JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Damit Deine Zukunft in Zukunft Zukunft hat

Context-aware Group-Interaction

Diplomarbeit zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Diplomstudium *Informatik*

Angefertigt am Institut für *Pervasive Computing*

Eingereicht von:

Stefan Oppl

Betreuung:

Univ.-Prof. Mag. Dr. Alois Ferscha

Beurteilung:

Univ.-Prof. Mag. Dr. Alois Ferscha

Linz, September 2004

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Linz, September 2004 Stefan Oppl

Kurzfassung

In der heutigen Zeit wird die Arbeit in Gruppen immer mehr zu einer zeitlich und räumlich verteilten Angelegenheit. Aus diesem Grund wird die Unterstützung der anfallenden Aufgaben durch Informationstechnologie immer unerlässlicher. Bereits verfügbare Systeme bieten ausgereifte Möglichkeiten zur Manipulation von Objekten und stellen auch Informationen über erfolgte Manipulationen zur Verfügung. Um jedoch Gruppenarbeit effektiv unterstützen zu können, reichen Informationen über die manipulierten Objekte alleine nicht aus. Vielmehr ist es notwendig, den beteiligten Personen auch Informationen über den aktuellen Kontext der anderen Gruppenmitglieder zur Verfügung zu stellen, um ihnen damit ein Gefühl für deren aktuelle Tätigkeit, deren Verfügbarkeit oder deren Aufenthaltsort zu geben. Derartige Informationen verbessern das Gefühl der Zusammengehörigkeit in einer Gruppe erheblich und tragen damit signifikant zur Verbesserung der computer-gestützten Gruppenarbeit bei. Zusätzlich können die verfügbaren Kontext-Informationen auch genutzt werden, um dem System selbst die Möglichkeit zu geben, sich den aktuellen Gegebenheiten und Bedürfnissen anzupassen und sogar proaktiv unterstützende Maßnahmen auszulösen.

Das Ziel dieser Arbeit ist es, die Möglichkeiten zur Unterstützung von Gruppeninteraktion am Universitäts-Campus durch Kontext-Information zu evaluieren. Die Erkenntnisse dieser Evaluation wurden prototypisch in einer Applikation (*GISS – Group Interaction Support System*) umgesetzt, die Anwender aktiv bei deren Interaktion in Gruppen unterstützt.

Aufbauend auf dem SiLiCon Context-Framework als Middleware und einem exisitierenden Instant Messenger als Benutzerschnittstelle, setzt das System Kontext-Informationen, wie Indentität, Aufenthaltsort und aktuelle Aktivität eines Benutzers sowie die aktuelle Zeit ein, um gemeinsam mit Informationen über vorhandene Gruppenstrukturen ein, um Dienste zur Ermöglichung und Verbesserung der Interaktion in Gruppen anzubieten.

Diese Dienste umfassen im einzelnen Mittel zur Unterstützung von Gruppenbildung, zur synchronen und asynchronen Kommunikation in Gruppen, zur Darstellung des Aufenthaltsortes anderer Benutzer sowie der Erkennung und Unterstützung von Gruppen-Besprechungen. Außerdem stehen noch ein Dienst zur kontext-sensitive Anpassung des öffentlich sichtbaren Verfügbarkeitsstatuses sowie ein kontext-sensitiver Erinnerungsdienst zur Verfügung.

Das System befindet sich gegenwärtig in einer stabilen Version und wurde bereits mit ausgewählten Szenarien getestet. Ein Feldtest mit mehreren Benutzern unter realen Bedingungen ist derzeit noch ausständig.

Abstract

As today's group work today becomes more and more distributed across space and time, the demand for supporting group tasks by means of information technology is constantly growing. Available systems provide means for shared manipulation of objects and awareness about these manipulations. For effective teamwork, this artefact-based awareness is not enough. Awareness of what the other group members are currently engaged with, where they are and if they could be reached significantly improves the experience of computer supported collaborative work. Going even one step further, this user-based awareness could be used to provide the computer system with the context of the users and of the whole groups they are working in. Using this context, applications can adapt to the user's current needs and even proactively trigger services appropiate for the current situation.

The goal of this thesis has been to evaluate how user contexts can be utilized to support interaction in groups in spatially bounded areas, namely university campuses, and to prototypically implement an application, which actively supports people in their interaction needs.

The result of this work is a fully implemented system for the context-aware support of group interaction that is referred to as *GISS (Group Interaction Support System)*. This system is based on the SiLiCon-Context-Framework (cf. chapter 5.2) and uses an already existing instant messenger as user interface (cf. chapter 5.4.3). The system is aware of user identities, locations and activity states and of the local time. Utilizing this knowledge about the user's context together with meta-information about group structures, the system is able to provide the users with pro-active services they might need to improve their interaction.

These services include means for group formation and management, synchronous and asynchronous group communication, location awareness and the recognition and support of gatherings as well as means for context-aware adaptation of one's availability status and a context-aware reminder service.

The system has reached a stable version and has already been tested in several artificial settings. A field test with more users in a real life setting is subject of future work.

To my parents

Contents

1 Introduction

Information is transmitted from HOST to HOST
in bundles called messages. […]
RFC 1: Host Software (April, 7th 1969)

1.1 *Motivation*

When in the early 1970s the foundations of the Internet were developed, nobody could imagine how this technology would evolve and completely change the way we interact with each other. Distances – both spatial and temporal – have lost their importance, there is no major difference if we communicate with a colleague sitting in the next room of with a business partner working thousands of kilometres away in his office in Japan.

In this brave new world, we are confronted with settings everyday where we have to work in geographically or temporally dispersed groups [Fers00]. As we all have experienced, working in such settings is not the same as working in traditional, co-located groups. The lack of information that is lost by only transmitting the message itself but not the implicit information normally available in face-to-face settings (like facial expression or intonation), is not compensated and makes the communication process somewhat unnatural.

