobile phone or a personal digital assistant (PDA). The different types of requests (i.e., SMS, WAP, HTTP) are converted into a uniform format and the user agent responds to these requests. The answer is then transformed back into the correct format and the user receives the answer to his request.

Shadow introduces a novel concept of implicit location-information usage. Common location-based systems require performing active-location detection, i.e., the system explicitly locates the user and returns the location information. In the Shadow system, the mobile agents reside always at the same location as the human user and use the locally present service. Thus, it is not necessary to explicitly locate the user as the mobile agent automatically uses local services. So the basic difference is that user agents and services move to the location where location information is present and evident in contrast to traditional systems where the location data is transferred to a centralized system and then processed. However, such a system requires location-change monitoring.

[1]In the paper the terms cell and location always refer to the position of the transceiver that server the user's communication device and the coverage area of this transceiver.

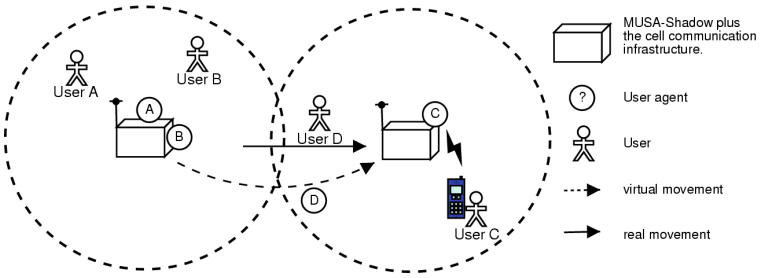


Figure 4. An agent moves with its user.

5.2. High-level Architecture of Shadow

Figure 5 provides an overview of the Shadow system showing the system divided into three layers plus a communication system. As the Shadow system uses implicit location information, it just receives user-location changes from the communication system and acts on such events. The interface between the two systems is wrapped, so the shadow system works, for instance, with 802.11b wireless local-area networks but also with global system for mobile communication (GSM) using the gateway mobile location center (GMLC).

A mobile agent system provides the basic infrastructure, which also includes an inter-agent messaging system (the *infrastructure layer).* The *service agent,* the *service dock,* and the local user agents work on top of this infrastructure layer. The service agent provides additional services on top of the infrastructure layer such as a service directory, an interface for the MUSA system.

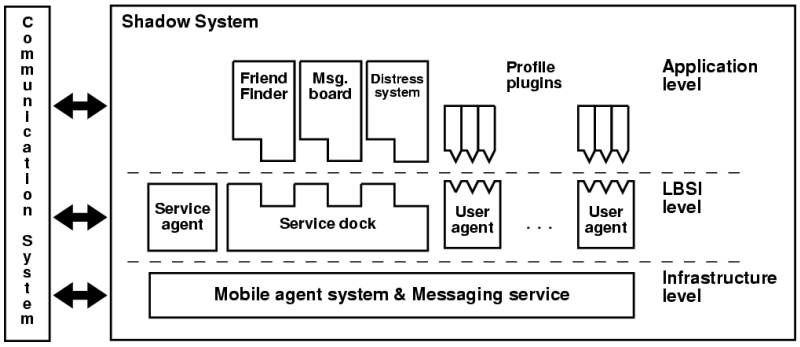


Figure 5. Architectural overview of Shadow.

The *service dock* is the main interface for location-based services. It is one of the two major hotspots [9] of the system. A new type of service can be plugged in anytime (such a plugin is called *service plugin).* Each service plugin has a *profile plugin* that the user agent needs to use a specific service. The user agents carry a number of profile plugins with them to make use of the local services. The plugin concept is the second major hotspot in the system. The service and profile plugins form the application level in the layered architecture. All layers below this application level offer the high-level abstractions necessary to ease developing location-based services.

6. Implementation

As a proof of concept and to show its technical feasibility, this system has been built as a prototype at the University of Constance. The system currently supports three communication types: SMS, WAP, and HTML. This section describes three applications that have been realized using the MUSA-Shadow system.

