JKU Learning Networks

eLearning – Offensive der Johannes Kepler Universität Linz

April 2004 - September 2005

Alois Ferscha, Clemens Holzmann, Stefan Oppl

Institut für Pervasive Computing

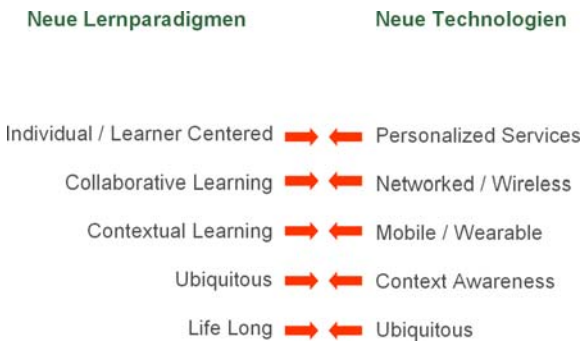
Johannes Kepler Universität Linz

Univ.-Prof. Dr. Alois Ferscha

Institut für Pervasive Computing
Johannes Kepler Universität Linz
A-4040 Linz, Altenberger Straße 69
www.pervasive.jku.at

Introduction

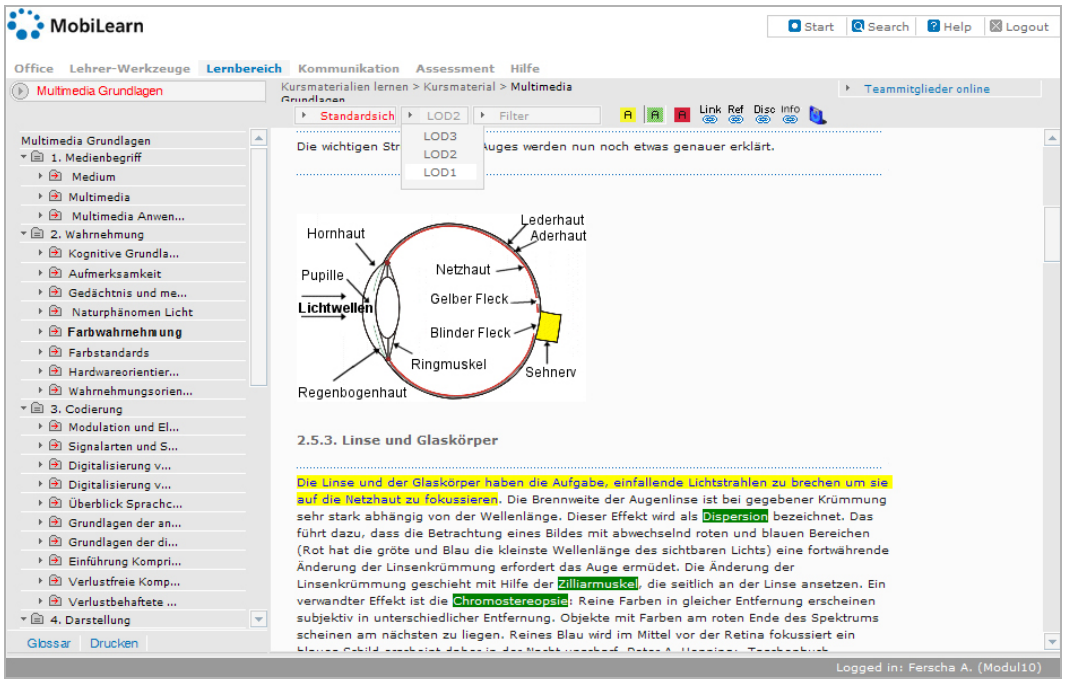
E-Learning is driven by technological developments in the areas of content creation, content delivery and content access. Besides technological possibilities like global networks, synchronous and asynchronous means of communication, wireless networks and mobile devices, there is also a paradigm change with regard to learning and teaching. Examples are individualized learning, collaborative learning and life-long learning. The project “JKU Learning Networks” aims at meeting the challenge which results from these two developments, as the following Figure illustrates.



Aim of the project is the investigation and realization of new teaching- and learning-paradigms (e.g. collaborative learning and individualized learning) by using the technological opportunities of the Johannes Kepler University of Linz, in particular the campus-wide wireless LAN infrastructure. This encompasses technological innovations concerning synchronous and asynchronous communication between teachers and students. Goal of “JKU Learning Networks” is to develop means for supporting learning processes with a global knowledge repository, a campus-wide awareness-system as well as means for interacting any time and any where.

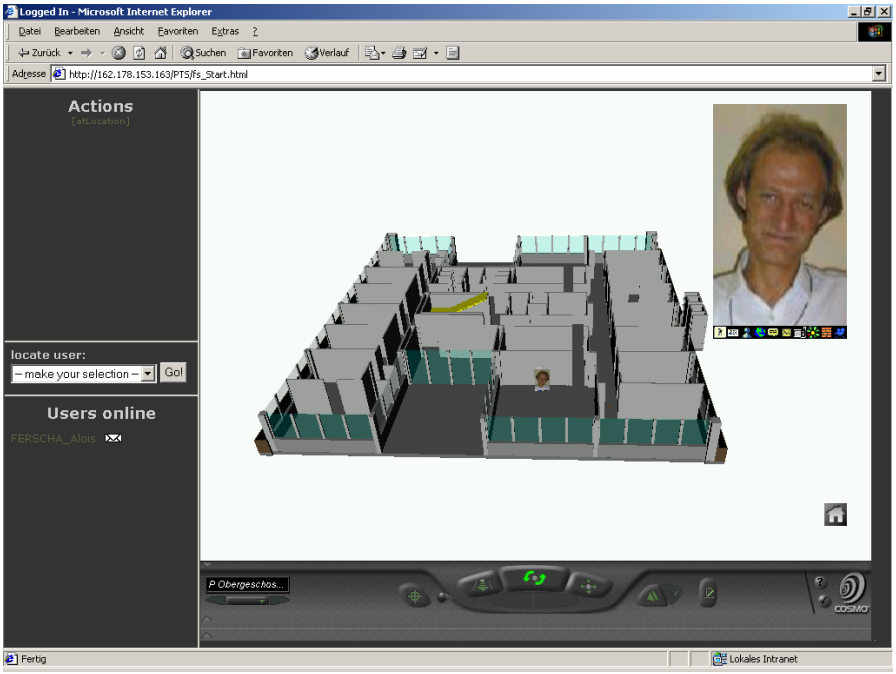
According to these goals, three main activities have been carried out. The first one is the development of a learning platform in the framework of the project “MobiLearn”, which enables learning any time, at any place and in any situation for each learner. The structure of the learning contents is compatible to the IMS Content Packaging format, which enables flexible authoring and delivery. Students can choose among three levels of detail while “consuming” the learning contents: highly-structured slides, detailed scripts and additional information. The following figure gives an example of the three levels of detail.

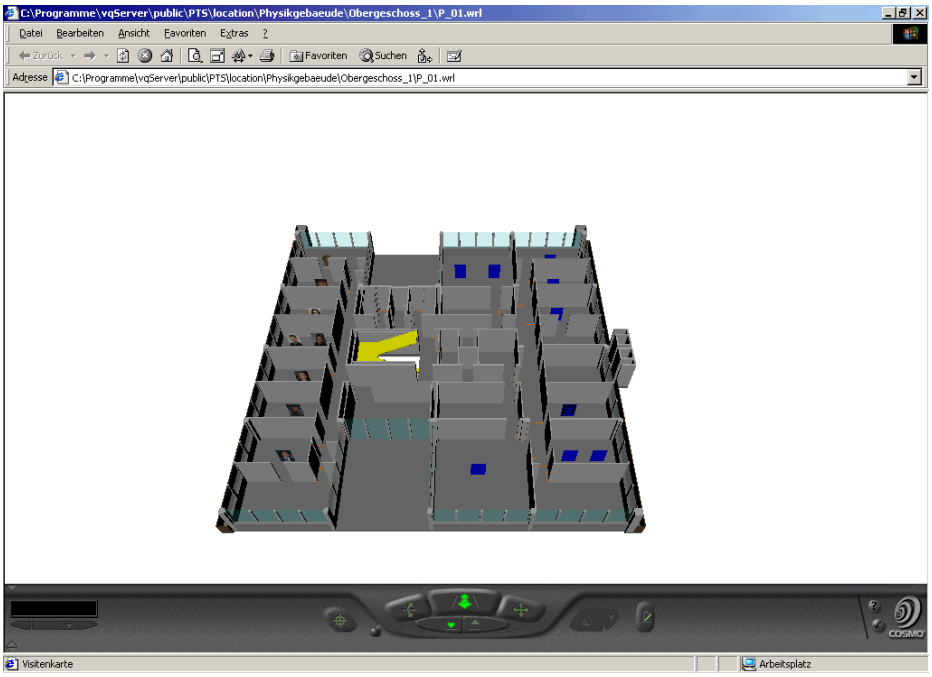
Similar to the levels of detail, the content presentation is adapted to the end device, which can be a notebook computer, PDA or a smartphone. Due to the flexible structure of the learning contents, their reuse is facilitated as they can be composed to new courses easily. The learning platform (cf. the following screenshot) provides tools for content annotation, team awareness and interaction among students and teachers. Learning contents for twelve courses of the media informatics domain have been developed.



One of several innovations beyond the provision of media informatics courses with a fully-functional learning platform is referred to as “physical hyperlinks”, which are visual markers in printed learning materials. They can be recognized with simple cameras as they are embedded in nowadays mobile phones, which in turn display additional information to the contents (e.g. related web sites). The MobiLearn website is accessible at http://www.mobilearn.at.

In the second area of activities, a campus-wide awareness system has been developed. The whole campus of the Johannes Kepler University Linz has been modelled in the VRML markup language, which enables navigation through the whole three-dimensional virtual campus. Based on this model, awareness features and location based services can be realized: the location of people and their presence can be visualized in the model (e.g. with pictures at the respective places, as it can be seen in the following figures), the way from a certain location to another location can be demonstrated with an animation of a flight through the virtual university campus, or virtual post-it messages can be placed at certain locations.

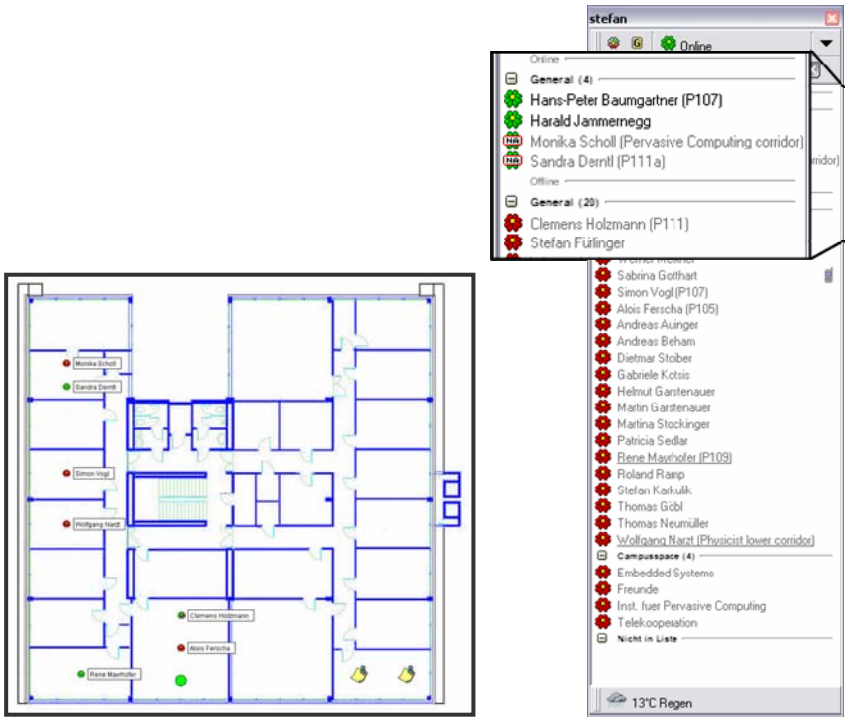




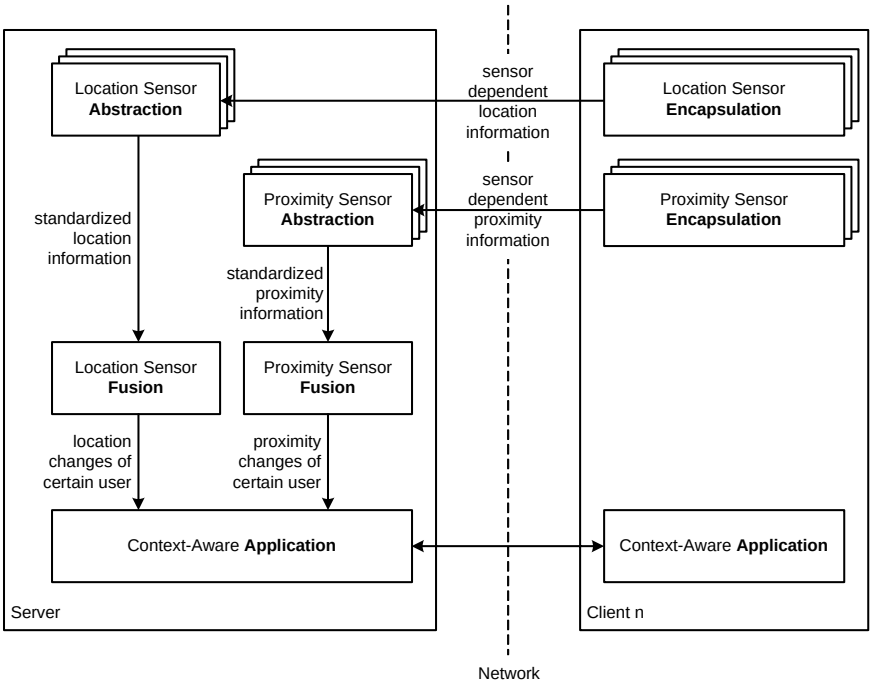
The third activity in the framework of the “JKU Learning Networks” project was the prototypical development of a “Group Interaction Support System” (GISS), whose aim was to support context-aware interaction in groups. Seven services have been implemented, where different awareness information (location L, time T, identity I, activity A and group-context G) is taken into account:

	L	*T*	*I*	*A*	*G*
location awareness	X		X		
forming and managing groups	X		X	X	
synchronous group communication			X	X	X
asynchronous group communication	X	X	X	X	X
availability management	X	X			
reminders	X	X	X		
group gathering support	X	X	X		X

These services have been integrated in an ICQ-compliant instant messenger, where locations of the colleagues are shown in the contact of the messenger or on a two-dimensional map of the respective campus area. Text messages can be bound to certain locations, and they automatically pop up on the computer screen of a student who enters this location. The instant messenger together with the map visualization of locations can be seen in the following figure.



The whole system is built upon a software framework, which allows the integration of various location sensors (e.g. based on wireless LAN or RFID) that are necessary for the services mentioned above. The architecture of the framework can be seen in the following diagram.



A more detailed description of context-aware interaction in groups, its prototypical implementations as well as the location sensing system is given in the following.

Context-Aware Interaction in Groups

Dipl.-Ing. Stefan Oppl

Univ.-Prof. Dr. Alois Ferscha

Institut für Pervasive Computing
Johannes Kepler Universität Linz
A-4040 Linz, Altenberger Straße 69
www.pervasive.jku.at

1 Context-aware Interaction in Groups

When talking about context-aware group-interaction, it is necessary to introduce the basic concepts that form the foundations of this work. For this reason, first the notion *group* is defined in the way it is used in this work. After this the relevant aspects of interaction and the support of those aspects in computer systems are presented.

As the term *context-aware* appears in the title of this thesis, it is also essential to give an introduction to the concepts of *awareness* in terms of computer science, and – building upon this – to define the notion of *context* and the characteristics of a *context-aware* system.

1.1 Traditional versus virtual Groups

Traditional groups tend to be co-located and are working in close proximity to each other. In contrast, virtual groups are in general composed of individuals that are dispersed geographically and/or temporally, as it is a common setting in the Internet. Because of this, virtual groups differ from traditional groups in many terms. They have different management, scheduling, communication and technical infrastructure needs.

Virtual groups face many unique challenges [John03], most of them stemming from reduced face-to-face interaction in distributed settings. They require, for example, more autonomy and longer cycles to complete certain tasks. It is also more difficult to achieve cohesion within the group and build up positive group-dynamics. Obviously, the technical support for virtual groups has to be much more sophisticated than for traditional groups. Unreliable or insufficient infrastructure will make failure much more likely.

In traditional teams it is fairly easy to become aware of the current task status, the forward progress or the availability and participation of other members. Not so in virtual teams; awareness information has to be delivered explicitly to provide at least the feeling of real group work. Connected to this, the lack of subtle communication cues and participation awareness can undermine trust between group members when working in a virtual group. Also this issue has to be taken into account, when designing awareness support systems for virtual groups.

1.2 Awareness

The notion of awareness in computer science is not a new one. It has been a field of intense research for over a decade now, since it has become an issue in the CSCW-community.

1.2.1 Definition

Dourish and Bellotti [Dour92] proposed one of the first and most widely accepted definitions of awareness in CSCW settings:

> *Awareness is an understanding of the activities of others, which provides a context for your own activity.*

This context is used to ensure that individual contributions are relevant to the group activity as a whole, and to evaluate individual actions with respect to group goals and progress. This information, then, allows groups to manage the process of collaborative working. Awareness information is always required to coordinate group activities, no matter, in which domain the group works or whether it is co-located or distributed.

1.2.2 Types of Awareness

Many researchers have collectively proposed a large number of definitions and classifications of awareness, trying to generalize or to focus on a certain aspect of awareness. Examples of those definition approaches have been collected by [Drur02] and [Liec00] and are summarized in the following list:

- *Concept awareness*: The participant's understanding of how their tasks will by completed.
- *Conversational awareness*: Who is communicating with whom.
- *Group-structural awareness:* Knowledge about people's roles and responsibilities, their positions on an issue, their status and group processes and similar issues.
- *Group awareness:* The ability that peers may have to stay in touch and to keep track of each others' activity.
- *Informal awareness:* the general sense of who is around and what others are up to.
- *Peripheral awareness:* showing people's location in the global context *or* where people know roughly what others are doing.
- *Social awareness:* the understanding that participants have about their social connections within their group.
- *Task awareness:* the participants' understanding of how their tasks will be completed.
- *Task-oriented awareness:* awareness focused on activities performed to achieve a shared task.

- *Workspace awareness:* the up-to-the-minute knowledge of other participants' interactions with the shared workspace *or* who is working on what.

It's clearly obvious, that most of those definitions are somewhat informal and overlap in certain areas. For this thesis, the definitions of *group awareness* and *workspace awareness* have been considered most relevant, as they describe the intended awareness information best and cover several other more special definitions like *informal awareness*. For this reasons, those two notions will be given a closer look in the following sections.

1.2.3 Group-Awareness

Group awareness can be defined as the ability that peers may have to stay in touch and to keep track of each other's activities [Liec00]. The information that group members maintain about each other may not be very consequent or very precise. Having only a general idea of what is happening, or merely that something is happening, is often already very valuable. It is obvious that the realization of group awareness may not require any technology support at all. When people share the same room, they naturally produce and interpret a constant flow of subtle cues with this aim. Glancing at someone may be enough to decide whether it is appropriate or not to start a conversation.

On the other hand, when the members of a group are scattered across space and time, technology may offer surrogates to this natural process – media spaces are an example for such technology. Obviously, groups formed in the context of the workplace greatly benefit from increased awareness. Communication and collaboration are easier, cooperation is smoother and, as a result, work processes are more efficient.

Groups formed in other situations, for instance in domestic settings, can also take advantage of awareness. The benefits may not be measured in terms of efficiency, but rather in terms of bonding, empathy or in a stronger sense of belonging to a community.

Steinberg et al. [Stei99] have defined the notion of group awareness in a broader sense (also including the group's tasks) and have identified four specific types of awareness deficiencies in today's virtual (distributed) groups in an empirical study. They have identified a lack of awareness about the others' *activities* (what are they doing?), about other's *availability* (when and how to reach them?), about *process* (where we are in the project?) and about the *perspective* (what are the others thinking and why?).

1.2.4 Workspace-Awareness

Workspace awareness is another well-known type of awareness, which has been defined first by Gutwin et al. [Gutw95] to be *up-to-the-minute knowledge of other participants' interactions with the shared space*. It emphasizes the fact that awareness generally emerges when people share a space, or at least when they share artefacts [Liec00]. This applies to both synchronous and asynchronous collaboration. Some of the research in the domain has focused on the design of shared editors, i.e. software tools that support the collaborative creation and manipulation of digital artefacts.

This effort has led to the design of novel user interface techniques and widgets, e.g. telepointers and radar views. Shared information spaces are other systems, which highlight the value of workspace awareness. Some of them are organized as document spaces, others are organized according to a spatial metaphor. In any case, awareness mechanisms can be introduced, allowing users to encounter each other on-line. Such mechanisms are clearly needed to ensure smooth collaboration within the information space, especially when users need access to shared artefacts.

1.3 Context

Being aware about a user's environment and the user's state enables a computer to build up a model of the user's current context. With knowledge about this context, the computer then may provide services, which take into account this information and adapt its behaviour accordingly.

1.3.1 Definition

In the work, in which the term 'context-aware' was introduced the first time, Schilit and Theimer [Schi94] refer to context as *location, identities of nearby people and objects, and changes to those objects.* In a similar definition, Brown et al. [Brow97] define context *as location, identities of the people around the user, the time of day, season, temperature, etc.* Ryan et al. [Ryan98] define context *as the user's location, environment, identity and time*.

Other definitions have simply provided synonyms for context, referring, for example, to context as the environment or situation. Some consider context to be the user's environment, while others consider it to be the application's environment.

Schilit et al. [Schi94a] claim that the important aspects of context are: *where you are, whom you are with, and what resources are nearby*. They define context to be the *constantly changing execution environment*. They include the following aspects of the environment:

- *computing environment* (e.g. available processors, devices accessible for user input and display, network capacity, connectivity and costs of computing)
- *user environment* (e.g. location, collection of nearby people and social situation)
- *physical environment* (e.g. lighting and noise level)

Dey et al. [Dey98] define context to be *the user's physical, social, emotional or informational state*. Finally, Pascoe [Pasc98] defines context to be *the subset of physical and conceptual states of interest to a particular entity*.

All of these definitions are too specific for our purpose. Context is all about the whole situation relevant to an application and its set of users. We cannot enumerate which aspects of all situations are important, as this will change from situation to situation. For example, in some cases, the physical environment may be important, while in others it may be completely immaterial. For this reason, in this work the definition of context according to Dey [Dey00a] will be used, which provides a more general notion of context and is widely used in literature today:

> *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves.*

As context can be gathered either implicitly (through sensors) or explicitly (through user input), virtually every application can be called context-aware, insofar as it reacts to user-input. However, in this case, focus is laid on implicit gathering of context by utilizing sensor input.

1.3.2 Types of Context – Context-Dimensions

To systematically design context-aware applications, it is useful to identify categories of context (so called context dimensions) that help designers to uncover the most likely pieces of context to be used. As we have already seen before, a manifold of classifications have been proposed in literature, but here we will present the categories Dey has identified [Dey00a], as these are the most common ones.

As stated before, there are nearly uncountable dimensions, that could be identified, but in practice, some of them are more important than others. These are *location*, *identity*, *activity* and *time*, which could be seen as the primary context types for characterizing the situation of an entity, as they can serve as indices into other sources of contextual information (secondary context types). If we know a person's identity, we can easily derive related information from several data sources such as birth date, list of friends or email addresses. Knowing the location of an entity, we can determine the nearby objects and people and what activity is occurring near the entity.

There are some situations in which multiple sources of primary context are required to index into an information space to acquire secondary context information. For example, the forecasted weather is a context dimension for outdoor activity and can be obtained by querying a forecast database with the desired location and time.

Dey himself denotes his classification to be incomplete. For example, it does not include hierarchical or containment information (as it may be useful for locations, where for example a room is part of a floor). To get around this at least for location, where the lack is obvious, a hierarchical model of location-contexts is used in this thesis.

1.3.3 Context-Awareness

As mentioned before, context-aware computing was first discussed by Schilit and Theimer [Schi94] in 1994 to be software that "*adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time.*" However, it is commonly agreed that the first research investigation of context-aware computing was the Olivetti Active Badge [Want92] work in 1992. Since then, there have been numerous attempts to define and explore context-aware computing.

Pascoe et al. [Pasc98] define context-aware computing to be the ability of computing devices *to detect and sense, interpret and respond to aspects of a user's local environment and the computing devices themselves.* Many researchers (among them [Schi94], [Brow97] and [Dey98]) define context-aware applications *to be applications that dynamically change or adapt their behaviour based on the context of the application and the user*.

As again these definitions are to special for this work, the definition of context-aware applications given by Dey in [Dey00] is used here, consequently following the path lead by the definition of context already given in chapter 1.3.1:

A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

By using this definition, not only applications that adapt to context but also applications that visualize context can be classified as being context-aware. This seems to meet a more intuitive notion of context-awareness, as an application has to be aware of a context in order to present it to the user.

1.3.4 Levels of Interactivity

In recent research, several attempts have been undertaken to classify the activities that can be triggered by context information. For example Barkhuus and Dey [Bark03] define three levels of interactivity for context-aware applications: *personalization*, *passive context-awareness* and *active context-awareness*. *Personalization* is where applications let the user specify his own settings for how the application should behave in a given situation; *passive context-awareness* presents updated context or sensor information to the user but lets the user decide how to change the application behaviour, whereas *active context-awareness* autonomously changes the application behaviour according to the sensed information [Bark03].

Table 1: Examples for the tree levels of interactivity (excerpt from [Bark03])

Service	*Personalization*	*Passive Context-Awareness*	*Active Context-Awareness*
Phone ringing	Different ringing profiles that are set manually	The phone prompts the user to adjust the profile when sensing a certain situation (e.g. meeting, restaurant)	The phone automatically changes profile when sensing the user is in a certain situation (e.g. meeting, restaurant)
Lecture slides	Manual search to see if lecture slides are available online	If signed up, the user is alerted of available slides for participants	Automatic alert every time the teacher updates lecture slide website
Location tracking	Manual location tracking of predefined friends.	Location tracking of friends and setting to alert when they are within a certain range	Location detection of friends that alerts when they are within 100 meters of user
Activity tracking	Display of potential communication partner's social situation (e.g. meeting, home, out)	In a new context, the user is prompted to display the user's situation to others.	Automatic switch to display of social situation when entering a new context.

Personalization, sometimes also referred to as customization or tailoring, is a common feature of computing applications. Researchers argue that the diversity and dynamics of applications

call for an increased level of tailoring in software, and that this emphasis on customized functionality will add to the user experience and smoothness of interaction [Stie97].

Active context-awareness describes applications that, on the basis of sensor data, change their content autonomously, where *passive context-aware applications* merely present the updated context to the user and let the user specify how the application should change, if at all [Chen00]. Examples for these levels can be found in Table 1.

In [Dey00a], another possible categorisation of context-aware features that applications may support is given. Three categories are identified:

- *Presentation* of information and services to a user
- *Automatic execution* of a service
- *Tagging of context* to information for later retrieval

While the first two categories can be easily mapped to the categorisation of [Chen00], it's not that easy with the third category. *Presentation* corresponds with *passive context-awareness*, while *automatic execution* is equivalent to *active context-awareness*. *Tagging of context* could be classified to belong to *active context-awareness*, but points out that history is also important when designing context-aware applications and adds the possibility to classify services, that maybe are only active in background collecting information until any triggering conditions are fulfilled and some action is set.

The classification of Dey is based on the works of Schilit et al. [Schi94a] and Pascoe [Pasc98], who both proposed a taxonomy of context-aware features. Both of them list the ability *to exploit resources relevant to the user's context*, the ability *to execute a command automatically based on the user's context* and the ability *to display relevant information to the user*. But beyond this, the taxonomies differ in some points. For example, Pascoe does not support the presentation of commands relevant to a user's context, while Schilit et al. do with so-called *contextual commands*. In contrary, Schilit's taxonomy does not include the ability to associate digital data with the user's context – this category is introduced by Pascoe and referred as *contextual augmentation*. Dey tries to unify those two taxonomies for example by not distinguishing between information and services. This results in the classification given above.

Due to the fine granularity of classification that can be reached with Dey's scheme, while staying on a level of generality, which allows easy classification, this one will be used in this thesis to classify the provided services.

1.3.5 Processing context

Between sensing context data and triggering context-aware services, several intermediate steps for processing this information have to be undertaken. When gathering data from the context sensors (either time-triggered or event-triggered), they do not contain any meaning at all. The *raw low-level context data* acquired has first to be interpreted and transformed in order to deduce *high-level context information* that can be used to make assumptions about a user's current context.

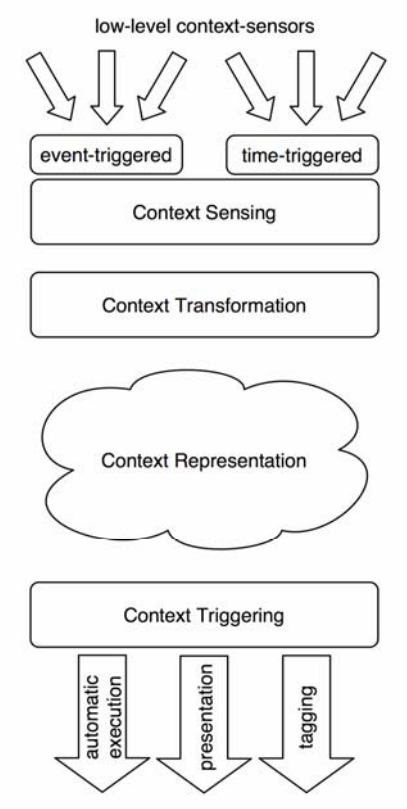


Figure 1: Processing context data

This high-level context information has to be represented in a common format in order to facilitate generic processing modules that make an application context-aware.

Every time the input of one of the context-sensors changes or is sensed due to an expired time-trigger, the *Context Sensing* component collects the raw sensor data, thus building an interface between external sensors (either implemented in hardware or in software only) and the higher-level context processing stages. It then passes on this data to the *Context Transformation* component, which interprets the raw data in order to retrieve context information. This may be done by simply mapping sensor values on high-level information but may also require more sophisticated activities like time-row-analysis or combination with other sensor values.

The context information gathered this way is then passed on to the *Context Representation* component, which on the one hand stores this information for later retrieval in a common format and on the other hand provides a generic interface for accessing the contained information. This interface is used by the *Context Triggering* component, which checks for changes in the user context and triggers context-aware services accordingly. These services can be classified into the three categories already mentioned in chapter 1.3.4.

1.4 Interaction

Every time people work in groups (from two members on), they are required to interact with each other. Interaction includes communication – either synchronous or asynchronous, explicit or implicit – between the group members, the sharing of common artefacts and cooperative working on those artefacts.

Since computers have become networked, they are used to support this interaction in groups with several services, which have lead to the development of a big research community that deals with *Computer Supported Cooperative Work* (CSCW). In this chapter, we will have a short look at the notions in this field, which are relevant for this work.

1.4.1 CSCW

Computer Supported Cooperative Work (CSCW) is the field of research, which is spanned over disciplines to examine the cooperation within a team. Its goal is the development of new computer technologies to support teamwork. Key issues of CSCW are group awareness, multi-user-interfaces, concurrency control, communication and coordination within the group, shared information spaces and the support of a heterogeneous, open environment which integrates existing single-user applications.

In 1988, Greif [Grei88] has defined CSCW to be *an identifiable research field focused on the role of the computer in group-work*. Greenberg gave a similar definition in 1991 [Gree91]: *CSCW is the specific discipline that motivates and validates groupware design. It is the study and theory of how people work together, and how the computer and related technologies affect group behaviour*.

CSCW systems are often categorized according to the time/location-matrix using the distinction between same time (synchronous) and different times (asynchronous) and between same place (face-to-face) and different places (distributed). The matrix (given in [Grun94]) is shown in Table 2.

Table 2: Classification of CSCW-Systems [Grun94]

Time / *Location*	*same time (synchronous)*	*different time (asynchronous)*	
		predictable	*not predictable*
same place	face-to-face	shift work	blackboard
different place (predictable)	video conference	email	collaborative document processing
different place (not predictable)	mobile radio conference	non realtime computer conference	workflow management

1.4.2 Communication

In literature, means of communication are often classified according to two criteria:

Table 3: Classification of means of communication [Diap93]

	explicit	*implicit*
synchronous	face-to-face telephone video- / audio-conference instant messaging	events status
asynchronous	letters email post-its	artefacts history

The first criterion is whether communication happens synchronous or asynchronous. In *synchronous communication* the communication parties are available at the same time and can have near real-time interaction. In *asynchronous communication* the communication parties are not available at the same time.

The second criterion is about whether the communication is explicit or implicit. *Explicit communication* requires direct action of the parties who want to communicate. In contrast, *implicit communication* is based on shared information. An example for this would be the percipience of the changes made to a common artefact.

2 Related Work

There are few systems that have the same or similar objectives as the presented system, namely the provision of context-aware interaction support for groups in spatially bounded areas (like campuses). On the other side, there is a manifold of systems that provide solutions for one or more issues addressed in this work.

For this reason, the first section will deal with the systems that have similar objectives. The following sections present systems that provide solutions for parts of this work:

- Systems for supporting awareness among groups (cf. chapter 2.2)
- Systems for context-aware synchronous communication (cf. chapter 2.3)
- Systems for context-aware asynchronous communication (cf. chapter 2.4)
- Visualisation of position information (cf. chapter 2.5)
- Context-aware reminders (cf. chapter 2.6)

2.1 Context-aware Support of Students on Campus

There is a manifold of systems supporting students in their daily live on the university campus. However, most of these systems appear either in the form of an (more or less enhanced) e-learning system or some sort of bulletin board system (BBS). Hardly any student-support-system supports any form of awareness about other students' activities. Issues of interaction and learning in groups are only addressed in very few systems, where support most of the time ends with the provision of a forum and a shared document repository.

Although the systems presented here do not use much more context information than location and (partially) time and identity and support group interaction rather poorly, they have at least similar objectives in the field of location-aware presentation of information.

2.1.1 ActiveCampus

ActiveCampus [Gris03] has been developed by the University of San Diego (UCSD) with the goal to provide a wireless location-aware computing environment on the university campus. Location of the users is obtained via the WiFi-infrastructure (signal strength based). On top of the *ActiveCampus*-Framework several applications have been built, including *ActiveClass* and *ActiveCampus Explorer*, both executable not only on PCs but also on PDAs.

ActiveClass enables collaboration between students and professors by serving as a visual moderator for classroom interaction. *ActiveCampus Explorer (ACE)* uses a person's context,

mainly location, to help engage them in campus life, which is related more closely with the objectives of this work.

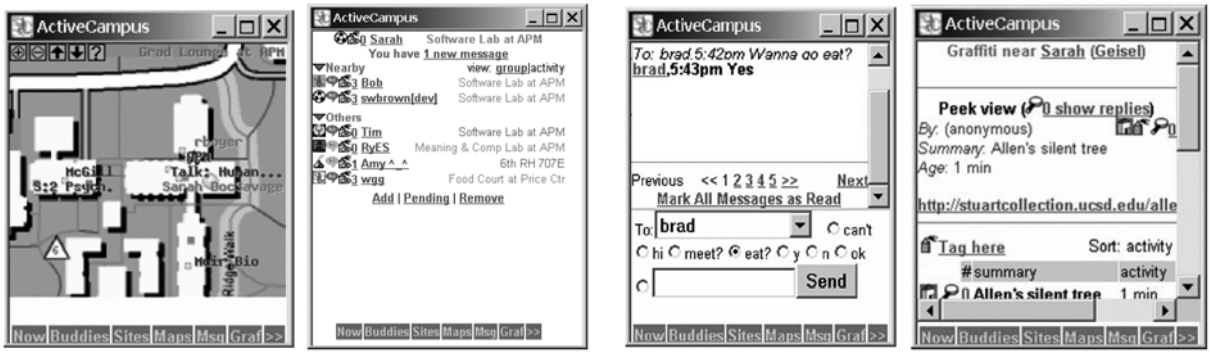


Figure 2: ActiveCampus Explorer User Interface [Gris03]

The *ACE* supports similar services like the system presented here. However, it uses location as the only explicit context dimension (although in some cases identity and time is also taken into account). Furthermore, no explicit support for group interaction is provided, as the communication is based on standard instant-messenger-like contact lists (here referred as buddy lists). *ACE* supports the visualization of one's buddies' positions, the placement of location-based messages and location-based and user-based reminders. Furthermore it can be integrated in an Instant Messenger (using Jabber [Jabb04] as a bridge to the most common IM-networks), thus being capable of all the features this messenger provides.

Although not being built in a very modular and extensible way, *ActiveCampus* provides many features that are crucial for successful interaction support in campus settings. As said before, it still lacks support for groups and their interaction needs. Nevertheless, as the system is still under development, the already provided services seem very promising. A comparision of the features of *GISS* and *ActiveCampus* is given in chapter Table 4.

Table 4: Comparison ActiveCampus vs. GISS

		ActiveCampus	*GISS*
usage of context dimension	*location*	yes	yes
	time	partially	yes
	identity	partially	yes
	activity	no	partially
	group	no	partially
extensibility		poor	good
scalability		good	presumably

			good***
reuse of existing messenger accounts		no	yes
synchronous communication	*individual*	yes	yes
	group	implicitly*	yes
asynchronous communication	*individual*	yes	implicitly**
	group	implicitly*	yes
context-aware reminders		partially	yes
group building & management		no	yes
gathering support		no	partially
context-aware visibility management		no	yes

* via selecting multiple recipients

** via delayed delivery in the underlying messenger network

*** not yet tested exhaustively

2.1.2 Cawar

The *Cawar-System* [Broy04] (context aware architectures @ Campus TU München-Garching) is a project that utilizes the existing WLAN-Infrastructure on TU München-Garching Campus to provide some context-aware services. These include navigation support, location-based information, context-aware messaging and reminders and a not clearly specified context-aware portal. The main research focus of this project lies on the assurance of privacy and not on the provision of services. Unfortunately, no further information is available currently, as the project is still in proposal stage, providing only a few concepts and usage scenarios.

2.2 Supporting Interaction in Groups

Several Systems have already addressed the importance of supporting awareness when working in groups, especially if they are spatially distributed. Those systems mainly support workspace awareness (cf. chapter 1.2.4), taking into account the results of the already cited study about the main awareness deficits in distributed groups [Stei99] (cf. chapter 1.2.3).

Looking at the presented systems, we can see that awareness information is presented in two ways. Most of the time, an object-driven approach is used, presenting for example information about an object (e.g. document) manipulation in history. More seldom, a subject-driven approach has been chosen. Often, nothing more than presence information is provided (in different granularities) here. As it is considered more expressive, in this work, the different

approaches of presenting awareness are referred to as *artefact-based awareness* (equivalent to object-oriented awareness) and *user-based awareness* (equivalent to subject-based awareness).

2.2.1 teamSCOPE

In 1999, teamSCOPE [Jang00] has been developed at the Michigan State University as a web-based collaborative system to respond to the awareness deficits in virtual groups mentioned in [Stei99].

teamSCOPE provides the following means to support group awareness:

- File Manager
- Message Board
- Calendar
- Activity Summary
- Activity Notification
- Team Member Login Status
- Team Member Usage Information
- Team Summary Site

The clear intention of *teamSCOPE* is to support group interaction as a whole. It does this pretty well but does neglect workspace awareness, as it does not provide any *up-to-the-minute* knowledge of what the other team members are doing (besides login status). Mainly information is logged here to provide awareness by making available the history of an object or a team member. As *GISS* focuses on different kinds of context-information, *teamSCOPE* and the research carried out in its surrounding provides a good background for this work in terms of showing possible group awareness services from a CSCW-point-of-view but can not compared in terms of functionality to the system presented here for the given reasons.

2.2.2 BSCW

BSCW (Basic Support for Cooperative Work) [BSCW04] [Bent97] enables collaboration of groups over the Internet. BSCW is a “shared workspace” system, which supports document upload, event notification and group management. It only provides artifact-based awareness and uses identity and activity as context-information to present an artifacts modification history.

BSCW has been chosen to be presented in this chapter because it is the most widely used system of its kind in real-life settings. *BSCW* also provides means of communication for users, as it allows the establishment of so-called discussions, a means mainly for asynchronous communication (like in forums).

The “shared space” metaphor supported by *BSCW* allows users to store documents and other objects into a folder, which can then be accessed by others. The access model which determines who can do what is very simple – each member of a workspace has access to all objects within it and all members have complete control over each object.

As *GISS* does not directly support the sharing of files and focuses on workspace awareness (in the real world), it is hard to compare to *BSCW*, which concentrates on a different aspect of interaction in groups. Thus *GISS* and *BSCW* rather extend each other’s functionality than trying to solve the same problems.

2.3 Context-Aware synchronous Communication

Instant Messaging as a form of synchronous communication is one of today’s most popular applications on the Internet. Nearly everybody uses some kind of instant messenger to keep in touch with colleagues and friends and have short, informal sessions of communication.

The concept of today’s instant messengers contains no or only marginal support for informing the user about the state of his communication partner. This raises big concerns that Instant Messaging may be to distracting when used on the workplace. As maybe everybody of us knows from his own experience, these concerns are well founded. Others disturb us in our current work by initiating a messaging session just because they do not know we are occupied right now [Tang03].

There is surely need for improving awareness in the upcoming generation of new instant messengers. As the necessary technology is available today, mainly the form of presentation of this additional information is an open research issue.

The presented systems try to solve the problem of appropriate presentation of awareness information and sometimes try to enhance communication itself by providing additional information.

2.3.1 ICQ / AIM / Yahoo Messenger / MSN Messenger

The widely used instant messenger applications, which for example include ICQ [ICQ04], AIM [AIM04], Yahoo Messenger [Yaho04] or MSN Messenger [MSN04], in the first place

provide means for synchronous communication. To support this form of communication, they offer some sort of very basic awareness information about the communication partner's current activity context.

The user is offered the possibility to set his current availability status to a certain value, providing others with awareness about his current availability. The messenger application additionally analyses the user's activity on his computer (mouse movements, typing etc.) and automatically adapts the status (*away*, *online*, etc.) according to the retrieved information, thus providing some simple sort of context aware behaviour.

As the user interface of *GISS* is based in a instant messenger application as can be seen in chapter 4.4, this context-aware feature is integrated into the *GISS*-application, providing some sort of simple activity context (cf. chapter 4.3.3).

2.3.2 Awarenex

Awarenex is an Instant Messenger and awareness prototype from SUN Labs that demonstrates additional real-time awareness information useful both for initiating contact and negotiating conversation [Tang01]. It doesn't provide any means of asynchronous communication.

Figure 3: Awarenex User Interface [Tang03]

Figure 3 illustrates the ways *Awarenex* integrates real-time awareness information to help users find good times to establish contact. For each entry in the Contact List, *Awarenex* shows not only the user's name, but also their so-called "locale" – an indication of whether they are in their office, at home, or another location. A locale usually corresponds to a physical location, but it is more intended to convey a high-level description of the person's context rather than their precise physical location. For example, the "mobile" locale covers many physical locations and lets others know when the person is "on the road."

An indication of the user's absence is reflected by displaying the inactive duration of the user's keyboard or other input device. Beyond this presence information, additional activity

indicators as whether the user has an appointment scheduled in their online calendar or is engaged in an IM or telephone call are included. Taken together, these indicators give cues about whether the user is preoccupied with some other activity, and thus is less receptive to additional incoming communication.