Some years ago, this hasn't been an issue simply because the technical prerequisites did not exist to easily and inexpensively communicate synchronously across spatial borders. Today, as these technical issues have been solved already and this kind of communication occurs in a manifold of settings in everyday work, the lack of awareness is becoming more and more obvious.

Those issues can only be solved if interaction support systems become aware of the context a user is currently in and provide potential communication partners with this information as well as adapt themselves and their services to this context. Working in groups even increases the requirements on such a system, as not only the state of the individual members has to be taken into account, but also the state of the whole group itself is relevant.

On the one hand, the location of communication partners has lost its importance today, as this issue does not restrict communication anymore. On the other hand, location has become important from another point of view, namely as an important part of the relevant context of the communication partner, together with a manifold of other information.

To provide a more intuitive infrastructure for interaction via computer systems is a big issue in research today. This work evaluates how context information can be used for this and what services are an appropriate means for context-aware interaction.

1.2 *Goals*

The goal of this thesis is to evaluate how user contexts can be utilized *to support interaction in groups in spatially bounded areas* and to prototypically *implement an application*, which actively supports people in their interaction needs.

Interaction aspects that have to be covered are *synchronous* and *asynchronous communication* as well as support for informal *gatherings* of groups. As the group is a central element in this work, there is need for support in questions of *forming groups* and *managing group membership*.

As mentioned before, the *use of context to support interaction* is subject of research in this work. The availability of context information has to be ascertained, before it is possible to design the context-aware services that support interaction.

Besides the explicit interaction services, the need for further services, especially in the field of *group-awareness* and *individual information management* has to be examined. This may include (but is not restricted to) *visualisation of position information* or *reminder services*.

The practical usability of the system naturally is a big issue in design and the possibility to easily extend it on both context sensing as well as user interface side by adding or exchanging components is crucial too.

1.3 *Results*

The main result of this work is a fully implemented system for the context-aware support of group interaction that is referred to as *GISS (Group Interaction Support System)*. This system is based on the SiLiCon-Context-Framework (cf. chapter 5.2) and uses an already existing instant messenger as user interface (cf. chapter 5.4.3), thus enabling users to continue using their existing instant-messaging account and not having to sign into another system. The use of the Context-Framework brings a highly modular architecture that is easily extendable, thus fitting the needs of this system.

The system is aware of user identities, locations and activity states and of the local time. Utilizing this knowledge about the user's context together with meta-information about group structures, the system is able to provide the users with pro-active services they might need to improve their interaction in a spatially restricted area, in the scenario of this work on a university campus.

The services provide the following means of group awareness and communication (cf. chapter 5.5 for a more detailed description):

- The *positions of all buddies* (the persons in one's contact list) are visualized in the contact list of the messenger without distracting the user from his main task. On demand, a 2D-floor-plan view is shown, on which the buddies' positions are shown graphically in the form of ICQ-floating-contacts (cf. chapter 5.5.2)
- There is *support for forming and managing groups*. Groups are a collection of users that are formed in different ways. *Dynamic groups* are formed spontaneously of users, who are located in the same region, thus enabling them to easily share thoughts or make appointments. *Static groups* are created manually and can be *open* (free for everybody to join and leave) or *closed* (a group-owner decides, who is a member) and still exist even if its members are not co-located. Users have the possibility to create new groups (either open or closed), to join and leave open groups, to add and remove members from their closed groups and even transform dynamic groups into open static groups (cf. chapter 5.5.3)
- As *synchronous interaction* via instant messengers is one of the most popular communication applications today, this means of interaction has been extended to groups. Users may send messages to groups they are member in and have their message delivered to all the members synchronously, thus enabling some sort of chat. The main advantage to standard messengers, which also provide similar features, is the fact that messages can also be sent to people who one maybe doesn't know (or at least doesn't have in his contact list) but who are members of a certain group (for example in lecture settings, where larger groups may occur) (cf. chapter 5.5.4).
- *Asynchronous interaction* today is largely covered by email-messages (one-to-one) or forums (one-to-many). However, in some cases it makes sense to leave a message in the context of a certain location. For this reason, so-called *virtual post-its* are introduced, which can be placed on any location that can be identified by the system. Furthermore, virtual post-its may have an expiry date or visibility restricted to a certain group of people. Virtual post-its remember who has already read them and can also be commented by others, making them much more powerful than "real-world" post-its.
- The system *supports groups when gatherings occur*. To successfully do this, gatherings have to be recognized first. Knowing basic context-information like location and time only in general isn't enough to reliably decide whether a gathering takes place at

a specified location and time or not. The chosen approach takes into account some additional meta-information to support this decision. However, the implemented algorithm still lacks the desired reliability and should be subject to rework in future (cf. chapter 8.3). Whenever a gathering is recognized, the system triggers some services like the *notification of absent group members* or the *creation of a meeting minutes*, in which an attendance list is automatically created (including members joining later and leaving earlier). After the gathering, the protocol – including the notes taken by the participants of the gathering – is sent to every group member (cf. chapter 5.5.6)

- The support of *reminders* is not part of group interaction support itself but it is considered useful in literature [Dey00], to provide tools for managing future tasks context-aware. For this reason, a reminder service has been implemented in which a user may leave messages for himself, which are triggered by freely definable conditions. These conditions may refer to the current location of a user, the current time, other users in proximity or any combination of these context sources (cf. chapter 5.5.7)
- *Visibility management* is also only loosely related to group interaction support but nevertheless an important feature in terms of providing activity awareness. The user is provided with a means of defining rules, in which conditions his messenger visibility state should be set to a certain value. Supported values include for example *away*, *online*, *busy* or *invisible*. Conditions can be freely combined out of time and location triggers. Furthermore recurrent time triggers are supported, so that users can define rules, which are for example applied at the same time every week (cf. chapter 5.5.8).

The implemented system and the presented services have been implemented for execution on a windows environment and is fully functional there. A transition to other platforms like Linux or Mac OS X should not cause much effort, as the used components are either implemented in Java or in platform-independent C++.