The programming language of choice for this system was Java, because it supports rapid development and offers fair support for networking. Currently the whole MUSA-Shadow system runs at one workstation that manages seven locations. The basic means for communication include remote method invocation, Aglets inter-agent communication protocol, and extended markup language (XML) messaging within MUSA. For XML parsing

the system uses the JAXP 1.1 package. The SMS gateway, required for short message communication the user and its agent, is a dedicated mobile phone that is connected to a separate workstation via the serial-line interface. The extended AT command set [1] provides basic functionality for the workstation to interact with the mobile phone (e.g., AT+CNMI=1,X,X,X,X,X to be notified of incoming short messages or AT+CMGS to send a short message). The system uses Jakarta Tomcat version 4.0b as WAP server and the MUSA request processor works as servlet on top of this servlet engine. The mobile-agent system of choice is Aglets 1.2. Currently the prototype is integrated into the 802.11b network at the University of Constance. In the 802.11b network the system detects location changes through listening to the inter-access point protocol.

6.1. Control Center

The control center allows monitoring locations, managing available services, executing operations on agents. It also allows modifying parameters in the MUSA part such as modifications of the event graph, changes in the PM repository or the transformer SPU.

Figure 6 shows a screen shot of the control center in action. Currently, the location "cell01" is selected and the table shows all agents that reside at that location. In the location overview a circle represents a user agent and the boxes represent system agents (i.e., agent such as status-reporting agents, the service agent).

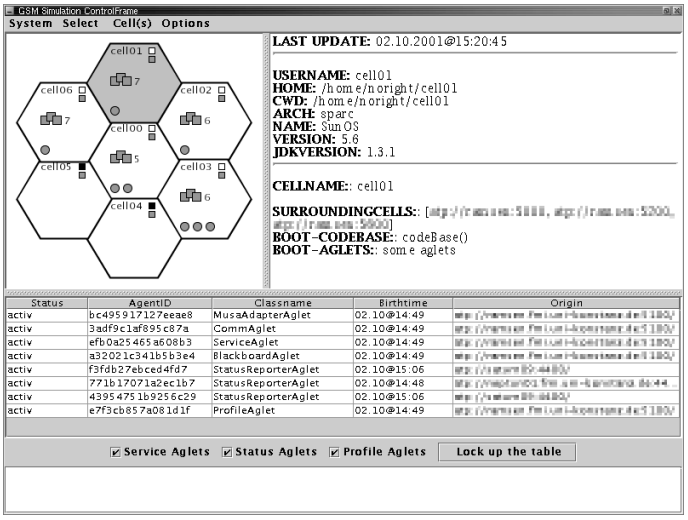
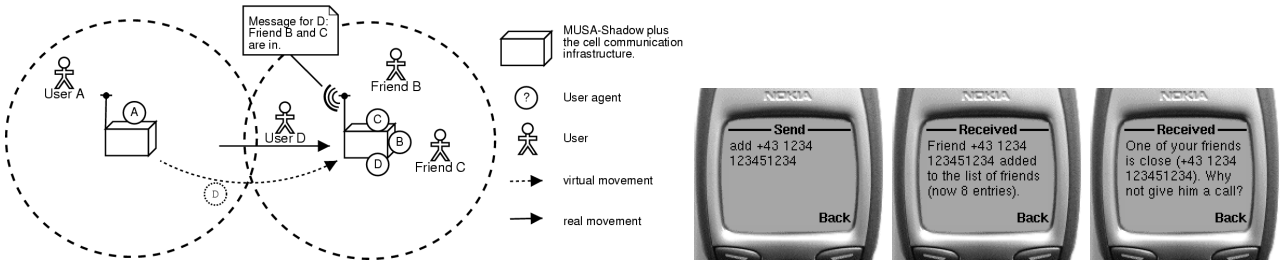


Figure 6. Control center of MUSA-Shadow.

6.2. Implemented Applications

Each of the three realized services emphasizes different strengths of the system. The FriendFinder shows that some applications do not even need a counterpart plugin in the service dock. The distress application makes use of the mobile-agent systems and spreads new agents that carry out their task at adjacent locations. And finally, the message-board application shows that different plugins can work together; in the case of the message board these are the message board itself and the FriendFinder that already contains the list of friends of the user. For the sake of brevity the sample screen shots only show one access method. Each application is accessible via SMS, WAP, and HTML.