Awarenex not only supports awareness while establishing a new interaction but also supports communication itself. For example, text input is transmitted character by character, thus supporting a sense of conversational flow and getting even closer to real-time communication than traditional instant messaging.

2.3.3 WebWho

WebWho [Ljun99] is a lightweight, web based awareness tool, that shows a schematic view of the workstations in a large university computer lab, and who is currently logged in where. Based on this data, means for Instant Messaging are provided. *WebWho* explicitly conveys place information and provides an overview to find out about other peoples locations as well as unoccupied computers in the lab. The messenger functionality is supplemented by information about the communication partner gathered from a central database. The system is mainly intended to support collaboration and coordination between distributed users, primarily within different rooms in a lab but also for people situated elsewhere.

WebWho is a lightweight service, being implemented straightforward with no underlying framework. As such, it is not really intended to be extensible but only to fulfil the tasks it has been designed for. It does this quite well but cannot serve as a model for *GISS* in terms of context-aware synchronous communication, as the design prerequisites are completely different (lightweight and straightforward vs. modular and extensible).

2.4 Context-aware asynchronous Communication

In chapter 1.4.2 the need for asynchronous communication within groups has already been motivated. While this sort of communication is generally covered by email, there are situations where one might want to limit the visibility of a message not only to a certain group of persons but also to a certain context – most of the time a location –, using the metaphor of a post-it, which can be stuck on any physical object.

As soon as the first more accurate positioning technologies became available in the mid-1990s, providing means for placing “virtual” post-its anywhere in the landscape became a big issue in research. One of the first approaches was *stick-e notes* by P. J. Brown [Brow96], which has even a wider scope, going beyond just location, will be presented in the first place.

Uncountable more or less similar approaches have followed, using different technologies to determine position and different strategies to visualize placed messages. A selection of those will also be presented shortly in the following.

2.4.1 Stick-e note

The concept of *Stick-e notes* has been developed by P.J. Brown [Brow96] and J. Pascoe [Pasc97]. *Stick-e notes* are messages that are attached to certain contexts, like location, nearby people, environmental states or time, simply following the metaphor of real-world post-it notes that can be written and stuck on any piece of furniture.

The metaphor of post-its as a means of providing context-aware information has been widely accepted. Even most of the time, the use is restricted to annotations attached to a certain location, there is no conceptual need for the restriction, *stick-e notes* can be attached to any context that is available to the application.

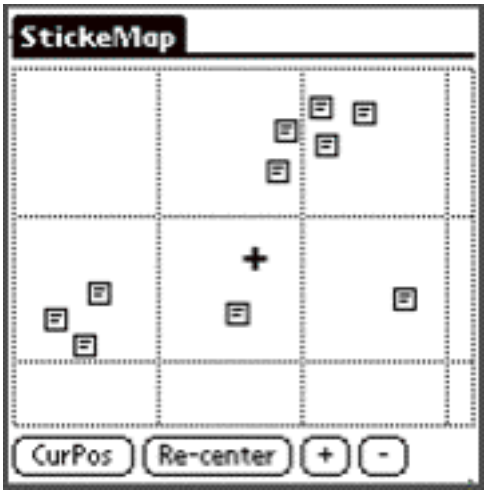


Figure 4: Stick-e note: Implementation on PDA [Pasc97]

There have been several implementations of the *stick-e notes* concept, which use location as the only context-dimension and are based on a palmtop-computer, thus providing some sort of virtual magnifying lens for real world objects, which displays attached information when pointed on them (or at least being in proximity to them).

As the *stick-e notes* system can be seen as the ancestor of all context- (or location-) aware applications that make use of the post-it-metaphor, the concept of virtual post-its in GISS as well as the concept of context-aware reminders is also based on this work.

2.4.2 eGraffiti

eGraffiti is a project carried out by the University of Cornell [Burr02]. Its main objectives lie in the determination of a user's location and the display of text messages based on the user's location and identity. The interface of the system can be seen in Figure 5.

Location determination is done rather roughly by taking into account the MAC-address of wireless access point one is connected to. Users can post messages, which are either visible for all others or just for a restricted group of users. Messages can only be posted and read at a user's current location, not providing the opportunity to get for example a list of all new messages all over the campus.

The *eGraffiti* system has been used to build a location-aware information system on Cornell University Campus, which is referred as *CampusAware* [Burr02] and is capable of providing visitors a guided tour on the whole campus, allowing them also to leave their own notes.

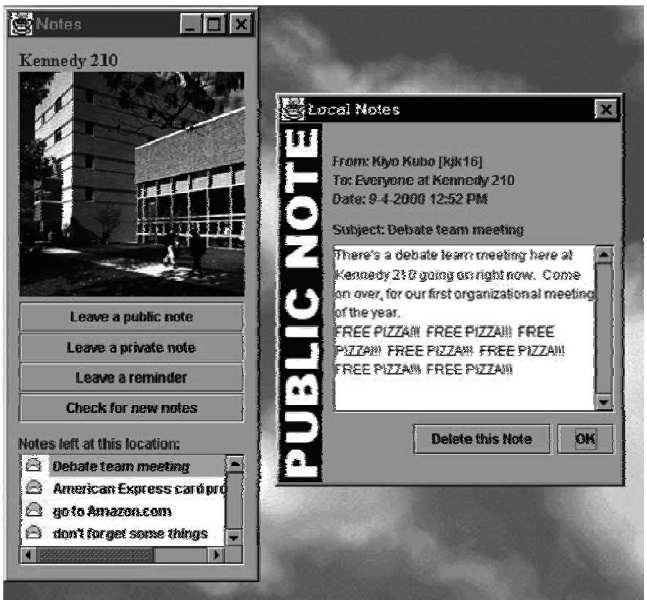


Figure 5: Screen capture of eGraffiti [Burr02]

2.4.3 GeoNotes

GeoNotes [Espi01] is a very specific research project from the Swedish Institute of Computer Science, because it does not mainly focus on the technical aspects of location-aware asynchronous communication through notes, like most of the other known projects do, but lays it research focus on the communicatory, social and navigational implications of the mass usage of a location-based information system of this kind. The authors state that in case of an open system, where everybody can post messages (in contrary to a closed system, where only con-

tent providers post messages and users are just consumers), the need for a well-structured and navigateable concept of displaying notes.

Although the results and concepts provided by this paper are very promising, they have been taken into account only partially for the design of GISS, because the number of post-its visible for one user at a certain location is not expected to exceed an amount that demands such a sophisticated navigation concept (authors of *GeoNotes* deal with several 100 notes at one location). In terms of the theoretical background, this paper has been considered really useful, especially the definition of requirements for interaction in large groups in general and for notes in this special case has provided much inspiration for the development of asynchronous interaction in GISS.

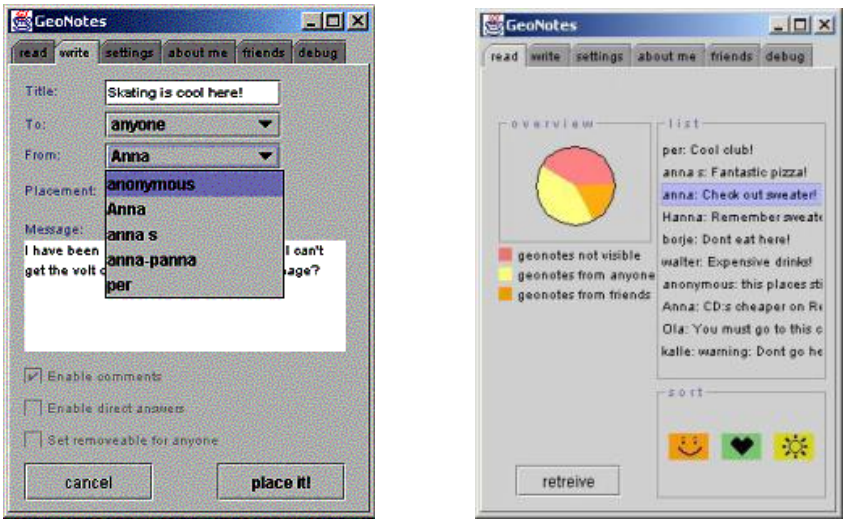


Figure 6: GeoNotes Posting (a) and Browsing (b) Interface [Espi01]

2.5 Visualisation of Location-Information

Visualisation of location-information has been subject of research in many projects. As can be found in literature, a two-dimensional, map-based view of users' positions is widely accepted to be a suitable solution for this issue. Only one project for this approach is presented as an example in the following. A different approach has been chosen in the second presented project, which uses a 3-dimension model of the world to present position information to the user.

2.5.1 ActiveMap

ActiveMap [McCa99] has been developed at the Centre for Strategic Technology Research in Northbrook, USA to be a visualization tool that enables users to gain greater awareness of the

location of people in their workplace environment. The position information is displayed by placing images of each person's face on a map of the region to visualize.

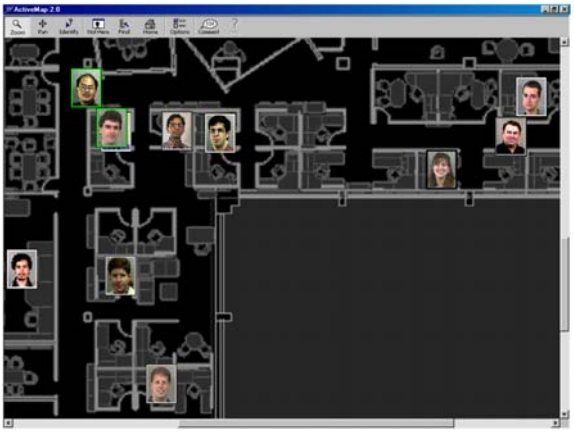


Figure 7: ActiveMap [McCa99]

Subject of research has not only been simply visualizing positions on the map but also showing the "freshness" (age) of position information as well as visualizing many people in a single room. For the first issue, two approaches have been examined, namely fading out the whole image of person as time passes by or just shading the border of the images, which has the advantage that the image quality is not degraded, if no new position information has been received recently (e.g. if the person is at work in his office).

The second issue – visualizing groups of people – has also been addressed in to ways. The first approach uses a tiled representation of people co-located in one room, where images partially cover each other, but faces still remain visible. The second approach uses a stacked view of users in a room, which has the disadvantage that only the uppermost image is clearly visible. The advantage is, that the number of people can be recognized more easily than in the first approach.

The viewer implemented in GISS has been partially inspired by *ActiveMap* and similar projects. As no images can be used for visualization simply due to the fact that they are not available in general, GISS uses a tiled view to present multiple users in one room.

2.5.2 CampusSpace

CampusSpace [Perv04] is a research project carried out by the Institute for Pervasive Computing at the Universtity of Linz and is based on an earlier work described in [Fers00]. Instead of using a 2-dimensional model of the region, in which user positions should be visualized, it follows a different approach by using a fully 3-dimensional model. The whole campus of the University of Linz has been modelled using VRML (Virtual Reality Markup Language).

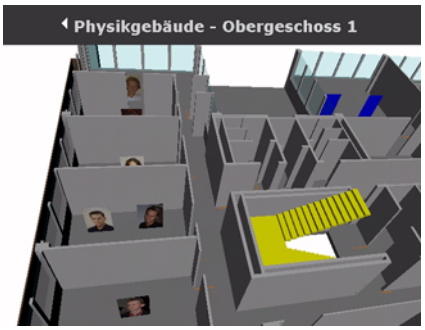


Figure 8: Screenshot of CampusSpace [Perv04]

This model has been evaluated to be used in GISS as location visualizing sub-system but has been considered unsuitable, as the provided means of navigation in the available browsers are to sophisticated for everyday use and navigation itself is challenging in terms of easily loosing orientation untextured environments like the given one. Furthermore, the development of VRML is discontinued, because it should be replaced by its XML-based successor X3D and therefore no support for the Java-APIs to dynamically change VRML-scenes is supported anymore. Development of VRML-browsers is also discontinued, the existing solutions require a large amount of computational resources and are fairly slow when rendering larger scenes.

2.6 Context-Aware Reminders

A reminder is a special type of message that one can send to himself or to others, to inform about some future activity that one should be engaged in. For example, a colleague might send us a reminder asking us to bring a copy of a paper to our next meeting. We use reminders to signal others and ourselves that a task still exists to be worked on and/or that a task is ready for further processing. We use reminders to re-establish needed information in short-term memory so that the trigger conditions for these reminders can be satisfied [Miya86].

Currently, there is a large number of tools and strategies to help one keeping track of his reminders. However, studies have shown that users still have difficulty dealing with reminders [Fert96]. Most of current reminder systems, acting as a form of externalized memory, do not present appropriate signals at appropriate times. More specifically, these tools are not sufficient because they are not proactive and do not make use of contextual information to trigger reminders at appropriate times in appropriate locations. Herstad et al. [Hers98] claim that in order to build useful, functional and powerful tools for supporting human-human interaction, context-information has to be taken into account. For example, to be most effective, a reminder to bring a paper to a meeting should be delivered when one is leaving his office and heading towards the meeting, and not when he happens to read his e-mail. The metaphor used

for this purpose in general is the use of post-its, which can be stuck on every context a system can sense [Brow96] [Pasc97].

Not very much research has been done in the area for context-aware reminders yet. In 2000, Dey and Abowd created *CybreMinder* as an application of their Context Toolkit. This system is described in the following chapter. The *MemoClip* developed by Beigl follows a more hardware-driven approach and will also be described shortly.

2.6.1 CybreMinder

CybreMinder is a context-aware Reminder System built upon the Context-Toolkit by Dey and Abowd [Dey00]. The goal of *CybreMinder* is to provide users with a tool that provides appropriate support for dealing with reminders. The particular objective is to support the following features of reminder tools:

- use of rich context for specifying reminders, beyond simple time and location and for proactively determining when to deliver them;
- ability for users and third parties to submit reminders;
- ability to create reminders using a variety of input devices;
- ability to receive reminders using a variety of devices, appropriate to the user's situation;
- use of reminders that include both a signal that something is to be remembered and a full description of what is to be remembered;
- allowing users to view a list of all active reminders.

The system provides some support for all of these features, except for the ability to create reminders using a variety of input devices. The maybe most important one is the allowing for the use of rich context in reminders. The interface of the application can be seen in Figure 9.

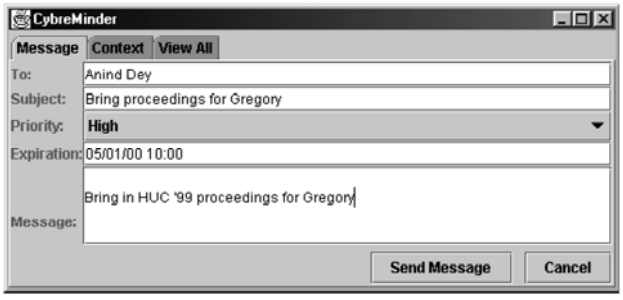


Figure 9: CybreMinder reminder creation tool [Dey00]

By using the features of the Context Toolkit, users are allowed to create arbitrarily complex situations to attach to reminders and to create custom rules for governing how reminders should be delivered to them. Users are not required to use templates, but can use any context that can be sensed and is available from their environment. However, this requires the users to use a fairly complex syntax to enter their reminders and the associated trigger conditions (cf. Figure 10).

On the delivery side, *CybreMinder* sends both the reminder and the associated situation to a service for display (via e-mail, available screen, or SMS). The quality of the reminder signal and the completeness of the reminder description depend on the service being used.

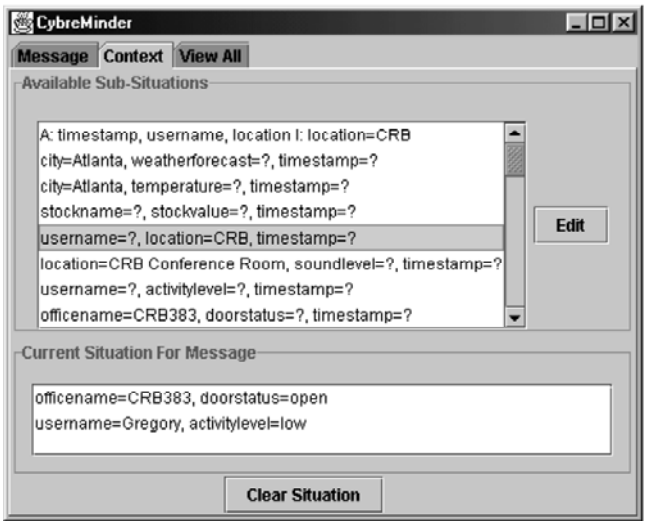


Figure 10: CybreMinder situation editor [Dey00]

CybreMinder delivers a reminder when the associated situation has been realized, and chooses the delivery mechanism/service based on the recipient's current context. However, it does not take into account how interruptible the recipient is.

2.6.2 MemoClip

MemoClip [Beig00] has been developed by Beigl to provide a comfortable and intuitive way to place context-aware (more specifically: location-aware) remembrance messages in the environment. For this purpose, the user carries a small piece of hardware, the so-called *MemoClip*, which contains a small embedded system, a display and some sensors and communication appliances. In the surrounding, so called location beacons are placed on all places of interest. These beacons are solar-powered and provide a means to determine a users location (cf. Figure 11). A central system, which communicates with both the beacons and the *MemoClips*, manages the placed reminders and is used to enter new reminders.

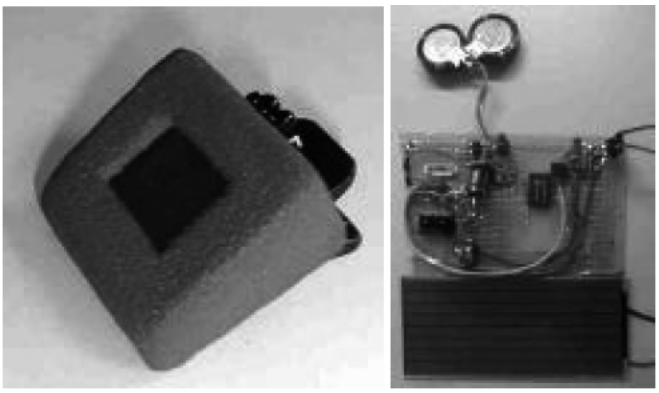


Figure 11: MemoClip and location beacon [Beig00]

To be independent from the availability of network connections when moving around, the placed reminders are downloaded onto the *MemoClip*, which determines its own position by searching for beacons whenever it is moved (recognized by internal accelerometers). The reminder information is represented on the *MemoClip* as a key/value-pair, where the location serves as the key. When a location is entered and reminders are found, the *MemoClip* tries to get the user's attention be playing a beep sound and displaying the messages belonging to the current location.

The *MemoClip* follows a promising approach in terms of being independent from possibly unavailable network connections. The major weakness of its approach is the restrictedness to location as the only context dimension. The use of time as another trigger would not have caused too much further effort and would have improved the system significantly. The concept GISS is designed after is more inspired by *CybreMinder*, as this concept is more open and supports arbitrary sources of context.

2.7 Summary

To get a quick overview of what the different described applications are capable of in comparison to *GISS*, two overview tables have been developed. The first table (cf. Table 5) shows the types of context information used and the types of services triggered. The chosen classification is oriented on the context types and types of services identified in [Dey00]. The used context dimensions are classified into *Location* (L), *Time* (T), *Identity* (I), *Activity* (A) and *Group* (G). Services can be assigned to one of the following categories: *context-aware presentation of information* (P), *automatic execution of services* (A) and *tagging of information for later retrieval* (T). For a more detailed description of this classification, cf. chapter 1.3.2 and 1.3.4.

Table 5: Comparison of related systems with regard to used types of context

	L	*T*	*I*	*A*	*G*	*P*	*A*	*T*
Context-aware Support of Students on Campus								
GISS	X	X	X	X	X	X	X	X
ActiveCampus	X	X	X			X		X
Carwar	X	?	X			X	?	
Supporting Awareness among Groups								
teamScope		X	X	X	X	X		
Awareness Database								
TeamPortal								
BSCW		X	X	X		X		
Context-aware synchronous communication								
Awarenex	X	X	X			X		
WebWho	X		X			X		
Context-aware asynchronous communication								
Stick-e note	X	X	X			X		
eGraffiti	X		X			X		
GeoNotes	X	X	X			X		
Visualisation of Location-Information								
ActiveMap	X					X		
CampusSpace	X					X		
Context-Aware Reminders								
CyberMinder	X	X	X	X		X		
MemoClip	X					X		

X … supported

? … unknown

The second table (cf. Table 6) is intended to give an overview over the aspects of group- and workspace-awareness used by the described systems for the provision of context-aware services. Those awareness-aspects have been divided into two categories: user-centred (where

awareness about a user's state is provided) and artefact-centred (where awareness about an artefact's state is provided).

The following user-centred awareness aspects have been considered:

- Location (L)
- Current Activity (CA)
- Membership in Groups (GM)
- Availability-State (AS)
- Intentions (I)

Looking at artefacts (such as files or notes), the following awareness aspects have been taken into account:

- Location (L)
- Creation date (CD)
- Last modified (LM)
- Modifiers (M)
- Modification History (MH)

Table 6: Comparison of related systems with regard to supported aspects of awareness

	user-centred					***artefact-centred***				
	L	*CA*	*GM*	*AS*	*I*	*L*	*CD*	*LM*	*M*	*MH*
Context-aware Support of Students on Campus										
GISS	X		X	X		X	X	X	X	X
ActiveCampus	X			X		X	X			
Carwar	X					X	X			
Supporting Awareness among Groups										
teamScope		X	X	X			X	X	X	X
Awareness Database										
TeamPortal										
BSCW			X				X	X	X	X

Context-aware synchronous communication										
Awarenex	X	X								
WebWho	X									
Context-aware asynchronous communication										
Stick-e note	X					X				
eGraffiti	X					X	X			
GeoNotes	X					X	X			
Visualisation of Location-Information										
ActiveMap	X									
CampusSpace	X									
Context-Aware Reminders										
CyberMinder	X					X	X			
MemoClip	X					X	X			

X … supported

3 Requirement Analysis

As a short retrospection, the goals for *GISS* have been defined as following:

- Context information has to be utilized to support interaction in groups in spatially bounded areas.
- Interaction aspects that have to be covered are synchronous and asynchronous communication as well as support for informal gatherings of groups.
- Besides the explicit interaction services, the need for further services, especially in the field of *group-awareness* and *individual information management* has to be evaluated.
- The practical usability of the system in terms of deploying it on a campus-like setting with a large number of users and lowering the learning and management effort for potential users naturally is a prerequisite for the successful deployment of the system.
- It has be assured that the system is easily extendable on both context sensing as well as user interface side by adding or exchanging components.

From this, the following basic requirements have been identified and will be specified more exactly in the following chapters:

- *Context-Awareness*: Design has to be oriented to the ability of sensing context and using this information to provide context-aware services for synchronous and asynchronous interaction as well as group-awareness.
- *Communication*: Synchronous as well as asynchronous communication has to be supported in both person-to-person and group settings.
- *Usability*: Common interaction paradigms should be used in order to lower barriers for usage. Failures on either client and server side should be transparent to the user or at least leave the system in a defined state.
- *Extensibility*: In order to enable a sustainable use of the system, it has to be easily adaptable to new circumstances both on context sensing side and user interface side.
- *Scalability*: For use in real world settings were largely heterogeneous utilization scenarios occur, scalability has to be assured.
- *Access*: The effort to get access to the system in terms of installing software as well as registering accounts and so on has to be as minimal as possible.

- *Client Resources*: The system should meet the requirements for a background application running all the time and thus has to utilize as less resources as possible.
- *Privacy*: Users have to be provided mechanisms to stop tracking and to restrict their visibility to a certain group of people.

3.1 Context-Awareness

The use of context information in the system is one of the main characteristics of the system and therefore is a very important requirement. Context, as already stated in the introducing chapters, is more than just location, therefore it is necessary to make the use of context as rich as possible.

To make use of rich context, it first has to be sensed. Besides the already available *location* information [Holz04], further types context information – like *time*, *identity* or *activity* – have to be gathered.

Group interaction support is also more than just displaying information that is more or less related to the user's current context. Supporting interaction in our case also means proactively triggering services, which will support the user with his current activity. As executing services by mistake because of misinterpretation of sensor data is annoying for the user, there should be mechanisms to make such mistakes unlikely by using a rather conservative interpretation of sensed data.

3.2 Communication

As communication is crucial for interaction in both person-to-person and group settings, there has to be support for synchronous and asynchronous communication in both cases. Going beyond the features of conventional instant messengers, there has to be context-aware support for synchronous communication in groups and means for leaving contextualized messages for other users (asynchronous communication).

Especially the possibility to leave messages bounded to a certain position (location-aware asynchronous communication) has to be provided, as this is considered a very important feature in literature (cf. chapter 2.4).

3.3 Usability

The usability of the system is the most critical success factor for GISS. Users should not have to learn new interaction paradigms to get along with the system and should be able to control it intuitively and comfortable.

Fast reactions to user inputs, predictability and fault tolerance have to be taken into account as well as stability on both server and client side. Usability is more than just a nice user interface and has to be treated accordingly to its importance for the success of the system.

3.4 Extensibility

As technologies change faster than ever before in these days, it is crucial to design systems in a way that they can be adapted and extended to make use of new available technologies or infrastructure. There are two main fields that have to be taken into account, when talking about extensibility.

First, extensibility on backend side, at the sensor layer, is important to have the chance to take into account further context-information as it becomes available through additional sensors.

Furthermore, extensibility has also to be provided on frontend side. This includes not only the possibility to adapt current user-interfaces to changed circumstances but also – and mainly – the possibility to add completely new user-interface-modules, which either may provide a different view on the available data or enable the usage of the system on another platform.

3.5 Scalability

The presented system is intended to be used on university campus with possibly several hundred users logged in concurrently. For this reason, it is crucial for the system to be able to cope with this amount of users or at least to be easily extendable when it becomes necessary.

The modules on the server have to be designed with scalability in mind – using data-structures and algorithms that are also eligible for heavy user load. Furthermore the single components should be coupled in a way that makes it possible to distribute them over several physically separated servers, if computational power of one server is not enough anymore.

3.6 Access

Getting access to the system should not involve too much effort for the possible users. This includes software that is necessary to be installed before using the system as well as the organisational effort one has to make when entering the system the first time (like registering accounts or configuring connections).

External components should only be used where absolutely necessary for the operation of the system. Configuration effort has to be reduced to a minimum; especially requesting the user to deal with network settings has to be avoided.

3.7 Client Resources

As the usage of the presented system only makes sense in the case where many clients are available most of the time, it is crucial to lower the usage of client resources as much as possible and so make it attractive for users to keep the system up and running most of the time.

The usage of client resources has to be as low as possible, where the most relevant factors are the usage of CPU- and network-resources. This leads to problems, as lower CPU-utilization leads to higher network-traffic, because more data has to be transmitted to the server, where the calculation takes place, and vice-versa. To solve this issue, a low CPU-utilization was considered more important, so that the design should be focused to meet this criterion.

3.8 Privacy

In an environment, where personal data about a user is collected and processed automatically, there is high demand to think about privacy issues and to protect one's vital interests to have control about which personal data is publicly available.

There are two issues that have to be considered, when talking about privacy. On the one hand, the user has to have control about the collection of data itself (which information is collected and even if any data is collected at all), on the other hand there is also demand for giving the user the possibility to control the accessibility of his private data by others (who has access to which kind of data).

4 Architecture

The main subject of this chapter is to present the architectural design of the application and the foundations that have led to the chosen architecture. Furthermore, sensing and representing context information will be a topic as well as the services that can be provided utilizing the sensed context.

4.1 Overview

GISS mainly consists of two parts that have been implemented in two separate diploma theses. The positioning component that can be seen in the upper part of Figure 12 is subject of [Holz04] and provides technology-independent information about a user's location. The lower part of Figure 12 shows the components that are subject of this work and provide users with a context-aware group-interaction.

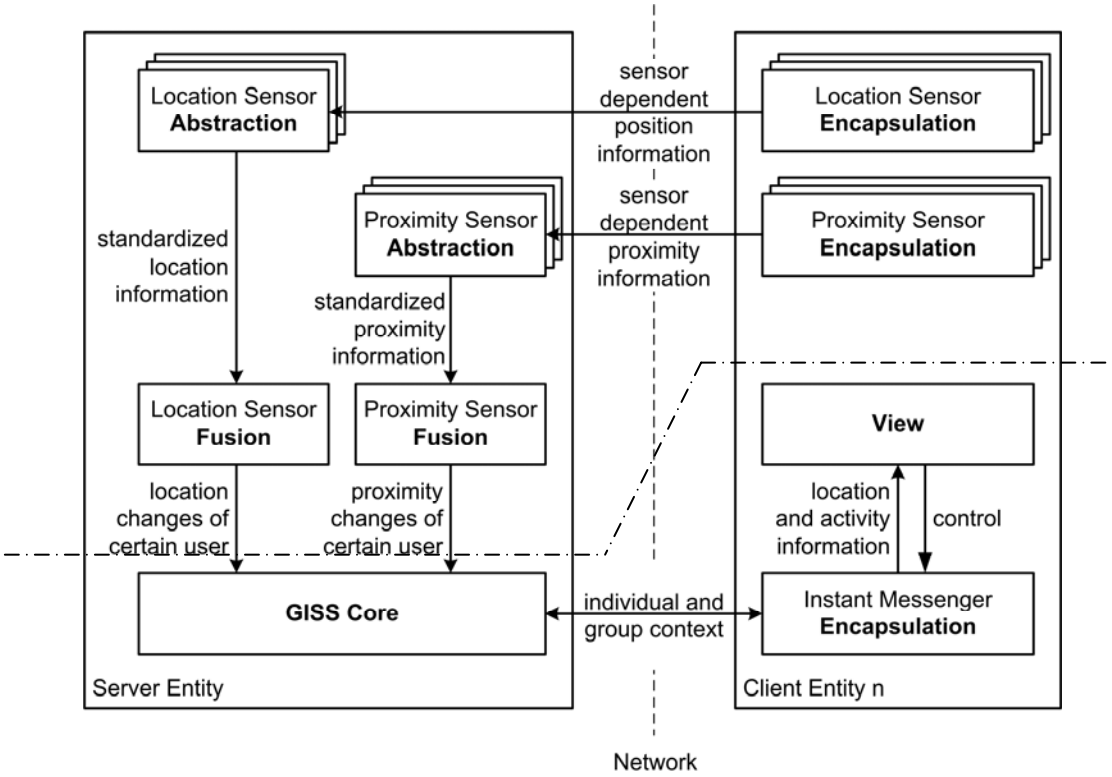


Figure 12: GISS architectural overview

As can be seen, the architecture chosen for this work is client/server based. This approach has been chosen to be able to provide a lightweight client-component that occupies as few resources as possible on a user's host. The whole system has been built upon the SiLiCon-Context-Framework, which provides means for modular, extensible and scalable design as well as network-transparent communication-features (cf. chapter 4.2).

On the server-side, the main component is *GISS Core*, which serves as the central coordinator for the whole application as well as the interface to the information provided by the positioning subsystem.

On the client side, the main module is *Instant Messenger Encapsulation*, which on the one hand serves as the interface to the server side and on the other hand handles communication with the external user interface (which is based on an Instant Messenger, cf. chapter 4.4). The *View*-Module is an exchangeable component, which is intended to provide the user with sophisticated location awareness (cf. chapter 4.5.2).

The individual context data of a user is sensed on client side. Interpretation of raw context data basically is done on server side in order to derive high-level context information (except for activity context, as we will see in chapter 4.3.3). For this reason, the raw context data has to be transmitted onto the server, where it is processed and also stored for possible later retrieval. In addition to the individual user context-information the context-information of the available groups is determined (cf. chapter 4.5.3 and 4.5.6). The derived high-level context information of both the single user and also the groups, the user is member in, is then transmitted back to the clients, where according services are triggered (cf. chapter 4.5).

4.2 The SiLiCon-Context-Framework

The *SiLiCon-Context-Framework* [Beer03] is a development of the Institut für Pervasive Computing in cooperation with Siemens Munich CT-SE 2, implemented in the years 2002 and 2003. The main focus in this work was to provide a stable and easy to use software middleware, in order to integrate context information into already existing application fields.

In this work, the *SiLiCon-Context-Framework* has served exactly as the middleware it was intended to be. Taking advantage of the event-based communication features it provides, a highly modular, flexible and distributed system design was easy to achieve.

4.2.1 Entities and Attributes

To describe whole context scenarios, it is necessary to create a description of all entities that are possibly important for a scenario. In the *SiLiCon-Context-Framework* a dynamic environment object is represented through a *context entity*, or short *entity*. From a conceptual point of view, an entity is a collection of *attributes*. An attribute is a software component that expresses one specific aspect of an entity in order to be able to provide services, which are related to this specific aspect. Every entity contains a set of attributes that are used to describe an object inside a certain scenario. These objects could model human users as well as mobile

digital devices, or non-digital objects, like places, rooms or furniture. Entities express their information and capabilities through the use of attributes that can be loaded into the entity.

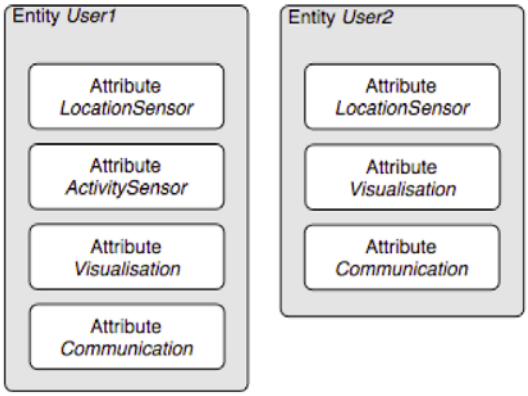


Figure 13: SiLiCon-Context-Framework Entities and Attributes

In Figure 13, an example of two entities presenting the digital counterpart of a user in an application is shown. The users are described by their attributes, which represent their capabilities. *User1* is capable of sensing the location and the current activity status of the person it represents, to visualize some kind of information and to communicate with others. *User2* has similar properties, but is not able to determine the current activity status. As the properties of an entity may change dynamically over time (e.g. if the person represented by *User2* plugs in a sensor that is able to sense his current activity status), the framework provides means for dynamically loading and unloading attributes.

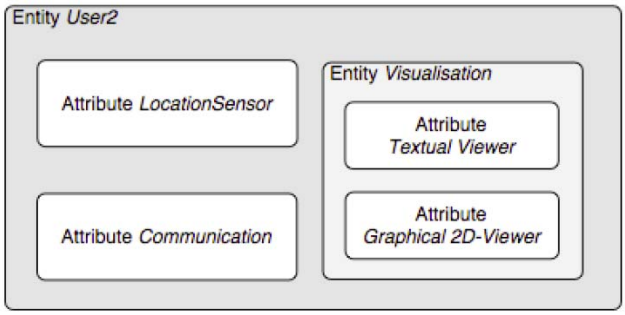


Figure 14: Recursive structure of Entities

In some cases, it may be useful to specify certain properties at a finer granularity. For this reasons, an entity may contain further entities that themselves are described by attributes. As can be seen in Figure 14, the visualisation component of *User2* is now capable of showing position information in a text-based view as well as in a graphical two-dimension view. The possibility of recursively embedding entities into other entities provides a powerful means of structuring application services in a modular and flexible way.

4.2.2 Rule-based Communication

To react to the change of an entity's context, each entity has its own set of context rules. Every attribute that senses a change of context in the aspect it covers, issues an according event, which consists of an event-name and arbitrary parameters. An example for an event that could be thrown by the position sensor attribute, when a user has changed his location is

```
Sensor.positionChanged(string newPos);
```

In this case, *positionChanged* is the name of the event, which carries one additional parameter – *newPos* – that determines the position the user has changed to. If the sensor could also determine the accuracy it provides at the moment, the event could look like

```
Sensor.positionChanged(string newPos, int accuracy);
```

This event, which describes a change of the user's context, may be used to trigger reactions to this change of context. This is described by a *rule*. If the new position should be displayed on a visualisation component, the rule would look like

```
on Sensor.positionChanged(string newPos, int accuracy) {
      Visualisation.showPosition(newPos, accuracy);
}
```

In the given case, the *Visualisation*-attribute is instructed to show the position given in *newPos*. Additionally it is provided with information about the *accuracy* of the position. It now could make sense not to display a position, whose accuracy is to low. To realize this constraint, the rule may contain a conditional clause:

```
on Sensor.positionChanged(string newPos, int accuracy) {
      if (accuracy > 5) {
            Visualisation.showPosition(newPos, accuracy);
      }
}
```

Here, the *Visualisation*-attribute is only notified, if the *accuracy* is greater than 5. Three parts of a rule can be identified. First, an *event* has to occur, which triggers the evaluation of a *condition*. If the condition is true, an *action* is triggered. Accordingly, the rules used here are called *ECA-rules*.

Often, events should not only trigger actions in the same entity but also in other entities, even if those other entities reside on another host in the network. To use the example setting again, maybe the change of a user's position should be displayed by another user's visualisation-attribute. For this reason, events can also be routed to different entities, where the network is

fully transparent for the application designer, i.e. local and remote entities are treated all the same, when writing a rule:

```
rules for User1 {
      on Sensor.positionChanged(string newPos) {
            User2.Visualisation.showPosition(newPos);
      }
}
```

In this case, the rules of *User1* determine, that in case of *positionChanged*-event, the *Visualisation*-attribute of *User2* should be notified. Sometimes it may be necessary to alter the defined rules. If *User1* decides, that for privacy-reasons, *User2* should not be notified about his current position anymore, the according rule has to be removed. This can be realized through *dynamic attribute loading*. With this mechanism, rules can be created, altered and removed dynamically at runtime.

The concept of rule-based notification of events provides a powerful means of dynamic communication between entities. Especially the feature of network-transparent triggering of events at remote entities is used extensively in the presented work.

4.2.3 Architecture

Figure 15 shows an overview of how the described concept is implemented in the framework. The basic object on each host is a *ContainerEntity*, which contains all entities that reside on this host. Furthermore it contains modules for processing events (*EventInterpreter*, *EventQueue*), for logging and for discovery of other hosts and the entities they contain as well as for communication with those other host via network.

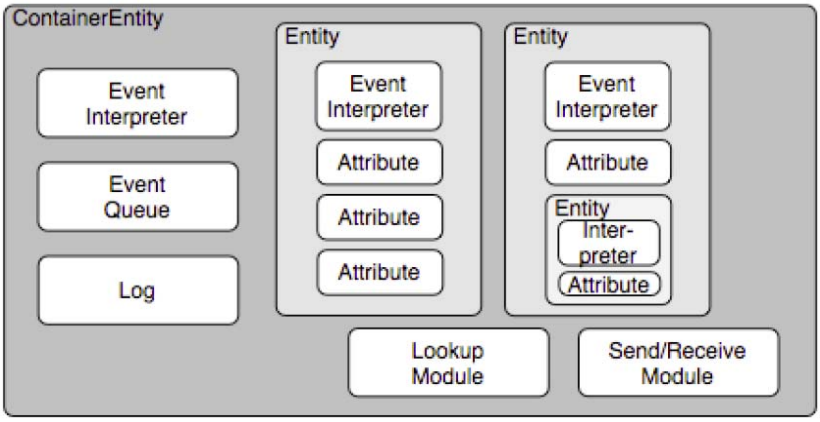


Figure 15: SiLiCon-Context-Framework architectural concept

As each entity has it's own set of rules, it contains an *EventInterpreter*, that checks for local applicable rules, when an event occurs. If no rule can be applied, the event is passed to the next outer level, at which the responsible *EventInterpreter* checks for applicable rules.

Whenever a rule is applied and the specified destination cannot be found on the local host, the *Send/Receive-Module* is used to deliver the event to the remote host, where the destination entity resides. When the destination entity cannot be found on the entire network, the event is dismissed.

The network layer of the framework in the current implementation works with IP-multicasts or -broadcast to discover remote hosts and their entities (in the *Lookup-Module*). Delivery of events across the network is configurable and can be carried out via a proprietary protocol or via standard SOAP-Calls (which is used in this application).

4.3 Sensing and representing Context

Before an application is able to act context-aware, it has to retrieve information about the current context from various sources. In this chapter, the chosen context dimensions for this work are presented, where special attention is given to sensing and representing context information.

4.3.1 Used Context-Dimensions

As already described in chapter 1.3.2, there are four primary types of context information that can be identified (*location*, *time*, *identity*, *activity*). At design time, it each aspect has been taken into account in order to be able to provide useful set of services and to have the possibility to derive secondary context information if needed. Furthermore, as this application is intended to support interaction in groups, it is necessary to have knowledge about the context of the group itself. This group-context is a secondary context-type and is derived from the contexts of the individual group-members.

It is obvious, that context can be sensed in different qualities, providing a finer or coarser granularity of the information sensed. In this application, the finest granularity is available in the context dimensions *location* and *time. Location* is considered the most relevant and important type of context information in literature [Chen00], so that the development of a location sensing system was subject of a related diploma thesis [Holz04]. As *time* is obviously an important context dimension too, as it enables a manifold of services dependent on the current time and furthermore is fairly easy to sense, it is also used extensively in *GISS.*

Identity and *activity* are sensed with a coarser granularity, thus demonstrating, that the use of context other than location and time makes sense and provides a real surplus value. A further enhancement of services depending especially on *activity*-context is surely possible by pro-

viding this kind of information with finer granularity, which can be achieved fairly easy by adding further sensors to the extensible architecture of *GISS.*

4.3.2 Sensing and representing Location and Proximity

As already stated before, *location* is the most prominent context information in our application (as it is in most of the other known context-aware systems). The process of sensing location is subject of a related diploma thesis [Holz04] developed at the Institute fuer Pervasive Computing at the University of Linz.

For this reason, the chosen approach for sensing location and proximity will only presented in an overview, whereas the chosen representation of location information will be presented more in detail due to its relevance for this work.

Location context is represented by so-called symbolic locations that are independent of the technology used to sense the location. Symbolic locations, in contrast to geometric locations, do not contain any geographical data like longitude or latitude. In point of fact, the only thing they do is a mapping between a certain location and its identifier. Symbolic locations are used whenever the used sensors are not able to provide geographical position information (like it is the case when using for example beacon-based systems in contrast to GPS).

The area in which positioning should take place, is represented by a hierarchical structure of symbolic locations. This structure is tree-based and corresponds to a set of *contains*-relations (as can be seen in Figure 16). Every location in this tree has a certain tag, which carries meta-information about the granularity and type of the tagged location. Four granularity levels are distinguished. Those levels have generic names to allow flexible use of this model. In this application, as mainly indoor settings are relevant, the levels are interpreted as follows:

- *Area:* is used to tag a room or a room-equivalent area (e.g. a part of a long corridor)
- *Level:* is used to tag floors (collections of room-equivalent areas, that reside on the same level)
- *Section:* is used to tag buildings (representing building-equivalent physical units)
- *Region:* is used to group multiple spatially related sections (like buildings on a campus)

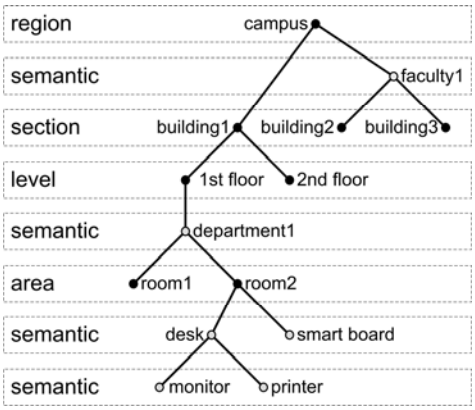


Figure 16: Symbolic location containment hierarchy [Holz04]

In addition, a fifth type of location level has been introduced to allow finer specification of the hierarchy. This fifth type – *semantic* – is used in two ways:

- to define locations below *area*-granularity (e.g. if a big room should be split up in several sub-areas or if a office should be divided into its occupants' workspaces)
- to group other locations on arbitrary levels (e.g. to define the area covered by a department or to group the buildings belonging to a faculty)

With this interpretation of the location model, a manifold of services is enabled. The utilisation of the embedded meta-information is described in chapter 4.5.2.