The tests have been carried out with the server running on a standard Windows-PC and up to three clients. In this setting, no major problems occurred. An exhaustive field test has not been undertaken up to now and is subject to future work (cf. chapter 8.3).

1.4 *Outline*

In *chapter 2*, the *basic concepts* needed for this work are defined. After looking at awareness and the notion of context, a closer look is given to the aspects of interaction, which are relevant for this work.

Chapter 3 presents *related research* in the areas covered by this work. As the topic of this work as hardly ever been addressed before as whole, mainly similar projects, which overlap only in some areas or cover certain aspects of this work, are presented here.

In *chapter 4* an *analysis of the requirements* for the system that has been implemented is carried out. This analysis is based on the goals specified in chapter 1.2 and on the lessons learned from the study of related work in chapter 3.

The *chosen architecture* of the system is presented in *chapter 5*. After an overview of the underlying software framework, the decision-process for a suitable user-interface is presented. After that, it is explained, which context information is relevant for this application and how it can be sensed. Taking this as a basis, the services that were chosen to fulfil the requirements are presented in the last part of this chapter.

In *chapter 6*, the implementation details are presented. While the first part of this chapter deals with the architectural issues like the flow of communication between the components of the system, the second part lays the focus on the conceptual issues and presents the algorithms that where chosen to realize the services presented in chapter 5.

Chapter 7 shows possible usage scenarios and presents the impacts the implemented system could have on the interaction processes in groups.

Last but not least *chapter* 8 summarizes the results and open issues of the work and gives an outlook for possible future directions of research.

2 Context-aware Group-Interaction

One cannot not communicate

Paul Watzlawick

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.

2.1 *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.

2.2 *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.

2.2.1 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.

2.2.2 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.

2.2.3 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?).

2.2.4 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.

2.3 *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.

2.3.1 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.

2.3.2 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.

2.3.3 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 2.3.1:

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.

2.3.4 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.

2.3.5 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.

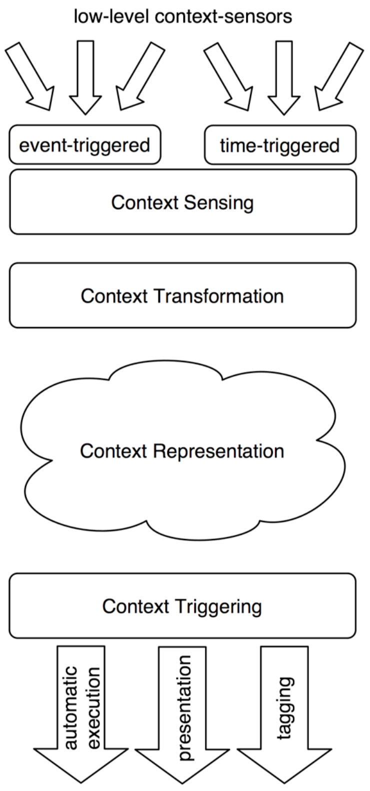


Figure 1: Processing context data

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 2.3.4.

2.4 *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.

2.4.1 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]

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

2.4.2 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.

3 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. These systems are classified according to the definition of goals given in chapter 1.2:

- Systems for supporting awareness among groups (cf. chapter 3.2)
- Systems for context-aware synchronous communication (cf. chapter 3.3)
- Systems for context-aware asynchronous communication (cf. chapter 3.4)
- Visualisation of position information (cf. chapter 3.5)
- Context-aware reminders (cf. chapter 3.6)

3.1 *Context-aware Support of Students on Campus*

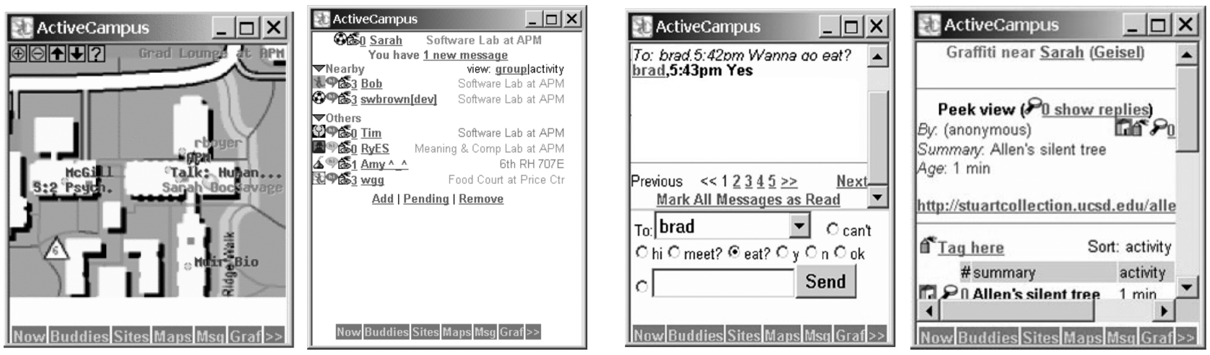
There is a manifold of systems supporting students in their daily live on the university campus. However, most of these systems appear either in the form of an (more or less enhanced) e-learning system or some sort of bulletin board system (BBS). Hardly any student-support-system supports any form of 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.

3.1.1 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.


ActiveCampus
Grad Lounge at APM
McGill
5:2 Psych.
Talk: Human...
Sarah Boc Savage
Muir Bio
Ridge Walk
Now Buddies Sites Maps Msg Graf >>
ActiveCampus
Now Buddies Sites Maps Msg Graf >>
ActiveCampus
Previous << 1 2 3 4 5 >> Next
Mark All Messages as Read
hi meet? eat? y n ok can't
Send
Now Buddies Sites Maps Msg Graf >>
ActiveCampus
Tag here
Sort: activity
Now Buddies Sites Maps Msg Graf >>

0 Sarah Software Lab at APM
You have 1 new message
Nearby view: group|activity
3 Bob Software Lab at APM
3 swbrown[dev] Software Lab at APM
Others
0 Tim Software Lab at APM
0 RyES Meaning & Comp Lab at APM
1 Amy ^_^ 6th RH 707E
3 wgg Food Court at Price Ctr
Add | Pending | Remove

To: brad.5:42pm Wanna go eat?
brad,5:43pm Yes
To: brad

Graffiti near Sarah (Geisel)
Peek view (0 show replies)
By: (anonymous)
Summary: Allen's silent tree
Age: 1 min
http://stuartcollection.ucsd.edu/alle

#	summary	activity
0	Allen's silent tree	1 min

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

3.1.2 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.