FriendFinder. The FriendFinder plugin notifies the user when she migrates to a new location in which known friends of her reside. So, after a user agent arrived at the new location, it scans if friends are at the same location and eventually will notify the user. If the agent finds a friend, it will take whatever action its user has specified (e.g., send both parties a short message). Figure 7 shows the concept of this plugin and some sample screen shots of interaction between the user and her agent via short messages. Various GSM network operators propagate such

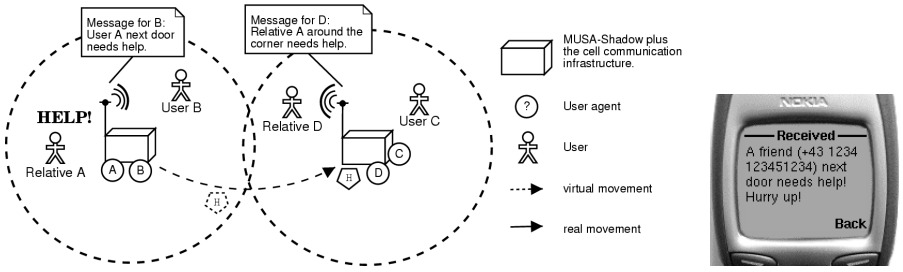


(a) The Model. (b) Add a friend. (c) System reply. (d) SMS notification.

Figure 7. Example of the FriendFinder plugin.

service or a variation of it. For this purpose they use a GMLC and implement services according to the evolving specification of the location inter-operability forum. So although the application idea is not novel, the simplicity of the implementation demonstrates how powerful MUSA-Shadow is. In contrast to using explicit location requests to find out whether or not a friend is in the close vicinity, the user agent simply checks whether a user agent of a friend resides at the same location.

Distress. The distress plugin provides special emergency functionality to its user. For example, a person that needs medical supervision triggers his emergency call in case of an accident (e.g., built in a wrist watch or locations in the house such as bathroom or kitchen). The user agent notifies the ambulance, but also relatives that reside in nearby locations. The user agent generates new mobile agents (helper agents) that move to nearby locations. These helper agents carry the information necessary to identify relatives of the injured person and to provide information about where this person is.



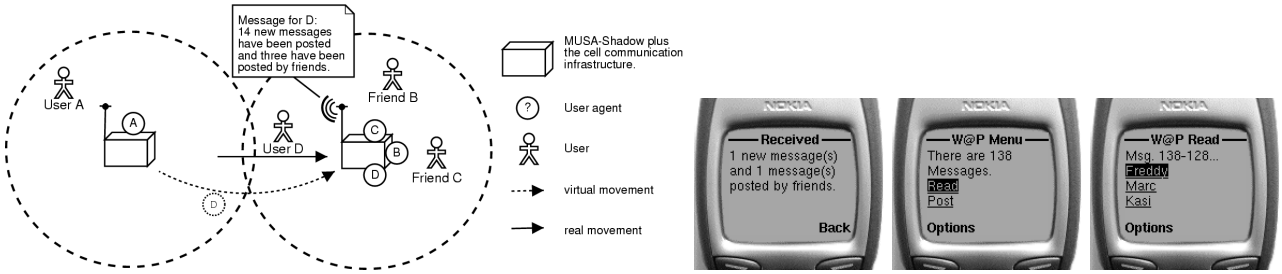
(a) The Model. (b) SMS interface.

Figure 8. Example of a distress call.

The major benefit is that the user agent not only notifies the ambulance but also migrates to nearby locations to look for relatives in the close vicinity, who can help until the ambulance arrives. Figure 8 visualizes a sample scenario of this concept and shows a sample short messages that is sent to the relatives. This concept introduces a complete new information dissemination model; it allows information to disseminate in form of waves throughout the system.

MessageBoard. Message boards are available at specific locations. The basic idea of this service is that a user leaves messages which are related to that specific location such as "this supermarket only sells old stuff" or "this pub rocks!". In addition to this, a user can configure its user agent to notify her in case a new message has been

posted by one of her friends. The user agent will then notify the user whenever she visits that location. Figure 9 shows the sample scenario and screen shots of the WAP interaction. This type of application—huge amounts of data combined with location information—is difficult to realize using a centralized approach. In contrast to a centralized approach the data never leave the location to which is it bound in the Shadow system.



(a) The Model.

(b) Notification via SMS.

(c) WAP start menu.

(d) WAP read interface.

Figure 9. Example of the MessageBoard system.

7. Related Work

This section provides a brief overview about work that also concentrates on defining and implementing an infrastructure for location-based systems.

Marsic presents an architecture using intelligent agents and XML to adapt data shared dynamically between devices with disparate capabilities in a collaborative environment [8]. The Disciple framework uses intelligent agents to implement network awareness. However, the agents are not mobile and do not exploit location information and location-based service. The clients accessing application in the Disiple framework are assumed to be clients with a Java virtual machine, since agents span all architectural layers of the system. Also, the devices participating in the system are powerful with the respect of processing power, memory, etc.