Determining of a user's location can be carried out via arbitrary sensors. In the presented system, sensors for wireless LANs, Bluetooth, RFID and for IP-subnets have been implemented. Every sensor delivers *raw context data* (e.g. MAC-addresses of reachable Bluetooth-devices or current IP-address of the user's host) to the server, where this raw context data is mapped onto symbolic locations, thus creating *high-level context information*.

As it is possible for a user to deploy several sensors, which could provide symbolic locations on different granularity levels, more than one location at a time (if sufficient accuracy cannot be reached) or even locations that are contradicting, there has to be a mechanism to join the different location context information in order to get the most likely location of the user. This is done by the *LocationSensorFusion*-attribute (cf. chapter 4.1).

Proximity to other users is also sensed by the system described in [Holz04]. Proximity information is also abstracted from the technology used to gather this information. In the present case, proximity is sensed through a Bluetooth-Sensor (proximity is defined by reachable Bluetooth device) and through a location-based sensor (proximity is defined by being in the same symbolic location at a configurable level, most likely *area* in this case). Both sources are merged in a *ProximitySensorFusion*-attribute accordingly.

4.3.3 Sensing and representing other Individual Context Information

As stated before, there are other dimensions of context besides location that are used in this application. In this chapter, the process of sensing *time*, *identity* and *activity* is described as well as the respective form of representation.

Obviously, *time* can be derived fairly easy from the computer's real time clock. Time is sensed in two ways. On the one hand, the current time is retrieved event-triggered, whenever a change of another type of context-information is recognized and the current time is needed to perform the evaluation of possible services. On the other hand, time is also sensed periodically (time-driven) every minute to enable triggering of services, which are solely dependent on the current time. Time is sensed in milliseconds, which also is the internal representation of this type of context. Although time is only needed in minute-granularity in this application to trigger services, it has been chosen to store and process time with the full available resolution in order to preserve generality and extensibility.

The other two context-dimensions are sensed through the user-interface of *GISS.* As can be seen in chapter 4.1, the user interface is based on a standard instant messenger application (the motivation for this decision will be given in chapter 4.4). For this application, information about the user's *identity* and to a certain amount also about his *activity* can be sensed.

Identity in this application is represented by the unique number that is also used by the instant messenger to identify the user (the so-called *UIN – Universal Internet Number* – in case of ICQ). As this number is inappropriate for users to identify other persons, a mapping between the UIN and the real name has to be provided. In our application, this mapping is not stored statically and centrally on the server but dynamically on the clients. This gives users the possibility the name their communication-partners in a way that is convenient and understandable for them. For example one might see the full name of a person, while others favour to see the nickname only. This local mapping of numbers and names is retrieved from to user's contact list in the instant messenger (cf. chapter 4.4.3) in order to provide a consistent usage of mapping throughout the whole communication system.

Sensing of *activity* is only used in a very coarse manner in this application, although it provides enough information to significantly improve the identification and adaptation of the services relevant in a certain context.

Activity is derived from the availability state of a user's instant messenger account. This state is set either manually (explicit context gathering) or automatically by the messenger application (implicit context gathering) when there has been a change in the utilisation of the com-

puter (e.g. no typing and mouse activity for a certain amount of time results in status automatically set to *away*). The distinction of activity context is limited to the following states in this approach:

- *Available:* User is online and ready for conversation
- *Offline:* User is offline (messenger application is not running)
- *Away:* User is currently not on his computer
- *N/A:* User is not available for a longer time
- *Busy (Occupied):* User is busy with another task
- *Free for chat:* User is willing to accept chat requests
- *Do not disturb:* User does not want to be disturbed
- *Invisible:* User is online but generally invisible to others (cf. to be offline) except for those explicitly added to a visibility list

As can be seen, these states provide a rather coarse classification of the user's current activity. The automatic state adaptation carried out by the messenger application is only based on utilisation of the computer and does not take into account external information like meeting situations, incoming or outgoing phone calls or similar situations. Only an explicit setting of the state by the user himself would provide more accurate activity context but is hardly ever done by the users and cannot be distinguished from an implicit state change (done by the messenger application) from *GISS*' point of view.

The current activity state is mapped to a unique identification number and also stored locally. In conformity to this current state, incoming and outgoing events are either forwarded or disposed (cf. chapter 4.5 for a detailed description of the use of this context dimension).

4.3.4 Deriving Group-Context

As already stated in chapter 4.3.1, the context of a group can be derived from the individual contexts of its members. As can be seen in Figure 17, this can only be done by taking into account additional knowledge; an obvious example would be a group's membership list.

This additional knowledge can be gathered either automatically or manually. Manual gathering has do be done by explicit user input, like joining or leaving a group (cf. chapter 4.5.3). Automatic gathering of knowledge about existing groups can be derived from the current group context itself. An example for this would be a recognized gathering of a certain group's members, which changes the context of the group and also implies different interpretation of a

individual member context, thus being a part of the knowledge to be taken into account when aggregating individual contexts to the group's context.

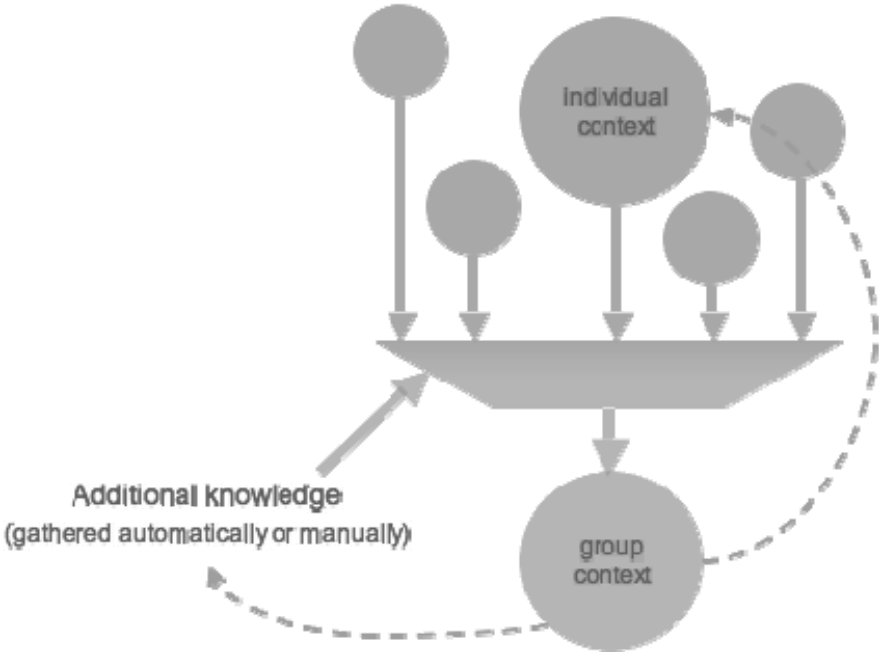


Figure 17: Gathering group context

The information considered most important about a group's context is its *gathering status*. A gathering is an informal, unplanned meeting of group members without agenda. As it is hard to distinguish between planned and unplanned meetings taking into account the available individual context only, in this application no distinction between a gathering and a meeting is made. This *gathering status* contains statements about the following issues:

- Is a gathering currently in progress?
- For each recognized gathering:
    - When did it start?
    - When did it end?
    - Who joined (at which time), who was absent and who left earlier?
    - Where did it take place?

The *gathering status* is the only information currently derived for existing groups from the group members' individual contexts. Furthermore, the formation of dynamic groups (cf. chapter 4.5.3) is recognized by spatial proximity and the users' current availability status.

4.4 User-Interface Design

As can already be seen in the requirements analysis (chapter 3), a proper design of user interface is a crucial issue in terms of a broad acceptance. Furthermore, as already stated in chapter 4.3.3, the user interface is also utilized as a source of context, as identity of the user and – to a certain amount – the activity of the user can be sensed.

Several approaches have been evaluated in terms of the specified requirements (especially usability, access and context awareness), before the decision for the now chosen implementation presented in chapters 4.4.2 and 4.4.3 has been made. The process of evaluation and the arguments for and against the considered approaches are subject of this chapter.

4.4.1 Design studies

The first decision that had to be made was the one between designing a user interface from scratch without integrating existing systems or taking an existing interface and extending it by the functionality *GISS* provides.

For high user acceptance, as few as proprietary application software as possible should have had to be installed, a new user-interface had to be web-based, using applet technology for the application logic (cf. Figure 18, left and below the VRML-visualisation). As a means for synchronous communication is a crucial part of the system (cf. chapter 3.2), it would be obvious to extend an instant messenger interface, as possible users are in general used to this sort of application and a part of functionality could be covered by already existing services (cf. Figure 19).

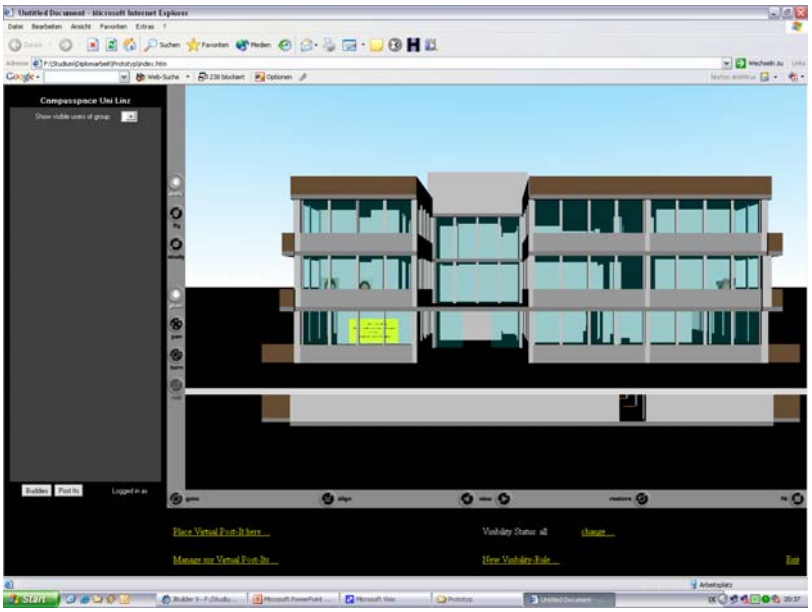


Figure 18: Proprietary user interface (design study)

A seamless integration of all context-aware services could have been reached only by designing a user interface from scratch. This approach would have given all freedom to adapt the interface to the current context as well as to integrate arbitrary services and visualisation components (like the CampusSpace-model depicted in Figure 18, see chapter 2.5.2). Anyhow, the second approach has been chosen due to the following reasons:

- Most of the possible users are used to an instant messenger interface, thus lowering the entrance hurdles for new users.
- It is possible to utilize an existing messenger account to identify the users, making it unnecessary to register another account and further enhancing acceptance by easy access.
- Existing features of the instant messenger network can be used and do not have to be reimplemented (possibly not being able to provide the functionality and ease of use of the original).
- The interface of instant messengers has been used by millions of users already and is broadly accepted by them. The (more or less common) design of such applications has reached a level of maturity, an interface designed from scratch without intense evaluation could never reach.
- The originally intended use of the 3D-VRML-model of the campus has shown to be unsuitable for this application, so that a browser based solution was not mandatory, because no VRML-browser has been needed anymore. The unsuitability basically results from two reasons. The first is the lack of up-to-date and stable programming interfaces for external access to the VRML-browser and the data displayed. The second – and more important – reason is the lack of proper navigation concepts for realizing a location visualisation system utilising the VRML-model only. It surely provides a good feeling of a region's real geography when navigating through a model of a real world, but experiences show that it is very likely to loose one's orientation sooner or later. Furthermore, providing a quick overview over a floor and the people currently sojourning there can be better realized with a two-dimensional view, because no problems with concealments of avatars by walls can occur there.

Figure 19: Example for a possibly suitable existing user interface

After having decided for extending an existing user interface by adding functionality to existing applications, the question is, how this extension should be realized. There are again several possibilities, which are subject of the next chapter.

4.4.2 How to extend existing Interfaces

When using an existing application as an interface, which has to be kept fully functional, ways have to be found to integrate the additional services into this interface without vitiating the original functionality. When designing *GISS*, again two approaches have been evaluated in terms of technical realisation and possibility of seamless integration of the provided services.

The first approach is based on a proxy solution, where an attribute in the context framework is designed to work as a proxy between an arbitrary, unmodified instant-messenger-client (in this case ICQ-Client) and the messenger network, namely the messaging-server, which is used in advanced versions of the ICQ-protocol OSCAR [Osca04]. This proxy-solution implies direct modifications in the communication-stream between messaging-client and -server, which is based on the binary, byte-oriented OSCAR-protocol created by ICQ-owner AOL. The resulting design can be seen in Figure 20. Using this approach, the services of *GISS* are realised by augmenting the original messenger-protocol-data with the context-information delivered by the *GISS*-application. On the other hand, user-commands for *GISS* or context-information is gathered by intercepting and analysing protocol-packets of both client and server side before forwarding them to the original recipient.

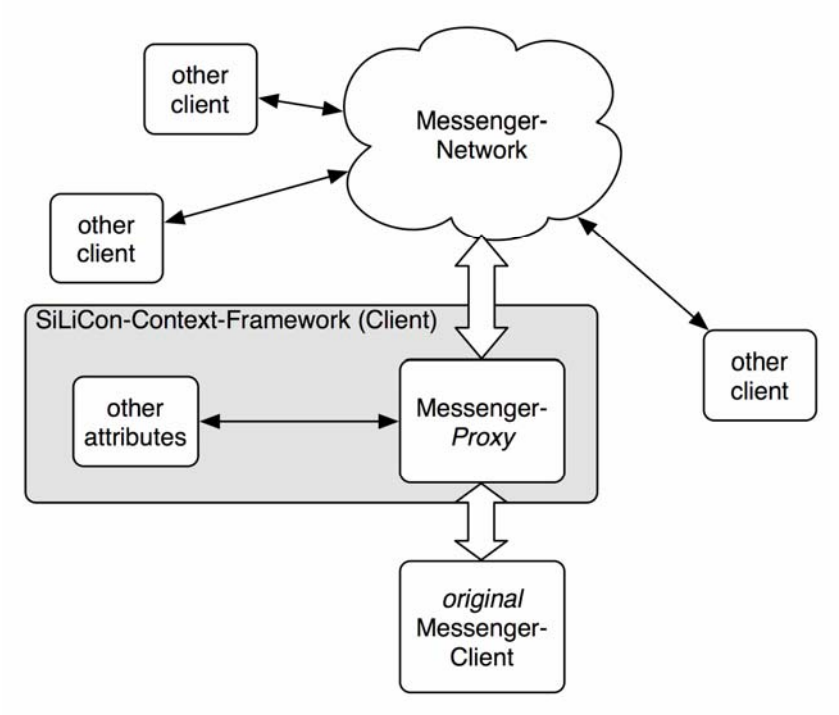


Figure 20: Interface using an original messenger-client

The biggest advantage of this approach is the possible use of unmodified, original messenger-clients "off the shelf", which nearly everybody has already installed on his computer today. Being minimal invasive, this approach completely fulfils the requirements of *usability* and *access* and could provide *GISS*-services without even noticing them at first sight. However, this approach also brings severe disadvantages. The maybe biggest problem is that OSCAR is a closed, undocumented protocol, which has only been reengineered by the open-source-community to be able to provide access to the messenger networks using this protocol. Therefore the specification is subject to change without notification or documentation by AOL. During prototypical implementations for the evaluation of this approach, such an unforeseeable modification occurred and caused major changes in the program code to work again. A second disadvantage is linked with the restricted possibilities of controlling the messenger client (the user interface) through the protocol. Needed features like changing contact-list-entries or the availability state of the messenger could not be realised through the OSCAR-protocol, causing the necessity of workarounds, which had very negative impact on *usability* and seamless integration of *GISS* services.

The second approach evaluated is based on a modified open-source instant messenger. In this case, the connection of the client to the messenger network is not affected and is used unmodified in the way it was intended to be.

To realize the *GISS*-functionality, some sort of extension mechanism has to be used to access and modify the data of the instant messenger as well as sense context from the *GISS* program modules. In order to provide largely seamless integration into the existing user-interface, it has been slightly extended to provide controls for the *GISS* services. The resulting application design can be seen in Figure 21.

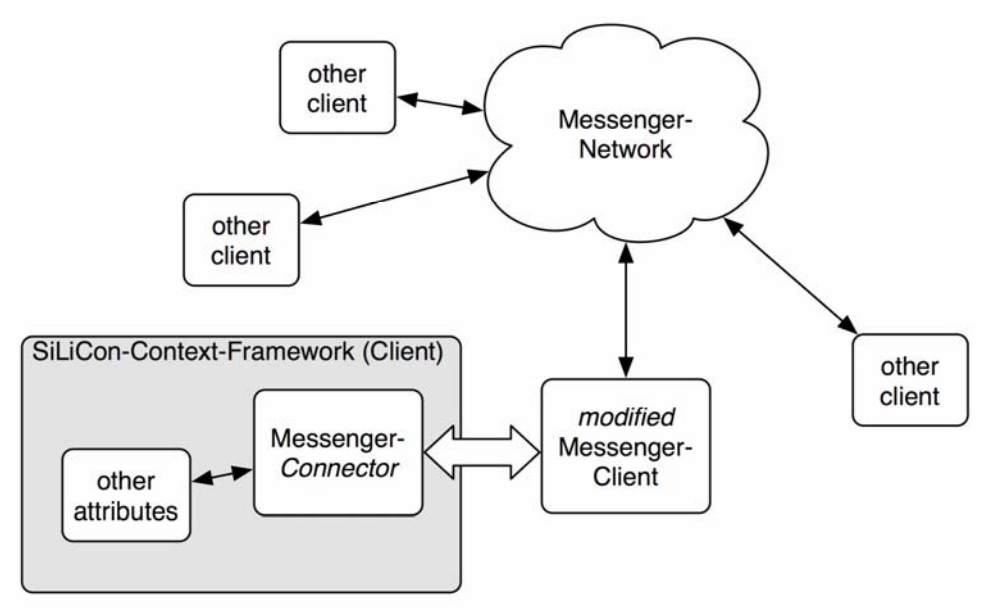


Figure 21: Interface using a modified messenger-client

Like before, this approach also has several advantages and disadvantages. To start with the disadvantages, *GISS* cannot be used with an arbitrary messenger-client as an interface anymore but has to be controlled via a modified, predefined client, thus having negative impact on the requirement of easy *access*. This also includes the impossibility of simply exchanging the messenger-client when a newer version becomes available, thus being not able to easily making use of new services, the messenger-network may provide. Furthermore, there is an additional open connection to connect the messenger to the context-framework (although only local), possibly causing problems with security and *privacy*.

The advantages of this approach clearly can be found in seamless integration of the *GISS*-services. Furthermore, when prototypically implementing this approach for evaluation, it has been found, that there are no major restrictions of what could be manipulated on the user interface. Also taking into account the independence from possible protocol changes, it has been decided to use this approach as a user interface for *GISS*.

4.4.3 Introducing SIM – Simple Instant Messenger

Now having decided for the approach to modify an existing messenger to be used as an interface for *GISS*, an open, extensible and possibly well-documented system had to be found to serve as a foundation for the context-aware services implemented in this work.

A suitable system has been found with *SIM* (Simple Instant Messenger), an open-source-project, which has the following advantages and limitations:

- Fully implemented in C++
- Cross-platform-compatibility (executable on Windows, Linux and MacOS X) based on the QT-library [Trol04]
- Support for different messaging protocols (including ICQ, AIM, MSN, Jabber, etc.)
- Modular, extensible architecture based on a plug-in-concept
- Possibility to control the messenger through a *remote-control*-plug-in
- Incomplete documentation of both architecture and implementation

SIM has been chosen mainly because of its plug-in-concept, which makes it easily extensible in every needed direction. Furthermore, as already stated in the list, *SIM* comes with a possibility to control its features and functionality from an external application via a remote control interface.

To adapt *SIM* to the needs of the *GISS* user-interface, several changes had to be made. First of all, the remote control interface had to be extended to support several commands, which had not been implemented in the original version (cf. chapter 5.1.2). Furthermore, a means of controlling the *GISS*-features had to be provided. This has been implemented as a plug-in, which is shown a toolbar integrated in the *GISS*-interface. As the remote-control-interface only supports a pull-scheme, i.e. it does not send information unless it is asked for them by the external application, a push-channel had to be implemented to be able to transmit user-commands from the user-interface to the context-framework without continuous polling from the context-framework-side (cf. chapter 5.1.2). A few minor changes that were needed to implement the services of *GISS* are described in chapter 5.2.

Figure 22: User-interface of SIM (with GISS-toolbar)

As can be seen in Figure 22, the user-interface of *SIM* is largely oriented on those of standard instant-messengers like ICQ. The second toolbar that can be seen in the uppermost part of the picture belongs to the implemented plug-in, thus providing the controls for the *GISS*-services.

4.5 Group Support Services

Having presented how to sense which kinds of context information and how to design a suitable user interface, it is time to look at the core functionality *GISS* provides, namely the services that support context-aware interaction in groups. Before describing those services in detail, a short overview is given, where the used types of context information as well as the category of context-aware features (cf. chapter 1.3.4) are identified.

4.5.1 Overview

Looking at the available types of context information, there exists a manifold of services that can be improved or are even enabled by taking into account this information. The focus of this work has been laid on the support of interaction in groups and here mainly on the aspects of communication. With respect to this, the seven services shown in Table 7 have been identified to be important and thus have been implemented using the available context information. As can be seen, the services use different combinations of available context information (*L*ocation, *T*ime, *I*dentity, *A*ctivity and *G*roup) to provide functionality in one or more categories

of context-aware features (context-aware *P*resentation, *A*utomatic execution of services, *T*agging of information for later retrieval).

Table 7: Classification of context-aware services

	L	*T*	*I*	*A*	*G*	*P*	*A*	*T*
location awareness	X		X			X		
forming and managing groups	X		X	X			X	
synchronous group communication			X	X	X		X	
asynchronous group communication	X	X	X	X	X	X		X
availability management	X	X					X	
reminders	X	X	X			X		
group gathering support	X	X	X		X	X	X	

In the following chapters, the services listed above are described in detail from a conceptual point-of-view. The details of implementation are presented in chapter 5.2.

4.5.2 Location Awareness

For interaction in groups which are co-located in a certain area, but do not currently meet, it is important to know about each other's location, as this provides some sort of feeling, what the others are doing (moving around, sitting in their office, etc.), thus improving the emotional bindings between group members.

As sensing a person's location has been the focus of a related thesis [Holz04], providing location awareness by visualizing the own and the others' position is a very prominent service in *GISS*. Currently, two ways of visualizing location information have been implemented. The chosen approaches have been inspired by various related work and provide both an unobtrusive way of keeping aware of others' locations on the one hand and a more room-taking, optional more detailed view on the other hand.

The first way of visualizing location information is based on augmenting a user's contact-list with the contact's current locations and can be seen in Figure 23. The sensed locations are visualised in brackets behind the contact name. This form of visualisation is fairly unobtrusive and kept in a user's peripheral perception most of the time. Due to the limited space in the contact list, a level-of-detail-approach has been implemented to keep this form of visualisation readable and clear. According to a person's location relative to the own location, the visualized location is adapted in terms of providing a more detailed description if the person

is near the own location. For example, if the searched person currently resides on the same floor as the user, the room description in displayed in the contact list. If he resides on a different floor or even in a different building, only the floor description or the building description is displayed. To get an idea of the algorithm that lies behind this, we have to take a look at the used location hierarchy depicted in Figure 16 (page 41). The location displayed is always the one just below the lowest common location node in the path of the person to be displayed. If there is no node below the lowest common location (as may be the case, when both persons reside in the same room), the common location is displayed.

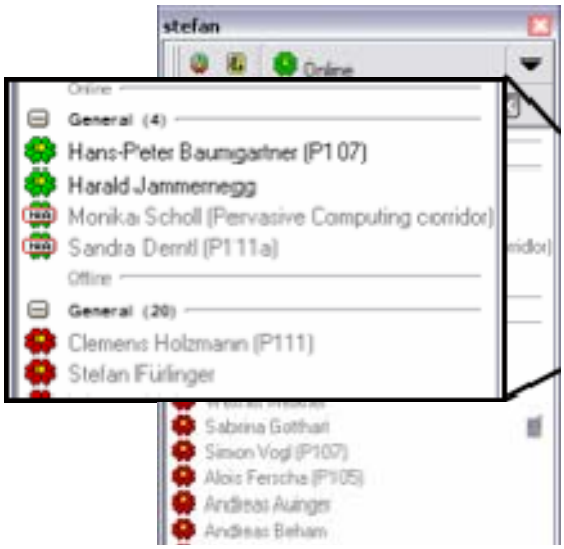


Figure 23: Location awareness via contact-list

In certain cases, the displayed information may not be enough for the user, because he for example wants to know the exact location, even if this person currently resides in a different building. For this reason, the "fully-qualified" location-path is displayed in a tool-tip, which shows up when the mouse is moved over the person's contact list entry. This "fully-qualified" location-path lists all nodes in the location hierarchy, which are above a person's current location and may for example read *P121 @ Institut fuer Pervasive Computing @ 1st floor @ Physics Building @ University of Linz.*

As it sometimes may be hard to keep an eye on several persons concurrently with the approach described above, a second means of visualisation has been implemented. This viewer is based on a 2D-floor-plan, where the own and the others' locations are displayed in manner of ICQ-floating-contacts (cf. Figure 24). Even not as unobtrusive as the first approach, this visualisation provides a quick and intuitive overview of where other persons currently reside. It can be shown and hidden with a simple mouse-click, thus providing quick access to a more

intuitive visualisation, if for example one cannot associate a room number with physical room at the moment.

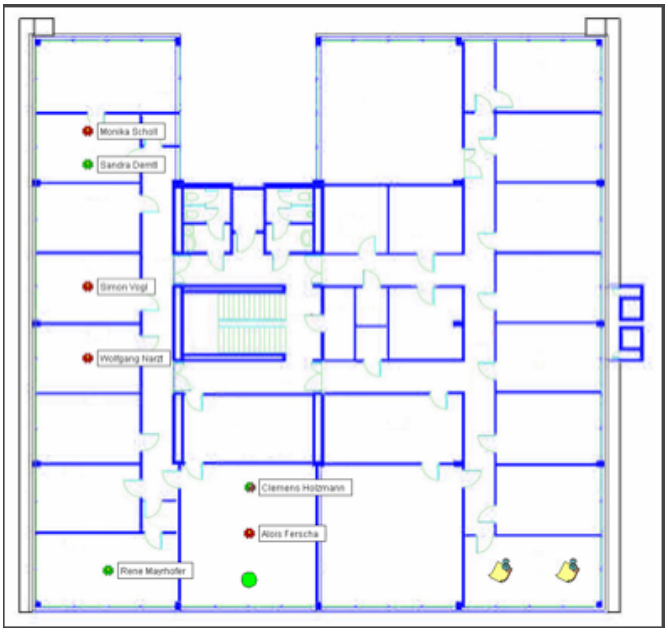


Figure 24: Location awareness via graphical viewer

The graphical viewer by default displays the floor the user currently is located in. The user itself is depicted as spot, as can be seen at the lower border of Figure 24. On demand, the user could zoom out and graphically view locations on building or even campus level, to get an idea of persons' locations that are currently farther away.

If more than one person resides in a room (in floor plan level mode), the contact visualisations are distributed in a tiled manner across the available space. The contact visualisations also include the current availability status coded into the ICQ-flower icon that is used like in the original application for this purpose.

4.5.3 Forming and Managing Groups

In a system intended to support the interaction in groups, there has to be a representation for groups and means to manipulate this representation. Because of the different characteristics and purposes of groups, *GISS* basically distinguishes between two different types of groups – *static* and *dynamic* ones. Static groups are used in case several people want to work on a common task or simply stay connected over a longer period of time. Dynamic groups are formed automatically when people are in spatial proximity and are dismissed again, when the proximity ends.

Static groups are again distinguished in two subtypes, which basically offer the same possibilities of interaction but differ in the way one can join them. In *open static groups*, every-

body can join and leave at any time without authorization process. In contrary, *closed static groups* have an owner that decides who is a member of a closed group. While the first type is suitable for settings like lectures or interest groups, the second type may be more suitable for groups of friends or working groups. As said before, open and closed static groups only differ in the way they are formed but provide the same functionality in terms of interaction possibilities.

Dynamic groups have a completely different purpose. They are mainly intended to enable quick and uncomplicated communication with nearby people. This sort of group does not offer the more sophisticated means of interactions supported by *GISS* (like recognition of gatherings) because they would make no sense in the context of dynamic groups. As already stated before, dynamic groups are defined by spatial proximity of several users. In the simplest case, a dynamic group would be created when people have the same location. This straightforward approach is too simple, as it does not take into account, that there may be locations, where creating a group would not make sense and that – now looking at the hierarchical location model – there may be locations on higher levels, where dynamic groups may be adequat.

For this reasons, the algorithm for building dynamic groups creates dynamic groups in every location, which is marked *suitable for dynamic groups* on an arbitrary level in the location hierarchy whenever more than two persons reside in this location. Locations where creation of a dynamic group does not make sense, are on the one hand those, which cover too large areas (like whole buildings or floors, depending on the use-case) or on the other hand those which are not suitable because of their semantic characteristics (like corridors or toilets). The common case will be the creation of dynamic groups on room level but also – and maybe even more important – on higher level locations that group several rooms (like a department). Also, in case a large room (like a lecture hall) is divided into several sub regions, maybe only one dynamic group should be created with all the people residing in those sub regions as members.

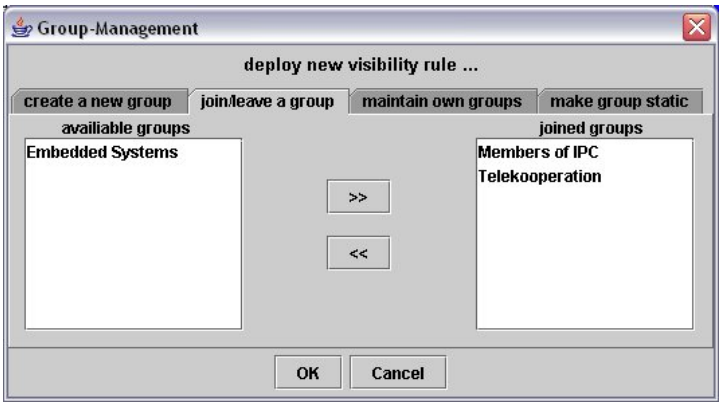


Figure 25: Group management interface (join/leave groups)

GISS offers an interface for the management of groups and individual memberships, which provides means for creating open and closed static groups, for joining and leaving open static groups and for managing memberships of owned closed groups (cf. Figure 25). Furthermore, to support the fast building of static groups, *GISS* supports the transformation of dynamic groups to static groups through this interface. An arbitrary dynamic group can be given a name and is such transformed into an open static group. All the current members of the dynamic group are asked, whether they want to join the newly created static group (cf. Figure 26). This feature for example is helpful in case a lecture is given the first time in a term and the lecturer wants to create a group of all students currently present in the lecture hall.

Figure 26: Confirmation request for joining a static group

As membership in closed static groups can only be altered by the group owner, there is a request mechanism, with which users can ask the owner for permission to join a closed group. As closed groups are often used for private purposes, they are not shown in any public accessible list, so that the user, who wants to join, has to know the group's identification number in order to send his request.

4.5.4 Synchronous Group Communication

Means of synchronous communication as defined in chapter 1.4.2 are mainly intended for spontaneous, informal interaction among people. Today, instant messengers are widely used for this purpose, at least in direct person-to-person communication. Most of today's messengers lack the possibility to easily send a message to a group of people. Although it is normally

possible to send a multi-recipient-message, it is necessary to add every recipient separately and, above all, every recipient has to be in the contact list of the sender.

GISS provides a means to easily and intuitively send messages to groups of people, where every type of group (static and dynamic) presented in the previous chapter is eligible for synchronous communication. Every group a user is member in is shown as an entry in the contact list of the instant messenger. By simply sending a message to this group contact, it is distributed to all members of the group, which are currently online and have not set their status to *busy*.

As can be seen, synchronous group communication – or instant group messaging – is seamlessly integrated in the user interface, simply extending the existing functionality of the instant messenger. In comparison with multi-recipient-messages, the advantages lie in the much lesser effort to send a message to more than one person simultaneously and in the fact, that not all recipients have to be in the contact list or even have to be known (like it might be in lecture settings, where questions could be sent to the whole group).

4.5.5 Asynchronous Group Communication

The support for asynchronous communication in *GISS* is based on means for leaving note-like messages for other users. In contrast to the most common means of asynchronous communication – email – the approach chosen here is based on a post-it metaphor, where messages can be left on arbitrary locations. So a user can leave a message on any location and everybody who is admitted to see this message and passes by, will be shown this *virtual post-it* – a means for *location-aware asynchronous communication* is provided (cf. Figure 27).

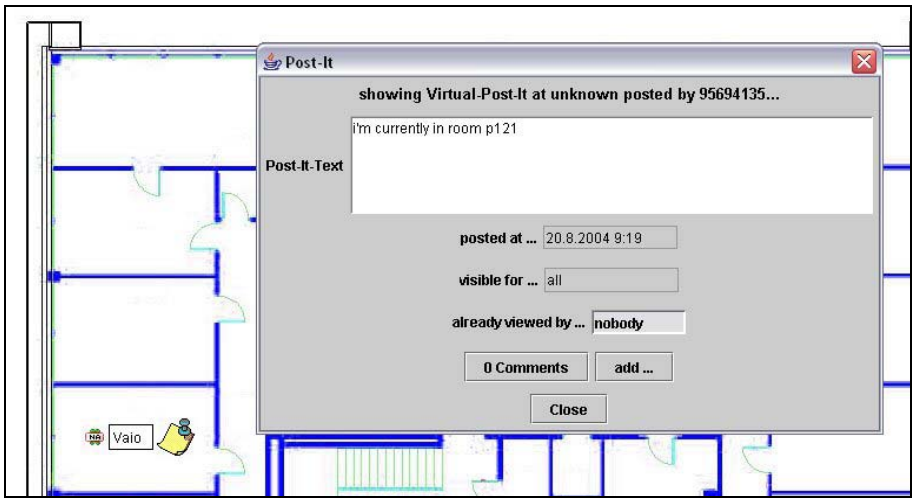


Figure 27: Opened virtual post-it

Virtual post-its can be placed on arbitrary locations, mostly at the position the poster currently resides at. Together with the location, several other pieces of information are stored to provide extended awareness about the post-its properties and its context. Besides the identity of the poster, the time of posting as well as an optional expiration time is stored. Virtual post-its also may have restricted visibility – the user is provided a means to restrict the post-it to be visible for a certain group he is a member in (cf. Figure 29). To be able to track the history of post-its, the identities of former readers and a list of optional comments (including poster and time of comment) are stored. Additionally, there is a field to distinguish "standard" post-its from those, which are used to store meeting minutes of recognized gatherings (cf. chapter 4.5.6), as those have to be handled in a different way in terms of visualisation. The structure of a virtual post-it can also be seen in Figure 28.

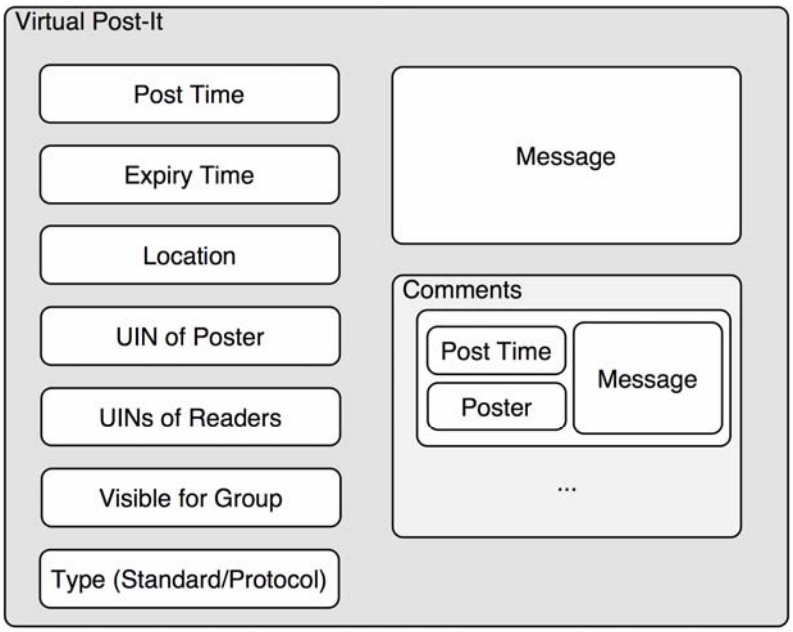


Figure 28: Structure of a Virtual Post-It

Whenever a post-it is placed, it is stored with a link to its location. When anybody enters this location, it is checked whether he is permitted to view the post-it or not (according to his group memberships and the visibility of the virtual post-it) and if the post-it has not expired yet. If the user has not yet watched the post-it and if his availability status is other than *busy*, the post-it will automatically pop-up and show the message as well as a button to open the associated comments.

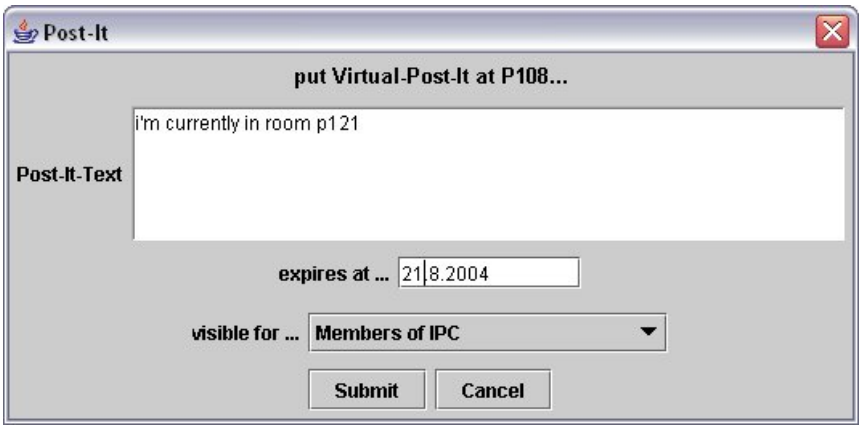


Figure 29: Interface for entering virtual post-its

Virtual post-its can also be opened via the 2D-viewer-interface where they are depicted as blue post-its. When they have already been read, their colour changes to yellow. Via this interface, the user is offered the possibility to access also already read post-its.

An application realised through virtual post-its is the so-called *virtual pin board*. The virtual pin board is a spatially rather small location (maybe a part of a wall), which is solely intended for placing post-its on it. This location is generally sensed via a technology that is dependent on spatial proximity to a reference point (like RFID or Bluetooth), to assure strict spatial limitedness. Standing in proximity to this reference point (residing at the location of the virtual pin board), users can read and post messages and announcements.

4.5.6 Group Gathering Support

GISS offers means to support gatherings of groups. A gathering is an informal, mostly spontaneous meeting without agenda. Formal, planned meetings are not explicitly supported but the available means are also suitable to support this sort of meeting to a certain extent.

Before a gathering can be supported, it first has to be recognized. *GISS* does not require the users to explicitly tell it that a gathering starts or ends, but tries to recognize these events from the current context of a group (cf. Figure 30). This recognition is based on spatial proximity of group members and their availability status. Whenever a certain percentage of available (not *busy*) group members (now chosen to be 66%) reside at the same room (recognition just happens on room level only), a gathering is assumed. When attendance falls below a certain percentage again (chosen to be 33%), the end of the gathering is assumed. This approach has major weaknesses especially for small groups, where gatherings could easily be recognized "accidentally" because of a few people being in proximity unintentionally.

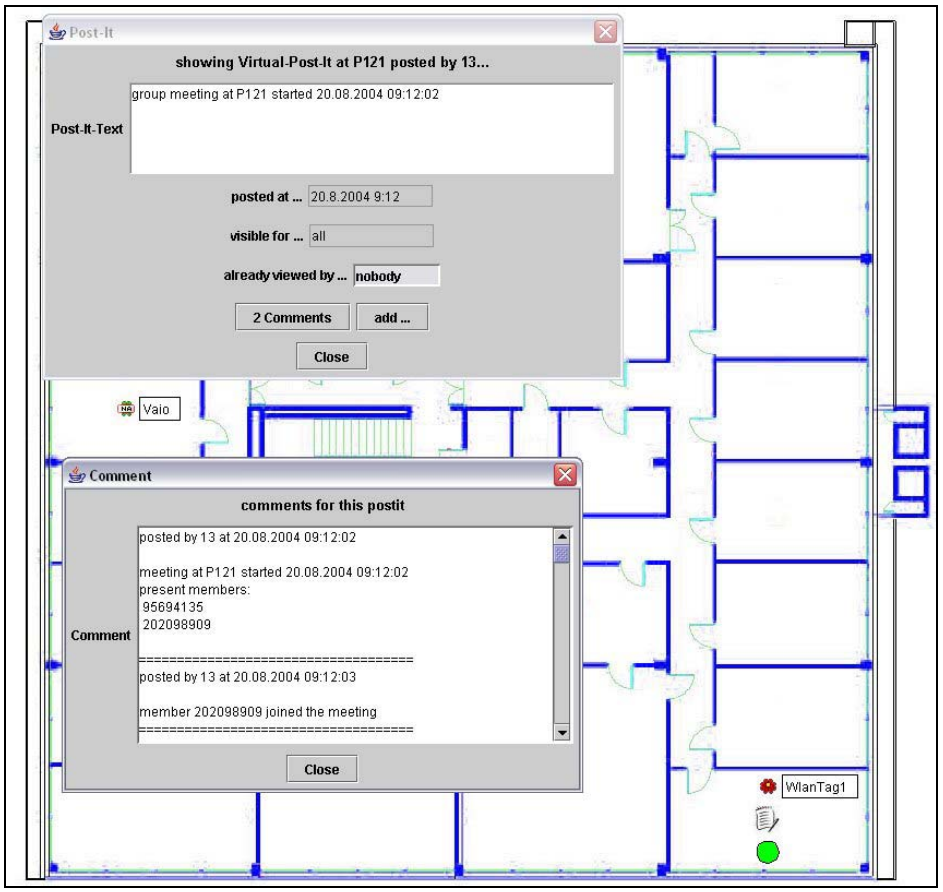


Figure 30: Screenshot for gathering recognized in room in the lower right corner

Whenever a gathering is recognized, several actions are triggered to support the group by carrying out routine work automatically. First, all absent (and not *busy*) group members are sent a message to notify them of the gathering and where it takes place (cf. Figure 31). This allows them to join, if they are nearby or at least contribute by synchronous communication via instant group messaging.

Figure 31: Notification of gathering for absent members

Furthermore, a special *meeting minutes post-it* is created at the location the gathering takes place. This post-it is used by *GISS* to record the start and end time of the gathering, the group members present at the beginning of the gathering as well as people joining and leaving during the gathering. Every of these events is recorded as an entry in the post-it's comment-field, which can be used by the group members to take notes at the same time (cf. Figure 32).