3.2 *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 2.2.4), taking into account the results of the already cited study about the main awareness deficits in distributed groups [Stei99] (cf. chapter 2.2.3).

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).

3.2.1 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].

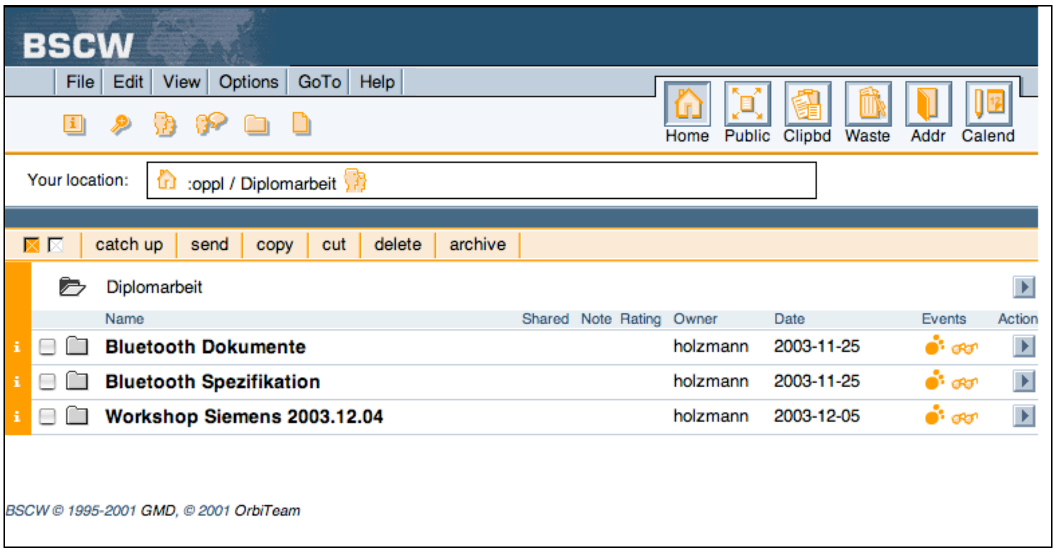
teamSCOPE provides the following means to support group awareness:

- 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.

3.2.2 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.



Name	Shared	Note	Rating	Owner	Date	Events	Action
Bluetooth Dokumente				holzmann	2003-11-25		
Bluetooth Spezifikation				holzmann	2003-11-25		
Workshop Siemens 2003.12.04				holzmann	2003-12-05		

Figure 3: Screenshot of BSCW-System

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.

3.3 *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.

3.3.1 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.

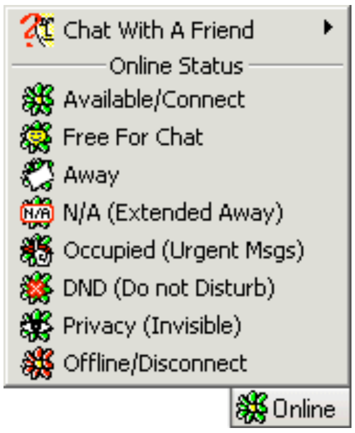


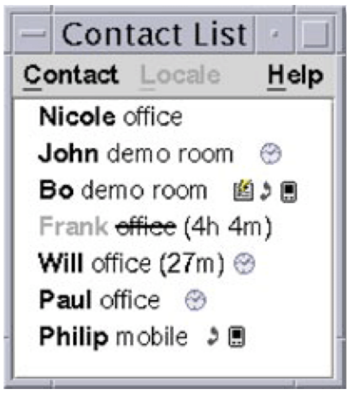
Figure 4: Activity awareness in ICQ [ICQ04]

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 5.4, this context-aware feature is integrated into the *GISS*-application, providing some sort of simple activity context (cf. chapter 5.3.3).

3.3.2 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.



Nicole office
John demo room
Bo demo room
Frank ~~office~~ (4h 4m)
Will office (27m)
Paul office
Philip mobile

Figure 5: Awarenex User Interface [Tang03]

Figure 5 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.

3.3.3 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).

3.4 *Context-aware asynchronous Communication*

In chapter 2.4.2 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.

3.4.1 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.

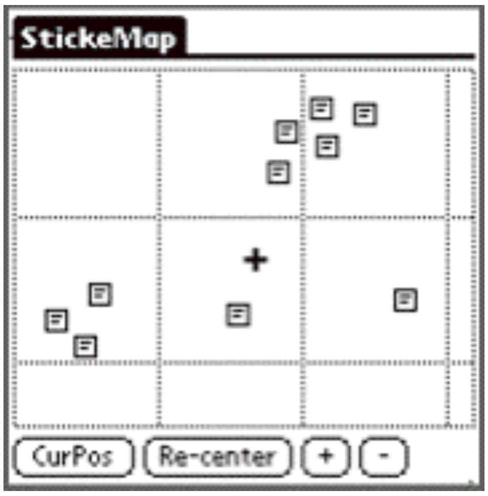


Figure 6: Stick-e note: Implementation on PDA [Pasc97]

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).