The Guide project [4] provides a tourist guide system for the city of Lancaster, UK. The Guide is built on top of Linux servers hosting 802.11b access points. The application uses a Java-based web browser enriched with communication and location controls. A user can take a tour trough Lancaster and can surf the local web for interesting information about particular locations. The major difference between the MUSA-Shadow project and the Guide project is that MUSA-Shadow supports multiple COTS devices in contrast to the Guide project that uses its single proprietary device. Furthermore the Guide technology concentrates on pull systems as it uses a browser (HTML) and a pull protocol (HTTP).

The European telecommunications standards institute (ETSI) provides standards so that different location-based systems are still compatible with each other; this is useful for international roaming. The complete specification of this serving mobile location center (SMLC) comprises several standards that include GSM 101.726, GSM 101.527, GSM 101.529, GSM 144.031 and GSM 148.071. Then applications use a dedicated interfaces to access this SMLC. Various custom applications already exist in the test phase in different countries (e.g., the FriendFinder of max.mobil in Austria, or the SieApplications developed by Siemens). However, the current implementations use only one GMLC. This fosters centralized solutions and is the major difference to the MUSA-Shadow system; Here the location information is not accessible only through one interface but at each location.

8. Conclusion

The increase of Internet-enabled and mobile devices has lead to a diversification of user interfaces. Besides this, the advent of location-sensing systems requires service-infrastructure support for application developers. The

MUSA-Shadow project aims to avoid the fragmentation of the web into spaces that are solely accessible with specific type of devices while providing an extensible and flexible infrastructure for location-based services.

The initial aims to provide a flexible, extensible, and scalable system have been met. The experiments with the three applications showed that the concepts are sound, the UI abstractions ease development for different devices, and that agent technology is capable to realize scalable and flexible location-based service infrastructures. Furthermore, implemented applications showed that the novel concept of implicit location-information handling provides means to realize existing applications in an efficient way and is capable of realizing a new class of location-based applications: applications that use wave-like information dissemination. The introduction of the event graph to exclusively describe the interaction between the service and the user interface and its separation from any style issues has proven to be a viable concept for future user-interface development.

Future work will focus on refining the concept of the abstraction model and the dynamic adaptation of event graphs as well as on exploring the consequences of implicit location handling and wave-like information dissemination in location-based systems.

References

[1] Technical Specification Group Terminals; AT Command Set for GSM Mobile Equipment. and 3GPP standard, 1998. Version 7.6.0, GSM 07.07.

[2] G. Abowd and E. Mynatt. Charting past, present, and future: Research in ubiquitous computing. *ACM Transaction on Computer-Human Interaction*, 7(1):29–58, 3 2000.

[3] T. Ball, P. Danielson, L. Jagadeesan, R. Jagadeesan, K. Laeufer, P. Mataga, and K. Rehor. Sisl: Several interfaces, single logic. *Internatinal Journal of Speech Technology*, 3:93–108, June 2000.

[4] N. Davies, K. Cheverst, K. Mitchell, and A. Efrat. Using and Determining Location in a Context-Sensitive Tour Guide. *IEEE Computer*, 34(8):35 to 41, Aug. 2001.

[5] A. Fuggetta, G. Picco, and G. Vigna. Understanding Code Mobility. *Transactions on Software Engineering*, 24(5):342 to 361, May 1998.

[6] J. Hightower and G. Borriello. Location Systems for Ubiquitous Computing. *IEEE Computer*, 34(8):50 to 56, Aug. 2001.

[7] J. Hong and Landay J. An infrastructure approach to context-aware computing. *Human-Computer Interaction*, 16 (to appear), 2001.

[8] Ivan Marsic. Adaptive collaboration for wired and wireless platform. *IEEE Internet Computing*, 53(4):26–35, 2001.

[9] W. Pree. *Building Application Frameworks, Object Oriented Foundations of Framework Design*, chapter Hot-Spot Driven Development, page 379 to 394. Wiley Computer Publishing, 1999. M. Fayad and D. Schmidt and R. Johnson (editors).

[10] C. Szyperski. *Component Software - Beyond Object-Oriented Programming*. Addison Wesley, Reading, Mass., 1998.

[11] Richard Taylor, Kari A. Nies, Gregory A. Bolcer, Craig A. Macfarlane, and Kenneth M. Anderson. Chiron-1: A software architecture for user interface development, maintenance, and run-time support. *ACM Transaction on Computer-Human Interaction*, 2(2):105–144, June 1995.