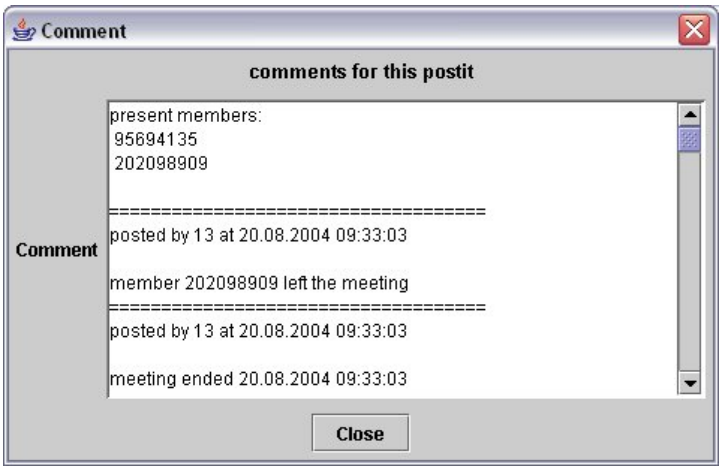


Figure 32: Automatically generated meeting minutes

If the group consequently takes notes during the gathering and enters them into the comment field, at the end a complete *meeting minutes* document including presence-list is available. When the end of the meeting is recognized, these meeting minutes are sent to each available group member (participating as well as absent) automatically to provide an overview of what had happened during the gathering. The protocol post-it – like a standard post-it – is also linked to the location, where the gathering took place and displayed in the 2D-viewer.

4.5.7 Reminders

Reminders provide the user with some kind of context-aware task management. A reminder can be seen as a special kind of virtual post-it, which is only visible for the poster himself and can not only be bound to a location but also to several other criteria, on which it should be shown.

In *GISS*, reminders can be triggered by the following criteria and any logical combination (AND/OR) of them:

- *Location*: show reminder, when user enters the specified location
- *Time*: show reminder, when the specified point in time has passed
- *Proximity*: show reminder, when the specified user is in spatial proximity

As said before, this criteria can also be combined, so that it is possible to set a trigger that shows a reminder, when the user currently is located in the area of a specified department and a specified colleague is nearby (if the user for example only wants to be reminded, if he meets the colleague in his office but not anywhere else on campus, because he needs to discuss some technical documentation only available in the office). For an overview of the structure of a reminder, see Figure 33, the interface for entering a reminder is shown in Figure 34.

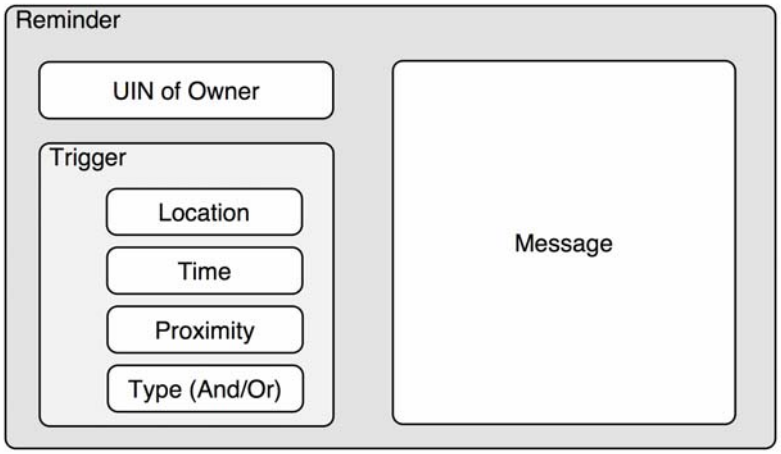


Figure 33: Structure of Reminder

When a time criterion is specified, and the determined point in time had passed while the user had been offline or busy, it is treated to be fulfilled the next time the user becomes available. So, if only the time criterion had been specified, the reminder would show up when going online, and if a combined trigger had been chosen, only the other specified criteria are used to trigger the reminder.

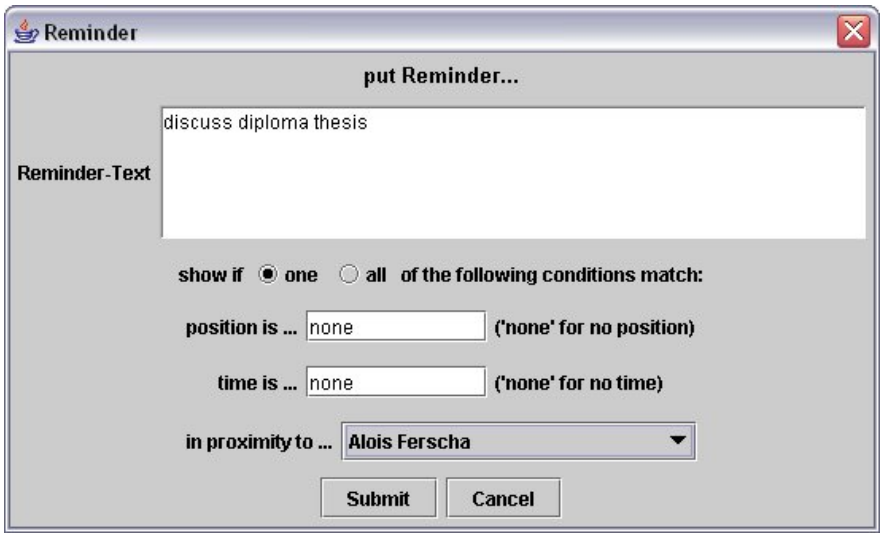


Figure 34: Interface for entering a reminder

When a reminder is triggered and shown, the user is given the opportunity to resubmit the reminder with newly defined criteria to allow him to get reminded later again (when the same criteria meet the next time or any other criterion if fulfilled). When the reminder is not resubmitted, it is dismissed and cannot be shown again.

4.5.8 Availability Management

Instant Messengers provide means for sharing one's availability in terms of setting a publicly visible status to *online, away, busy, offline, etc. GISS* extends this feature by adding support

for context-aware modification of this status. In the group settings *GISS* is intended to support, it would be desirable to set one's availability status individually for each group one is member in. An extension in this way would however have lead to a very sophisticated interface, which would have been hard to handle intuitively and furthermore would have separated the messenger-network-availability-status from the *GISS*-availability-status, thus confusing the user and causing him most likely to not use this feature at all.

To support context-aware modification of his availability status, the user is provided with a means to enter rules that define in which context which availability status should be applied. The context dimensions that can be taken into account here are location and time. As it is fairly likely that a rule should be applied repetitive by time, the user may specify the type of time trigger he wants to use. This type may be one of the following:

- *Once*: the time trigger is only checked once
- *Daily*: the specified time is triggered every day
- *Weekly*: the specified time and day of week are triggered every week
- *Monthly*: the specified time and day of month are triggered every month

Together with the location criterion, the user is provided a powerful means of defining availability rules, for example to set his availability during a lecture to *busy* (where a lecture is defined by the lecture hall where it takes place and it's starting and end time every week). The structure of an availability rule can also be seen in Figure 35.

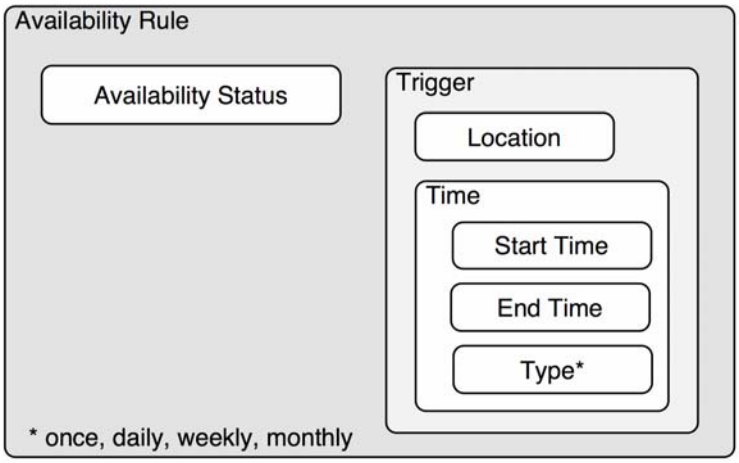


Figure 35: Structure of Availability Rule

To enter the rules, the user is provided with an interface, where he is able to choose the location and time criteria (or only one of them) and the according availability status, which may

be one of the following (in conformity with the states supported by the used messenger network):

- Online
- Offline
- Away
- Extended Away
- Busy
- Do not disturb
- Free for Chat

Furthermore, the user additionally may select *Invisible* so that he appears to be offline to all people except those he has put on his so-called *visibility list* (a feature of the messenger network) (cf. Figure 36).

The modification of rules is not yet supported in the current version of *GISS* but would be carried out via the same interface.

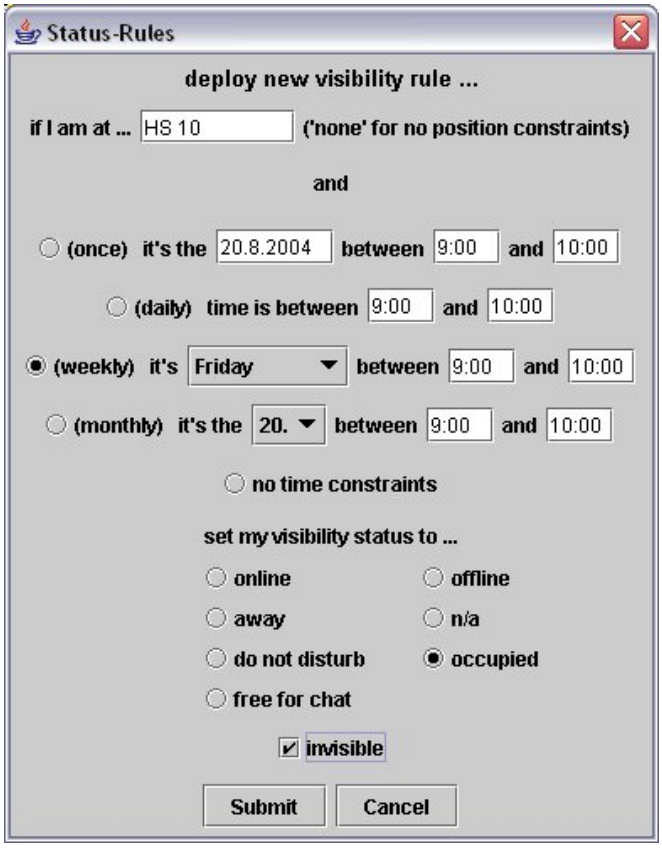


Figure 36: Interface for entering availability rules

Whenever the availability status is modified automatically, the user has the possibility to override this modification by simply choosing a different availability status in his messenger. When the automatic selection is not overridden and the end criterion of the rule has been reached, the status is set back to the one which had been chosen before the rule was applied. If the selection had been overridden, the user-selected status remains unchanged instead of setting it back to the value chosen before the rule was applied, because overriding the automatic selection is an intentional act and thus are considered not to be changed automatically.

5 Implementation Details

In this chapter, the details of implementation will be discussed. The first section deals with the architectural issues of the system, namely how communication between the components is realized and how the application data is kept in a persistent and consistent state. The second section presents the conceptual details of the system and shows, how the services presented in chapter 4.5 are realized.

5.1 Architectural Issues

This section presents the general architectural issues of the *GISS* implementation. It provides information of how the communication between the components is realised and how persistent and consistent storage of application data can be guaranteed.

5.1.1 Communication between Entities

As already stated in chapter 4.2.2, the SiLiCon Context Framework offers transparent communication via a network in terms of providing the entities with a common interface, no matter if the receiving entity located on the same or on the different host. This feature is extensively used in *GISS* to communicate between all kinds of entities. In fact, there is no network communication at all carried out by *GISS* directly.

The communication between entities is carried out via events and rules (cf. chapter 4.2.2), which can only handle the following basic data types:

- String
- Long
- Double
- Boolean

As *GISS* often needs to transmit more complex data structures (like a virtual post-it, that contains several fields of different data type), the classes which encapsulate these data structures have been equipped with serialization- and deserialization-methods, which create CSV-strings (comma-separated-value strings). This concept also works for lists of dynamic length embedded in these data structures, where a separate delimiter is used to identify the fields of the list.

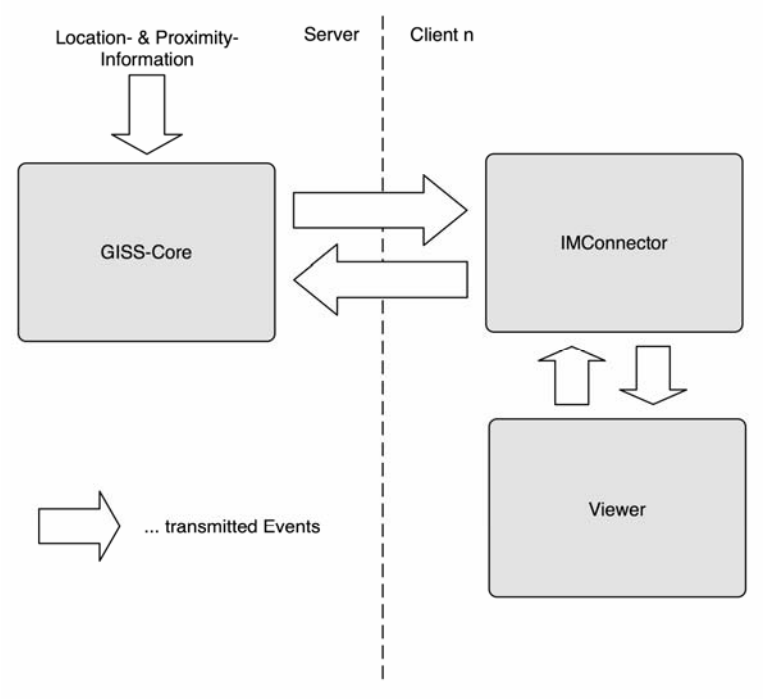


Figure 37: Flow of Events between Server and Clients

The flow of events between the attributes is depicted in Figure 37. As can be seen, communication between client and server side is realized involving just two attributes. The *Viewer* attribute request the information it needs from the *IMConnector*-Attribute, which has a local buffer for data already transmitted by the server. Only if the requested information cannot be found, the *GISS-Core* attribute is asked to send the needed information.

The main communication load is caused by the data that has to be distributed from the server to the clients for the update of location- and proximity information for each logged-in user. For privacy reasons and to minimize network traffic, new location information is not broadcast to all clients, but only transmitted to those, who have registered for this information in advance. As soon as they go online and whenever their messenger contact list changes, the clients send notification requests for every contact list entry (= buddy) to the server. The server then delivers location and proximity information for each requested buddy, as soon as this information is available. If a buddy is offline or does not deliver any location information at all, its location is set to *away*.

Other data transmitted from the server to the clients and vice-versa are information about and modification-requests for group-memberships and the submission and delivery of post-its, their comments and reminders.

Intra-client communication is limited to control- and location-data-transmission- and -request-events to and from the *Viewer* attribute. There is no direct communication between the client-entities, the communication is strictly based on a server/client-architecture.

5.1.2 Communication with SIM

As the user-interface of *GISS* is based on an external application, namely the instant messenger *SIM*, there has to be connection between the *GISS* components, which are executed within the SiLiCon Context Framework and the external application, which is executed natively by the operation system without any middleware (cf. Figure 38). As already explained, *SIM* has been chosen as user-interface, because it already provides a means for remote control by other applications.

This functionality is encapsulated in the *Remote Control*-plug-in, which is part of the standard distribution of *SIM*. The plug-in listens on a definable TCP-port for commands, which have to be sent as strings and take up to three parameters that are separated by spaces. With these commands, most of the basic features of *SIM* can be controlled and certain data can be queried. *SIM* natively supports the following commands (only those that are used by *GISS)*:

- *Status*: Query the current availability status or set a new one
- *Invisibe*: Query, if the user is currently invisible or set the invisible-state
- *Add*: Add an entry to the contact-list
- *Delete*: Delete an entry from the contact list
- *Open*: Open the message window of a contact
- *Contacts*: Query contact list
- *Show*: Open an unread message

Because not all needed commands have been provided by the original implementation, several others have been added when implementing *GISS*:

- *Message*: Send a message to another user
- *Setloc*: Set the location of a user (show location in brackets behind contact name)
- *Recvmessage*: Pretend a message of a contact has been received
- *Myuin*: Query the user’s UIN

Through the *Remote Control Connection*, all communication originating in the *IMConnector*-attribute is carried out. This includes commands to *SIM* that cause no feedback (like setting the availability status) and commands that request data from *SIM* (like querying the contact list). In the second case, the same TCP-connection is used by *SIM* to send query results.

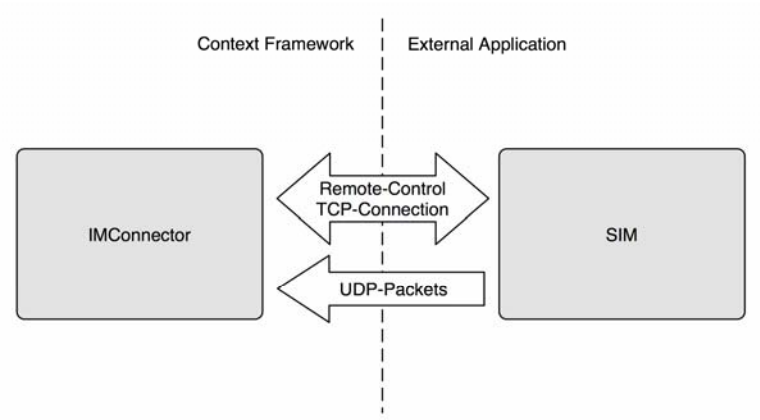


Figure 38: Flow of communication between Context-Framework and *SIM*

However, there are cases that are not covered by the communication facility mentioned above. As the *Remote Interface* does only support pulling data from *SIM* but does not allow messages to be sent from *SIM* to the *IMConnector*-attribute without request, another communication-channel is set up to allow the sending of messages originating from *SIM*, like user commands issued through the *GISS*-plug-in.

This second channel is realised with UDP-packets that are sent only from *SIM* to the *IMConnector*-attribute when necessary. There are two cases, when this channel is used:

- Transmitting user-commands from the user-interface to the *GISS*-components in the Context Framework
- Transmitting synchronous groups messages (cf. chapter 5.2.2)

Via those two channels the complete external communication of the *GISS* client is carried out. Another external connection is needed on server side to connect to the database (cf. chapter 5.1.3); all of these external communication channels are opened only locally, as said before, the complete network communication is realized via the SiLiCon Context Framework.

5.1.3 Database

The application data on server side is stored persistent in a database (*MySQL*) according to the data-model presented in Figure 39. The data-model is divided into two parts, of which the *backend*-part is used by the location-sensing application [Holz04] and the *frontend*-part is

used by the *GISS*-system. Storing the application data (including context information) in a database provides several advantages:

- Persistent storage of context data for later retrieval via a defined interface (SQL)
- Possible restart the system after failure in a defined state
- Defined interface between location sensing part and application (in this case *GISS*)

In the implementation, the ER-model has been transformed in a database-scheme, which is accessed trough a defined interface realized as a static class on server side.

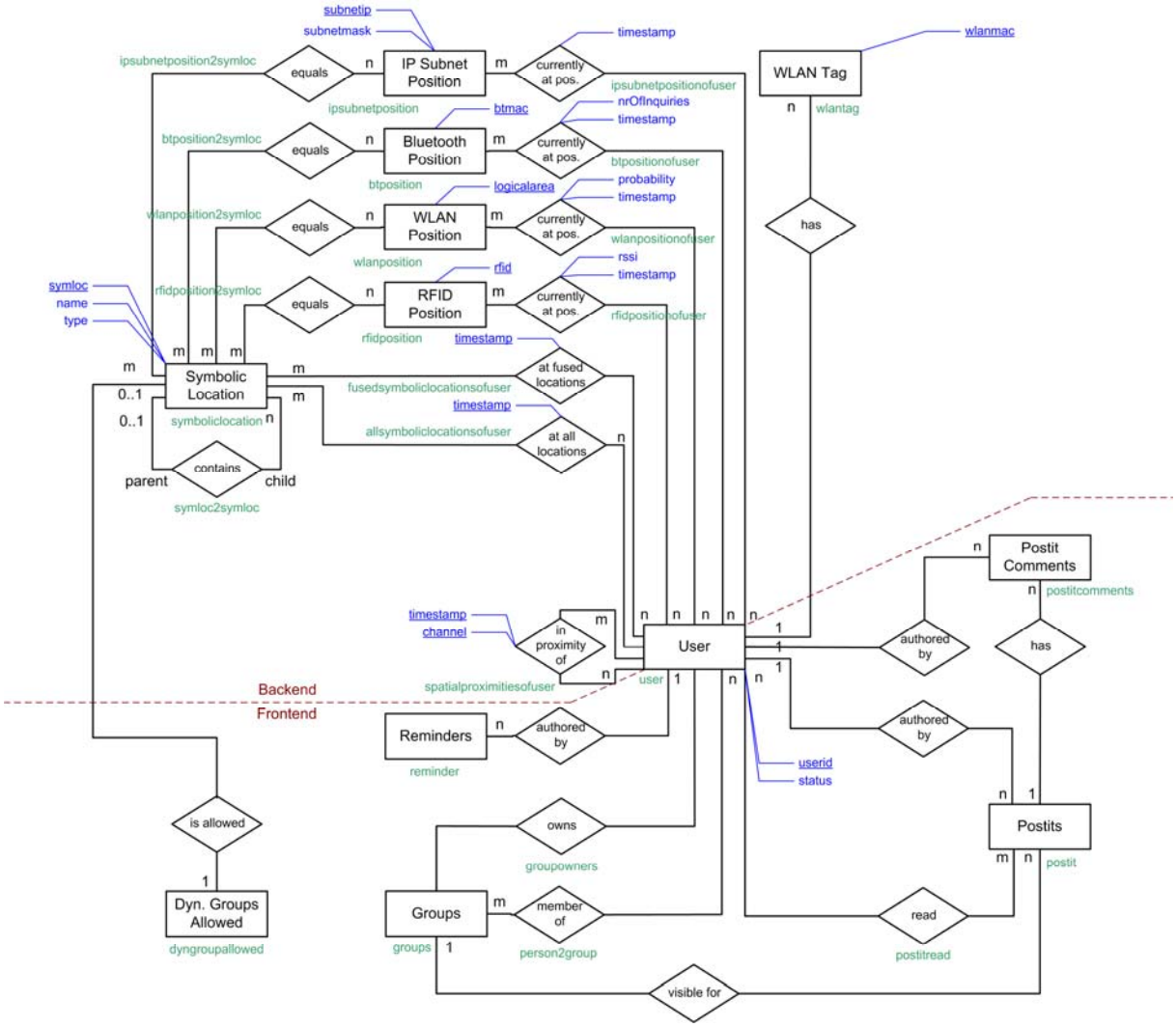


Figure 39: ER-Model of *GISS*-Data-Structures

This class encapsulates all creation, alteration and deletion of data in the database, provides transparent connection management (connections are opened, closed and checked automatically whenever needed) and offers methods for querying data also on higher abstraction levels (including necessary aggregation, selection or even recursive queries).

From the *frontend*-point-of-view, there is a database entity for each type of data used in *GISS*:

- Users
- Groups
- Post-Its
- Post-It-Comments
- Reminders

Those entities are linked by several relations that represent the internal structure of the data model. The central entity is *User*, to which all but one other entities are linked. In addition to the mentioned entities, every user also has one or more according locations that are represented in the *backend*-part of the ER-model. The only special case is *DynGroupsAllowed*, which is a application-specific extension of the location sensing part and marks all the locations, at which the building of dynamic groups is considered to make sense and thus is allowed.

5.2 Conceptual Issues

Having described the realisation of communication between the *GISS* components in the last chapter, the presentation of the service implementation is subject of this chapter. Before describing the services in detail, the use of context information to trigger services is presented shortly. From a conceptual point of view, the triggering of context-aware services in *GISS* can be caused by three cases:

- Change of a user's location
- Change in users' proximities
- Periodical time trigger

While the upper two cases are represented by two events issued by the location sensing module [Holz04] (`UserLocationChanged` and `UserProximityChanged`), the third case, namely periodically checking for fulfilled context criteria every minute, is done by the *GISS-Core*-attribute internal.

5.2.1 Managing Delivery of Virtual Post-Its

The check for virtual post-its and their delivery is executed in two cases:

- If the location of a user changes.
- If the client explicitly asks for post-its at a certain location

When the location of a user changes (`UserLocationChanged` received), the method for checking for virtual post-its is called. This method then performs the follow steps:

- Get the groups the user is member in
- Get all post-its visible for those groups at the current location of the user (including post-its at all locations that are below the current one in the location hierarchy)
- Check which post-its have already been read
- Deliver all unread post-its to the client.
- On client side, those post-its are shown automatically, when the user's availability status is other than *busy*

The second case when checking for post-its is performed, is the explicit request for this action from the client side. This request occurs every time the *2D-Viewer*-attribute changes the visualised floor. It then requests the delivery of all visible post-its on this floor (read as well as unread) to show them on the graphical floor plan. The sequence of steps basically remains the same as before except that all post-its instead of only the unread ones are delivered. On client side, namely in the *IMConnector*-attribute, those post-its are not shown automatically but buffered for later retrieval, when the user request the display by clicking on the post-it-icon in the *2D-Viewer*. The *2D-Viewer* is notified of the existence of a post-it by the *IMConnector*-attribute, where this notification only includes the position of the post-it and its status (read/unread), as this is the only information necessary to display the post-it-icon.

Whenever a post-it is chosen to be shown (either automatically or explicitly by user-request), several further information is requested from the server:

- The list of readers
- The list of comments

These data is not transmitted together with the basic post-it data, because it is highly dynamic and has to be up-to-date, when a post-it is shown. As post-its often remain in the client-side buffer for a longer time, the list of readers and comments is retrieved just in time when the post-it finally is shown.

As soon as a post-it is shown, the client sends an event to the server which causes the user to be added to the post-it's readers-list. Similarly, when a user enters a new comment for a post-it, this comment is transmitted via an event to the server, which adds it to the post-it's comment-list.

5.2.2 Delivery of synchronous Group-Messages

Synchronous group-messages appear to the user like standard messages that are delivered through the instant messenger network. But from an implementation point-of-view, they are completely different. In short, synchronous groups-messages are intercepted by *SIM* before they can be transmitted through the instant messenger network, sent to the Context Framework, which distributes them to all online recipients. The recipient's *SIM*-instances then show the message, like it would have been received through the messenger network.

In more detail, the delivery of synchronous group-messages is performed as follows:

- The ICQ-protocol knows several ways to transmit a message (among others delivery via the ICQ-server or direct peer-to-peer-transmission). *SIM* automatically chooses the most suitable way of transmission. The method, which realizes this functionality has been extended by another alternative, namely routing messages to the Context Framework components of *GISS*. This alternative is chosen every time a message is sent to a group account. Group accounts have a special UIN (identification number) that is not used by the ICQ-network and thus messages to these accounts can be easily identified.
- The recognized group-message is passed to the *GISS*-plug-in in *SIM* (coming from the ICQ-plug-in), which manages the UDP-channel to the *IMConnector*-attribute within the Context Framework
- By sending an UDP-packet, the group message is transmitted to the *IMConnector*-attribute, which constructs an event out of the message, including information about the sender and the receiving group.
- This event is sent to the server, which queries a list of all possible recipients (= members of the receiving group) from the database and forwards the message to all of them again via an event.
- The receiving clients first check their current availability status. When this status is set to *busy*, the message is dismissed. Otherwise, the message is transmitted to the *SIM*-instance through the *Remote Control*-plug-in, which opens a message window and displays the message in the form "*sender's name: message"*.

This solution is only suitable for the transmission of messages. Sharing files within a group according to the known ICQ-functionality is not supported at this time and is subject of future work.

5.2.3 Triggering of Reminders

As presented in chapter 4.5.7, reminders can be triggered by the following criteria:

- User at a certain location
- Point in Time has passed
- Another user in spatial proximity
- Any logical combination of the above (AND/OR)

This variety of possible triggers implies that the check for reminders has to be performed on changes in a user's location, changes of the nearby people and periodically every minute (cf. chapter 5.2). The process of checking for reminders is divided into several queries, which are used according to the type of trigger a reminder has:

- *Pure Proximity*-triggers are checked whenever a change in the proximity-list of a user occurs (on event `UserProximityChanged`)
- *Pure Location*-triggers are checked whenever a user changes his location (on event `UserLocationChanged`)
- *Pure Time*-triggers are checked periodically every minute.
- *Combined triggers* can be distinguished in two classes:
    - Combination with *OR*: Reminders are conceptually treated like multiple reminders, each with a pure trigger for one of the set criteria.
    - Combination with *AND*: Again, two cases can be distinguished:
        - *All criteria except time fulfilled*: checked periodically every minute (like a pure time trigger).
        - *All other cases*: checked whenever location or proximity changes, time trigger is checked here in the course of this process.

All of these cases are covered by two methods (one for *pure time triggers* and *all criteria except time fulfilled* and one for all other cases, where the check itself is coded into the SQL-statement used to retrieve the suitable reminders from the database.

5.2.4 Formation of Dynamic Groups

In chapter 4.5.3, dynamic groups have been defined to represent users in spatial proximity. Although it might appear obvious, not the proximity-sensing functionality is used for this ser-

vice, because a different quality of proximity is needed here, namely not only spatial (on room level) but additional some sort of organisational proximity is taken into account (as needed, when the persons located within the area of one department are forming a dynamic group).

For this reason, the check for modifications in dynamic groups is carried out on receiving a `UserLocationChanged`-event. A dynamic group is formed at the current location of a user and all the locations above in the location hierarchy, when the building of dynamic groups is considered to make sense at this location (determined by a preconfigured table in the database) and at least two users reside within the location (including locations that lie below in the location hierarchy).

Checking for dynamic groups is a two-stage process. First, the user is deleted from all dynamic groups he had been a member in because of his former location. All dynamic groups that happen to have only one member left after this deletion are dismissed. In the second step the user is added to all dynamic groups, he is a member of because of his current location. If a dynamic group does not yet exist, because only one user has resided at the according location before, it is created. Whenever a dynamic group is created or dismissed, all affected users are notified by an according event.

The formation of dynamic groups is clarified with the example depicted in Figure 40. The dashed boxes attached to the locations show the users currently residing at this location. A filled circle represents a location, at which formation of dynamic groups is allowed, whereas not filled circles represent locations, where not dynamic groups are built.

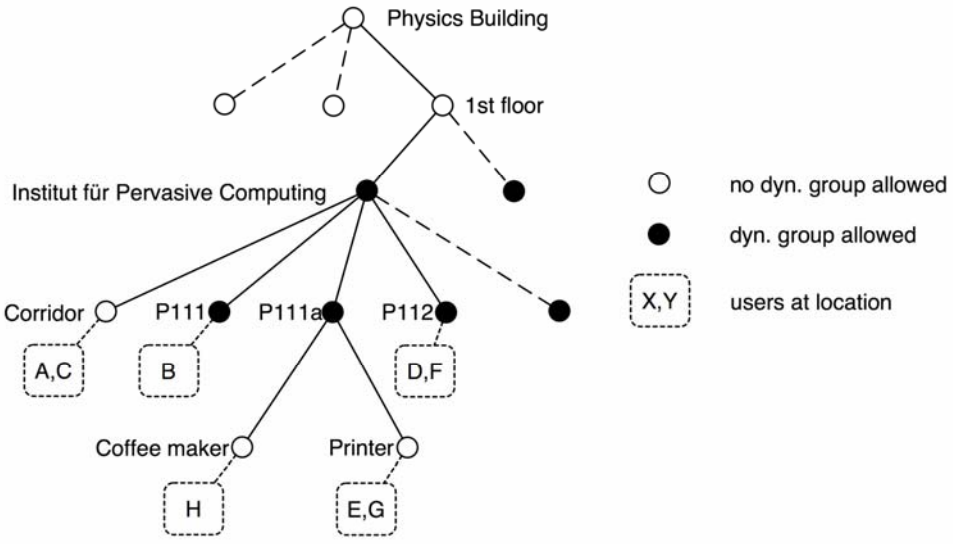


Figure 40: Formation of dynamic groups (example)

With the given settings, the dynamic groups at the following locations are formed with the stated users:

- *P112*: D, F
- *P111a*: E, G, H
- *Institut für Pervasive Computing*: A, B, C, D, E, F, G, H

No other groups are formed, because in location *Corridor* and *Printer* no dynamic groups are allowed and location *P111* currently contains only one user.

5.2.5 Recognition of Group-Gatherings

The recognition algorithm for group gatherings is executed every time a user changes his location. One of the four following cases may occur for each static group the user is member in:

- There is currently no gathering at the new location but now more than a certain percentage of the group members reside there and the start of a gathering is recognized.
- A gathering is already in progress at the new location and the user joins it.
- A gathering is already in progress at the former location and the user leaves it.
- A gathering is already in progress at the former location but now less than a certain percentage of the group members reside there and the end of the gathering is recognized.

When a `UserLocationChanged`-event is received, several queries are carried out. First, a list of all users that reside in the same room (location-level-ID 1, cf. chapter 4.3.2) is retrieved (for the new and the former location). Additional, by the membership-lists of all groups the user is member of, are made available. By merging those lists, the percentage of absent and nearby members at the new and the former location for each group can be found out. By using these percentages, the start (at the new location) and end (at the former location) of gatherings is recognized.

For each of cases mentioned above several activities have to be accomplished:

- When a group gathering starts, a *meeting minutes*-post-it is created at the location of the gathering. The starting time of the gathering as well as the present group members are stored as a comment in this post-it. The present group members are presented these *meeting minutes* automatically to be encouraged to take notes. Additionally, all absent group members are notified of the gathering by sending them an according event.

- When a user joins an ongoing gathering, an according entry in the *meeting minutes* is generated. The joining user is presented to already available *meeting minutes*, to get a feeling of the gathering progress so far.
- When a user leaves an ongoing gathering, an according entry in the *meeting minutes* is generated.
- When a group gathering ends, the end time is stored in the *meeting minutes* and all group members are shown these *meeting minutes* automatically, as long as they are online and their availability status is other than *busy*.

The generated *meeting minutes* are conceptually equivalent to virtual post-its and differ only in a set flag, which causes them to be displayed with a different icon in the 2D-viewer.

5.2.6 Availability Management

Availability management uses rules to define the contexts, in which a user's availability status should be set to a certain value. This service is a perfect use case for the context rules, on which the SiLiCon Context Framework is based on. The rules a user enters are translated into Context Framework-compliant rules and then are loaded into the rule base of the according client entity using the Context Frameworks feature to dynamically load and unload rules.

The details of this service are now presented by using a simple example. It is assumed, that the user has entered the following availability rule:

When I am in room HS10 *and time is* every week on Monday *between* 10:15 *and* 11:45, *set my availability state to* Busy.

This rule can now be translated into the following Context Framework-compliant ECA-rule:

```
on IMConnector.checkStatusRules(long time, long timeinweek,
                                long timeinmonth, long pos) {
      if (pos == 32 && timeinweek > 36900000 && timeinweek < 42300000) {
            Viewer.IMConnector.setStatus(4,0,2);
      }
      else {
            Viewer.IMConnector.resetStatus(2);
      }
}
```

The *Event*, the evaluation of this rule is triggered with, is `checkStatusRules`. This event is issued by the local *IMConnector*-attribute periodically every minute and causes the evaluation of all availability rules (as the *Event* is the same for all those rules). Different versions of

the current time (differing in the reference point) and the current location are provided as parameters:

- `time`: current system time
- `timeinweek`: current system time with the reference point at the beginning of the current week (Monday, 00:00 o'clock)
- `timeinmonth`: current system time with the reference point at the beginning of the current month (1st of month, 00:00 o'clock)
- `pos`: current location of the user (location id used in the database)

The *Condition* that is checked is assembled according to the specified criteria. There may be a location condition (as in this example `pos == 32`, where 32 corresponds to *HS10*) and time conditions according to the selected type of time trigger (once, daily, weekly, monthly). For the evaluation of time triggers of type *once* and *daily*, the parameter `time` is used, for *weekly* and *monthly*, `timeinweek` and `timeinmonth` are used respectively.

The *Action* part of the rule consists of two parts, which are used for setting the availability status, when the criteria are fulfilled for the first time and to restore the former availability state, when the criteria do not meet anymore. This is done by sending two events back to the *IMConnector*-attribute:

- `setStatus(status,invisible,rule)`: The parameter `status` holds the availability status to set, coded as a number. `invisible` is a Boolean parameter, which is used to specify whether *Invisible* should be set or not in addition to the new availability state. `rule` is a unique identification number for this rule that is used by *IMConnector* to store the former availability status, thus being able to restore it, when the criteria do not meet anymore.
- `resetStatus(rule)`: only takes `rule` as a parameter, which holds the unique identification number of the rule that is used to restore the former availability status.

Although the `setStatus`-event is issued every time, the check is performed during the criteria meet, the availability status is only set the first time it is issued. Before the modification of the status, the former status is stored with the unique identification number of the rule, to be able to restore it in `resetStatus`. In the subsequent calls, it is checked whether the user has modified the availability state manually. If this is the case, the `resetStatus` event has

no effect, because the former availability status stored at the first occurrence of `setStatus` is overridden by the new status set by the user.

6 Usage Scenarios

In this chapter, possible usage scenarios of *GISS* are identified and described with respect to learning on a university campus. The presented settings are intended to show the use of one or more *GISS* services that improve the learning and working experience on campus.

6.1 Learning on Campus

Student A has just had his last lecture for today and has to wait another two hours before his train home leaves. So he decides to repeat the lecture slides relevant for the exams he takes next week in the university e-learning system.

At the fifth slide, he finds an unclear issue, which he is unable to understand even after having searched for further explanation in the referenced related work. So he decides to ask the other attendees of the lecture for help. He does this by sending his question to the *group account* of the lecture in his instant messenger, which is shown to all online and available members of the group.

Student B is currently having a coffee break in the cafeteria when the question of Student A arrives through the lecture's group account. He remembers that he had overcome the same problem a few days ago. As Student A is a colleague he already has worked with is several project groups, he has already added him to his contact list, via which he now sends a message to Student A offering his help.

Student B tries to explain the fundamental points of the unclear issue in a message he sends to Student A. Student A still doesn't really understand but from *the little note behind the contact list entry* he can see that Student B currently resides fairly nearby in the cafeteria. So he decides to join Student B and gets on his way to the cafeteria.

The cafeteria is fairly big, so Student A cannot see Student B instantly, when he enters. Additionally, their last meeting has happened several months ago, so that Student A hardly remembers how Student B looked like. Fortunately, all the tables in the cafeteria are equipped with a location tag, so that after Student A had entered, the location description of Student B has refined from "cafeteria" to "third table from the back on window side". As he is unclear what is meant with "from the back", he opens the *graphical location viewer* and easily recognizes on which table Student B is sitting. With this information, Student A is able to join Student B, who is now able to explain the unclear issue face-to-face in a more interactive way.

6.2 Interaction in Lectures

Lecturer A gives his lecture the first time in this term. After the students have entered the lecture hall and have taken a seat, he starts by presenting the organisational issues. As a support for his lecture, he would like to create a group out of the participants in *GISS*, which can be used to ask question about unclear issues even out of lecture time, to post organisational information or to provide links to the lecture slides and related work.

As he had experienced, it is better to confront students with an opt-out-option rather than an opt-in-option if you want them to do anything voluntary. So he has decided not to create an empty *open static group*, which every participant has to join but to *transform the existing dynamic group* in the lecture hall into a static group, so that the students only have to confirm their membership. The newly created group is given the name of the lecture to be identifiable and contains now all members of the dynamic group of the lecture hall, who agree to join.

To provide the students with the lecture slides, Lecturer A puts a *virtual post it* in the lecture hall, in which he enters the URL to the PDF-slide set. As he wants his slides not to be publicly available for copyright reasons, he restricts the visibility of the post it to the lecture group. The post-it expires one day before the next lecture, thus always assuring that the most current slide set is available.

During the next lecture, Student B has not understood the content of the current slide and wants to ask a question. As the overcoming to ask is much lower, he uses the account of the *dynamic group* in the lecture hall *to post* his message. The question is shown to all available participants, which now can answer through the same channel. Because nobody could help, Lecturer A interrupts his current explanation and answers the question himself. Asking questions via the group account also has the advantage that questions are queued automatically, so that Lecturer A doesn’t easily miss an unanswered question.

Student B does not want to be bothered by others during the lecture because he finds it hard to concentrate then. As he has experienced, setting his availability status to *busy* helps others to recognize he is currently occupied. Furthermore, he does not receive those annoying group messages from people that write before they think. So Student B enters an *availability rule* to have his availability status automatically set to *busy* during the lecture. He specifies location and time of the lecture, so that the rule is not applied when he does not attend.

6.3 Working in Project Groups

Student A, Student B, Student C and Student D have *formed a closed static group* to work together on a practical in Java-programming. They have found each other to form a group by a *virtual post-it* Student B had put on the *virtual pin board* of the department, which supervises the practical. The group regularly use *synchronous group messaging* to discuss problems and solve problems with common interfaces.

Student A is working on a program module that has originally been implemented by Student C. After having adapted a part of the code to fit a new interface design, he encounters some code he is not able to understand and thus does not want to make modifications blindly. Unfortunately, Student C is currently offline, so that he cannot ask him for support. To prevent him from forgetting this issue, he enters a *reminder* which he defines to show when Student C is in *spatial proximity* and continues adaptation at a different part of the code.

A few days later, Student A, who does not remember his implementation questions anymore, meets Student C. The reminder pops up and brings the issue back into mind of Student A. While discussing the unclear part of the code, Student B comes around and joins them in the discussion. Now the system *recognizes a gathering* and automatically creates meeting minutes, where they start to write down the agreed key facts. Additionally, Student D, who is currently sitting in the cafeteria and surfing in the WWW, is notified about the gathering. Via the group account, he asks his colleagues if he is required to join them, which they confirm.

As Student D gets on his way to join them, he is not able to link the displayed room number to the physical room. So he quickly opens the *graphical location display* and checks for the room where the other three reside at currently on the visualized floor plan. As he joins them, the current version of the meeting minutes automatically pops up, providing him a quick overview of what had been discussed so far. After having agreed on all open issues, they continue their interrupted work and leave the room they had met in. As the end of the gathering is recognized, all of them are delivered the final meeting minutes. Student A, who is responsible for keeping track of the project progress, saved the document on his local computer, while the other three quickly overlook the notes and check them for correctness before closing them.

Bibliography

[Abow02] Abowd, G.D., Battestini, A., O'Connell, T. *"The Location Service: A framework for handling multiple location sensing technologies"*, College of Computing & GVU Center, Georgia Institute of Technology, Atlanta, Georgia, USA, 2002.

[Addl97] Addlesee, M.D., Jones, A.H., Livesey, F., Samarita, F.S. *"The ORL Active Floor"*, IEEE Personal Communications, vol. 4, no. 5, pp. 33-41, October 1997.

[Aero04] *Aeroscout*. Internet: http://www.aeroscout.com (31. August 2004).

[Anti02] Antifakos, S., Schiele, B. *„Beyond Position Awareness"*, Personal and Ubiquitous Computing, vol. 6, no. 5/6, pp. 313-317, December 2002.

[Bahl00] Bahl, P., Padmanabhan, V. *"RADAR: An in-building RF-based user location and tracking system"*, Proceedings of IEEE INFOCOM 2000. (Tel Aviv, Israel, March 2000). IEEE Computer Society Press, Los Alamitos, CA, 2000, vol. 2, pp. 775-784.

[Bal90] Bal, H.E. *„Programming Distributed Systems"*, Prentice Hall International, 1990.

[Barb04] Barbeau, M., Kranakis, E., Krizanc, D., Morin, P. *"Improving Distance Based Geographic Location Techniques in Sensor Networks"*, Proceedings of the 3rd International Conference on AD-HOC Networks & Wireless (ADHOC-NOW'04). (Vancouver, British Columbia, July 22-24, 2004). Springer Verlag, LNCS 3158, 2004, pp. 192-210.