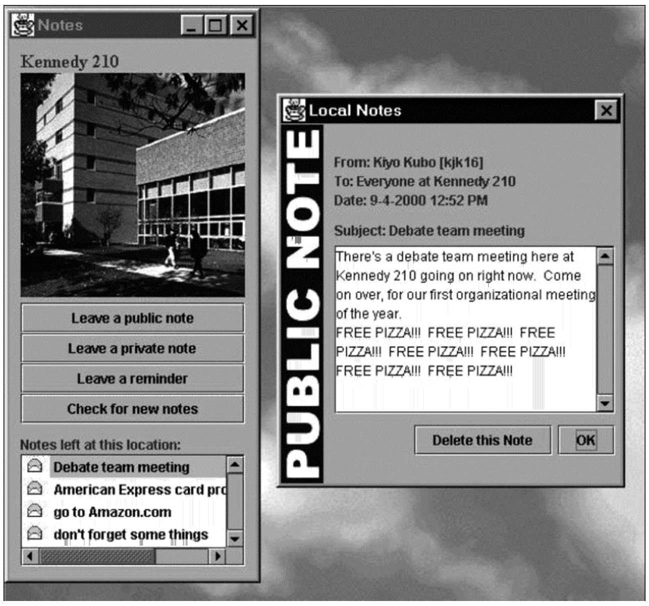
As the *stick-e notes* system can be seen as the ancestor of all context- (or location-) aware applications that make use of the post-it-metaphor, the concept of virtual post-its in GISS as well as the concept of context-aware reminders is also based on this work.

3.4.2 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 7.

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.



Notes left at this location:
Debate team meeting
American Express card pr
go to Amazon.com
don't forget some things

From: Kiyo Kubo [kjk16]
To: Everyone at Kennedy 210
Date: 9-4-2000 12:52 PM
Subject: Debate team meeting

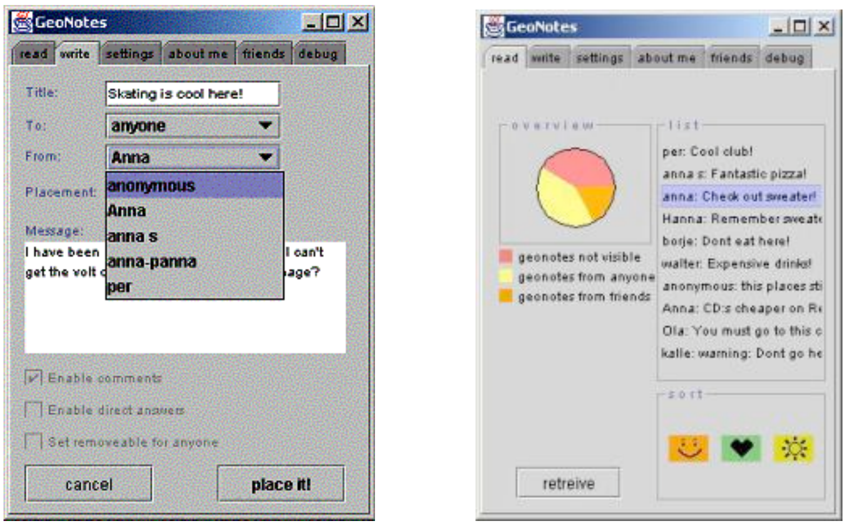
There's a debate team meeting here at Kennedy 210 going on right now. Come on over, for our first organizational meeting of the year.
FREE PIZZA!!! FREE PIZZA!!! FREE PIZZA!!! FREE PIZZA!!! FREE PIZZA!!! FREE PIZZA!!! FREE PIZZA!!!

Figure 7: Screen capture of eGraffiti [Burr02]

3.4.3 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 content 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.



Title: Skating is cool here!
To: anyone
From: Anna
anonymous
Anna
anna s
anna-panna
per
Placement:
Message:
I have been I can't
get the volt c age?

[✓] Enable comments

🟥 geonotes not visible
🟨 geonotes from anyone
🟧 geonotes from friends

per: Cool club!
anna s: Fantastic pizza!
anna: Check out sweater!
Hanna: Remember sweat
borje: Dont eat here!
walter: Expensive drinks!
anonymous: this places sti
Anna: CD:s cheaper on R
Ola: You must go to this c
kalle: warning: Dont go he

Figure 8: GeoNotes Posting (a) and Browsing (b) Interface [Espi01]

3.5 *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.

3.5.1 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.

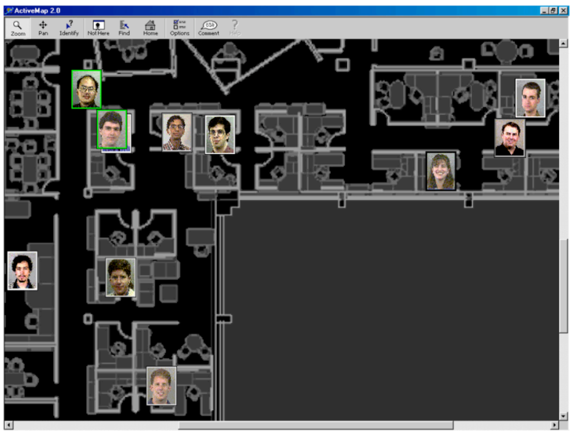


Figure 9: ActiveMap [McCa99]

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.

3.5.2 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).

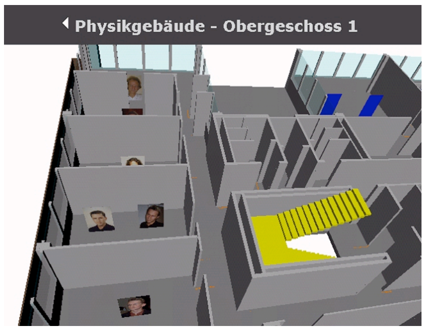


Figure 10: Screenshot of CampusSpace [Perv04]

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.

3.6 *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].