[Bead97] Beadle, H.W.P., Harper, B., Marguire, G.Q., Judge, J. *„Location Aware Mobile Computing"*, Proceedings of the IEEE International Conference on Telecommunications (ITC'97). (Melbourne, Australia, April 1997).

[Beer03] Beer, W., Christian, V., Ferscha, A., Mehrmann, L. *„Modeling Context-Aware Behavior by Interpreted ECA Rules"*, Proceedings of the 9th International Conference on Parallel and Distributed Computing (Euro-Par'03). (Klagenfurt, Austria, August 26-29, 2003). Springer Verlag, Austria, LNCS2790, 2003, pp. 1064-1073.

[Beer04] Beer, W. *"SILICON Context Framework"*, Technical Report, Context Framework for Mobile User Applications, Siemens Munich CT-SE 2, February 2004.

[Blue04a] *The Official Bluetooth SIG Membership Site*. Internet: https://www.bluetooth.org (31. August 2004).

[Blue04b] *BlueTags*. Internet: http://www.bluetags.com (31. August 2004).

[Bohn03] Bohn, J., Vogt, H. *"Robust Probabilistic Positioning based on High-Level Sensor-Fusion and Map Knowledge"*, Technical Report No. 421, Institute for Pervasive Computing, Swiss Federal Institute of Technology, ETH Zurich, Switzerland, April 2003.

[Bonn01] Bonnifait, P., Bouron, P., Crubillé, P., Meziel, D. *"Data Fusion of Four ABS Sensors and GPS for an Enhanced Localization of Car-like Vehicles"*, Proceedings of the IEEE International Conference on Robotics and Automation (ICA'01). (Séoul, Korea, 21.-26. May 2001). IEEE, 2001, pp. 1597-1602.

[Bray02] Bray, J., Sturman, C.F. *"Bluetooth 1.1 – Connect Without Cables, Second Edition"*, Prentice Hall, 2002.

[Brow97] Brown, P.J., Bovey, J.D., Chen X. "*Context-Aware Applications: From the Laboratory to the Marketplace"*, IEEE Personal Communications, vol. 4, no.5, pp. 58-64, October 1997.

[Brum00] Brumitt, B., Meyers, B., Krumm, J., Kern, A., Shafer, S. *"EasyLiving: Technologies for Intelligent Environments"*, Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC 2000). (Bristol, UK, 25.-27. September 2000). Springer Verlag, London, UK, LNCS1927, 2000, pp. 12-29.

[Butz01] Butz, A., Baus, J., Krüger, A., Lohse, M. *"A hybrid indoor navigation system"*, Proceedings of the 6th International Conference on Intelligent User Interfaces (IUI 2001). (Santa Fe, NM, USA, 2001). ACM Press, New York, NY, USA, 2001, pp. 25-32.

[Chen00] Chen, G., Kotz, D. *„A Survey of Context-Aware Mobile Computing Research"*, Technical Report TR2000-381, Computer Science Department, Dartmouth College, Hanover, New Hampshire, November 2000.

[Chen03] Chen, R.Y.F., Petrie, C. *„Ubiquitous Mobile Computing"*, IEEE Internet Computing, vol. 7, no. 2, March-April 2003.

[Dey00] Dey, A.K. *„Providing Architectural Support for Building Context-Aware Applications"*, Ph.D. Thesis, Department of Computer Science, Georgia Institute of Technology, November 2000.

[Dobs04] Dobson, S. *„Where's Waldo? – or – A taxonomy for thinking about location in pervasive computing"*, Technical Report TCD-CS-2004-05, Computer Science Department, Trinity College Dublin, Ireland, May 2004.

[Domn01] Domnitcheva, S. *"Location Modeling: State of the Art and Challenges"*, Proceedings of the Workshop on Location Modeling for Ubiquitous Computing (Ubicomp 2001). (Atlanta, Georgia, 30. September 2001). pp. 13-20.

[Ekah04] *Ekahau Inc*. Internet: http://www.ekahau.com (31. August 2004).

[Estr02] Estrin, D., Culler, D., Pister, K., Sukhatme, G. *„Connecting the Physical World with Pervasive Networks"*, IEEE Pervasive Computing, vol. 1, No. 1, pp. 59-66, March 2002.

[FCC04] *Federal Communications Commission*. FCC enhanced 911. Internet: http://www.fcc.gov/e911 (31. August 2004).

[Fers04a] Ferscha, A., Holzmann, C., Oppl, S. *„Team Awareness in Personalized Learning Environments"*, Proceedings of the 3rd European Workshop on Mobile and Contextual Learning (MLEARN'04). (Rome, Italy, July 5-6, 2004).

[Fers04b] Ferscha, A., Holzmann, C., Oppl, S. *„Context-Awareness for Group Interaction Support"*, Proceedings of the 2nd ACM International Workshop on Mobility Management and Wireless Access (MobiWac'04). (Philadelphia, PA, USA, September 26-October 1, 2004). ACM Press, 2004, to be published.

[Fink03] Finkenzeller, K. *"RFID-Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification"*, John Wiley & Sons, October 2003.

[Form94] Forman, G.H., Zahorjan, J. *„The Challenges of Mobile Computing"*, IEEE Computer, vol. 27, no. 4, pp. 38-47, April 1994.

[Fox03] Fox, D., Hightower, J., Liao, L., Schulz, D. *"Bayesian Filtering for Location Estimation"*, IEEE Computer, vol. 2, no. 3, pp. 24-33, July-September 2003.

[Hall01] Hall, D., Llina, J. *"Handbook of Multisensor Data Fusion"*, CRC Press, 2001.

[Hall02] Hallberg, J., Nilsson, M. *"Positioning with Bluetooth, IrDA and RFID"*, Master Thesis, Department of Computer Science and Electrical Engineering, Division of Computer Engineering, Luleå University of Technology, January 2002.

[Hall03] Hallberg, J., Nilsson, M., Synnes, K. *"Positioning with Bluetooth"*, Proceedings of the 10th International IEEE Conference on Telecommunications (ICT'2003), (Tahiti, Papeete – French Polynesia, 23. February-1. March 2003). IEEE, 2003, pp.954-958.

[Hart99] Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P. *„The Anatomy of a Context-Aware Application"*, Proceedings of the fifth annual ACM/IEEE Conference on Mobile Computing and Networking (Mobicom'99). (Seattle, WA, USA, August 1999). ACM Press, 1999, pp. 59-86.

[Haza04] Hazas, M., Scott, J., Krumm, J. *„Location-Aware Computing Comes of Age"*, IEEE Computer vol. 37, no. 2, pp. 95-97, February 2004.

[High00] Hightower, J. *"SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength"*, Technical Report #2000-02-02, Computer Science and Engineering Department, University of Washington, Seattle, WA, USA, February 2000.

[High01a] Hightower, J., Boriello, G. *„A Survey and Taxonomy of Location Systems for Ubiquitous Computing"*, Extended paper from IEEE Computer, vol. 34, no. 8, pp. 57-66, August 2001.

[High01b] Hightower, J., Vakili, C., Borriello, G., Want, R. *"Design and Calibration of the SponON Ad-Hoc Location Sensing System"*, unpublished, August 2001.

[High02] Hightower, J., Brumitt, B., Boriello, G. *„The Location Stack: A Layered Model for Location in Ubiquitous Computing"*, Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02). (Callicoon, NY, USA, 20-21 June 2002). IEEE Computer Society, June 2002, pp. 22-30.

[Hu04] Hu, H., Lee, D.L. *"Semantic Location Modeling for Location Navigation in Mobile Environment"*, Proceedings of the 5th IEEE International Conference on Mobile Data Management (MDM'04). (Berkeley, California, 19.-22. January 2004). IEEE Computer Society Press, 2004, pp. 52-61.

[IDEN04] *IDENTEC SOLUTIONS AG*. Internet: http://www.identecsolutions.com (31. August 2004).

[IEEE04] *IEEE 802 LAN/MAN Standards Committee*. Internet: http://www.ieee802.org (31. August 2004).

[Indu03] Indulska, J., Sutton, P. *„Location Management in Pervasive Systems“*, Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003. (Adelaide, Australia, 2003). Australian Computer Socienty, Darlinghurst, Australia, 2003, pp. 143-151.

[Krum00] Krumm, J., Harris, S., Mayers, B., Brummitt, B., Hale, M., Shafer, S. *“Multi-Camera Multi-Person Tracking for EasyLiving”*, Proceedings of the 3rd IEEE International Workshop on Visual Surveillance (VS’2000). (Dublin, Ireland, 1. July 2000). IEEE Computer Society Press, Washington, DC, USA, 2000, pp. 3-10.

[Kuma03] Kumar, V., Das, S.R. *„Performance of Dead Reckoning-Based Location Service for Mobile Ad Hoc Networks”*, Wireless Communications and Mobile Computing Journal, December 2003.

[Leon98a] Leonhardt, U. *„Supporting Location-Awareness in Open Distributed Systems“*, Ph.D. Thesis, Department of Computing, Imperial College, London, May 1998.

[Leon98b] Leonhardt, U. Magee, J. *„Multi-Sensor Location Tracking“*, Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom’98). (Dallas, Texas, United States, 25.-30. October 1998). ACM Press, New York, NY, USA, 1998, pp. 203-214.

[Lin] Lin, J., *“Personal Location Agents for Communication Entities (PLACE)”*, Master Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Cambridge, UK, May2002.

[Matt03] Mattern, F., Sturm, P. *„From Distributed Systems to Ubiquitous Computing – State of the Art, Trends and Prospects of Future Networked Systems“*, Proceedings of the 13th Fachtagung Kommunikation in Verteilten Systemen (KiVS’03). (Leipzig, Germany, February 2003). Springer Verlag, Leipzig, Germany, 2003, pp. 3-25.

[Morl04] Morla, R., Davies, N. *“Evaluating a Location-Based Application: A Hybrid Test and Simulation Environment”*, IEEE Pervasive Computing, vol. 3, no. 3, pp. 45-56, July-September 2004.

[Moor65] Moor, G.E. *“Cramming more components onto integrated circuits”*, Electronics, vol. 38, no. 8, 19. April 1965.

[Nels98] Nelson, G.J. *„Context-Aware and Location Systems“*, Ph.D. Thesis, University of Cambridge, Computer Laboratory, Cambridge, UK, January 1998.

[Ni03] Ni, L.M., Liu, Y., Lau, Y.C., Patil, A.P. *“LANDMARC: Indoor Location Sensing Using Active RFID”*, Proceedings of the 1st International IEEE Conference on Pervasive Computing and Communications (PerCom’03). (Forth Worth, TX, USA, 23.-26. March 2003). IEEE Computer Society Press, 2003, pp. 407-415.

[Nibb01] *The Nibble Location System*. Internet: http://mmsl.cs.ucla.edu/nibble (31. August 2004).

[Oppl04] Oppl, S. *"Context-Sensitive Interaction in Groups"*, Master Thesis, University of Linz, Institut für Pervasive Computing, 2004, to be published.

[Orr00] Orr, R.J., Abowd, G.D. *"The Smart Floor: A Mechanism for Natural User Identification and Tracking"*, Proceedings of the International Conference on Human Factors in Computing Systems (CHI 2000). (The Hague, Netherlands, 1.-6. April 2000). ACM Press, New York, NY, USA, 2000, pp. 275-276.

[Patt03] Patterson, C.A., Muntz, R.R., Pancake, C.M. *"Challenges in Location-Aware Computing"*, IEEE Pervasive Computing, vol. 2, no. 2, pp. 80-89, April-June 2003.

[Priy00] Priyantha, N.B., Chakraborty, A., Balakrishnan, H. *"The Cricket Location-Support System"*, Proceedings of the 6th International ACM Conference on Mobile Computing and Networking (MobiCom 2000). (Boston, MA, USA, 6.-11. August 2000). ACM Press, 2000, pp. 23-43.

[Ray03] Ray, A., Kurdovsky, S. *"A Survey of Intelligent Pervasive Computing"*, Proceedings of the International Conference on Artificial Intelligence (IC-AI'03). (Las Vegas, Nevada, USA, 23.-26. June 2003). CSREA press, 2003, pp. 30-35.

[Röde04] Röder, P., Hoffmann, M., Ritscher, M. *"A distributed framework for location sensing systems"*, Proceedings of the 1st Workshop on Positioning, navigation and Communication (WPNC'04). (University of Hannover, Germany, 26. March 2004). Shaker Verlag, Germany, 2004.

[Roth02] Roth, J. *„Mobile Computing"*, dpunkt.verlag GmbH, Heidelberg, Germany, 2002.

[Rous02] Roussos, G. *"Location Sensing Technologies and Applications"*, Technical Report TSW 02-08, School of Computer Science and Information Systems, Birbeck College, University of London, England, November 2002.

[Ryan98] Ryan, N.S., Pascoe, J., Morse, D.R., *"Enhanced Reality Fieldwork:: the Context-aware Archaeological Assistant"*. Gaffney, V., Lesuen, M.van., Exxon, S. (edt.), Computer Applications in Archaeology 1997, British Archeological Reports, Tempus Reparatum, Oxford, October 1998.

[Saty96] Satyanarayanan, M. *„Fundamental Challenges i Mobile Computing"*, Proceedings of the Fifteenth ACM Symposium on Principles of Distributed Computing. (Philadelphia, PA, USA, May 1996). ACM Press, New York, 1996, pp. 1-7.

[Saty01] Satyanarayanan, M. *„Pervasive Computing – Vision and Challenges"*, IEEE Personal Communications, vol. 8, no. 4, pp. 10-17, August 2001.

[Schi94] Schilit, B., Adams, N., Want, R. *„Context-Aware Computing Applications"*, Proceedings of the IEEE Workshop on Mobile Systems and Applications (WMCSA'94). (Santa Cruz, CA, USA, December 1994). IEEE Computer Society Press, 1994, pp. 85-90.

[Schi95] Schilit, B.N. *„A System Architecture for Context-Aware Mobile Computing"*, Ph.D. Thesis, Columbia University, Department of Computer Science, May 1995.

[Schi02] Schilit, B.N., Hilbert, D.M., Trevor, J. *"Context-Aware Communication"*, IEEE Wireless Communications, vol. 9, no. 5, pp. 46-55, October 2002.

[Schi03] Schiller, J. *"Mobilkommunikation"*, Pearson Studium, Munich, Germany, 2003.

[Schm98] Schmidt, A., Beigl, M., Gellersen, H.W. *"There is more to Context than Location"*, Proceedings of the International Workshop on Interactive Applications of Mobile Computing (IMC'98). (Rostock, Germany, Fraunhofer IGD, November 1998).

[Schm02] Schmidt, A. *"Ubiquitous Computing – Computing in Context"*, Ph.D. Thesis, Lancaster Universit, U.K., November 2002.

[SETI04] *SETI@home: Search for Extraterrestrial Intelligence at home*. Internet: http://setiathome.ssl.berkeley.edu (31. August 2004).

[Sing91] Singhal, M., Casavant L.C. *"Distributed Computing Systems"*, IEEE Computer, vol. 24, no. 8, pp. 12-15, August 1991.

[Sriv01] Srivastava, M. "*Mobile and Wireless Networked Systems"*, Lecture EE206A, University of California, Department for Electrical Engineering, Internet: http://www.cs.wmich.edu/wsn/doc/loc/loc.ppt (31. August 2004).

[Star97] Starner, T., Mann, S., Rhodes, B., Healey, J., Kirsch, D., Picard, R., Pentland, A. *"Augmented reality through wearable computing"*, Presence, vol. 6, no. 4, pp. 386-398, August 1997.

[Tane02] Tanenbaum, A.S., Steen, M. *"Distributed Systems: Principles and Paradigms"*, Prentice Hall, 2002.

[Want92] Want, R., Hopper, A., Falcão, V., Gibbons, J. *"The Active Badge Location System"*, ACM Transactions on Information Systems (TOIS), vol. 10, no. 1, pp. 92-102, January 1992.

[Ward97] Ward, A., Jones, A., Hopper, A. *"A New Location Technique for the Active Office"*, IEEE Personal Communications, vol. 4, no. 5, pp. 42-47, October 1997.

[Ward98] Ward, A.M.R. *"Sensor-driven Computing"*, Ph.D. Thesis, Corpus Christi College, University of Cambridge, August 1998.

[Webs04] *Merriam-Webster Online Dictionary*. Internet: http://www.m-w.com (31. August 2004).

[Well93] Wellner, P., Mackay, W., Gold, R. *"Computer-Augmented Environments: Back to the Real World"*, Communications of the ACM, vol. 34, no. 7, pp. 24-26, July 1993.

[Wemm03] Wemmer, A. *"Design and Implementation of a Bluetooth Tag"*, Master Thesis, Institut für Technische Information, Technische Universität Graz, May 2003.

[Wert01] Werthimer, D., Cobb, J., Lebofsky, M., Anderson, D., Korpela, E. *"SETI@home-Massively Distributed Computing for SETI"*, IEEE Computing in Science & Engineering, vol. 3, no.1, pp. 18-83, January/February 2001.

[Weis91] Weiser, M. *"The Computer for the 21st Century"*, Scientific American, vol, 265, no. 3, pp. 94-104, September 1991.

[Weis93] Weiser, M. *"Some Computer Science Issues in Ubiquitous Computing"*, Communications of the ACM, vol. 36, no. 7, pp. 75-84, July 1993.

[WIDC04] *WIDCOMM Inc*. Internet: http://www.widcomm.com (31. August 2004).

[Wifi04] *Wi-fi Alliance*. Internet: http://www.wi-fi.org (31. August 2004).

[Yosh00] Yoshimi, B. *"On Sensor Frameworks for Pervasive Systems"*, Proceedings of the Workshop on Software Engineering for Wearable and Pervasive Computing (ICSE 2000). (Limerick, Ireland, 6. June 2000).

Location Sensing System

Dipl.-Ing. Clemens Holzmann

Univ.-Prof. Dr. Alois Ferscha

Institut für Pervasive Computing
Johannes Kepler Universität Linz
A-4040 Linz, Altenberger Straße 69
www.pervasive.jku.at

7 Background and Related Fields

Designing a distributed framework that automatically acquires location-information from multiple sensors attached to mobile clients and makes it available for distributed context-aware applications in a standardized form draws from several fields of research (cf. Figure 1).

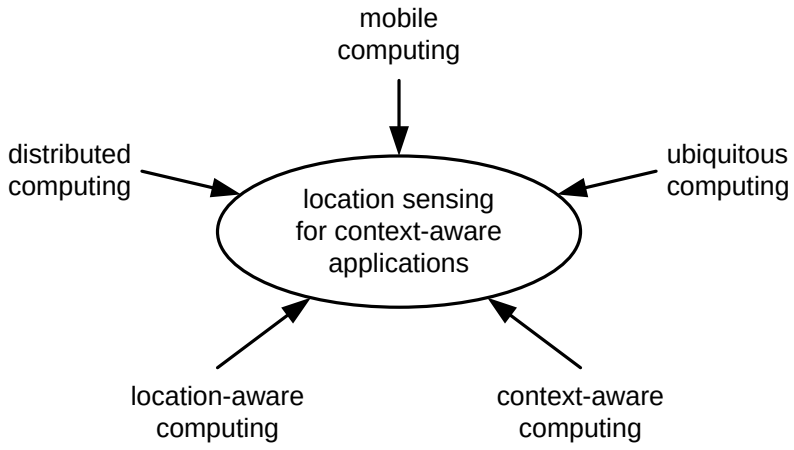


Figure 1: Interrelated computing paradigms

Distributed computing, a type of computing where different components of a single application are located on multiple heterogeneous computers connected to a network, as well as *mobile computing*, the use of portable computers capable of wireless networking [Form94], are computing paradigms which set the stage for running an application distributed among mobile clients moving through an environment.

On top of these paradigms is *ubiquitous computing*, the vision of environments that are saturated with invisible computing and communication capabilities. Ubiquitous computing (also referred to as pervasive computing) comprises the research areas of distributed and mobile computing [Saty01], but it goes much further and raises numerous new questions concerning invisibility of technology and proactive behaviour.

A fundamental, enabling technology for ubiquitous computing is *context-awareness*, which is defined as the property of a system to have information about the environment and adapt its behaviour based on this information. It provides a means for realizing minimally intru-

sive systems, which again is crucial for creating systems that disappear at least in the user's perception. [Schm02]

Finally, *location-aware computing* is a crucial subset of context-aware computing and thus of particular interest. Location-awareness refers to applications which take advantage of their own and other objects' physical locations [Leon98a]. The field of location-aware computing spans numerous research issues like sensing of physical positions, merging location information acquired from multiple sensors and methods for representing locations for location-aware applications [Haza04].

7.1 Distributed Computing

Distributed computing is defined as the process of dividing a computational task into smaller sub-tasks that are separated out to devices across a distributed system [Ray03]. A distributed system can be seen to consist of several autonomous computers that cooperate by message passing over a communication network in order to coordinate the action and processes of an application [Bal90, Matt03]. A more general definition is given in [Tane02], where a distributed system is defined as *"a collection of independent computers that appears to its users as a single coherent system"*. This virtuality of a system can be based on different forms of transparency like location and mobility of components.

[Sing91] identified two different reasons for using distributed systems. The first reason is to *find better or more efficient solutions for problems* regarding high performance, fault tolerance or coping with real-time requirements. An innovative example for increasing performance is SETI@home [SETI04, Wert01], a worldwide-distributed computing project whose goal is to determine if there is intelligent life outside the Earth. In order to cope with the immense amount of data collected by the world's biggest radio telescope in Puerto Rico, the data stream is divided into small chunks and distributed among millions of personal computers all over the world having a particular client program installed[1]. The second reason for using distributed systems is to *address problems that are inherently distributed* (i.e. many applications involve spatially separated machines) [Tane02]. An example is the location sensing architecture presented in this thesis, which is based upon a distributed

[1] With July 2004, approximately 60 TeraFLOPs/sec are performed. More than 5 million users have contributed and 2 million years of computation time have been spent [SEIT04].

client/server-architecture, consisting of multiple clients that are being tracked and provided with location information as well as a central server on which the location service is running.

Distributed systems involve numerous challenges like heterogeneity concerning hardware and operating systems, scalability with regard to the number of users and components, integrity of resources that are shared among multiple clients and different forms of transparency [Tane02]. Driven by such challenges as well as potentials coming along with distributed systems, the research issues comprise remote communication, fault tolerance, high availability, remote information access and security [Saty01]. This has created a theoretical and algorithmic basis which is useful in all settings where multiple computers are connected by any kind of network, and thus being of particular importance for ubiquitous computing.

7.2 Mobile Computing

The next stage towards ubiquitous computing is mobile computing. It refers to the use of portable devices like PDAs, smartphones and notebook computers that are capable of wireless networking, and it enables access to digital resources – anytime and anywhere.

The research area of mobile computing emerged from building distributed systems with mobile clients [Saty01] who are characterized by three basic properties [Form94]: wireless communication, mobility and portability. *Wireless communication* is a pre-requisite for communication among mobile computers, and it can be based either on modulated radio waves or on pulsed infrared light [Form94]. Currently, there are two predominant trends [Ray03]: protocols for short-range wireless communication like Bluetooth and IrDA, and those for long-range wireless communication like IEEE 802.11x. The second property is *mobility*, which refers to the ability of changing locations while staying connected to the network. Finally, *portability* means that mobile devices are small, light and requiring minimal power usage for long battery life.

The properties stated above lead to several constraints, which are characteristic for mobile computing. First, mobile devices are resource-restricted regarding processor speed, memory size and communication bandwidth for example. Second, mobile computing is inherently hazardous. A main problem in this context is wireless communication, which comes

along with packet loss, high battery consumption, areas with no reception and security risks. Moreover, portable devices are more likely to be lost, damaged or stolen. Another constraint is the highly variable performance and reliability. Finally yet importantly, mobile devices rely on limited energy sources. This makes it necessary to use more efficient algorithms (e.g., certain tasks must be accomplished with less amount of processing) as well as communication protocols (e.g. by turning off the receiver during long periods without communication). [Form94, Ray03, Saty96]

Caused by the constraints stated above, the research on mobile computing led to the development of particular techniques, which can be divided into the following areas [Saty01]: additional mobile networking, mobile information access, system-level energy-saving techniques and the support for adaptive applications and location sensitivity, including location sensing and location-aware behaviour [Chen00]. As with distributed computing, mobile computing forms an essential basis for ubiquitous computing.

7.3 Ubiquitous Computing

The vision of ubiquitous computing was described by Mark Weiser 1991 in a pioneering paper [Weis91] in Scientific American, which he began with the following statement: *"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."* [Weis91]

The scope of ubiquitous computing is the fusion of computing and communication technologies with the environment in a way that the technology disappears. This implies two key objectives for ubiquitous systems – *ubiquity*, the constant availability of computation and communication, and *invisibility*, a masking of the presence of the system from the user [Estr02]. Driving factors are Moore's Law, namely the observation that computing power doubles every 18 months (Moore originally observed that the *"complexity for minimum component cost has increased at a rate of roughly of two per year"* [Moor65], it was later amended to 18 months), progress in communication technologies and recent developments in sensor technology, among other things. As motion is an integral part of everyday life, ubiquitous computing comprises the research areas of distributed and mobile computing, but it goes further. [Matt03, Saty01]

A main objective of ubiquitous computing is *invisibility*, which refers to the disappearance of computing technology from users' consciousness. This can be achieved by minimization and embedding of computational logic into the environment. Invisibility can also be reached by proactive and automatically adjusting behaviour, which requires that ubiquitous systems have information about a certain user and his environment. Thus, context-awareness, and in particular location-awareness, is of crucial importance for ubiquitous computing. [Saty01, Schm02]

A further research issue are *smart spaces*, which are areas with an embedded computing infrastructure. Smart spaces bring together artificial and physical worlds and thus enable sensing and control of one world by each other [Estr02, Weis91, Well93]. For example, light or heating in a room may adapt according to the profile of a certain user. However, it is also possible that applications behave differently depending on the user's location [Saty01] (e.g. an office application, which automatically uses the nearest printer). These examples show the importance of remote identification (e.g. who is present in a certain room) and location sensing (e.g. where is someone located) techniques for ubiquitous computing. Mark Weiser wrote in [Weis91]: *"If a computer merely knows what room it is in, it can adapt its behaviour in significant ways, without requiring even a hint of artificial intelligence."*

7.4 Context-Aware Computing

Context-aware computing is a paradigm in which applications can recognize and exploit real-world contextual information. It arose under the vision of ubiquitous computing about a decade ago, since then many other researchers have studied this topic. Context-awareness is a crucial contributor to the invisibility-fundament in ubiquitous computing. [Chen00, Haza04, Patt03]

A general definition of context is given by Merriam-Webster's Online Dictionary [Webs04], where *context* is defined as the *"interrelated conditions in which something exists or occurs"*. As this definition is very loose, many approaches have been made to define the notion of context with respect to computing environments.

Most definitions are done by enumerating examples or by choosing synonyms for context. The term context-aware has been introduced first in [Schi94], where context is referred to

as location, identities of nearby people and objects, and changes to those objects. In [Brow97], context is also defined by an enumeration of examples, namely location, identities of the people around the user, the time of the day, season, temperature etc. [Ryan98] defines context as the user's location, environment, identity and time.

A more formal and often used definition is given in [Dey00], where context is defined as *"any information than can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves"*. [Dey00] identifies four primary types of context information, which are more important compared to others with respect to characterizing the situation of an entity. These are location, identity, time and activity, and they can be used to derive other sources of contextual information (so-called secondary context types or high-level contexts [Chen00]).

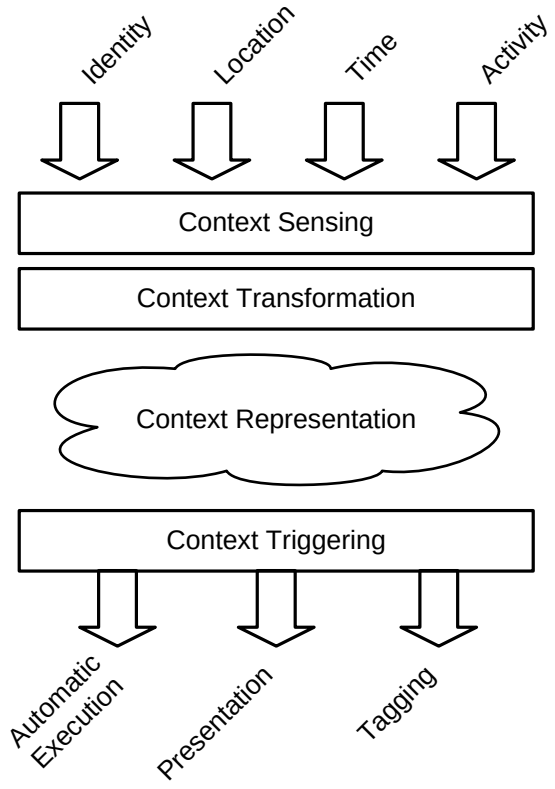


Figure 2: Layers of a context-aware system [Fers04b]

Context computing focuses on two major issues, namely *sensing* and *using* context. As location is an important context that changes whenever the user moves, location sensing is

fundamental for most context-aware applications [Chen00], and location is considered as one of the most important types of context information by many researchers [Abow02, Haza04, Patt03, Weis91]. However, there are many types of context other than location, like time, proximity of persons, temperature, pressure or light level [Schm98]. The second issue is context use. [Dey00] defines a system to be context-aware *"if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task"*. [Dey00] also gives a classification of features for context-aware applications, which comprises presentation of context information to a user, automatic execution of a service based on the user's context and tagging information of context to information for later retrieval.

In order to use sensed context information in context-aware applications, a few layers of abstraction are needed (cf. Figure 2). First, the obtained low-level context information has to be transformed, aggregated and interpreted (*context transformation*) and represented in an abstract context world model (*context representation*), either centralized or decentralized. Finally, the stored context information is used to trigger certain context events (*context triggering*). With respect to providing location context for context-aware applications, location information acquired by location sensors has to be transformed and represented in a certain location model, and there must be a mechanism to notify context-aware applications about location and proximity information of users.

7.5 Location-Aware Computing

A location-aware system is defined as one that *"knows where each system component is located and can use the information to enhance the overall system operation"* [Bead97]. Components can be people, computers and virtually any other object of interest. Driven by recent advances in location sensing technologies and their widespread deployment, the emergence of mobile computing as well as the importance of location for the growing field of ubiquitous computing, location-aware computing became a distinct field of research in the past decade. [Haza04]

The research issues of location-aware computing are manifold. A central problem is the *acquisition of location information*, wherefore numerous location-sensing systems have been developed. They use different sensor technologies and sensing techniques differing in

accuracy, coverage, frequency of location updates, the ability to locate certain types of objects, and cost of installation and maintenance [Haza04, High02]. Unfortunately, there is no technology currently available which provides high accuracy and coverage at the same time [Bead97] and complies with the broad spectrum of ubiquitous applications [High02]. For this reason, the *fusion of location information* from multiple sensors has become a major challenge in location-aware computing.

In order to provide support for a variety of location-aware applications, methods for *abstracting location* are needed. In this regard, two approaches for representing and correlating location-information can be distinguished [High01a, Leon98a]: geometric models, which represent location as coordinates, and symbolic models, which represent locations as abstract symbols. Much research has focused on the *development of location-aware systems and applications* for numerous everyday scenarios as well as military training [Chen00, Haza04, High01a], where an important issue is the drawing of higher-level contexts from location information, for example by classifying a visitor's movement patterns or by combining it with other types of contextual information.

8 Location Sensing Fundamentals

This chapter gives an overview of location sensing fundamentals, independently from concrete location systems and technologies. First, the *concept of location* is discussed from a ubiquitous computing viewpoint and common *location-sensing techniques* are presented. The second part deals with abstracting location by representing location information with *location models*, and *sensor fusion*, namely the combination of location information acquired from multiple different location sensors. Finally, a taxonomy for *location system properties* is given.

8.1 What is Location?

Location typically refers to where someone or something is located in the real world, but it is a quite subtle term with many different meanings.

First, locations can be classified in absolute and relative locations. *Absolute locations* refer to an idealized view of space for all *located objects* (i.e. entities that can be localized by a certain location system [Leon98a]), and they in turn can be classified in physical positions and symbolic locations. *Physical positions* provide geometric coordinates (e.g. latitude and longitude from a GPS receiver), areas or volumes within one or more reference coordinate systems [Domn01, Leon98a]. Such locations do no have an immediate connection to the real world, which means, that special maps are needed for interpretation [Dobs04]. *Symbolic locations* (sometimes also called semantic locations) provide human-meaningful, abstract symbols like “Physics Building” or “Room P121”, which depend on a certain application domain. They are typically structured in a hierarchy of locations like <building, floor, room> [Haza04], which matches the natural structure of the respective spaces. [Leon98a] defines a symbolic location as *“a name referring to a well-defined geographical area which does not need to be constant over time”*.

The second type are *relative locations*, which refer to spaces that are identified by their relationship to another subject, space or artefact. A person’s location can be *related to a static space*, namely a location where they have a known relationship to (e.g., a person is in his office, but the office’s location may change with time), or it can be *related to a dynamic space*, which refers to a space that moves (e.g. a car, which has a dynamic relation with other spaces while representing a space of its own). Other forms of related locations

are *related associations*, namely the knowledge that someone is with someone else. [Dobs04]

Besides these *known locations* as described above (absolute, relative), [Dobs04] also identified a category of *approximate* locations which are classified in temporal and spatial. *Temporal approximations* refer to locations where someone was in the past or where he will be in the future, although the current location is unknown.

Spatial approximations refer to locations where someone is approximately located, e.g. in *spatial proximity* of somebody else (where proximity is interpreted as *"close enough to still be of relevance"* [Dobs04]) or on a *path* (between two locations). A further interesting aspect is the fact that locations are mostly determined by some artefact (e.g. a GPS receiver or a mobile phone) which is associated with a certain person; however, this assumption may be false (e.g. if the person forgets the artefact at home). [Dobs04]

Locations can also be *task-based*, which means that the location of someone is estimated by determining *in which task he is involved*. This, indeed, is only possible if the task is associated with a location. Even if the task as well as the location is not explicitly known, it may be possible to determine where someone *usually* is located at a certain time. [Dobs04]

Instead of declaring where someone is, the number of possible locations can eventually be reduced by determining *where someone is not at this time* [Dobs04] (e.g. he has been detected to leave the office by an electronic check clock).

Orientation and *velocity* are context information related to location, which can significantly enhance location-aware applications [Patt03, Ward98] (e.g. for indoor or car navigation systems [Butz01] or smart spaces which automatically activate the display a certain user is looking at). *Velocity* can be measured directly by taking advantage of the Doppler Effect or indirectly by delayed determination of position [Roth02] for example. As with velocity, *orientation* can also be derived from location information [Ward97] or by using special sensors like gyroscopes or ultrasonic tracking systems [Ward98].

8.2 Location Sensing Techniques

There are many types of location information and they again are acquired by different *location sensors*, which can be classified in three general categories [Indu03]:

- *Physical location sensors* provide information about the position of a physical device (e.g. a GPS receiver) and there are many different techniques for determining the position (e.g. triangulation, proximity and scene analysis).
- *Virtual location sensors* extract location information from virtual space (e.g. the IP address of a mobile device).
- *Logical location sensors* combine information from physical or virtual sensors with information from other sources (e.g. a database which maps RFID numbers to symbolic locations) in order to deduce the physical location.

In the following, some important techniques for determining location information are described.

8.2.1 Triangulation

Triangulation uses the geometric properties of triangles to calculate locations. Two types of triangulation can be distinguished: *lateration*, which uses distance measurements, and *angulation*, which uses primarily angle or bearing measurements [High01a].

8.2.1.1 Lateration

Lateration calculates the position of an object by measuring its distance from multiple non-collinear (i.e. not lying on the same straight line) reference points. As Figure 3 illustrates, distance measurements (r1, r2 and r3) from three such reference points (x1, x2 and x3) with known positions are necessary in order to calculate the position X, which lies at the intersection of the resulting three circles, in two dimensions [High01a]. In three dimensions, a fourth reference station is necessary [Ward97].

There are basically three approaches to measure the distance r. First, the distance can be measured *directly* by a physical action or movement. This is difficult to determine automatically because of the need for coordinating autonomous physical movements [High01a]. A related approach is dead reckoning, which is mentioned in chapter 8.2.4.

The second approach is *time-of-flight* measurement, where the time of a transmission to travel between the object of interest and a certain reference point at known velocity is measured. For example, the velocity of ultrasonic is approximately 344 m/sec at 21°C air and that of light and radio waves is 299.792.458 m/sec in vacuum. To measure the time, a

time agreement between transmitter and receiver is needed. This is a simple task with only one measurement (e.g. radar reflection) where transmitter and receiver is the same object. Otherwise, if reference stations emit signals and the distance is measured by the time of arrival (TOA), clock synchronization is required. When using light or radio, very high resolution and therefore expensive atomic clocks are required. A further challenge is filtering of reflections in the environment, since direct and reflected pulses cannot be distinguished otherwise [High01a, Sriv01].

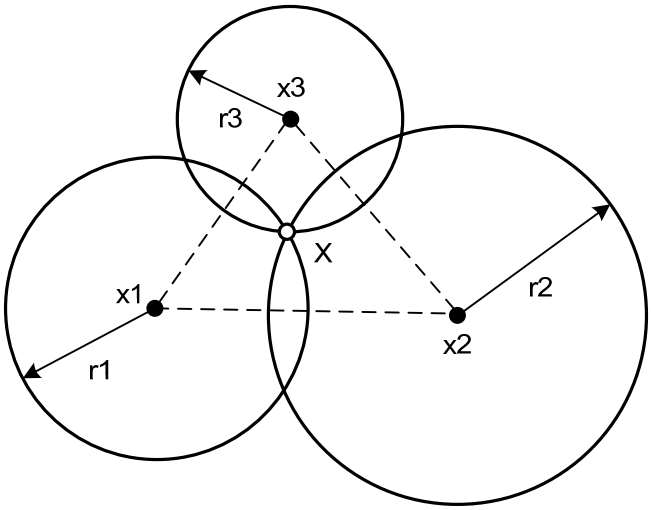


Figure 3: Two-dimensional lateration [High01a]

A related approach is the determination of location by measuring the time differences of arrival (TDOA) of signals emitted simultaneously by three reference stations. This leads to three hyperbolas with constant time differences between two reference stations; the position of X is then defined as the intersection of two such hyperbolas [Barb04].

Last, *attenuation* exploits the fact that an emitted signal's intensity decreases as a function of distance from the emission source. If the strength of an emission as well as the mathematical model that describes the signal attenuation with distance is known, the distance r of an object at a certain point X to the emitting source can be estimated by measuring the signal strength at this point (referred to as received signal strength indicator, RSSI). The attenuation function of radio signals is proportional to $1/r^2$ [High01a]. However, this approach is very problematic as multipath fading and shadowing due to obstacles can cause high variations of the signal strength [Roth02, Sriv01].

8.2.1.2 Angulation

Angulation is similar to lateration, but it uses angles instead of distances for determining the position X of a certain object. For angulation in two dimensions, two angle measurements ($\alpha 1$ and $\alpha 2$) and the distance d between the reference-stations with known position (x1 and x2) are needed (cf. Figure 4). In three dimensions, an additional azimuth measurement is required. The direction from where a signal comes (referred to as angle of arrival, AOA) is determined by antenna arrays consisting of multiple antennas with known separation. Given the geometry and differences in arrival times at these antennas, it is possible to calculate AOA [Barb04, High01a]. Accuracy can be reduced by scattering objects near the reference stations and the object of interest [Sriv01].

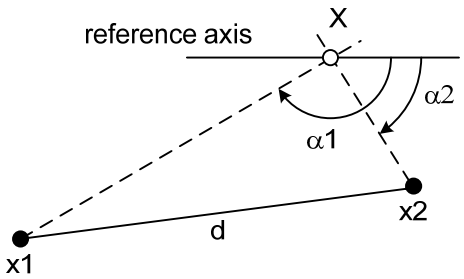


Figure 4: Two-dimensional angulation [Barb04]

8.2.2 Proximity

Proximity sensing techniques determine the object's location when it is "near" a known location or within a certain region. The presence of the object is sensed by exploiting physical phenomena with limited range (from a few centimetres with RFID up to several kilometres with GSM) [Indu03].

[High01a] distinguishes several approaches for sensing proximity. Common approaches are the detection of *physical contact*, e.g. by using pressure- or touch-sensors, or the use of *automatic identification (auto-ID) systems* like RFID- or Barcode-systems. In both cases, the object's location is associated with the known location of the sensing system.

Another approach uses *wireless cellular access points*, whereas the location of a mobile device is associated with that of a base station in range (this range is referred to as cell of origin, COO). This is a very coarse location sensing technique, which heavily depends on the used technology, for example GSM, IEEE 802.11, Bluetooth or infrared. Accuracy can

be improved by the use of *overlapping cell geometries* in a wireless cellular network, where the location of a mobile device can be estimated by the intersection of the geometries of all reachable access points. Figure 5 shows the cell geometries of three access points (x1, x2 and x3), where X is in range of two of them (x1 and x2) and thus is assumed to be in the shaded area. [High01a]

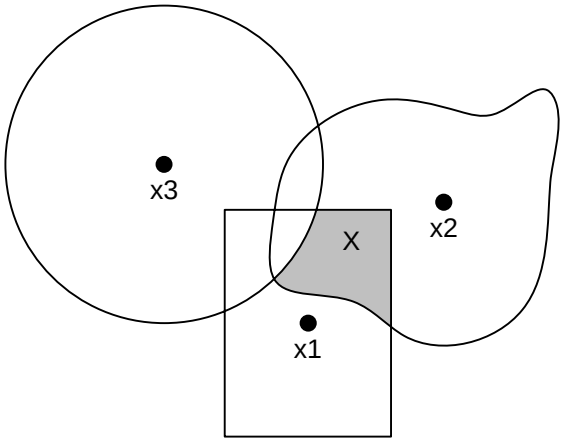


Figure 5: Cell Geometries of access points (cf. [High01a])

8.2.3 Scene Analysis

The scene analysis technique *extracts features of a (usually simplified) scene*, observed from a particular vantage point so as to determine the location of the observer or of objects within the scene. Two types of scene analysis can be distinguished: *static scene analysis*, where observed features are mapped to object locations by the help of databases, and *differential scene analysis*, where differences in successive scenes are used for location estimation. A scene can consist of any physical phenomena, such as visual images captured by mobile or stationary digital cameras or electromagnetic characteristics of IEEE 802.11 access points. The location of an 802.11 device can then be computed by performing a table lookup on the recorded dataset of signal strength values correlating with particular locations. [High01a]

Advantage of this technique is that locations of objects can be inferred without the need for distances or angles. However, a serious disadvantage is that the observer must have access to the features of the scene and that environmental changes may require a reconstruction of the stored features. For example, scene analysis based on signal strength maps requires creating maps in every location where localization should be possible, and these maps must

be recreated whenever the location's physical topology is modified significantly. [High01a, Patt03]

8.2.4 Dead Reckoning

Dead reckoning is a method for *measuring distance and direction from one point to the next along a travel path*, starting from an initial point with known location. It follows from an ancient navigation technique used by sailors in the fifteenth century, before more accurate navigation techniques were developed. This technique is based on the principle that a mobile device continuously samples its own location (e.g. by the use of accelerometers and gyroscopes), and iteratively calculates a model of its movement. [Kuma03]

A key problem with dead reckoning is the inherent bias drift, but it is independent of external signals and thus useful as supplement to a primary location sensing technique. For example, GPS signal outages can be compensated and its accuracy can be improved by taking advantage of wheel- and steering-information in vehicles [Bonn01].