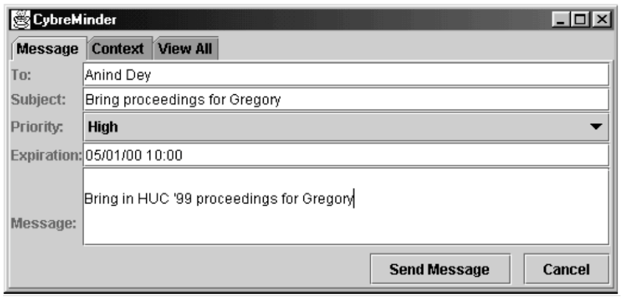
Not very much research has been done in the area for context-aware reminders yet. In 2000, Dey and Abowd created *CybreMinder* as an application of their Context Toolkit. This system is described in the following chapter. The *MemoClip* developed by Beigl follows a more hardware-driven approach and will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

3.6.1 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 11.

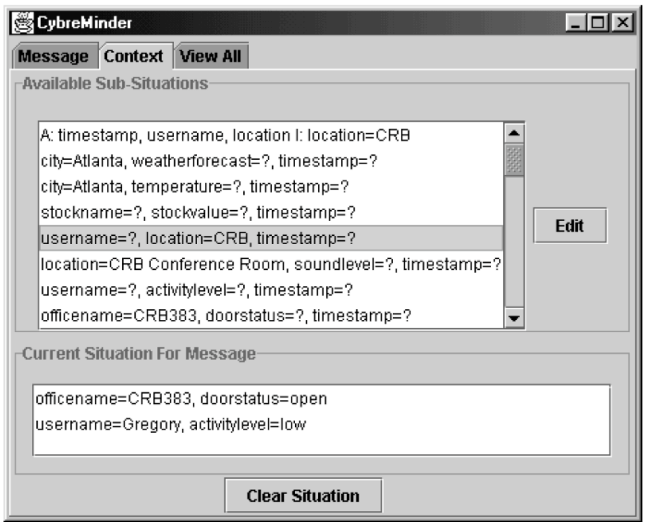


To: Anind Dey
Subject: Bring proceedings for Gregory
Priority: High
Expiration: 05/01/00 10:00
Message: Bring in HUC '99 proceedings for Gregory

Figure 11: 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 12).

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.



A: timestamp, username, location I: location=CRB
city=Atlanta, weatherforecast=?, timestamp=?
city=Atlanta, temperature=?, timestamp=?
stockname=?, stockvalue=?, timestamp=?
username=?, location=CRB, timestamp=?
location=CRB Conference Room, soundlevel=?, timestamp=?
username=?, activitylevel=?, timestamp=?
officename=CRB383, doorstatus=?, timestamp=?

officename=CRB383, doorstatus=open
username=Gregory, activitylevel=low

Figure 12: 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.

3.6.2 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 13). A central system, which communicates with both the beacons and the *MemoClips*, manages the placed reminders and is used to enter new reminders.

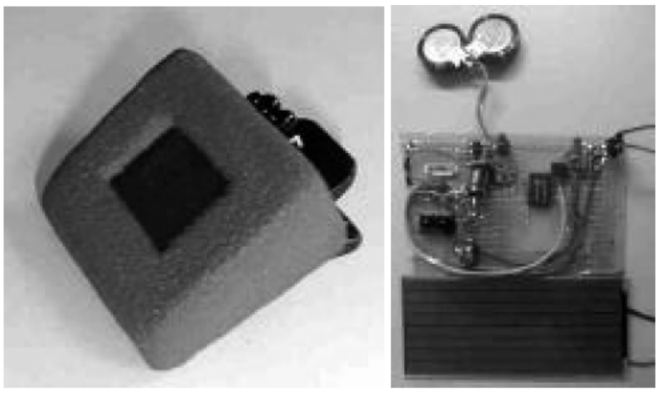


Figure 13: MemoClip and location beacon [Beig00]

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.

3.7 *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 2.3.2 and 2.3.4.

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

4 Requirement Analysis

From the goals given in chapter 1.2 we can derive some requirements the implemented system has to comply. These requirements are a prerequisite in order to create an architecture, which is capable of achieving the defined goals.

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.

4.1 *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.

4.2 *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 3.4).

4.3 *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.

4.4 *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.

4.5 *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.

4.6 *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.

4.7 *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.

4.8 *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).

5 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.

5.1 *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 14 is subject of [Holz04] and provides technology-independent information about a user's location. The lower part of Figure 14 shows the components that are subject of this work and provide users with a context-aware group-interaction.

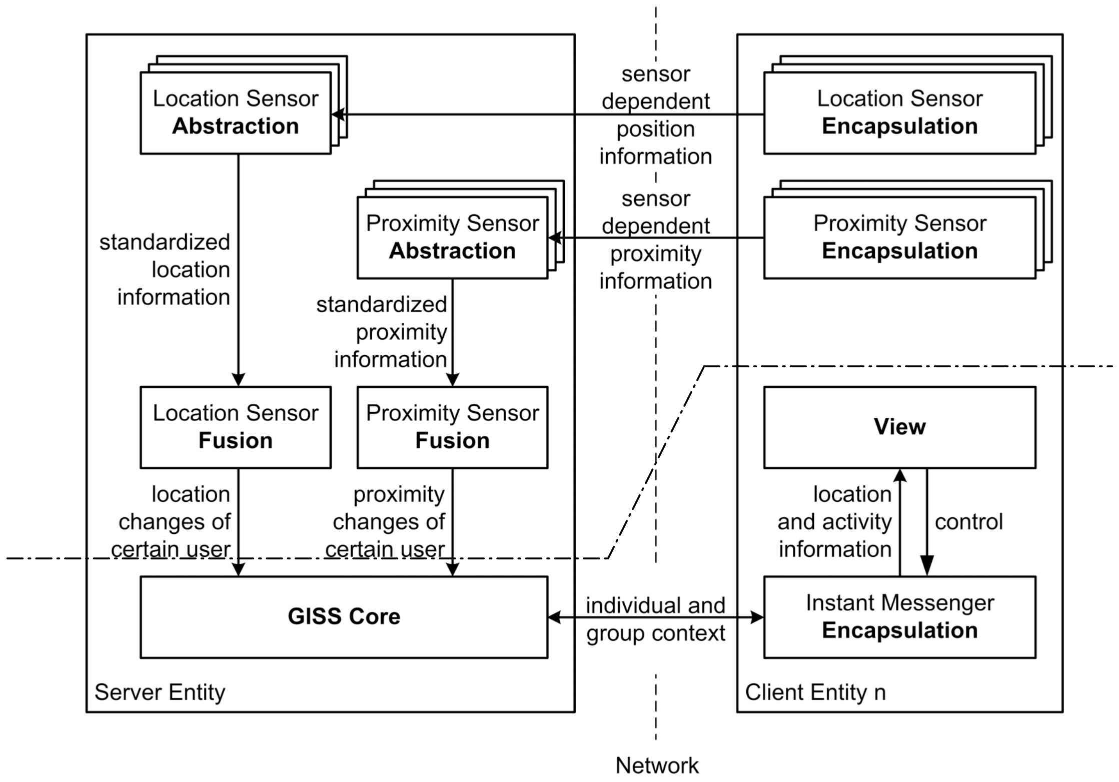

Location Sensor Abstraction
sensor dependent position information
Location Sensor Encapsulation
standardized location information
Proximity Sensor Abstraction
sensor dependent proximity information
Proximity Sensor Encapsulation
standardized proximity information
Location Sensor Fusion
Proximity Sensor Fusion
View
location changes of certain user
proximity changes of certain user
location and activity information
control
GISS Core
individual and group context
Instant Messenger Encapsulation
Server Entity
Client Entity n
Network

Figure 14: 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 5.2).

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 5.4). The *View*-Module is an exchangeable component, which is intended to provide the user with sophisticated location awareness (cf. chapter 5.5.2).

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 5.3.3). 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 5.5.3 and 5.5.6). 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 5.5).

5.2 *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.

5.2.1 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.

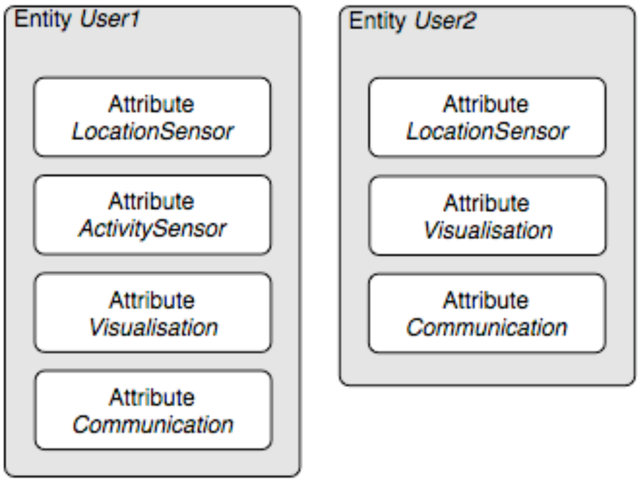

Entity *User1*
Attribute *LocationSensor*
Attribute *ActivitySensor*
Attribute *Visualisation*
Attribute *Communication*
Entity *User2*
Attribute *LocationSensor*
Attribute *Visualisation*
Attribute *Communication*

Figure 15: SiLiCon-Context-Framework Entities and Attributes

In Figure 15, 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.

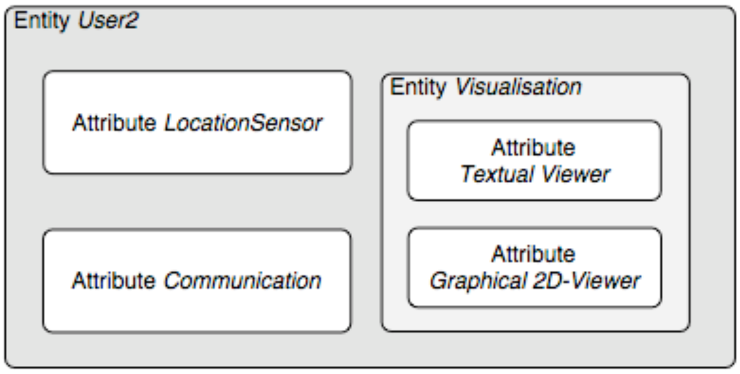

Entity *User2*
Attribute *LocationSensor*
Attribute *Communication*
Entity *Visualisation*
Attribute *Textual Viewer*
Attribute *Graphical 2D-Viewer*

Figure 16: 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 16, 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.

5.2.2 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.

5.2.3 Architecture

Figure 17 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.

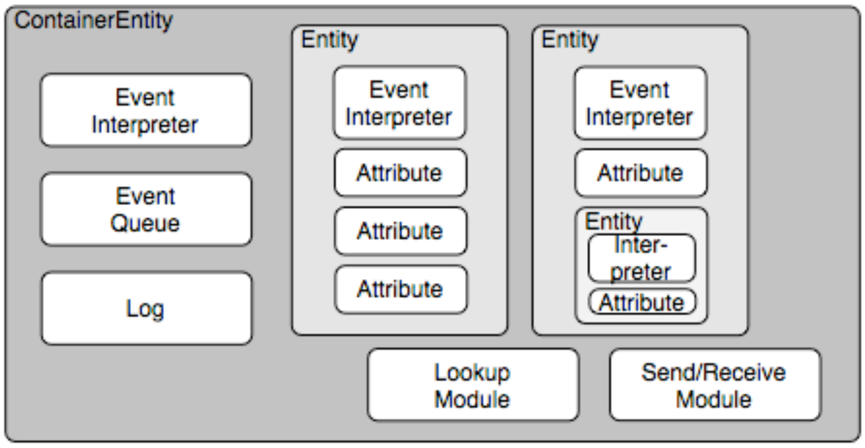


Figure 17: SiLiCon-Context-Framework architectural concept

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).

5.3 *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.

5.3.1 Used Context-Dimensions

As already described in chapter 2.3.2, 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 providing this kind of information with finer granularity, which can be achieved fairly easy by adding further sensors to the extensible architecture of *GISS*.