8.2.5 Virtual Space

Location information can also be gathered from virtual space, where *software processes* are used instead of physical location sensors. For example, *applications* like networked calendar systems often hold information about the location of certain people or objects and *operating systems* are aware of directly or remotely logged-in users. Further examples are *networks*, which are mostly organized in IP subnets that correspond to particular physical positions. By the help of the currently active IP address of a stationary or mobile device, the location of this device can be determined. [Indu03]

8.3 Abstracting Location

The preceding chapters showed that there are many types (e.g. symbolic locations or spatial proximities) and sources (e.g. different kinds of physical and virtual sensors) of locations, which differ in accuracy and coverage, among other things. This diversity makes it difficult to develop location-aware applications, wherefore a *middleware layer* is needed which decouples processing of low-level sensor data from high-level applications [Chen00, Indu03].

In order to provide heterogeneous location information (e.g. IP address, serial number of an RFID tag or MAC address of a Bluetooth reference station) acquired from a large number of different sensors (physical, virtual or logical sensors) at application level in a sensor-independent format, most location systems developed so far have a *layered architecture* [Bohn03, Indu03, Leon98a]. Essential components are the *location model* (i.e. a *context representation* for the context-information location, also referred to as location space [Leon98a]), which is needed for representing and correlation locations, as well as the *sensor fusion*, which merges location information of a certain entity into a single, coherent format.

8.3.1 Location Models

There are generally two kinds of location models: *symbolic models* for representing symbolic locations (i.e. abstract symbols) and *geometric models* for representing physical positions (i.e. geometric coordinates). The best choice is determined by the output format of the location sensor [Chen00] (e.g. triangulation is based on the geometric model and proximity on the symbolic model) as well as the application needs (e.g. semantic representation is more effective for reasoning about users' locations [Haza04, Yosh00]).

In both geometric and symbolic models, locations are typically *structured hierarchically* [Chen00]. [Schi95] uses a containment hierarchy in the active map service, where all symbolic locations have a containment relation (e.g., rooms are contained in buildings and buildings are contained in regions). [Nels98] uses a so-called R-tree index, which forms a hierarchy with the smallest rectangles of each object. In [Brum00], a flat geometric model is used, where the relationship between entities is defined by position- and orientation-information.

A helpful *taxonomy of symbolic location models* is given in [Leon98a], where locations are modelled as *sets* and located objects as *members* of these sets; that is, a located object is member of a location if it is physically within the area associated with this location. Such a model is referred to as *simple symbolic model* (e.g. cell model). A symbolic location model, where locations are not allowed to overlap, is referred to as *exclusive symbolic model* (e.g. zone model). In both cases, the model may include a partial ordering of loca-

tion symbols (based on spatial inclusions) which results in an *acyclic location graph* (e.g. domain model) and a *location tree* respectively (cf. Figure 6).

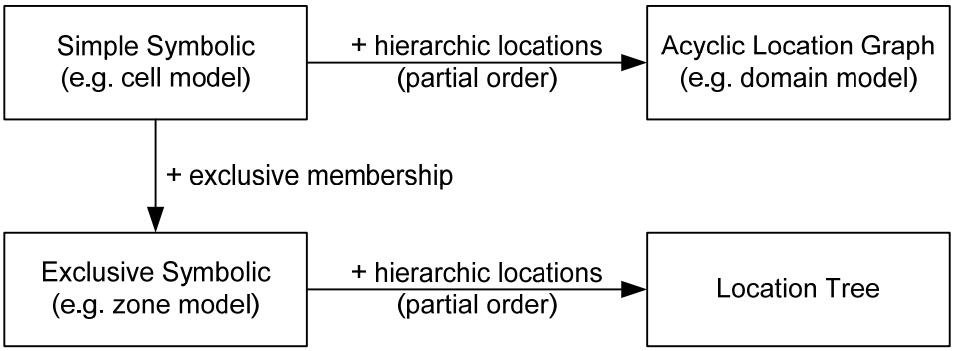


Figure 6: Classification of symbolic models [Leon98a]

Figure 7 shows an example space where the pentagon represents a GSM cell, the circle a Bluetooth cell and the rectangle an IR cell. A *cell* in this context is defined as a specific area representing the coverage of a certain sensor technology. In the *cell space*, cells represent symbolic locations and they are allowed to overlap, whereas the *zone space* contains non-overlapping zones representing symbolic locations, which result from an intersection of all cells and thus potentially increase accuracy (compared with the simple cell model). As zones do not overlap, a located object can only be in one zone at a time. [Leon98a]

[Leon98a] identified some *advantages and disadvantages* of geometric and symbolic models. *Geometric models* have the advantages that they provide a flexible means of retrieving location information and they can typically be reused in many environments without customization. On the other hand, applications have to deal with geometric data and computations, and they often require an additional directory for mapping coordinates into data that is meaningful for applications and humans. *Symbolic models* facilitate application-level reasoning if locations can be referred to by names and they are better suited for proximity-sensors like RFID-systems. Drawbacks are the restricted spatial resolution, which is reduced to the granularity of physical areas associated with particular symbolic locations as well as the bigger administrative effort for creating and managing a large number of symbolic locations.

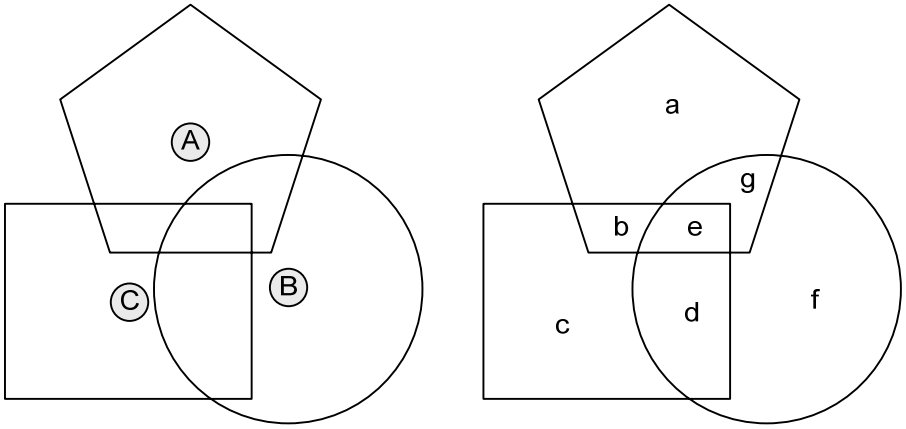


Figure 7: Simple cell space (left) and zone space

[Leon98a] proposes an additional *combined model* (also referred to as semi-symbolic or hybrid model [Domn01]), in which a location contains symbolic names as well as geometric coordinates. A located object is associated with both a symbolic location and a corresponding geometric area. Such a combined model has the advantage that it supports applications requiring both types of location information (e.g. travel planners, which need geometric coordinates for calculating distances and symbolic locations for providing information about the topology [Haza04]) as well as location sensors supporting one representation only [Leon98a].

A combined model similar to the one mentioned above, which preserves topology and distance information between locations, is presented in [Hu04]. It consists of a *location hierarchy* for modelling topological relations and a so-called *exit-hierarchy* for modelling semantic distances.

8.3.2 Sensor Fusion

Since no location sensor takes perfect measurements and is ubiquitously available, the combination of data originating from multiple location sensors (the so-called *sensor fusion*) is a big research issue and of great importance when developing location systems for ubiquitous computing environments, which offer a multitude of location sources (e.g. RFID, Bluetooth and WLAN) where each one has its limitations. Sensor fusion aims to improve accuracy, precision, coverage and robustness to environmental conditions [Lin02], provide seamless transition from one sensor-technology to another one, and reduce costs

by exploiting different kinds of sensors that are often already existent in ubiquitous computing environments (e.g. a WLAN infrastructure) [Bohn03].

[Hall01] distinguishes three levels on which sensor fusion can be performed:

- *On raw sensor data*: Fusion at the level of raw sensor data is potentially the most accurate method, but it requires a high amount of computation and is only possible if the domains of all sensors are of the same type and measure the same quantity.
- *On features extracted from raw sensor data*: Feature extraction in the context of location sensing means abstracting away data provided by a location sensor, which is irrelevant for determining the locations of objects. Thereby, sensors working on different domains can be combined in some cases, but this is not always possible (e.g., serial numbers provided by RFID sensors cannot be fused with signal strength information from WLAN-sensors).
- *On the decision level*: The decision level refers to sensor data that is combined with other data or a-priori knowledge. Each sensor makes an independent *decision* based on its own observations and passes these decisions to a central fusion module where a global decision is made. An example of such a decision is the mapping of RFID tag numbers to symbolic locations. Most existing location systems perform sensor fusion at the decision level [Bohn03, Haza04, High02, Leon98a], where each sensor provides standardized locations represented as a probability distribution; these distributions are combined to compute a new distribution which represents the most probable location of an object. An established technique is *Bayesian filtering* [Haza04], which provides a probabilistic framework for state estimation (e.g. estimating an object's location). However, implementing Bayes filters for location estimation requires time-consuming specification of a *perceptual model* (e.g. the probability of observing certain sensor measurements at a particular location, which is based upon a map of the environment and information about the sensor noise), and a *dynamics model* describing how the system's state changes over time (e.g. a motion model specifying where an object might be at a time t, given that it was previously at a certain location) [Fox03]. Fusion at this level provides an extensible system architecture as the number and types of sensors are not restricted.

Sensor fusion can be quite challenging, because sensors vary in accuracy (e.g. granularity of rooms versus granularity of buildings), sample-rate (synchronous versus asynchronous location notification) and coverage (e.g. available within a building versus available around a few objects only) among others.

Due to the heterogeneity of location sensors in ubiquitous computing environments, sensor fusion at decision level is of particular interest in the context of this thesis. As it operates on standardized (symbolic) locations, its main issue is to cope with overlaps (e.g. the location information provided by one sensor is contained in or overlaps with those provided by another sensor) and inconsistencies (e.g., two sensors provide contradictory location information) [Leon98a, Ward98]. Spatially overlapping location information, which is likely if many different sensors are used, facilitates the detection of inconsistencies and removing outliers on the one hand, and it improves accuracy on the other hand [Leon98b].

8.4 Location System Properties

Location-sensing systems can be classified by numerous properties, which are generally *independent on the techniques and technologies used*. According to [High01a, Indu03, Leon98b, Roth02], a location system can be characterized by the following properties:

- *Physical position versus symbolic location*: A system can provide physical as well as symbolic information. Systems, which provide physical positions, can usually be augmented to provide symbolic location information as well. Purely symbolic location systems are mostly very coarse-grained and they often require multiple readings – e.g. from overlapping proximity sensors – for improving accuracy.
- *Absolute versus relative*: Absolute location systems use a single reference grid for all located objects (e.g. latitude and longitude of GPS receivers) whereas in relative systems each object has its own reference grid (e.g. the rescuer's device reports the avalanche victim's transceiver position relative to itself).
- *Positioning versus tracking*: A positioning system measures its own location with the help of a certain infrastructure. In contrast, a tracking system measures the location of other objects. The latter approach is involved with security issues, as the position information is potentially accessible by any user.

- *Accuracy and precision*: If a system locates a certain object within a range of 10 meters for at least 95 percent of measurements for example, then this distance is referred to as the accuracy of the system and the percentage denotes the precision. Additional to these probabilistic errors, systematic errors can occur, which introduce bias into the measurement and thus reduce accuracy. Accuracy as well as precision can be improved by using multiple sensors.
- *Coverage*: Coverage concerns the size of the area in which the location system works (e.g. worldwide or within a particular room) as well as the number ob objects it is able to locate within that area.
- *Recognition*: Refers to the property of a system to identify located objects and take specific actions based on their location.
- *Limitations*: Limitations describe particular constraints of a system (e.g. that GPS does not work indoors).
- *Openness and Generality*: Openness means that new sensor technologies can be integrated as they are developed, and generality means that the most important approaches to location sensing are covered.
- *Cost*: This does not only address monetary costs for hardware and software, but also time-costs for installation and maintenance of the location system as well as energy- and space-costs.
- *Location-rate*: Refers to the rate at which locations can be calculated.

9 Location Systems and Technologies

Location-aware applications have been developed for a variety of scenarios. Examples are office-applications (e.g. nearest printer-service), museum guides (e.g. provision of information about the nearest exhibit), medical facilities (e.g. tracking of medical staff and patients [Morl04] in case of emergency), home automation (e.g. rooms that adjust light and temperature) and safety (e.g. localizing fire fighters in burning buildings). A multitude of further scenarios comprises navigation systems, tracking vehicles and containers for supply chain management and emergency calls from cellular phones for example [Hart99, Haza04, Sriv01].

In order to *acquire location information* for such location-aware applications, numerous location-sensing systems differing in accuracy, coverage, maintenance costs and rate of location updates have been developed by researches in academy and industry. Goal of this chapter is to present an overview of location sensing technologies and to give a survey and taxonomy of location sensing systems.

9.1 Location Sensing Technologies

Location sensing technologies can be distinguished in coarse-grained and fine-grained. Examples for *coarse-grained systems* are *GPS*, which provides worldwide coverage and accuracy of up to 1 m (with terrestrial reference-stations), and *infrared*-based systems that provide room-grained accuracy with low power. Nevertheless, *GPS* works outdoors only, and infrared requires line-of-sight and does not work under certain lighting conditions.

In the past decade, the wireless communication technologies *IEEE 802.11 (Wi-Fi)* and *Bluetooth* emerged and forced the development of location systems due to their increasing deployment. Localization can be done by exploiting station visibility (i.e. which reference-stations are in range) or by taking advantage of signal strength information. A further emerging technology suitable for location sensing is *RFID*; for example, people with attached *RFID* tags can be identified as they pass strategic points that are equipped with *RFID* readers. Moreover, infrastructures for *mobile phones* and *TV broadcasts* can also be used for localization. [Haza04, Sriv01]

The examples above show that many of the technologies used for location sensing were developed for other tasks like wireless communication or remote identification. *Fine-grained systems* on the other hand are based on technologies explicitly designed for location sensing, and they provide location accuracy of up to a few centimetres. Prominent examples are based on *ultrasonic* (for accurate determination of distances between a certain infrastructure and mobile tags) and *computer vision* (users are not necessarily required to wear any tags, but such systems have difficulties identifying and tracking multiple objects at the same time). Another approach is based on *ultrawideband (UWB) radio*, which has very short pulse durations allowing accurate TOA and AOA measurements. Further advantages are that they do not interfere with the sine wave-spectrum and that they can be produced very small and lightweight. [Haza04, Sriv01]

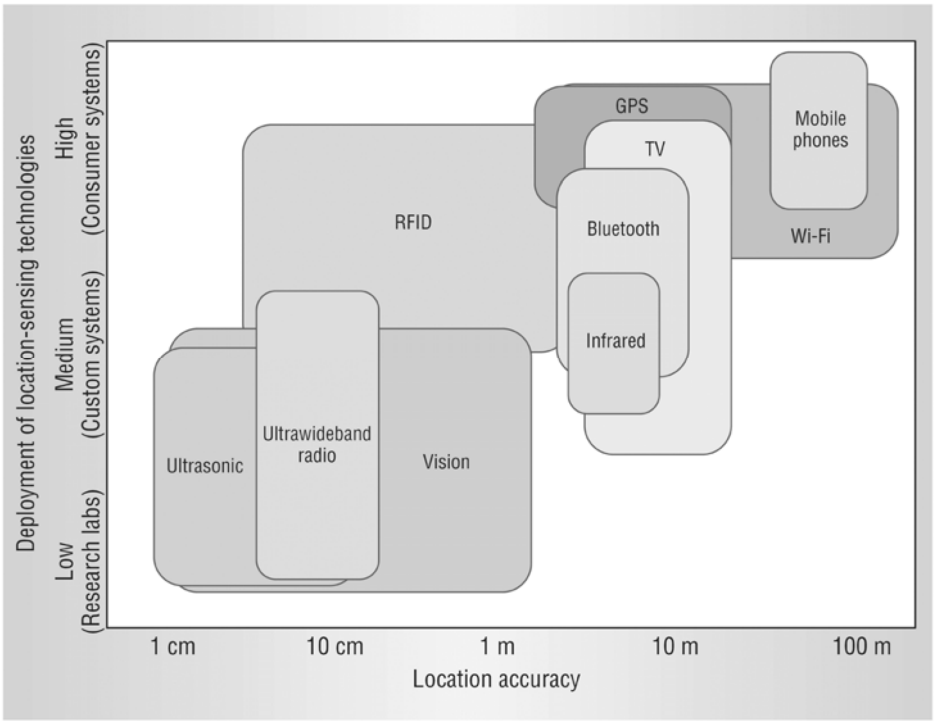


Figure 8: Deployment of location sensing technologies [Haza04]

Figure 8 shows the *deployment of location sensing technologies* as a function of their accuracy. It can be seen that fine-grained systems are less deployed, which can be traced back to the fact that they require expensive infrastructures and have limited coverage respectively. As the location-system presented in this thesis should support granularity at the level of buildings, rooms and positions within rooms, the highly deployed technologies

IEEE 802.11 *wireless LAN (Wi-Fi)*, *Bluetooth* and *RFID* have been chosen to meet this requirement:

- *Wi-Fi* provides high coverage as it is seamlessly available at the whole university campus. Due to the high density of access points, localization at the granularity of rooms is actually possible.
- *Bluetooth* unfortunately is very coarse-grained if no signal strength information is used, but accuracy can be increased to room-level by using many low-power Bluetooth reference stations. Bluetooth is also suitable for detecting proximity between persons.
- *RFID* provides accuracy of several centimetres and is therefore a good choice for localization within rooms. RFID tags are attached to objects; mobile devices equipped with an RFID reader passing by are associated with the known location of the nearest tag. Drawbacks of this technology are poor coverage and high cost.

A brief description of these technologies is given in the following three subchapters.

9.1.1 IEEE 802.11

IEEE 802.11 refers to a set of wireless LAN (WLAN) standards developed by the working group 11 of the IEEE 802 LAN/MAN Standards Committee [IEEE04], which creates standards for local area networks (LANs) and metropolitan area networks (MANs). The original 802.11 standard was released in 1997. It specifies an operating frequency in the 2.4 MHz ISM (Industrial, Scientific, Medical) frequency band and data rates of 1 and 2 Mb/sec. In 1999, the standard was extended with 802.11a (5.2 GHz and 54 Mb/sec) and 802.11b (2.4 GHz and 11 Mb/sec). In 2003, 802.11g was approved, which also operates at 54 Mb/sec, but uses the 2.4 GHz ISM band and is backwards compatible to 802.11b. [Roth02, Schi03]

IEEE 802.11 provides two different modes of operation, namely ad-hoc and infrastructure. The *ad-hoc mode* supports the creation of spontaneous peer-to-peer networks between nearby WLAN devices without the need for any infrastructure. The *infrastructure mode* (cf. Figure 9) allows the creation of large WLAN-based networks. It consists of an interconnection network (the so-called *distribution system*), which is mostly realized as a *wired*

LAN, and a large number of *access points* that are integrated in the wireless LAN and the distribution system as well. In infrastructure mode, mobile 802.11 *stations* (STA) are associated with access points and cannot connect directly to other stations. All stations connected with the same access point build up a *basic service set* (BSS), whose range is determined by that of the respective access point and typically lies between 10 and 50 m. Multiple BSS, which are interconnected by the distribution system, build up an *extended service set* (ESS) which is identified by a service set identifier (SSID). Stations within an ESS can communicate with each other, roaming (i.e. moving from one BSS to another without losing connectivity) is only possible within an ESS. [Roth02, Schi03]

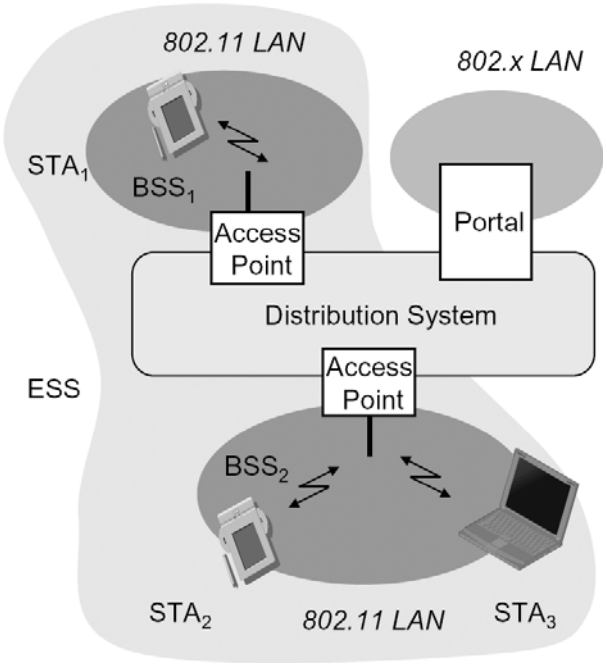


Figure 9: IEEE 802.11 architecture - infrastructure mode [Schi03]

As the IEEE 802.11 standard is very complex and thus allows interpretations, the *Wi-Fi Alliance* [Wifi04] – a non-profit international association with over 200 member companies – certifies interoperability of products based on the IEEE 802.11 specification.

Wireless LANs are a very promising technology for *location sensing* as they provide existing infrastructures without the need for expensive location sensors. This idea has been realized in research projects (e.g. [Bahl00]), and commercial systems with a location accuracy of up to a few meters are available [Aero04, Ekah04].

9.1.2 Bluetooth

Bluetooth is a *wireless ad-hoc communication technology* with low-cost, low power and small form factors, and it is mainly used for portable and personal devices like mobile phones and PDAs. As it uses the 2.4 GHz ISM band like wireless LAN 802.11b/g or microwave ovens, pseudo random frequency hopping between 79 1-MHz-channels is performed with an interval of 625 µs in order to increase robustness against interferences with other devices using the same frequency band. Bluetooth supports both synchronous (e.g. streaming audio) and asynchronous (e.g. TCP/IP) connections and provides optional authentication and encryption. The data rate over the air is about 1 Mb/sec. [Bray02]

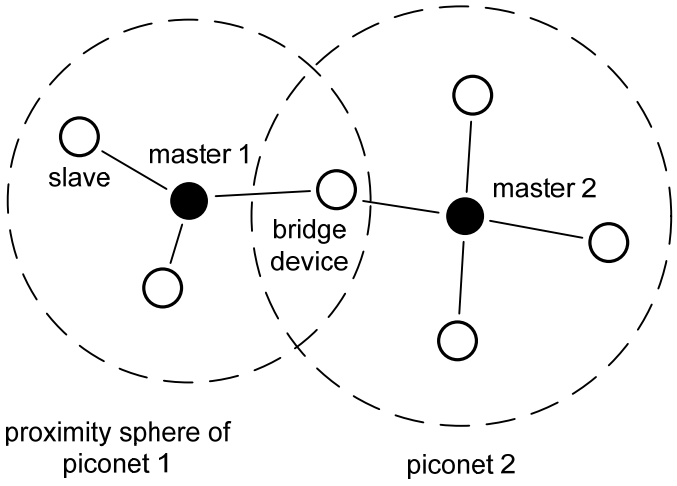


Figure 10: Bluetooth scatternet

Bluetooth provides *ad-hoc communication*, which means, that no a-priori central coordinator is required. Devices in proximity can build an ad-hoc network called *piconet*, which has one master (i.e. the initiator of the piconet) and up to 7 communicating and 255 parked slaves (they can become active whenever necessary) at the same time. Slaves communicate with other slaves via the master only. Several piconets with overlapping devices can form a *scatternet* (cf. Figure 10), where a single Bluetooth unit shares different piconets and bridges the traffic between them ("bridge device"). Any Bluetooth device can perform the role of a slave and/or master. Each Bluetooth device is identified by a unique *48 Bit MAC address*. Additionally, active devices get a 3 Bit locally unique active member address (AMA) and parked members an 8 Bit parked member address (PMA). [Bray02]

A Bluetooth device can discover other devices in proximity by the so-called *inquiry* process. The proximity sphere is determined by the *transmit power classes* of the affected Bluetooth devices: class 3 (1 mW) with a typical range of about 10 m, class 2 (2.5 mW) with a range of 20 m or class 1 (100 mW) with a range of up to 100 m [Hall02, Wemm03]. A master transmits an inquiry sequence on a new frequency every 312.5 µs and listens after each transmission. At the same time, slaves enter *inquiry scan* mode periodically to make themselves available to inquiring devices. If the process succeeds, the master learns the MAC addresses and power classes of all slaves in its piconet. Afterwards, the master can connect to a specific slave with known MAC address (*page process*). Unfortunately, the *inquiry process may take more than 10 s*. [Bray02]

Market segments for Bluetooth are manifold, but its main application area is *cable replacement* (e.g. wireless headsets and synchronization of mobile phones with PCs) [Roth02]. Nevertheless, Bluetooth is also suitable for *location sensing*, as several projects [Blue04b, Hall03] have demonstrated. Location can be determined simply by detecting proximity to a Bluetooth device with known location or by performing triangulation based on signal strength information. The Bluetooth Special Interest Group (SIG) has created a separate working group named “Local Positioning Work Group” which develops a Bluetooth profile (i.e. a specification ensuring application-interoperability between Bluetooth devices supporting the same profile) that allows Bluetooth devices to exchange positioning information [Blue04a].

9.1.3 RFID

RFID (radio frequency identification) is a technology for *contactless identification* and it is quite simple compared with WLAN and Bluetooth. RFID systems are currently used for electronic article surveillance (EAS), baggage labels, contactless access control and product tracking in logistics among others, and they are envisioned to replace UPC and EAN bar codes in the future. They consist of *transponders* (also referred to as *tags*), which store information, and RFID *readers*, which are able to read this information. Both transponder and reader are equipped with an antenna. Numerous types of tags are available, for example so-called “smart labels” fitting on an 80 µm thick paper or glass transponders designed for implantation under the skin of animals. [Fink03]

Figure 11 shows the *functional principle* of an RFID system. The reader sends out radio waves within a range of a few centimetres up to many meters, depending on the power and frequency used. Once a tag comes in a reader's range, it sends its data to the reader. Besides this very simple behaviour, there are also more sophisticated systems with writeable tags and encrypted transmission for example. [Fink03]

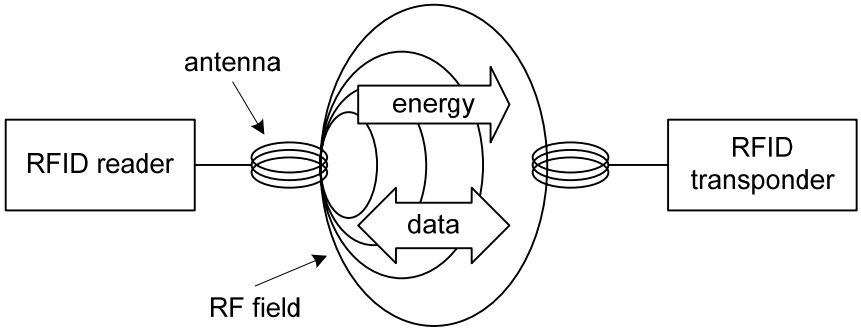


Figure 11: Functional principle of an RFID system [Fink03]

RFID tags can basically be categorized in passive and active. *Passive tags* do not have an own energy source wherefore they obtain energy from the RF field of the reader, either inductively (typically at 125 kHz) or electromagnetically (typically at 13.56 MHz). Once powered, they identify themselves to the reader using various techniques for influencing the RF field. As they require no battery, they have a very long life cycle and are cheaper and smaller compared with active tags, but they have a shorter range. [Fink03, Sriv01]

Active tags on the other hand require a battery. For this reason, they are more expensive, less durable and they have bigger form factors. Active tags operate at higher frequencies, typically in the ISM bands at 868 MHz, 915 MHz or 2.4 GHz. For communication, they often use *modulated backscatter*, where the tags change the reflection characteristics of the RF field by switching on and off a resistor parallel to the transponder antenna in a way that identifies themselves to the reader. They are typically readable and writeable as well. Big advantage is their *higher range of up to 10 m* compared with passive tags. [Fink03, Sriv01]

With respect to *location sensing*, RFID technology has several advantages. First, it is contactless and does not require line-of-sight, which allows tags to be hidden in the environment. Another advantage is its remarkable speed; for example, current active readers can detect up to 100 tags/sec and the response time of a single tag is below 10 ms [IDEN04]. Moreover, some active readers support signal strength measurement, which allows local-

ization that is much more accurate. Several projects [Abow02, Ni03] have successfully used RFID for location sensing, both with and without exploiting RSSI.

9.2 A Survey and Taxonomy of Location Systems

Figure 12 shows a *classification of location systems* in three categories, which is based on the one given in [Roth02]: *Satellite-based* systems for outdoor usage, systems specifically designed for *indoor* usage and systems based on existing *infrastructures* like GSM or WLAN. With regard to this thesis, especially *radio-based indoor location systems* and *WLAN-based infrastructure location system* are of interest. Nevertheless, the developed system is extensible with any location system and technology.

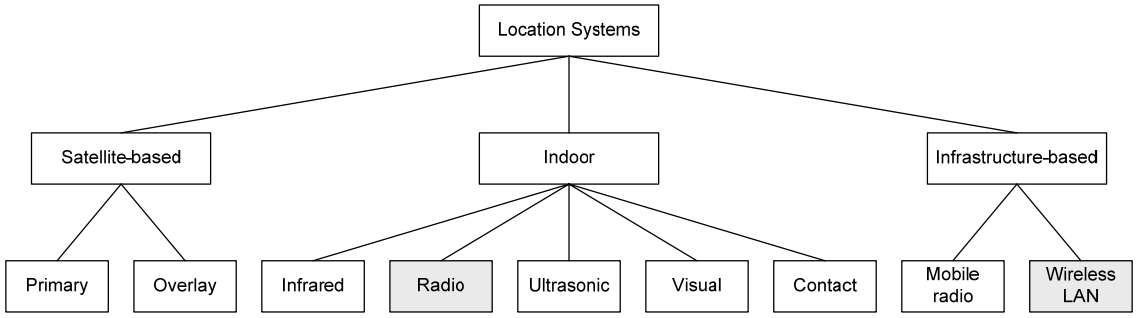


Figure 12: Classification of location systems

The following subchapters present concrete systems that are representative for these categories. However, this classification is not intended to be complete. For example, *electromagnetic or mechanic position tracking*, which are common in common in virtual reality environments [High01a], are not considered in the following.

9.2.1 Satellite-Based Location Systems

The *Global Positioning System (GPS)* is a satellite navigation system developed by the U.S. Department of Defence (DoD). The system consists of 24 satellites in six orbital planes at a height of 20.200 km; at least five satellites are potentially visible anywhere on earth. The GPS satellites continuously transmit digital radio signals, containing information about the satellites' location and the exact time, which is given by on-board atomic clocks. Based on this information, GPS receivers determine their distance to three satellites via *time-of-flight*-calculation and perform *triangulation* in order to calculate their own position; a fourth satellite is used for time synchronization. GPS provides an accuracy of

about 25 m horizontal and 43 m vertical. A European satellite-based navigation system called *Galileo*, which works independently from GPS, is expected to be launched in 2008. [Roth02]

To improve accuracy, several *overlay-systems* have been developed as a supplement to *primary systems* like GPS. For example, *differential GPS (dGPS)* uses additional fixed reference stations, which broadcast differences between internally computed and measured distances to GPS satellites, by which accuracy is increased to 1-3 meters. A similar system is the *Wide Area Augmentation System (WAAS)*, but it uses additional satellites instead of terrestrial stations. [Roth02]

9.2.2 Indoor Location Systems

Drawback of all satellite-based systems is the *required line-of-sight to the satellites*, which makes indoor usage impossible. In order to localize objects indoors, dedicated indoor location systems are required. Various systems have been developed, which differ in accuracy and costs among others, and can be categorized by the technology used. In the following subchapters, representatives of each category are presented.

9.2.2.1 Infrared

The *Active Badge Location System* [Want92] is one of the first indoor location systems. Each person wears a small badge (i.e. the infrared transmitter) which emits a unique user-identifier every 15 seconds. Infrared sensors around the building collect this data and send it to a central server, which provides an API (application programming interface) for using this data. The Active Badge Location System is a proximity-based system and it provides symbolic location information at the granularity of rooms. As with all infrared-based systems, drawbacks are the required line-of-sight and difficulties with fluorescent lighting or direct sunlight [High01a], wherefore a large number of readers are required to achieve line-of-sight to each tag [Sriv01].

9.2.2.2 Radio

Radio-based systems have a major difference compared to infrared systems, namely the property that *radio signals penetrate walls*. Moreover, radio signals allow *measurement of signal strength*, a measure for the distance between sender and receiver, which allows lo-

calization even in three dimensions if the sensors are mounted in different heights [Roth02]. A project using custom hardware is *SpotON* [High00], a three-dimensional location sensing system based on radio signals. The SpotON system uses *ad-hoc lateration* [High01b], which means that it locates objects without central control by performing RSSI measurement for estimating distances between tags. In order to get absolute locations, some of the tags must be configured with their known absolute positions. [High01a]

Another radio-based system is *LANDMARC* [Ni03], which uses *RFID* for locating objects inside buildings. It is based on the principle that a large number of RFID readers is placed in the environment at known positions and the location of a tag passing by is associated with that of the reader in range. In order to increase accuracy and robustness against environmental dynamics, additional fixed location reference tags are placed in the environment and thus helping to calibrate the location of tags by exploiting RSSI. Drawbacks are the need for a large number of expensive RFID readers and the accurate placement of readers and reference tags.

Due to its high deployment and comparatively little costs, *Bluetooth* became interesting for location sensing. Several issues have to be considered when using Bluetooth for localization. First, its range is between 10 m and 100 m, depending on the power class of the device. Second, it may take more than 10 s for a Bluetooth device to identify others in range; moreover, devices are sometimes not found during inquiry. Accuracy could be increased by using the optional RSSI function, but tests have shown that it is an unreliable source of information [Hall03]. A commercial coarse-grained location system is the *BlueTags System* [Blue04b], which consists of small Bluetooth tags, a certain tracking software and Bluetooth access points. If tags move around in the environment, they are detected by the access points and unique IDs are transmitted to a central location server.

9.2.2.3 Ultrasonic

By using ultrasonic, the distance between transmitter and receiver can be determined very accurately. *Active Bat* [Hart99] for example uses an ultrasonic *time-of-flight* lateration technique, which provides accuracy of about 10 cm. Each user wears a device (the so-called bat) which emits a short ultrasonic impulse to a grid of sensors mounted on the ceiling if it is requested via radio by the controller. At the same time, the controller also sends

a reset-signal to the receivers, which measure the time interval from the reset to the ultrasonic impulse, and compute their distances to the bat out of it. These distance measurements are then sent to a special location server, which afterwards performs lateration. Although such a system is very accurate, it requires a large and expensive sensor infrastructure around the ceiling. [High01a, Roth02]

An inverse approach is the *Cricket Location-Support System* [Priy00], which uses an infrastructure with ultrasonic emitters (so-called beacons) and integrates receivers in the mobile devices. Accuracy is in the range of few meters. In order to acquire *time-of-flight* data, each beacon simultaneously transmits a radio signal, which uniquely identifies its location, and an ultrasonic impulse. Determination of time-of-flight and respective distance as well as *lateration* are performed on the mobile devices. If just one beacon is in range, proximity information can be used for localization. As with all ultrasonic-based systems, disadvantages are their lower range compared with radio-based systems as well as the fact that ultrasonic pulses are reflected on obstacles. [High01a]

9.2.2.4 Visual

A further class of systems is based on *computer vision* to localize objects. The *EasyLiving* project [Brum00] for example uses multiple stereo cameras to identify and localize people within rooms [Krum00]. Advantage of vision-based systems is that people do not need to wear any devices, but such systems require high processing power and provide low scalability with respect to increasing scene complexity and motion of tracked objects [High01a].

Another approach is the use of *Visual Tags* [Star97], which are patterns of red and green squares. Information (e.g. the IDs of users) is coded in the arrangement of squares, which have a fixed size and thus enable determination of distance and orientation to fixed cameras with known position [Roth02]. Drawbacks besides high processing power are aesthetic considerations regarding the tags.

9.2.2.5 Contact

Less common approaches of localization are systems *detecting proximity by direct contact*. For example, the *Smart Floor* [Orr00] detects footsteps by pressure sensors embedded in the floor and hence identifies and localizes people by means of their footstep force profiles.

Although the Smart Floor has the advantage that people do not need to wear any tags, it has a poor scalability and requires expensive pressure sensor grids [High01a]. A similar approach is the *Active Floor* [Addl97].

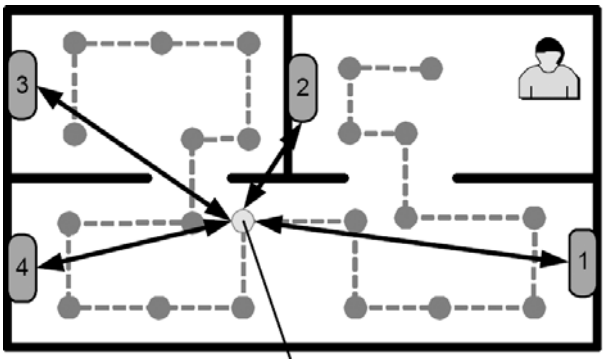
9.2.3 Infrastructure-Based Location Systems

Infrastructure-based systems take advantage of *already existing infrastructures*, without the need for expensive sensor hardware. Especially cellular networks are suitable for location sensing as they provide at least coarse COO measurement or TOA and AOA for localization that is more accurate. [Roth02]

Important examples are *mobile radio infrastructures* because they provide nearly complete coverage. As the mobile network operator exactly knows which mobile phone is connected with which base station, COO is always possible and therefore provides coarse localization with an accuracy of about 100 m within congested areas and up to 35 km in the country. Localization that is more accurate is possible by exploiting additional features of GSM, as it is done by the *Mobile Positioning System (MPS)*. It requires no modifications to mobile phones and only marginal infrastructural changes. Besides *cell identity*, *direction* of antennas and *timing advance* (i.e. a mechanism, which forces mobile devices to start their transmission the sooner the farther they are away from the base station) are exploited. If at least four stations are in range, location can be determined with an accuracy of 50-150 m [Roth02]. A related approach is *Enhanced 911 (E911)*, which enables localization of 911 emergency calls with an accuracy of 50 to 100 m within the USA in most cases, and it should be completed by fall 2005 [FCC04].

More interesting with respect to this thesis are *wireless LAN infrastructures* as they provide significantly higher accuracy. An example is *RADAR* [Bahl00], a *scene-analysis* tracking-system based on an 802.11 WLAN infrastructure, which has been realized with the *Nibble Location System* [Nibb01]. Drawback of such systems is that they have to be calibrated. In the whole area where localization should be possible, signal strength measurements of all WLAN access points in range must be performed in short distances. This data is then saved in a table, where the signal strength information of each access point is stored with the coordinates <x,y> and (eventually) the orientation d of the measuring device (cf. Figure 13).

As the orientation has a significant impact on the measured signal strength [Bahl02], measurements should be performed in multiple directions. For each combination of location and orientation, [Bahl02] collected at least 20 signal strength samples. In order to localize a mobile device, the table entry that matches the observed signal strength of this device best is selected. A possible approach is to select the entry whose *Euclidean distance* between the observed and recorded signal strength measurements is a minimum [Bahl02]. Although this approach allows quite accurate localization, it requires time-consuming creation of such tables and they furthermore must be recreated whenever the location's physical topology is modified significantly [Roth02].



x/ m	y/ m	d/°	SS_1/ dBm	SS_2/ dBm	SS_3/ dBm	SS_4/ dBm
1.0	3.5	0	20	10	18	25
2.0	3.5	90	25	15	17	25
2.5	3.0	90	15	18	16	16
2.5	1.5	180	6	35	18	20
2.0	2.5	0	12	10	22	14
...	...	...	...	...	...	...

Figure 13: The Nibble Location System [Roth02]

A similar commercial system is the *Ekahau Positioning Engine* [Ekah04], which works in three dimensions <x,y,floor> and achieves an accuracy of up to 1 m. Ekahau uses a patented *scene-analysis* technique based on RSSI and supports most available Wi-Fi adapters. As the Ekahau Positioning Engine is used for the location-system presented in this thesis, it is discussed in chapter 11.4.1 in more detail. A similar commercial system is the *Aeroscout System* [Aero04].

9.3 Comparison and Motivation

The following table according to [High01a, Rous02] gives an overview of some interesting properties of the location systems presented in the preceding chapter.

Table 1: Location systems properties

System	Technique	Accuracy	Type	Coverage	Limitations
dGPS	lateration (time-of-flight)	1 - 3 m	physical	global	not indoors
Active Badge	proximity (cellular)	room	symbolic	rooms	sunlight and fluorescent interference
SpotON	lateration (attenuation)	depends on density of tags	physical	cluster coverage	attenuation less accurate than time-of-flight
BlueTags	proximity (cellular)	20 m	symbolic	network coverage	Bluetooth access points required
Active Bat	lateration (time-of-flight)	10 cm	physical	rooms	low range, reflection of ultrasonic pulses
Cricket	lateration (time-of-flight)	few meters	symbolic	single location	low range, reflection of ultrasonic pulses
EasyLiving	scene analysis (vision)	variable	symbolic	rooms	ubiquitous public cameras
Smart Floor	proximity (physical contact)	spacing of pressure sensors	physical	rooms	recognition may not scale to large populations
RADAR	infrastructure (wireless LAN)	3 - 4,3 m	physical	network coverage	wireless network and training required
MPS	infrastructure (mobile radio)	50 - 150 m	physical	network coverage	coarse localization in the country

The location systems presented in chapter 9.2 are all based on a *single technique and technology*. As Table 1 shows, such systems differ in accuracy and coverage and they have certain limitations. For example, GPS works outdoors only and ultrasonic-based systems require an expensive infrastructure. In order to provide *ubiquitous location sensing at different granularities*, an easily extensible system that supports an arbitrary number of differing

location-sensors at the same time is needed. Moreover, the system should not be restricted to a particular field of application.

In the past few years, some research has been done to *facilitate location-aware computing by the development of software-infrastructures*, which hide sensor-details from the application developer and provide reliable location services that do not suffer from deficiencies of single location sensing technologies.

The *Location Service* [Abow02] for example is a software-framework addressing these issues. It works on physical positions, namely three-dimensional coordinates and orientation, and it provides a uniform way for handling multiple location sensing technologies at the same time. It is divided into the three layers *acquisition* of location information from multiple sensors, their *collection* and fusion separately for each entity and *monitoring*, which refers to a mechanism that transforms the uniform location information to an application-specific form (e.g. symbolic locations). A probabilistic positioning service is presented in [Bohn03], which is also based on a hybrid location model with fixed cell size and decision-level sensor fusion. A further ongoing project is described in [Röde04], where a four-layered framework based on a hybrid location model is presented. Core of this system are distributed location servers (e.g. one per organization), which handle database access and perform complex computations.

Although many ideas of the systems mentioned above have influenced the development of the location system presented in this thesis, none of them meets all requirements: No system supports dedicated proximity sensors and their combination, a location history as well as a location model and fusion based on symbolic locations. Moreover, these systems are frameworks especially designed for providing location information via certain APIs. In contrast, the location system presented in this thesis is fully integrated in a context middleware described in chapter 11.1, which provides a flexible and transparent way of communication between the location systems and context-aware applications on the one hand, and facilitates scalability on the other hand.

Of particular interest with regard to the architecture of the developed location-system is the *layered model* presented in [Indu03, Leon98a], which consists of the following five layers:

- *Sensors*: Represents multiple logical, virtual and physical location sensors

- *Reception*: Delivers the location information from the sensor-layer to the abstraction-layer
- *Abstraction*: Transforms sensor-dependent location information into a uniform format
- *Fusion*: Merges uniform location information of a particular entity into a single, coherent location
- *Application*: Provides flexible and scalable interaction mechanisms for interactions between the location system and its clients

The *Location Stack* [High02] refers to an abstract layered model intended to facilitate software engineering for location sensing, similar to the seven-layer-Open Systems Interconnection (OSI)-model for networks. It consists of a seven-layer design with the capability of handling different sensors and fusing their readings by exchangeable fusion algorithms.

10 Architecture for Seamless Integration of Location Information

This chapter gives an *architectural overview of the developed location system*, which is derived from a set of requirements. Afterwards, the used location model is presented and the integration of sensors as well as the fusion of sensor data is discussed. This chapter should give an overview only, wherefore implementation details are presented in the subsequent chapter. At the end of this chapter, properties of the system are listed.

10.1 Requirements Analysis

As mentioned in the problem statement given in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.**, goal of this work is the development and prototypical implementation of a location system that supports automated acquisition of location and proximity information acquired from multiple sensors, as well as their uniform provision at application level. Several *requirements* can be derived from these goals, which are listed in the following:

- *Location information*: As knowledge of nearby people is also of great importance for context-aware applications, the location system should not only provide absolute location of users, but also relative locations to other users (i.e. spatial proximities between them).
- *Sensor technology*: It should be possible to integrate arbitrary location and proximity sensors into the location system.
- *Multiple sensors*: Because no single sensor technology offers all capabilities (e.g. accuracy at the granularity of positions within rooms and coverage of the whole university campus at the same time) to support the broad spectrum of ubiquitous applications [High02], it is necessary to consider multiple differing location and proximity sensors. Their measurements have to be merged by certain sensor fusion algorithms.
- *Context-aware applications*: There must be mechanisms to notify context-aware applications about context changes, namely if the locations of users or spatial proximities between them have changed.

- *Location model*: A flexible model for representing locations, which is independent of application domains and sensor technologies, is needed. Such a model should correlate locations to each other and it should support queries like “return a list of all objects at a certain location” or “return the current location of an object”.
- *Extensibility and flexibility*: There should be the possibility to extend the location system with arbitrary future location sensors. Moreover, the system must be flexible with respect to exchanging certain algorithms (e.g. fusion algorithms) and adding further locations.
- *Transparent provision*: Context-aware applications must be able to use location information without having knowledge about the underlying sensor techniques and technologies. To achieve this, sensor-dependent positions must be transformed into a uniform, sensor-independent format. This is also a prerequisite for extending the system with additional sensors.
- *Scalability*: Several kinds of scalability can be distinguished. First, the system should be scalable with regard to an extension of the region where localization is possible. Second, the system should also scale with the number of sensors integrated as well as the number of users using the location service. Moreover, the implementation of the server-side modules has to be scalable, which mean s that they must be efficient and coupled in a way that they can be distributed among multiple servers.
- *Accuracy and Coverage*: Coverage is given by the union of the ranges and accuracy by the number and types of integrated sensor technologies.
- *Robustness*: As clients can go online and offline, sensors can be plugged and unplugged and environmental conditions (e.g. availability of WLAN) can change any time, the system must be robust enough to work under these conditions in a stable way.
- *Client resources*: In order to save power of mobile devices, hide location sensing from the user at the best and allow clients with limited processing power to participate in the system, the client-side must be kept as simple as possible in order to

minimize CPU utilization, and the communication between client and server should be reduced to a minimum.

- *Administration*: Efforts for administrating and maintaining the system should be kept as little as possible.
- *Context history*: Context-aware applications may require a context-history, wherefore location and proximity history should be stored.
- *Platform independency*: It should be possible to execute the location system on various platforms, e.g. Windows and Linux.
- *Privacy*: The user must have full control about provision of location information, which means, that he cannot be localized if the location service is not running on his computer.

10.2 Architecture Overview

With these requirements in mind, the software architecture of the location system has been developed. The whole system is divided into *modules*, which are integrated in a Java-based *context middleware* described in chapter 11.1. It provides a flexible, distributed architecture, which hides communication details from developers. Communication between the layers is event-triggered and performed via the middleware. As it is Java-based, the location system is basically *independent of a certain platform*. Nevertheless, native sensor implementations require a re-implementation for each platform.

In order to reduce complexity and facilitate extensibility, a *layered architecture* with clearly defined interfaces has been chosen. Because of the context middleware, communication between these interfaces is based on asynchronous events. Inspired by [Indu03] and [Leon98a], the architecture consists of the following four layers:

- *Encapsulation*: This layer contains the integration of arbitrary native, proprietary sensor implementations into the context middleware and the transmission of sensor-dependent position and proximity information to the server. Transmission is performed asynchronously every time a sensor records new position information. For each type of client-side sensor such a module is necessary. However, some sensors may only need a server-side implementation.

- *Abstraction*: The abstraction layer is responsible for receiving sensor-data of all clients, transforming them into standardized location and proximity information, and transmitting them to the corresponding *sensor fusion*-modules. For each type of sensor, such a server-side module is necessary; they are typically the counterparts of the client-side *sensor encapsulations*. After abstraction, sensor-details are hidden. The abstraction layer relies on a *location model*, which represents and correlates locations. Because of the fact that location information of numerous clients has to be processed, scalability with regard to the number of sensors and users is crucial at this layer.
- *Fusion*: This layer collects and merges standardized location and proximity information acquired by multiple differing sensors, separately for each client. As the fusion works on standardized data, it is independent of the underlying sensor technology. Every time the location or proximity of a certain client changes, the context-aware application is notified. As with the abstraction layer, scalability of fusion algorithms is very important.
- *Application*: This is the highest layer and it is notified about location- and proximity-changes of particular users. An example context-aware application is described in [Fers04a, Fers04b, Oppl04], which takes advantage of location and proximity information in order to facilitate group-interaction.

Figure 14 shows the interaction of the four layers. Each rectangle represents a single module in the context middleware. In order to keep the client-side as simple as possible, a *client/server-architecture* has been chosen. Clients just acquire location information from various sensors, which is done by the *encapsulation* layer. Sensor-dependent data is then transmitted to the server, where *abstraction* and *fusion* are performed. Every time the location or proximity of a certain user changes, respective events are thrown; context-aware applications, which are integrated in this context middleware, are therewith informed about context changes. As the interfaces between the layers are well-defined, modules such as the location sensor fusion can be replaced by ones that are more sophisticated. For scalability reasons, it is possible to distribute server-side modules among multiple servers. Finally, the *application* layer typically consists of a client- and a server-part.

Abstraction requires a *database*, where the mapping of physical positions to standardized locations as well as the location model for correlating these locations is stored. Because of the abstraction of sensor-dependent position information, the system can easily be extended by *additional sensor technologies*, just by implementing the corresponding abstraction- and encapsulation-modules. The database also saves a *context history*, namely a recording of all locations for each user and spatial proximities between users.

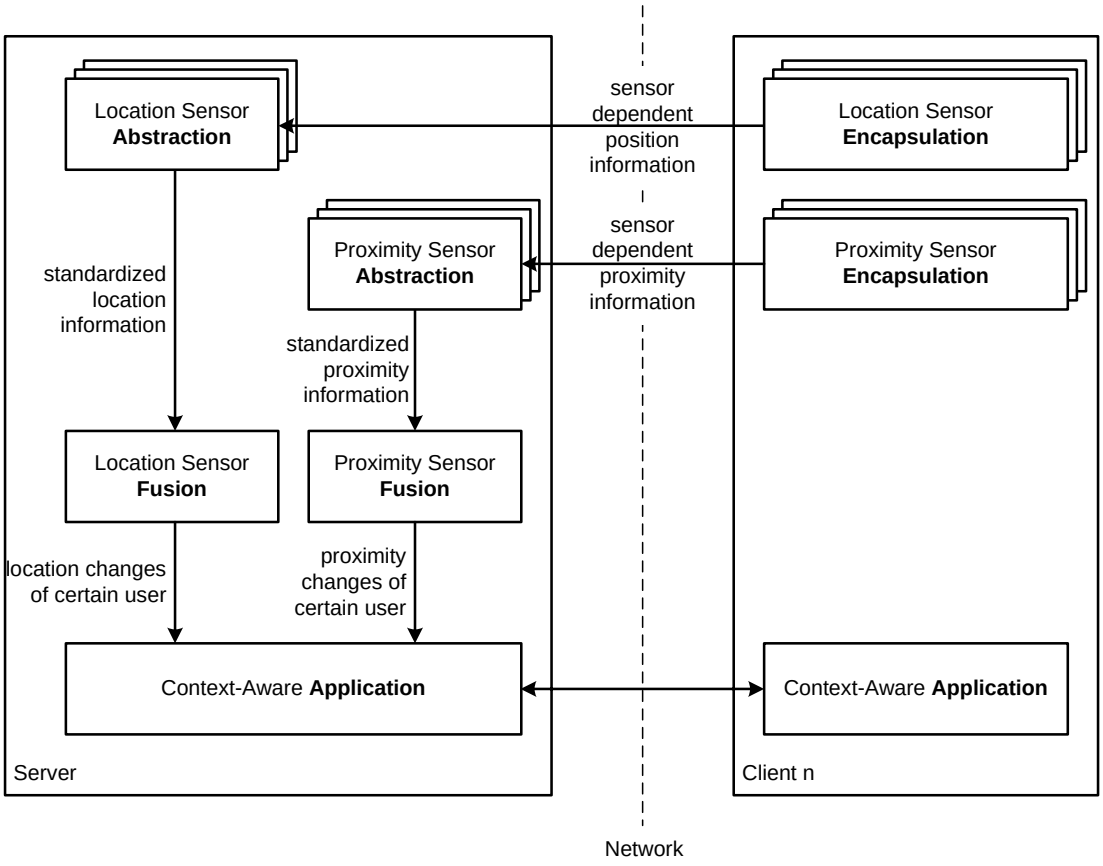


Figure 14: Architecture overview

10.3 Location Model

A flexible data model is required for *representing locations* of objects. An important requirement is its independence of application domains and sensor technologies. As described in chapter 8.3.1, two types of location models can be distinguished: *symbolic models*, which represent locations as abstract symbols, and *geometric models*, which represent locations as geometric coordinates.

A *symbolic model* has been chosen for the location system presented in this thesis, because pure geometric locations are not suitable for application level reasoning and geometric information is not needed for the addressed scenarios. Moreover, various location sensors are not able to provide geometric coordinates (e.g. proximity-based systems like RFID). A *zone-model* is used which contains non-overlapping named zones representing symbolic locations that are not allowed to overlap.

In order to reduce complexity and facilitate administration, a *hierarchical tree-based structure* similar to the one introduced in [Schi95] has been chosen. Locations are represented at multiple granularities and they have a containment relation (e.g., workplaces are contained in rooms and rooms are contained in buildings).

The hierarchy consists of *four levels of granularity*, which have general names in order to be suitable for both indoor and outdoor environments:

- *Area*: Refers to a room within a building or a room-equivalent area like a part of a corridor or a specific area outdoors.
- *Level*: Refers to a floor within a building or a floor-equivalent level outdoors.
- *Section*: Refers to a building or a building-equivalent outdoor section (e.g. a park).
- *Region*: Refers to a union of multiple spatially related sections (e.g. a university campus comprising multiple buildings and outdoor sections).

An additional fifth type of location is introduced. These so-called *semantic locations* can appear at any level in the hierarchy and they can be nested. Semantic locations are used in two ways, namely to *represent locations below area-granularity* (e.g. if a room is split up in sub-areas like workspaces) on the one hand and to *group locations* to units with a certain meaning on the other hand (e.g. a department which consists of several rooms or a faculty which comprises several buildings). As a *location tree* is used, locations are not allowed to be contained in multiple semantic locations. The hierarchy of locations as well as their types is stored in the database.

Figure 15 shows an example location model, which comprises all five types of locations. The bars on the right side mark the accuracy of different sensor technologies, which will be discussed in the subsequent chapter.

Drawback of this model is that *no distance information* is represented, wherefore it is not possible to determine how far two located objects are away. However, this information is considered less important, because the location system addresses limited regions like a university campus. Distance information could be provided by saving distances between all locations as it is described in [Hu04]. Nevertheless, this would lead to much higher administrative effort, as the adding of a single location would require a determination of distances to all other already saved locations. An alternative solution would be to associate all symbolic locations with geometric coordinates.

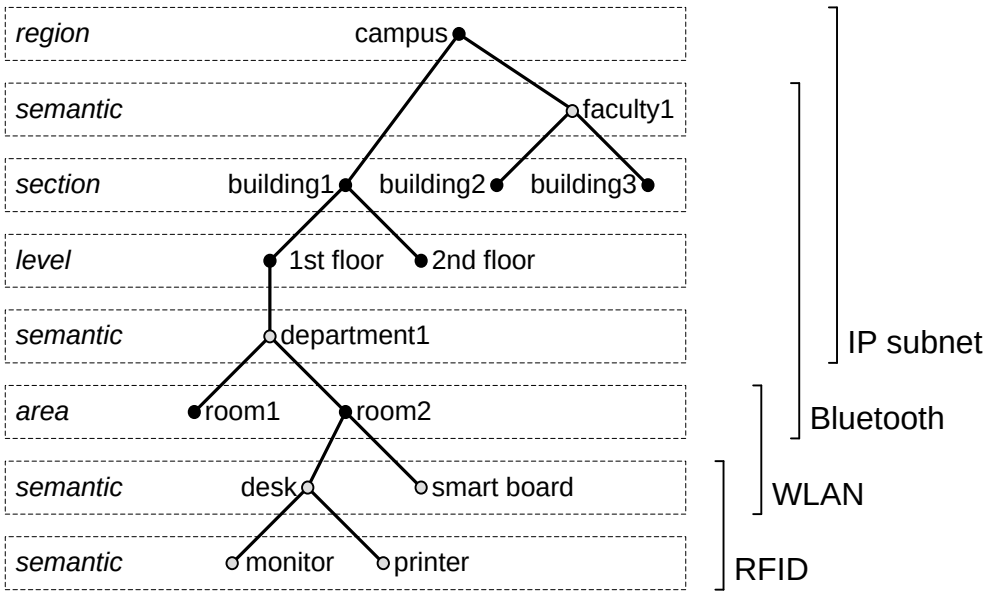


Figure 15: Exemplary symbolic location containment hierarchy

10.4 Sensors

The location system supports two different types of sensors: *location sensors* for acquiring location information and *proximity sensors* for detecting spatial proximities between users.

Each sensor implementation typically consists of two parts: a client-side implementation (*sensor encapsulation*), which is responsible for acquiring low-level sensor-data and transmitting it to the server (e.g. MAC addresses of Bluetooth reference stations), and a server-side implementation (*sensor abstraction*), which transforms them into a *uniform and sensor-independent format*. In the case of location sensors, sensor-dependent positions are mapped to *symbolic locations*, where some positions (e.g. a Bluetooth reference station) may cover multiple symbolic locations. In the case of proximity sensors, low-level sensor data are transformed to *IDs of users* that are in spatial proximity. Due to the trans-

formation into a standardized format, arbitrary location- and proximity-sensors differing in technique and technology (outdoor- and indoor-sensors, positioning- and tracking-systems etc.) can be integrated, just by implementing these parts. From this point on, the underlying sensor-technology is hidden.

The respective *sensor fusion algorithms* (one for location- and another one for proximity information) are then triggered with this sensor-independent information of a certain user, detected by a particular sensor. Such notifications are performed for each sensor measurement, which occur *asynchronously* (i.e. measurements may arrive at irregular intervals).

With respect to *robustness*, sensor abstractions are responsible to detect when a certain sensor is no longer available on the client side (e.g. it has been unplugged by the user or did not deliver measurements for a certain time), or when position- or proximity-information could not be determined any longer (e.g. an RFID reader cannot detect tags or the Bluetooth device of another user could not be found again during inquiry). In that case, the *sensor fusion* is notified about the stoppage of this sensor of a particular client. Special cases like plugging and unplugging sensors on the client-side also affect native sensor implementations as well as *sensor encapsulations*, which have to detect such special cases and react accordingly (e.g. by starting localization if a certain sensor is plugged). Sensor implementations are also a *privacy* issue, as they must ensure to stop tracking a certain user if his location service is not running any longer (e.g. if a WLAN tracking system is integrated which is able to track users that do not have the client-side location service running).

10.4.1 Location Sensors

Any sensor that is able to determine the location of a particular user is referred to as *location sensor*. In order to sense physical positions, the *location sensor encapsulation* asynchronously transmits sensor-dependent position information to the server. The corresponding *location sensor abstraction*-modules collect these physical positions delivered by the sensors of all users, and perform database-lookups in order to retrieve the associated symbolic locations. At this point, only the most probable measurements of the sensors are taken into account (e.g. the symbolic location of the RFID tag with the highest RSSI only). One physical position may have multiple symbolic locations at different levels of granular-

ity in the location hierarchy assigned to, for example if a sensor covers several rooms. If such a mapping could be found, an event is thrown in order to notify the *location sensor fusion*-module about the symbolic locations a certain sensor of a particular user has recognized.

Four different location sensors have been implemented, which differ in accuracy, coverage, update-rate and administrative effort and thus provide a means for evaluating the versatility of the location system, namely its *seamless integration of various location sensors*:

- *RFID sensor*: This *proximity-based* sensor is the most accurate one. It is based on RFID readers, which are plugged into client computers, and active RFID tags that are placed anywhere in the region. The RFID readers periodically scan for tags and transmit the serial numbers of the found tags to the server. This type of sensor provides accuracy of approximately *10 centimetres*. Administrative effort is low, as for each new RFID tag its serial number has to be assigned with the according symbolic location only. Drawbacks are high hardware costs and little coverage. The location-rate, i.e. the time delay until a tag is detected by the reader, is about *one second*.

- *WLAN sensor*: This type of sensor provides region-wide coverage if WLAN is ubiquitously available. A commercial *scene analysis-based* Wi-Fi location-tracking system has been integrated, which works with all common Wi-Fi devices and provides accuracy of about *2 meters*. As it is purely software-based, no additional hardware is needed. Localization of clients is done on the server-side by the help of special client software, which transmits signal strength information to the tracking system. Besides high license costs for a large number of clients, a big disadvantage of this approach is the very time-consuming calibration, as the tracking system requires to take calibration points with an interval of less than five meters and anywhere where signal characteristics change heavily (e.g. in front of and behind a door). The interval between two location updates is approximately *two seconds*.

- *Bluetooth sensor*: This sensor is based on the principle that clients continuously perform inquiries and transmit the found Bluetooth MAC addresses to the server. It relies on Bluetooth references stations distributed in the environment and Blue-

tooth-enabled clients (e.g. with plugged Bluetooth dongles). As no signal strength-information is exploited, the Bluetooth sensor is *proximity-based*. All MAC addresses found during inquiry, which belong to reference stations, are mapped to according symbolic locations. For this reason, accuracy is quite low, but location sensor fusion can increase it to *granularity of a few rooms* by using multiple overlapping reference stations. However, in the worst case, a single reference station covers multiple buildings. A big disadvantage of Bluetooth technology in general is the long inquiry time, which increases the interval between location updates to *up to 30 seconds* in the current implementation. A further disadvantage is the need for calibrating the environment where Bluetooth localization should be possible, similar to the WLAN approach (i.e. for each symbolic location all reachable Bluetooth reference stations must be determined; each station is then associated with the symbolic locations it covers).

- *IP subnet sensor*: This *virtual sensor* has complete coverage, because it is available wherever the location service is available. The location of clients is identified just by determining which IP subnet their IP address, which is sent periodically to the server, belongs to. If just WLAN connection is available, this sensor is not very useful as wireless LANs typically have one IP subnet per ESS; in that case, the IP subnet-sensor only provides the information that somebody is anywhere where WLAN is available (e.g. on the university campus). If wired LAN is used, usually a more accurate localization is possible (e.g. anywhere on a certain *department* which owns a dedicated IP subnet). Administration is easy, as each subnet has to be assigned with the symbolic locations it covers only. The location-rate depends on how often a client transmits its IP address to the server.

The accuracy of each sensor is visualized in Figure 15. Table 2 subsumes the properties of these four technologies, whereas a more detailed description of the implementation is given in chapter 11.4.

Table 2: Comparison of implemented location sensors

Sensors	Technique	Max. accuracy	Coverage	Administration	Location-rate
RFID	proximity (auto-ID)	~10 cm	poor	easy	1 - 2 s
802.11b	scene analysis	~2 m	very well	very time-consuming	2 - 4 s
Bluetooth	proximity (COO)	few rooms	well	time-consuming	10 - 30 s
IP Subnet	virtual space	departments	very well	easy	5 s

10.4.2 Proximity Sensors

Any sensor that is able to detect whether two users are in spatial proximity is referred to as *proximity sensor*. In order to detect spatial proximities between users, the *proximity sensor encapsulation* asynchronously transmits sensor-dependent proximity information to the server. The corresponding *proximity sensor abstraction*-modules collect physical proximity information delivered by the sensors of all users, and transform them to mappings of user-IDs. For each detected proximity or proximity change between users, an event is thrown in order to notify the *proximity sensor fusion*-module.

Two different proximity sensors have been implemented, which differ in sensor type, range and delay:

- *Bluetooth*: The Bluetooth proximity sensor is based on the implementation of the Bluetooth location sensor; abstractions of location and proximity information are integrated into a single module. Instead of MAC addresses of reference stations, addresses of other users are detected. Each address indicates proximity. An advantage of this approach is that Bluetooth modules are highly deployed and that the detection of proximities is independent of the knowledge of absolute locations.
- *Location-based*: This can be seen as a logical sensor in the sense that it exploits standardized location information of all users and determines – by the help of the location model – which users currently stay at the same area or level. Advantages of the location-based proximity sensor are that they do not require a separate client-side implementation and the range can be restricted to a certain layer of the location hierarchy. A disadvantage is that it requires location information.

Table 3: Comparison of implemented proximity sensors

Sensors	Type	Range	Delay
Bluetooth	physical	10-100m	5-15 s
Location-based	logical	area or level	~ 4 s

Table 3 subsumes the properties of these two sensors, whereas a more detailed description of the implementation is given in chapter 11.5.

10.5 Sensor Fusion

Core of the location system is the *sensor fusion*, which operates on the *decision-level*. Its job is to merge the measurements of various sensors, while coping with differences concerning accuracy and sample-rate. The algorithms collect all (asynchronously arriving) events triggered by the *sensor abstraction*-modules, merge the standardized information and trigger events in order to notify context-aware applications if the symbolic location of a certain user or the spatial proximity between users has changed. As fusion operates on standardized information, additional location- and proximity-sensors can be added easily or the fusion algorithms themselves can be replaced.

According to the two kinds of sensors described in the preceding chapter, *two types of sensor fusion* can be distinguished:

- *Location sensor fusion*: Takes all symbolic locations acquired by various location sensors of a certain user and selects the ones representing his location at the best. If only one or few coarse sensors (e.g. Bluetooth) are available on a certain client, where each measurement (e.g. a Bluetooth reference station) is associated with multiple symbolic locations, fusion may not be able to determine a single location. Because of such cases, an additional single location is calculated by selecting the *least upper bound* of the fused locations in the location hierarchy. Fusion is only done for users whose symbolic locations (delivered by the *location sensor abstraction*) have changed. Any time the fused location of a certain user changes, an event with his new location is thrown. Fusion is performed for each user separately and takes into account the latest measurements of each location sensor only (i.e. no history information is used for determining the current location of a particular user). Due to performance reasons, no probability distributions are combined; each sensor only delivers the most accurate locations without weighting them with a certain probability. Beyond *fused locations* and the *single location*, context-aware applica-

tions are notified if there is any change in the set of *all locations* (i.e. all symbolic locations provided by the sensors of a certain client).

- *Proximity sensor fusion*: Takes all mappings of user-IDs acquired by various proximity-sensors of a certain user and detects whether two of them are in spatial proximity or if they are no longer nearby. Two users are considered to be nearby if at least one sensor of these users detects proximity, and they are considered to be nearby no longer if no sensor of them detects proximity.

An important feature is the persistent storage of *location- and proximity-history* in a database in order to allow future retrieval. This enables context-aware applications, which are integrated in the context-middleware, to visualize the movement of users for example. History information can also be used for more complex fusion algorithms that do not only take into account the latest measurements. A detailed description of the fusion algorithms is given is chapter 11.6.

10.6 Properties of the Presented Location System

By means of the classification of location systems given in chapter 8.4, the location system presented in this thesis can be evaluated:

- *Physical position versus symbolic location*: Only symbolic locations are supported, no physical (geometric) position information is provided.
- *Absolute versus relative*: Absolute locations as well as relative locations (i.e. spatial proximities) are supported.
- *Positioning versus tracking*: This is only indirectly a matter of the presented location system, as both positioning and tracking systems can be integrated. However, the user has always the possibility to shut down the location client; the server-side *sensor abstractions* must ensure that localization stops in that case.
- *Accuracy and precision*: This again depends on the integrated technologies. If RFID sensors are integrated, accuracy is in the range of several centimetres (cf. Table 2).
- *Coverage*: The effective range of the location system is defined as the union of the individual ranges of all integrated sensors. For example, complete coverage is pro-

vided if WLAN-based localization is supported and WLAN is ubiquitously available (cf. Table 2).

- *Recognition*: Each entity, which is being localized by the presented location service, has a unique user-ID and can thus be identified by context-aware applications using the location system.
- *Limitations*: A serious limitation is that the used context middleware *requires multicast*, which restricts the operational area of the location system to "multicast domains". Further restrictions depend on the integrated sensors (e.g. little coverage with RFID, the need for WLAN access points, required training with scene analyses-based WLAN systems etc.). However, limitations of one technology can be compensated by others if multiple sensors are used.
- *Openness and Generality*: Both openness and generality are fulfilled, as sensor-dependent location information is transformed to a uniform format and thus arbitrary sensor technologies and techniques can be integrated.
- *Cost*: Costs again depend on the integrated sensor technologies. As arbitrary sensors can be integrated, the administrator can choose which technologies to use where in the environment.
- *Location-rate*: The location-rate depends on the integrated sensors (cf. Table 2 and Table 3) and sensor implementations, and it is defined as the minimum of the location-rates of all sensors.

11 Location System Implementation

Goal of this chapter is to present the developed location system in more detail. First, the *context middleware*, in which the whole location system is integrated, is introduced and a detailed view of the *architecture* and the underlying *data model* is given. Afterwards, the implemented *location- and proximity-sensors* as well as the corresponding *sensor fusions* are presented in detail.

11.1 Context Middleware

The location system is integrated in a software-framework, which serves as a middleware for developing context-aware applications. This so-called *Context Framework* [Beer03, Beer04] is based on a peer-to-peer communication architecture and it supports different kinds of transport protocols.

One important task of this framework is the *abstraction of context information retrieval* via various sensors from application designers. This is accomplished by so-called *entities*, which describe objects – e.g. human users – that are important for a certain context scenario. Entities express their functionality by the use of *context attributes*, which can be loaded into entities. These attributes are complex pieces of software, which are implemented as Java classes. Typical attributes are *encapsulations of sensors*, but they can also be used to implement *context services*, for example to notify other entities about location changes of users.

Each entity can contain a collection of such attributes, where an entity itself is an attribute (cf. Figure 16). For example, an entity that is associated with a certain user can contain an entity for a location service and another one for a context-aware application. The initial set of attributes an entity contains can change dynamically at runtime, if it loads or unloads attributes from a local storage or over the network. In order to load and deploy new attributes, an entity has to reference a *class loader* and a *transport and lookup* layer, which manages the lookup mechanism for discovering other entities and performs transport. Discovery of entities relies on *IP-multicast*. *XML configuration files* specify which initial set of entities should be loaded at start-up and which attributes these entities own.

The communication between entities and attributes is based on *context events*. Each attribute is able to trigger such events, which are addressed to other attributes or entities, independently on which physical computer they are running. Among other things, an event contains the *name of the event* and a *list of parameters* delivering information about the event itself. For an attribute, the transport of context events is completely transparent, which means that there is no difference if the receiving attribute resides on the same host or on another one, connected with any kind of network. The *hiding of external communication details* from the application developer is an important feature of the context framework with regard to the integration of the location system.

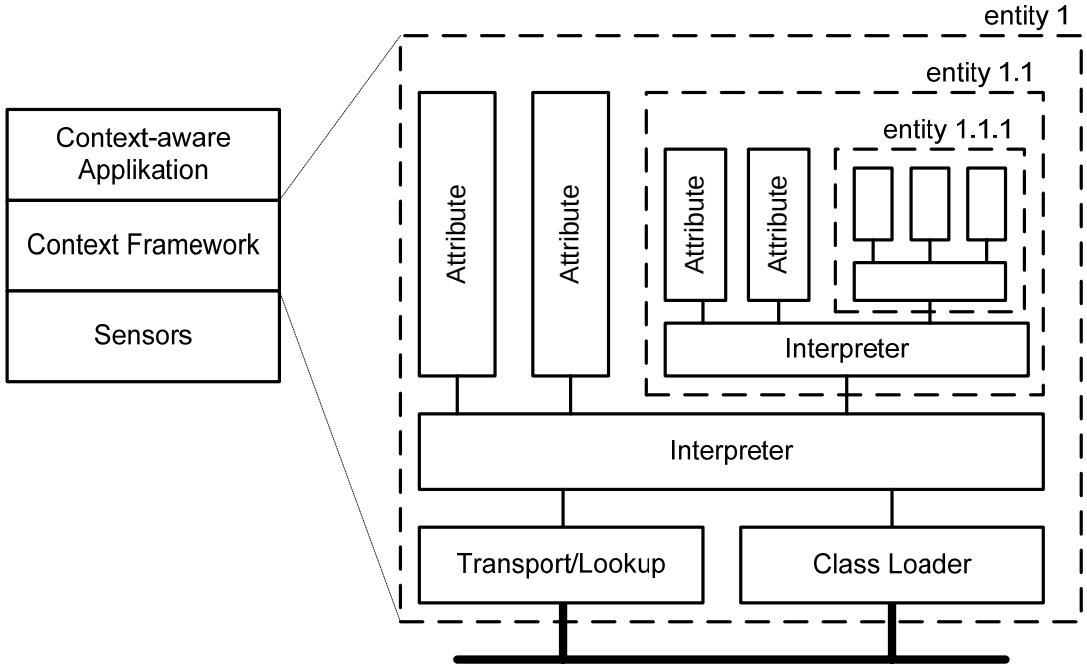


Figure 16: Recursive attribute-entity-architecture [Beer03]

Related with this event-based architecture is the use of *ECA-rules* for defining the behaviour of the whole system, namely how to react on specific events. Therefore, every entity has a *rule-interpreter*, which catches triggered events, checks conditions associated with them and causes certain actions (e.g. to deliver the event to a certain attribute). Each entity has its own set of rules, which are referenced by the entity's XML configuration. A rule is able to trigger the insertion of new rules or the unloading of existing rules at runtime in order to change the behaviour of the context system dynamically.

11.2 Architecture in Detail

Figure 17 gives a more detailed view of the architecture (see also Figure 14). It shows the integration of all modules of the location system – except native code for accessing sensors – in the *Context Framework* and the event-based communication between them. Server-side event notifications are encapsulated by a dedicated Java-class. The shaded boxes refer to modules of the location system; the others are either commercial software or their implementation is not part of this thesis.

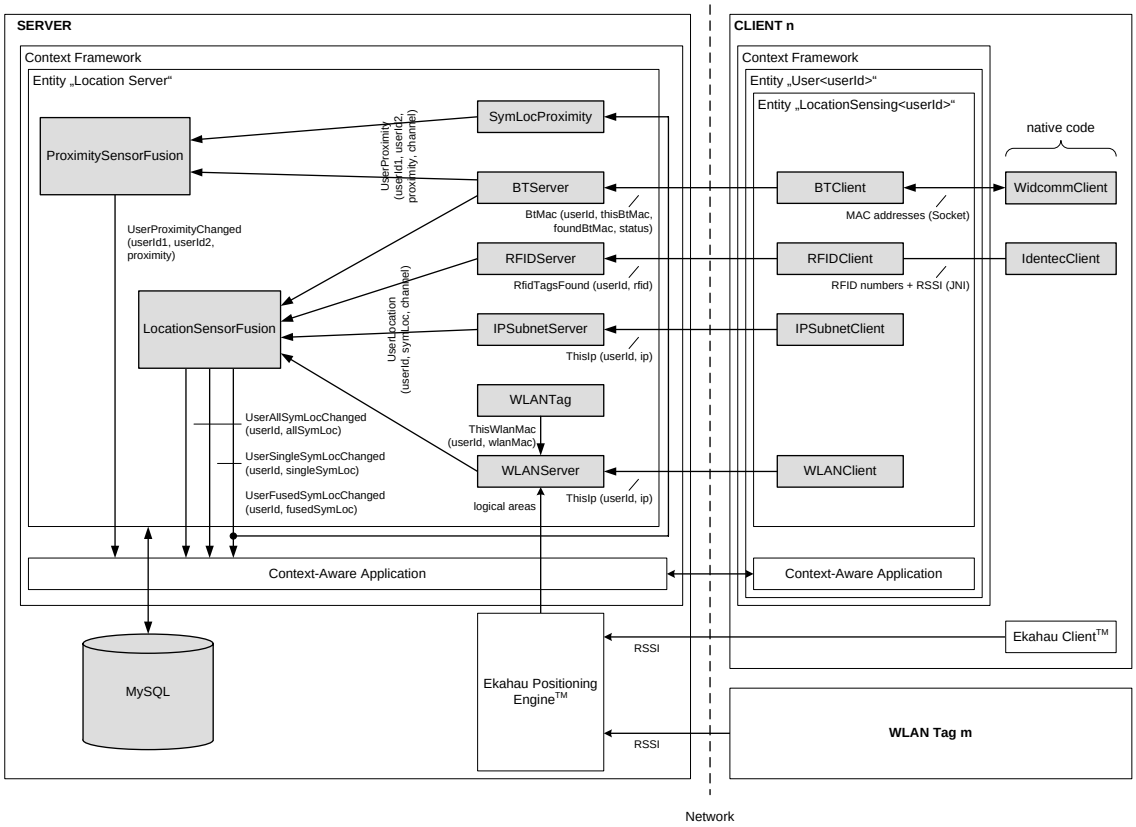


Figure 17: Detailed view of the architecture

The location system consists of *clients* acquiring location information and transmitting sensor-dependent data to a single *server*, which performs abstraction and fusion, and notifies context-aware applications if location or proximity of a certain user has changed. The server relies on a *MySQL database* for persistent storage. In addition to the clients, so-called *WLAN tags* can be tracked by the location system (cf. chapter 11.4.1).

On both client and server, an instance of the Context Framework is running. On the *client-side*, there is an entity `User<userId>`, which contains the nested entity

`LocationSensing<userId>`; `userId` denotes a unique ID, the number of an instant messenger account for example [Oppl04]. *Sensor encapsulations* are implemented as context attributes (grey boxes within the entity `LocationSensing<userId>`), where two of them require native implementations (grey boxes outside the Context Framework). They send events with sensor-dependent position information to the server. To which server-side attribute a certain event has to be delivered is defined by client-side context rules. As the Context Framework does no allow sending serialized objects via events, complex data is transformed to comma separated value- (CSV-) strings.

On the *server-side*, there is a single entity `LocationServer` containing context attributes for *sensor abstraction* and *sensor fusion* (grey boxes within the entity `LocationServer`). *Location sensor abstractions* collect events of the *sensor encapsulations*, transform them to symbolic locations and release uniform `UserLocation`-events containing the symbolic locations of a certain user detected by a particular location sensor. Similar to the location sensor abstractions, *proximity sensor abstractions* trigger uniform `UserProximity`-events containing information about which users are newly or which are no longer in proximity. However, not all sensor abstractions require client-side encapsulations. *Sensor-encapsulations* and *sensor abstractions* will be described in the chapters 11.4 and 11.5.

The two *sensor fusion modules*, which are discussed in chapter 11.6 in depth, are also implemented as context attributes. `LocationSensorFusion` collects all `UserLocation`-events of the *location sensor abstractions* and merges the received symbolic locations separately for each user. It can trigger three types of context events, independently of each other, in order to notify *context-aware applications* about location changes of a certain user: `UserAllSymLocChanged` (if the set of *all symbolic locations* acquired by all sensors has changed), `UserFusedSymLocChanged` (if the *fused symbolic locations* of the respective user have changed) and `UserSingleSymLocChanged` (if the *single symbolic location*, namely the least upper bound of the fused locations in the location tree, has changed).

`ProximitySensorFusion` receives `UserProximity`-events, merges the proximity-information received from all *proximity sensor abstractions* and generates

`UserProximityChanged`-events in order to notify *context-aware applications* if the proximity of two users has changed.

11.3 Data Model

The location system uses the *open source database MySQL* for persistent storage of static and dynamic data. Database access is encapsulated by a single Java class, which is used by all server-side context attributes.

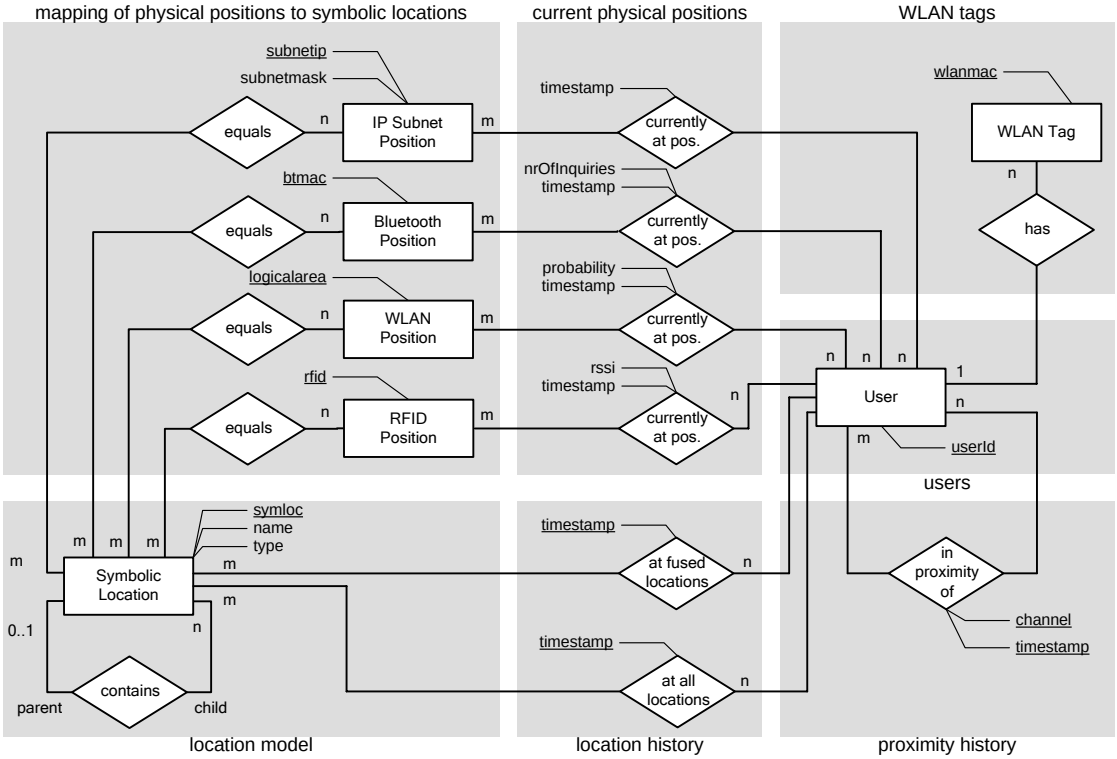


Figure 18: Entity-Relationship model

Figure 18 shows the representation of the *data model* as an *Entity-Relationship (ER) model*, which contains entities and relations for representing the current physical positions of all users, the mapping of physical positions to symbolic locations, the location model for correlating these locations, the location- and proximity-history and WLAN tags. The meaning of the entities and relations will be described in the subsequent chapters.

11.4 Location Sensors

Location sensors typically require two implementations: a client-side attribute (*location sensor encapsulation*) for acquiring sensor-dependent physical positions and transmitting

them to the server, and a server-side attribute (*location sensor abstraction*) for collecting these physical positions, storing them to the database and transforming the *most probable* of them (e.g. only the RFID numbers with the highest RSSI), for which a mapping to symbolic locations exists, to *uniform symbolic locations*. From this point on, low-level sensor details are hidden. An integration of additional sensors only requires the implementation of a server- and in most cases also a corresponding client-side context-attribute.

This transformation is done by the help of database tables that map physical positions (e.g. RFID numbers) to symbolic locations, where each physical position may be associated with multiple symbolic locations. Symbolic locations are stored as unique numbers with names (i.e. textual representations of the locations) and types (i.e. the types in the location model, cf. chapter 10.3) and they are hierarchically structured by a contains-relationship. The current physical positions of users are stored in tables (one for each sensor) containing sensor-dependent information (e.g. RSSI) and timestamps, which are needed for detecting timeouts (e.g. if the computer is switched off). Additionally to the sensors available at a certain client, users can also be associated with WLAN tags in the current implementation, which are identified by their MAC address; Bluetooth tags [Blue04] would also be an interesting option.

All *location sensor abstractions* trigger uniform `UserLocation`-events with the following parameters, every time a particular *location sensor abstraction* has acquired a *location update*:

- `userId`: ID of the user who has been localized
- `channel`: A number indicating which location-sensor has triggered the event
- `symLoc`: The *symbolic locations* acquired by the sensor with the given `channel`-number coded as CSV string, or an *empty string* if the respective sensor of this user could not acquire location information any more (e.g. if the sensor has been removed or it cannot find reference stations any more).

However, location changes are not only triggered by location updates from certain sensors, but also by *timeouts*. Timeouts occur if the client-side location service is terminated; in such a case, all physical positions (e.g. Bluetooth MAC addresses) of the affected user are

removed from the database and a `UserLocation`-event with empty parameter `symLoc` is triggered.

11.4.1 WLAN Sensor

The WLAN location sensor uses the *Ekahau Positioning Engine* [Ekah04], a scene analysis-based Wi-Fi location-tracking system. It is a software-only solution for localizing Wi-Fi-enabled PCs and PDAs.

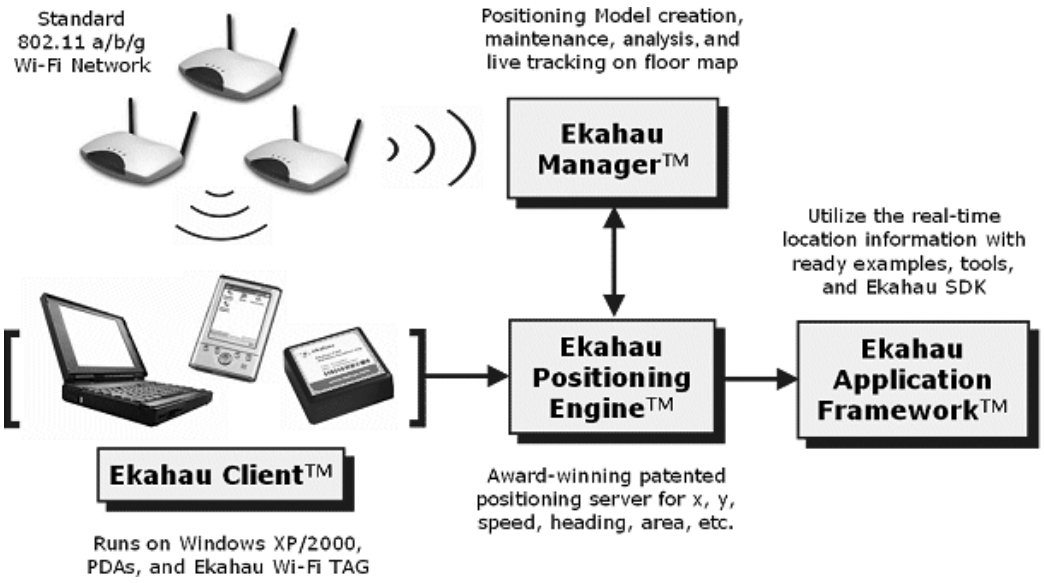


Figure 19: Ekahau Positioning Engine software concept [Ekah04]

Figure 19 shows the software concept, which consists of the following components:

- *Ekahau Client*: A small program that runs on a client device and collects signal strength information from all Wi-Fi access points in range and transmits this information periodically to the Ekahau Positioning Engine.
- *Ekahau Manager*: The Ekahau Manager is an application for recording the field data in order to create the positioning model, namely a floor plan with recorded access point-signal strength values assigned to certain places.
- *Ekahau Positioning Engine*: This Java-based server software collects signal strength information acquired by Ekahau Clients and determines their locations by comparing them with the calibrated positioning model. It supports multi-floor tracking, both zone-based (reporting the device location by pre-defined zone

names, so-called *logical areas*) and as location coordinates (x, y, floor). It provides accuracy of up to 1-2 meters [Ekah04].

- *Ekahau Application Framework and SDK*: Refers to a Java API for utilizing location information, which is accessed by the server-side attribute `WLANServer`.

Tracking is only possible if the *Ekahau Client* is running on the client side. In order to start tracking, the *Ekahau Positioning Engine* must know the IP address of the client. Therefore, the client-side attribute `WLANClient` periodically triggers `ThisIp`-events transmitting the IP address along with the `userId` to the attribute `WLANServer`. `WLANServer` registers each new client with the Positioning Engine, starts tracking and periodically receives a list of best-matching *logical areas* along with their respective probabilities for all registered clients. In addition, the attribute `WLANServer` is able to register *WLAN tags* with the *Positioning Engine*, Wi-Fi-enabled PDAs or *Ekahau T101 Wi-Fi Location Tags* [Ekah04] for example. Each WLAN tag is identified by its MAC address that is mapped to a certain user-ID in the database. The attribute `WLANTag` periodically reads the MAC addresses of all tags from the database and transmits them – along with the `userId` – to the attribute `WLANServer`.

Every time `WLANServer` receives the current logical areas of a certain user from the Positioning Engine (e.g. with an interval of 2 seconds), it stores them along with their probabilities in the database. Afterwards, the logical area with highest probability, for which a mapping to a symbolic location exists, is transformed to this symbolic location and an according `UserLocation`-event is triggered.

If the IP- or MAC-address of a tracked device has not been received for a certain time (e.g. if the client has been switched off) or if the *Positioning Engine* could not track a registered device for a predefined time (e.g. if the *Ekahau Client* of the respective device has been shut down), `WLANServer` deregisters this client from the Positioning Engine and throws a `UserLocation`-event with empty parameter `symLoc`.

11.4.2 RFID Sensor

The RFID sensor has the highest spatial resolution among the implemented ones. It is based on an *i-CARD III Long-Range RFID reader* of IDENTEC SOLUTIONS Inc. [IDEN04], which has a PCMCIA card format and operates in the 915 MHz ISM band. It can read up to 35 tags/s and has a response time of less than 150 ms.

IDENTEC provides a C++ SDK for its products, which has been used for the native implementation `IdentecClient` that reads serial numbers and RSSI of RFID tags in range. The context attribute `RFIDClient` periodically scans for tags via the native implementation `IdentecClient` by the help of the *Java Native Interface (JNI)*. Plugging and unplugging of the PCMCIA card is possible anytime; `RFIDClient` periodically checks if the card is connected and starts scanning for tags as soon as the card is plugged. By reducing the power level of the RFID reader, its range is limited to about one meter. In order to determine which one of multiple tags in range is nearest, signal strength information is exploited. Therefore, if only one tag is in range of the RFID reader, accuracy of about one meter is reached. If multiple tags are in range, the nearest one is be chosen with an accuracy of up to 10 centimetres. As the signal strength information varies heavily, the *moving average* of the latest three RSSI of each tag is calculated.

`RFIDClient` transmits context events named `RfidTagsFound` to the server-side attribute `RFIDServer`, which contain the following parameters:

- `userId`: ID of the user, which is extracted from the name of the client-entity
- `rfid`: CSV-string with tag-IDs and the according RSSIs

This attribute stores the received physical positions to the database, transforms the ones with highest RSSI to symbolic locations (if such a mapping exists) and triggers uniform `UserLocation`-events. If no `RfidTagsFound`-event has been received from a certain client for a predefined time, a `UserLocation`-event with empty `symLoc` is triggered.

11.4.3 Bluetooth Sensor

This sensor requires a native implementation named `WidcommClient`, which is based upon the widely deployed *WIDCOMM Bluetooth protocol stack* [WIDC04]. WIDCOMM provides a C++ SDK for Windows supporting developers with direct access to all protocol-

layers, which has been used for the native implementation `WidcommClient`. Unfortunately, a JNI implementation was not feasible because of a failure in the implementation of the Bluetooth stack, which causes a delay of more than 60 sec when loading the Dynamic Link Library (DLL) of `WidcommClient` into a Java class. As a workaround, `WidcommClient` has been compiled to an executable file and *local UDP socket connections* are used in order to transmit data to the context-attribute `BTClient`.

`WidcommClient` *periodically performs inquiry* and sends found Bluetooth MAC addresses along with information about the Bluetooth device itself (e.g. its MAC address or if it has been unplugged) via socket connection to the context attribute `BTClient`. MAC addresses are not only transmitted if inquiry has completed, but also during inquiry for each found device. A Bluetooth dongle can be removed anytime; as soon as it is plugged in, the native implementation starts inquiry and transmits found MAC addresses to the context attribute. In order to terminate `WidcommClient` when the context framework is being terminated, `BTClient` periodically sends *"alive-packages"* via a second socket connection to `WidcommClient`, which receives these packages in a concurrent thread and terminates the native application if these packets have not been received for a predefined time.

Every time `BTClient` receives a packet via the socket connection, it transmits a context event `BtMac` to the server-side attribute `BTServer`, which contains the following parameters:

- `userId`: ID of the user, which is extracted from the name of the client-entity
- `thisBtMac`: MAC address of the Bluetooth sensor or empty if it has been removed (status=0)
- `foundBtMac`: MAC address of a single device found while inquiry is in progress (status=1), addresses of all devices found during inquiry (status=2) or empty if the Bluetooth sensor has been removed (status=0)
- `status`: Used to distinguish which kind of data is transmitted

If `BTServer` receives a particular MAC address belonging to a reference station *two times in series*, it stores the address to the database, transforms it to symbolic locations

(typically multiple symbolic locations per MAC address) and triggers uniform `UserLocation`-events. If a particular address has not been found during the latest *two inquiries* (saved in the database as attribute *nrOfInquiries*), if no `BtMac`-events have been received for a certain time or if the Bluetooth dongle has been removed, the physical positions (i.e. the MAC addresses) of the affected user are deleted from the database and a `UserLocation`-event with empty `symLoc` is released. The constraint that a particular device must be found twice and not found twice respectively is necessary because of the fact that empirical studies have shown that Bluetooth devices sometimes do not answer upon inquiries, although they are in range. However, this approach causes low location-rates of up to 30 seconds.

11.4.4 IP Subnet Sensor

The IP subnet sensor requires two implementations. On the one hand, the client-side attribute `IPSubnetClient` periodically sends `ThisIp`-events with the IP address of a certain client along with its associated `userId` to the server. The server-side attribute `IPSubnetServer` collects all `ThisIp`-events, determines which IP subnets the client with `userId` belongs to, stores these subnet addresses (only one in most cases) to the database, transforms them to symbolic locations if possible, and triggers according `UserLocation`-events. The mapping of subnet addresses to IP addresses is done by static database entries saving all known subnet addresses and corresponding subnet masks.

11.5 Proximity Sensors

Like the location sensors, proximity sensors typically require two implementations: a client-side attribute (*proximity sensor encapsulation*) for acquiring sensor-dependent proximity information and transmitting it to the server, and a server-side attribute (*proximity sensor abstraction*) for collecting this information and transforming it to *uniform mappings of user-IDs*; as with location sensors, low-level sensor details are hidden from this point on. How the transformation is performed depends mainly on the used proximity sensor.

The *proximity sensor abstractions* trigger uniform `UserProximity`-events with the following parameters, every time proximity between two users has been detected:

- `userId1` and `userId2`: User with `userId2` has been detected to be in proximity of user with `userId1`
- `proximity`: A flag indicating if new proximity between these two users has been detected (`proximity`=1) or if they are no longer nearby (`proximity`=0)
- `channel`: A number indicating which proximity-sensor has triggered the event

11.5.1 Bluetooth Sensor

The implementation of the Bluetooth proximity sensor goes along with the Bluetooth location sensor, and it is integrated in the same server-side attribute `BTServer` (cf. chapter 11.4.3). The event `BtMac`, which is transmitted for each Bluetooth device found during inquiry, contains a parameter `thisBtMac` (i.e. the Bluetooth MAC address belonging to the client that has triggered this event). This parameter enables `BTServer` to map the MAC addresses of all Bluetooth sensors to the respective user-IDs.

Proximity between clients is detected by exploiting the second parameter `foundBtMac`, which contains MAC addresses of found devices. Every time such a MAC address found by the client with `userId1` equals that of a Bluetooth sensor plugged to the client with `userId2`, `BTServer` triggers a `UserProximity`-event with these two user-IDs and a `proximity`-value of 1. If the Bluetooth dongle of the client with `userId1` is removed, `UserProximity`-events with `proximity`-values of 0 will be triggered for each client which is known to be in proximity of user with `userId1`. If the MAC address of a certain client in proximity has not been found during the *last two inquiries*, a `UserProximity`-event with a `proximity`-value of 0 is triggered, too.

11.5.2 Location-Based Sensor

The second sensor is based on *symbolic locations* processed by the attribute `LocationSensorFusion`, which will be described in chapter 11.6.1. For this reason, the sensor only consists of the server-side abstraction `SymLocProximity`; a dedicated client-side implementation is not needed.

Every time the attribute `LocationSensorFusion` releases a `UserFusedSymLocChanged`-event (i.e. the fused symbolic locations of a certain user with `userId1` have changed), `SymLocProximity` checks the history of fused locations if another user with `userId2` is currently associated with a symbolic location nearby. Proximity is recognized if these two users have the *same parent symbolic location of a predefined type* in the location model (e.g. if they are in the same room or on the same floor). If a user is associated with multiple fused locations at a time, at least one must fulfil this criterion. For each detected proximity, a `UserProximity`-event with a `proximity`-value of 1 and the corresponding user-IDs is triggered. If proximity could not be detected the next time, a `UserProximity`-event with a `proximity`-value of 0 is thrown.

11.6 Sensor Fusion

According to the two classes of sensor families, two sensor fusion attributes are distinguished, namely `LocationSensorFusion` for merging symbolic locations, and `ProximitySensorFusion` for merging mappings of user-IDs.

The `LocationSensorFusion` collects all `UserLocation`-events of the *location sensor abstractions* and stores them in an internal data structure, separately for each user (parameter `userId`) and sensor (parameter `channel`).

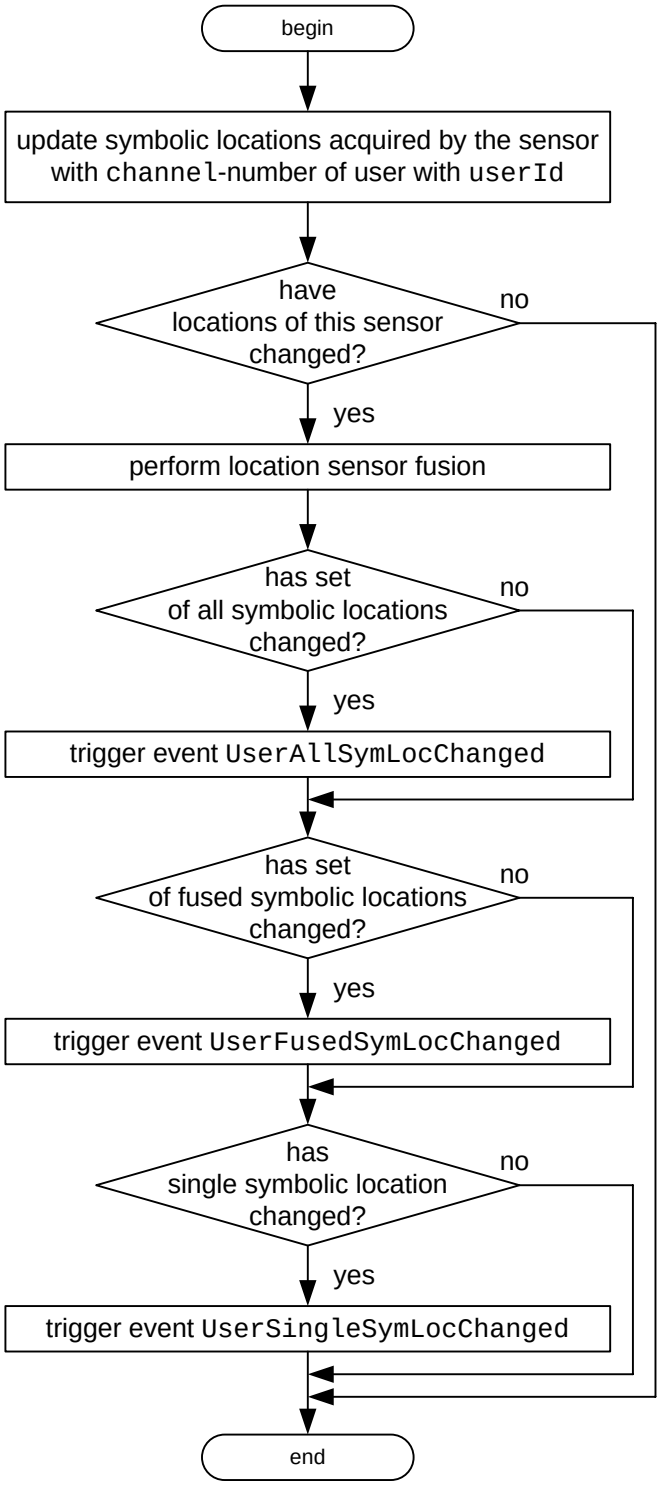


Figure 20: Per-user location sensor fusion flow chart

Every time such an event occurs, it checks whether `symLoc` contains *new symbolic locations*, which are not already associated with the given sensor of the user with `userId` (if this is the case, they are added to the data structure), or if it contains an *empty string*, which

means that this sensor could not acquire location information any more (all locations of this sensor are then removed from the respective data structure in this case). If and only if this is the case, `LocationSensorFusion` merges the symbolic locations acquired by all sensors of this user (cf. Figure 20).

As Figure 20 shows, three types of events can be triggered after fusion:

- `UserAllSymLocChanged`: Contains *all different symbolic locations* (parameter `allSymLoc`) acquired by all sensors of the client with the given `userId`, and it is only triggered if this set of locations has changed. Symbolic locations that are found several times (e.g. by different sensors) only occur once in this set. In order to provide a location history, they are written to the database together with a time-stamp and the user-ID every time they have changed.
- `UserFusedSymLocChanged`: Contains the *fused symbolic locations* (parameter `fusedSymLoc`), namely those locations that remain after merging the symbolic locations acquired by all sensors of a certain user. This event is only triggered if the fused locations of the respective user have changed. Similar to the set of all symbolic locations, the fused ones are also written to the database, together with time-stamps and the corresponding user-IDs. The fusion algorithm, which merges the symbolic locations, will be described in chapter 11.6.1.
- `UserSingleSymLocChanged`: Contains a *single symbolic location* (parameter `singleSymLoc`), namely the least upper bound of the fused locations in the location hierarchy. As with the other two events, it is only triggered if the single location of the user with `userId` has changed. Single symbolic locations are not stored in the database as they can easily be derived from the fused symbolic locations.

`ProximitySensorFusion` receives `UserProximity`-events, merges the proximity-information received from all *proximity sensor abstractions* (distinguished by the `channel`-parameter) and generates `UserProximityChanged`-events with the parameters `userId1`, `userId2` and a `proximity`-flag indicating if these two users are *newly in proximity* (`proximity`-value of 1) or if they are *no longer nearby* (`proximity`-value of 0). The fusion algorithm, which detects proximity changes, will be described in chapter

11.6.2. With respect to the *proximity history*, any time a particular sensor detects a proximity change between two users, the mapping of `userId1` to `userId2` is saved together with a timestamp (the time when the change has been detected) and the channel-number (identifying which proximity sensor has detected the change) in the database. As clients in proximity usually find each other, both a mapping of `userId1` to `userId2` and the inverse mapping of `userId2` to `userId1` are stored in most cases.

11.6.1 Location Sensor Fusion-Algorithm

It is the task of the attribute `LocationSensorFusion` to merge the asynchronously arriving `UserLocation`-events separately for each user and trigger `UserFusedSymLocChanged`-events if the symbolic locations of a certain user have changed.

The implemented fusion algorithm tries to *reduce the number of all symbolic locations* associated with each user:

- Each symbolic location is associated with its *number of occurrences*, which is a measure for the probability that a certain user stays at a particular location. A symbolic location may occur several times if it is referred to by more than one sensor or if a single sensor detects multiple reference stations with overlapping coverage.
- In a second step, this *number of occurrences of each location is inherited*, which means, that it is added to all numbers of symbolic locations, which are child-locations of the considered one in the location tree (cf. Figure 15). For example, if “room2” occurs twice and “desk” occurs once, the value 2 of “room2” is added to the value 1 of “desk”, which finally gets the number 3. This procedure relies on the consideration that locations are the more probable the more often they are detected.
- In a last step, the maximum number of occurrences is determined and all locations, which do not have this *maximum number of occurrences*, are removed. The remaining ones are referred to as the *fused symbolic locations*.
- If multiple sensors are used which deliver locations that do not overlap or are not contained in each other, a reduction of symbolic locations based on the steps above is not possible, as no location is more probable than another one. Therefore, *priori-*

ties are assigned to location sensors, which are the higher the more *confident* a sensor is. For the implemented sensors, RFID has the highest priority 4, WLAN has priority 3, Bluetooth has priority 2 and IP subnet has priority of 1. Instead of just accumulating the number of occurrences of locations, the priorities of sensors that have detected a certain location are summed up as described above.

An advantage of this algorithm is that it performs an *intersection* as locations covered by multiple overlapping reference stations are weighted stronger. For example, the locations of a client, which is in the range of multiple Bluetooth reference stations, can be reduced to a few symbolic locations that are covered by all stations. A further advantage is a kind of *fault tolerance*, as outliers (e.g. locations detected by a single one of several sensors only) are automatically removed.

Figure 21 shows some fusion examples with four sensors A, B, C and D (their priorities are given in brackets) by means of an example location model. The ellipses show the coverage of each sensor, the numbers beneath the symbolic locations show the accumulated priorities. Fused symbolic locations are those with the highest accumulated priorities (shown as shaded circles). The first example shows two sensors A (priority of 4) and B (priority of 3), where sensor A covers two symbolic locations and sensor B covers one. As the priority of sensor B is inherited, only one symbolic location covered by sensor A gets the maximum value of 7 which is the resulting fused symbolic location.

In the second example, sensor D provides a symbolic location three levels above in the location tree, which is a parent location of the two covered by sensor A. In this case, the set of fused symbolic locations contains two locations having both a maximum value of 5. Therefore, the single symbolic location (i.e. the least common symbolic location in the location tree) does not equal the fused one.

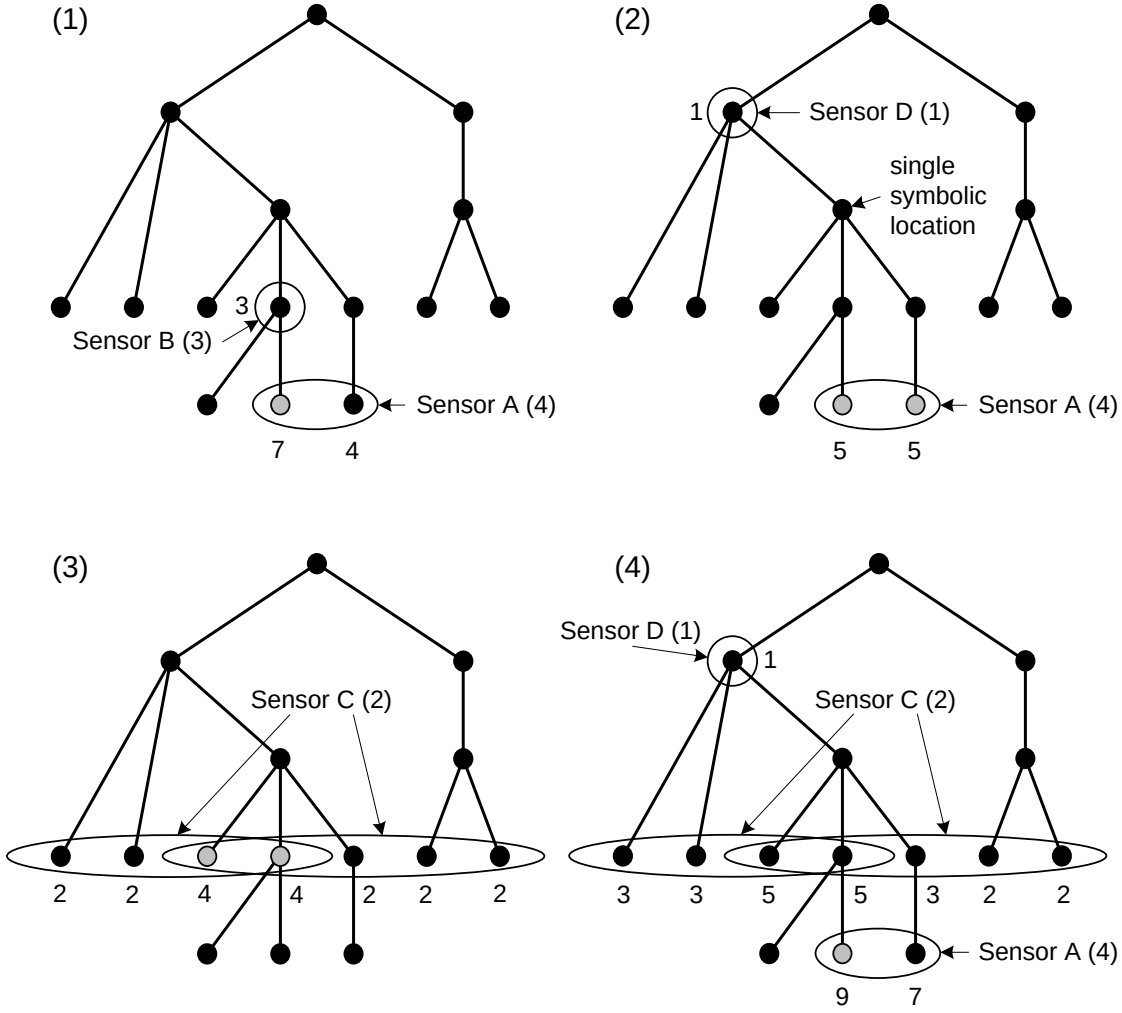


Figure 21: Location Sensor Fusion Examples

The third example shows the intersection of the ranges of two reference stations with overlapping symbolic locations (e.g. Bluetooth reference stations), which are both detected by sensor C (e.g. a Bluetooth dongle). As a result, the two symbolic locations with a maximum value of 4 are thus the resulting fused locations.

The last example shows the locations acquired by three sensors A, C and D, where the resulting fused location is that with the maximum value of 9.

11.6.2 Proximity Sensor Fusion-Algorithm

The task of the attribute `ProximitySensorFusion` is to merge the asynchronously arriving `UserProximity`-events and trigger `UserProximityChanged`-events if the proximity between certain users has changed.

If the parameter `proximity` of a received `UserProximity`-event has a value of 1, `ProximitySensorFusion` checks the *proximity history* whether these two users have already been detected to be in proximity (i.e. user with `userId2` is in proximity of user with `userId1` or vice versa), either by this or by another sensor. If this is not the case, these two users are *newly in proximity*.

If the parameter `proximity` has a value of 0, `ProximitySensorFusion` sets the according proximity, namely that user with `userId2` is in proximity of user with `userId1`, in the *proximity history* to a proximity-value of 0 and checks if these two users are still known to be nearby afterwards (e.g. if user with `userId1` is in proximity of user with `userId2` or if another sensor still detects proximity). If this is not the case, these two users are considered as being *no longer nearby*.

Bibliography

[Abow02] Abowd, G.D., Battestini, A., O'Connell, T. *"The Location Service: A framework for handling multiple location sensing technologies"*, College of Computing & GVU Center, Georgia Institute of Technology, Atlanta, Georgia, USA, 2002.

[Addl97] Addlesee, M.D., Jones, A.H., Livesey, F., Samarita, F.S. *"The ORL Active Floor"*, IEEE Personal Communications, vol. 4, no. 5, pp. 33-41, October 1997.

[Aero04] *Aeroscout*. Internet: http://www.aeroscout.com (31. August 2004).

[Anti02] Antifakos, S., Schiele, B. *„Beyond Position Awareness"*, Personal and Ubiquitous Computing, vol. 6, no. 5/6, pp. 313-317, December 2002.

[Bahl00] Bahl, P., Padmanabhan, V. *"RADAR: An in-building RF-based user location and tracking system"*, Proceedings of IEEE INFOCOM 2000. (Tel Aviv, Israel, March 2000). IEEE Computer Society Press, Los Alamitos, CA, 2000, vol. 2, pp. 775-784.

[Bal90] Bal, H.E. *„Programming Distributed Systems"*, Prentice Hall International, 1990.

[Barb04] Barbeau, M., Kranakis, E., Krizanc, D., Morin, P. *"Improving Distance Based Geographic Location Techniques in Sensor Networks"*, Proceedings of the 3rd International Conference on AD-HOC Networks & Wireless (ADHOC-NOW'04). (Vancouver, British Columbia, July 22-24, 2004). Springer Verlag, LNCS 3158, 2004, pp. 192-210.

[Bead97] Beadle, H.W.P., Harper, B., Marguire, G.Q., Judge, J. *„Location Aware Mobile Computing"*, Proceedings of the IEEE International Conference on Telecommunications (ITC'97). (Melbourne, Australia, April 1997).

[Beer03] Beer, W., Christian, V., Ferscha, A., Mehrmann, L. *„Modeling Context-Aware Behavior by Interpreted ECA Rules"*, Proceedings of the 9th International Conference on Parallel and Distributed Computing (Euro-Par'03). (Klagenfurt, Austria, August 26-29, 2003). Springer Verlag, Austria, LNCS 2790, 2003, pp. 1064-1073.

[Beer04] Beer, W. *"SILICON Context Framework"*, Technical Report, Context Framework for Mobile User Applications, Siemens Munich CT-SE 2, February 2004.

[Blue04a] *The Official Bluetooth SIG Membership Site*. Internet: https://www.bluetooth.org (31. August 2004).

[Blue04b] *BlueTags*. Internet: http://www.bluetags.com (31. August 2004).

[Bohn03] Bohn, J., Vogt, H. *"Robust Probabilistic Positioning based on High-Level Sensor-Fusion and Map Knowledge"*, Technical Report No. 421, Institute for Pervasive Computing, Swiss Federal Institute of Technology, ETH Zurich, Switzerland, April 2003.

[Bonn01] Bonnifait, P., Bouron, P., Crubillé, P., Meziel, D. *"Data Fusion of Four ABS Sensors and GPS for an Enhanced Localization of Car-like Vehicles"*, Proceedings of the IEEE International Conference on Robotics and Automation (ICA'01). (Séoul, Korea, 21.-26. May 2001). IEEE, 2001, pp. 1597-1602.

[Bray02] Bray, J., Sturman, C.F. *"Bluetooth 1.1 – Connect Without Cables, Second Edition"*, Prentice Hall, 2002.

[Brow97] Brown, P.J., Bovey, J.D., Chen X. "*Context-Aware Applications: From the Laboratory to the Marketplace"*, IEEE Personal Communications, vol. 4, no.5, pp. 58-64, October 1997.

[Brum00] Brumitt, B., Meyers, B., Krumm, J., Kern, A., Shafer, S. *"EasyLiving: Technologies for Intelligent Environments"*, Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC 2000). (Bristol, UK, 25.-27. September 2000). Springer Verlag, London, UK, LNCS 1927, 2000, pp. 12-29.

[Butz01] Butz, A., Baus, J., Krüger, A., Lohse, M. *"A hybrid indoor navigation system"*, Proceedings of the 6th International Conference on Intelligent User Interfaces (IUI 2001). (Santa Fe, NM, USA, 2001). ACM Press, New York, NY, USA, 2001, pp. 25-32.

[Chen00] Chen, G., Kotz, D. *„A Survey of Context-Aware Mobile Computing Research"*, Technical Report TR2000-381, Computer Science Department, Dartmouth College, Hanover, New Hampshire, November 2000.

[Chen03] Chen, R.Y.F., Petrie, C. *„Ubiquitous Mobile Computing"*, IEEE Internet Computing, vol. 7, no. 2, March-April 2003.

[Dey00] Dey, A.K. *„Providing Architectural Support for Building Context-Aware Applications"*, Ph.D. Thesis, Department of Computer Science, Georgia Institute of Technology, November 2000.

[Dobs04] Dobson, S. *„Where's Waldo? – or – A taxonomy for thinking about location in pervasive computing"*, Technical Report TCD-CS-2004-05, Computer Science Department, Trinity College Dublin, Ireland, May 2004.

[Domn01] Domnitcheva, S. *"Location Modeling: State of the Art and Challenges"*, Proceedings of the Workshop on Location Modeling for Ubiquitous Computing (Ubicomp 2001). (Atlanta, Georgia, 30. September 2001). pp. 13-20.

[Ekah04] *Ekahau Inc*. Internet: http://www.ekahau.com (31. August 2004).

[Estr02] Estrin, D., Culler, D., Pister, K., Sukhatme, G. *„Connecting the Physical World with Pervasive Networks"*, IEEE Pervasive Computing, vol. 1, no. 1, pp. 59-66, March 2002.

[FCC04] *Federal Communications Commission*. FCC enhanced 911. Internet: http://www.fcc.gov/e911 (31. August 2004).

[Fers04a] Ferscha, A., Holzmann, C., Oppl, S. *„Team Awareness in Personalized Learning Environments“*, Proceedings of the 3rd European Workshop on Mobile and Contextual Learning (MLEARN'04). (Rome, Italy, July 5-6, 2004).

[Fers04b] Ferscha, A., Holzmann, C., Oppl, S. *„Context-Awareness for Group Interaction Support“*, Proceedings of the 2nd ACM International Workshop on Mobility Management and Wireless Access (MobiWac'04). (Philadelphia, PA, USA, September 26-October 1, 2004). ACM Press, 2004, to be published.

[Fink03] Finkenzeller, K. *"RFID-Handbook: Fundamentals and Applications in Contact-less Smart Cards and Identification"*, John Wiley & Sons, October 2003.

[Form94] Forman, G.H., Zahorjan, J. *„The Challenges of Mobile Computing“*, IEEE Computer, vol. 27, no. 4, pp. 38-47, April 1994.

[Fox03] Fox, D., Hightower, J., Liao, L., Schulz, D. *"Bayesian Filtering for Location Estimation"*, IEEE Computer, vol. 2, no. 3, pp. 24-33, July-September 2003.

[Hall01] Hall, D., Llina, J. *"Handbook of Multisensor Data Fusion"*, CRC Press, 2001.

[Hall02] Hallberg, J., Nilsson, M. *"Positioning with Bluetooth, IrDA and RFID"*, Master Thesis, Department of Computer Science and Electrical Engineering, Division of Computer Engineering, Luleå University of Technology, January 2002.

[Hall03] Hallberg, J., Nilsson, M., Synnes, K. *"Positioning with Bluetooth"*, Proceedings of the 10th International IEEE Conference on Telecommunications (ICT'2003), (Tahiti, Papeete – French Polynesia, 23. February-1. March 2003). IEEE, 2003, pp.954-958.

[Hart99] Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P. *„The Anatomy of a Context-Aware Application“*, Proceedings of the fifth annual ACM/IEEE Conference on Mobile Computing and Networking (Mobicom'99). (Seattle, WA, USA, August 1999). ACM Press, 1999, pp. 59-86.

[Haza04] Hazas, M., Scott, J., Krumm, J. *„Location-Aware Computing Comes of Age“*, IEEE Computer, vol. 37, no. 2, pp. 95-97, February 2004.

[High00] Hightower, J. *"SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength"*, Technical Report #2000-02-02, Computer Science and Engineering Department, University of Washington, Seattle, WA, USA, February 2000.

[High01a] Hightower, J., Boriello, G. *„A Survey and Taxonomy of Location Systems for Ubiquitous Computing“*, Extended paper from IEEE Computer, vol. 34, no. 8, pp. 57-66, August 2001.

[High01b] Hightower, J., Vakili, C., Borriello, G., Want, R. *"Design and Calibration of the SponON Ad-Hoc Location Sensing System"*, unpublished, August 2001.

[High02] Hightower, J., Brumitt, B., Boriello, G. *„The Location Stack: A Layered Model for Location in Ubiquitous Computing"*, Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02). (Callicoon, NY, USA, 20.-21. June 2002). IEEE Computer Society, June 2002, pp. 22-30.

[Hu04] Hu, H., Lee, D.L. *"Semantic Location Modeling for Location Navigation in Mobile Environment"*, Proceedings of the 5th IEEE International Conference on Mobile Data Management (MDM'04). (Berkeley, California, 19.-22. January 2004). IEEE Computer Society Press, 2004, pp. 52-61.

[IDEN04] *IDENTEC SOLUTIONS AG*. Internet: http://www.identecsolutions.com (31. August 2004).

[IEEE04] *IEEE 802 LAN/MAN Standards Committee*. Internet: http://www.ieee802.org (31. August 2004).

[Indu03] Indulska, J., Sutton, P. *„Location Management in Pervasive Systems"*, Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003. (Adelaide, Australia, 2003). Australian Computer Socienty, Darlinghurst, Australia, 2003, pp. 143-151.

[Krum00] Krumm, J., Harris, S., Mayers, B., Brummitt, B., Hale, M., Shafer, S. *"Multi-Camera Multi-Person Tracking for EasyLiving"*, Proceedings of the 3rd IEEE International Workshop on Visual Surveillance (VS'2000). (Dublin, Ireland, 1. July 2000). IEEE Computer Society Press, Washington, DC, USA, 2000, pp. 3-10.

[Kuma03] Kumar, V., Das, S.R. *„Performance of Dead Reckoning-Based Location Service for Mobile Ad Hoc Networks"*, Wireless Communications and Mobile Computing Journal, December 2003.

[Leon98a] Leonhardt, U. *„Supporting Location-Awareness in Open Distributed Systems"*, Ph.D. Thesis, Department of Computing, Imperial College, London, May 1998.

[Leon98b] Leonhardt, U. Magee, J. *„Multi-Sensor Location Tracking"*, Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98). (Dallas, Texas, United States, 25.-30. October 1998). ACM Press, New York, NY, USA, 1998, pp. 203-214.

[Lin02] Lin, J., *"Personal Location Agents for Communication Entities (PLACE)"*, Master Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Cambridge, UK, May2002.

[Matt03] Mattern, F., Sturm, P. *„From Distributed Systems to Ubiquitous Computing – State of the Art, Trends and Prospects of Future Networked Systems“*, Proceedings of the 13th Fachtagung Kommunikation in Verteilten Systemen (KiVS'03). (Leipzig, Germany, February 2003). Springer Verlag, Leipzig, Germany, 2003, pp. 3-25.

[Morl04] Morla, R., Davies, N. *"Evaluating a Location-Based Application: A Hybrid Test and Simulation Environment"*, IEEE Pervasive Computing, vol. 3, no. 3, pp. 45-56, July-September 2004.

[Moor65] Moor, G.E. *"Cramming more components onto integrated circuits"*, Electronics, vol. 38, no. 8, 19. April 1965.

[Nels98] Nelson, G.J. *„Context-Aware and Location Systems“*, Ph.D. Thesis, University of Cambridge, Computer Laboratory, Cambridge, UK, January 1998.

[Ni03] Ni, L.M., Liu, Y., Lau, Y.C., Patil, A.P. *"LANDMARC: Indoor Location Sensing Using Active RFID"*, Proceedings of the 1st International IEEE Conference on Pervasive Computing and Communications (PerCom'03). (Forth Worth, TX, USA, 23.-26. March 2003). IEEE Computer Society Press, 2003, pp. 407-415.

[Nibb01] *The Nibble Location System*. Internet: http://mmsl.cs.ucla.edu/nibble (31. August 2004).

[Oppl04] Oppl, S. *"Context-Sensitive Interaction in Groups"*, Master Thesis, University of Linz, Institut für Pervasive Computing, 2004, to be published.

[Orr00] Orr, R.J., Abowd, G.D. *"The Smart Floor: A Mechanism for Natural User Identification and Tracking"*, Proceedings of the International Conference on Human Factors in Computing Systems (CHI 2000). (The Hague, Netherlands, 1.-6. April 2000). ACM Press, New York, NY, USA, 2000, pp. 275-276.

[Patt03] Patterson, C.A., Muntz, R.R., Pancake, C.M. *"Challenges in Location-Aware Computing"*, IEEE Pervasive Computing, vol. 2, no. 2, pp. 80-89, April-June 2003.

[Priy00] Priyantha, N.B., Chakraborty, A., Balakrishnan, H. *"The Cricket Location-Support System"*, Proceedings of the 6th International ACM Conference on Mobile Computing and Networking (MobiCom 2000). (Boston, MA, USA, 6.-11. August 2000). ACM Press, 2000, pp. 23-43.

[Ray03] Ray, A., Kurdovsky, S. *"A Survey of Intelligent Pervasive Computing"*, Proceedings of the International Conference on Artificial Intelligence (IC-AI'03). (Las Vegas, Nevada, USA, 23.-26. June 2003). CSREA press, 2003, pp. 30-35.

[Röde04] Röder, P., Hoffmann, M., Ritscher, M. *"A distributed framework for location sensing systems"*, Proceedings of the 1st Workshop on Positioning, Navigation and Communication (WPNC'04). (University of Hannover, Germany, 26. March 2004). Shaker Verlag, Germany, 2004.

[Roth02] Roth, J. *„Mobile Computing“*, dpunkt.verlag GmbH, Heidelberg, Germany, 2002.

[Rous02] Roussos, G. *“Location Sensing Technologies and Applications”*, Technical Report TSW 02-08, School of Computer Science and Information Systems, Birbeck College, University of London, England, November 2002.

[Ryan98] Ryan, N.S., Pascoe, J., Morse, D.R., *“Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant”*. Gaffney, V., Lesuen, M.van., Exxon, S. (edt.), Computer Applications in Archaeology 1997, British Archeological Reports, Tempus Reparatum, Oxford, October 1998.

[Saty96] Satyanarayanan, M. *„Fundamental Challenges in Mobile Computing“*, Proceedings of the Fifteenth ACM Symposium on Principles of Distributed Computing. (Philadelphia, PA, USA, May 1996). ACM Press, New York, 1996, pp. 1-7.

[Saty01] Satyanarayanan, M. *„Pervasive Computing – Vision and Challenges“*, IEEE Personal Communications, vol. 8, no. 4, pp. 10-17, August 2001.

[Schi94] Schilit, B., Adams, N., Want, R. *„Context-Aware Computing Applications“*, Proceedings of the IEEE Workshop on Mobile Systems and Applications (WMCSA’94). (Santa Cruz, CA, USA, December 1994). IEEE Computer Society Press, 1994, pp. 85-90.

[Schi95] Schilit, B.N. *„A System Architecture for Context-Aware Mobile Computing“*, Ph.D. Thesis, Columbia University, Department of Computer Science, May 1995.

[Schi02] Schilit, B.N., Hilbert, D.M., Trevor, J. *“Context-Aware Communication”*, IEEE Wireless Communications, vol. 9, no. 5, pp. 46-55, October 2002.

[Schi03] Schiller, J. *“Mobilkommunikation”*, Pearson Studium, Munich, Germany, 2003.

[Schm98] Schmidt, A., Beigl, M., Gellersen, H.W. *“There is more to Context than Location”*, Proceedings of the International Workshop on Interactive Applications of Mobile Computing (IMC’98). (Rostock, Germany, Fraunhofer IGD, November 1998).

[Schm02] Schmidt, A. *“Ubiquitous Computing – Computing in Context”*, Ph.D. Thesis, Lancaster Universit, U.K., November 2002.

[SETI04] *SETI@home: Search for Extraterrestrial Intelligence at home*. Internet: http://setiathome.ssl.berkeley.edu (31. August 2004).

[Sing91] Singhal, M., Casavant L.C. *“Distributed Computing Systems”*, IEEE Computer, vol. 24, no. 8, pp. 12-15, August 1991.

[Sriv01] Srivastava, M. *“Mobile and Wireless Networked Systems”*, Lecture EE206A, University of California, Department for Electrical Engineering, Internet: http://www.cs.wmich.edu/wsn/doc/loc/loc.ppt (31. August 2004).

[Star97] Starner, T., Mann, S., Rhodes, B., Healey, J., Kirsch, D., Picard, R., Pentland, A. *"Augmented reality through wearable computing"*, Presence, vol. 6, no. 4, pp. 386-398, August 1997.

[Tane02] Tanenbaum, A.S., Steen, M. *"Distributed Systems: Principles and Paradigms"*, Prentice Hall, 2002.

[Want92] Want, R., Hopper, A., Falcão, V., Gibbons, J. *"The Active Badge Location System"*, ACM Transactions on Information Systems (TOIS), vol. 10, no. 1, pp. 92-102, January 1992.

[Ward97] Ward, A., Jones, A., Hopper, A. *"A New Location Technique for the Active Office"*, IEEE Personal Communications, vol. 4, no. 5, pp. 42-47, October 1997.

[Ward98] Ward, A.M.R. *"Sensor-driven Computing"*, Ph.D. Thesis, Corpus Christi College, University of Cambridge, August 1998.

[Webs04] *Merriam-Webster Online Dictionary*. Internet: http://www.m-w.com (31. August 2004).

[Well93] Wellner, P., Mackay, W., Gold, R. *"Computer-Augmented Environments: Back to the Real World"*, Communications of the ACM, vol. 34, no. 7, pp. 24-26, July 1993.

[Wemm03] Wemmer, A. *"Design and Implementation of a Bluetooth Tag"*, Master Thesis, Institut für Technische Information, Technische Universität Graz, May 2003.

[Wert01] Werthimer, D., Cobb, J., Lebofsky, M., Anderson, D., Korpela, E. *"SETI@home-Massively Distributed Computing for SETI"*, IEEE Computing in Science & Engineering, vol. 3, no.1, pp. 18-83, January/February 2001.

[Weis91] Weiser, M. *"The Computer for the 21st Century"*, Scientific American, vol. 265, no. 3, pp. 94-104, September 1991.

[Weis93] Weiser, M. *"Some Computer Science Issues in Ubiquitous Computing"*, Communications of the ACM, vol. 36, no. 7, pp. 75-84, July 1993.

[WIDC04] *WIDCOMM Inc*. Internet: http://www.widcomm.com (31. August 2004).

[Wifi04] *Wi-fi Alliance*. Internet: http://www.wi-fi.org (31. August 2004).

[Yosh00] Yoshimi, B. *"On Sensor Frameworks for Pervasive Systems"*, Proceedings of the Workshop on Software Engineering for Wearable and Pervasive Computing (ICSE 2000). (Limerick, Ireland, 6. June 2000).