5.3.2 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 18). 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)

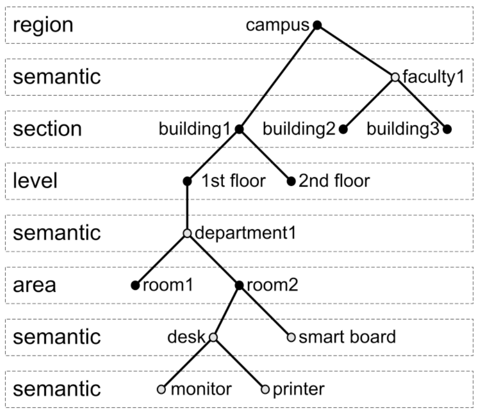


Figure 18: Symbolic location containment hierarchy [Holz04]

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 5.5.2.

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 5.1).

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.

5.3.3 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 5.1, the user interface is based on a standard instant messenger application (the motivation for this decision will be given in chapter 5.4). 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 5.4.3) 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 computer (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 5.5 for a detailed description of the use of this context dimension).

5.3.4 Deriving Group-Context

As already stated in chapter 5.3.1, the context of a group can be derived from the individual contexts of its members. As can be seen in Figure 19, 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 5.5.3). 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.

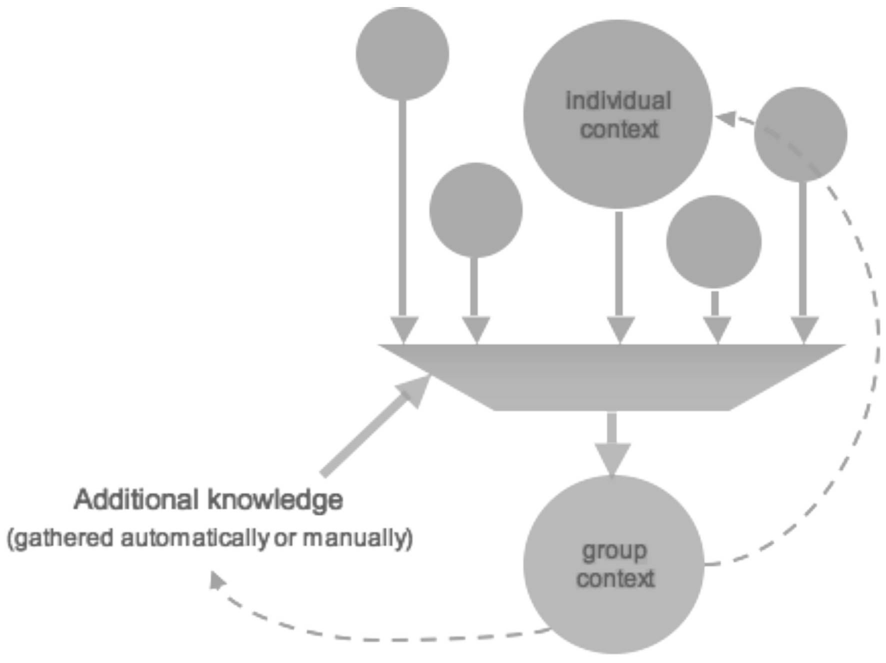


Figure 19: Gathering group context

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 5.5.3) is recognized by spatial proximity and the users' current availability status.

5.4 *User-Interface Design*

As can already be seen in the requirements analysis (chapter 4), a proper design of user interface is a crucial issue in terms of a broad acceptance. Furthermore, as already stated in chapter 5.3.3, 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 5.4.2 and 5.4.3 has been made. The process of evaluation and the arguments for and against the considered approaches are subject of this chapter.

5.4.1 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 20, left and below the VRML-visualisation). As a means for synchronous communication is a crucial part of the system (cf. chapter 4.2), 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 21).

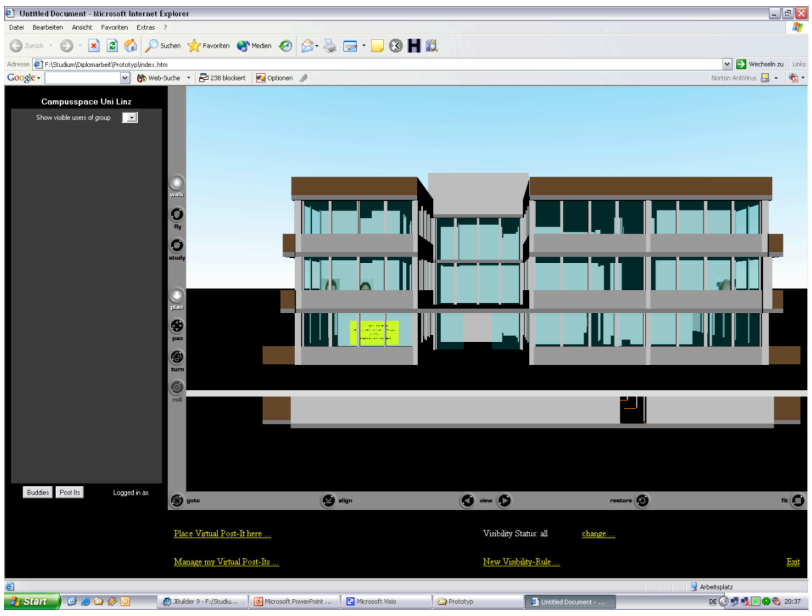


Figure 20: Proprietary user interface (design study)

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 20, see chapter 3.5.2). 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.

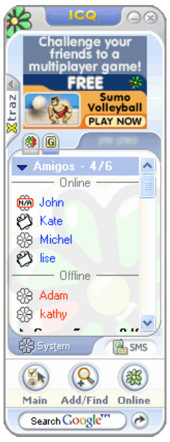


Figure 21: Example for a possibly suitable existing user interface

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.

5.4.2 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 22. 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.

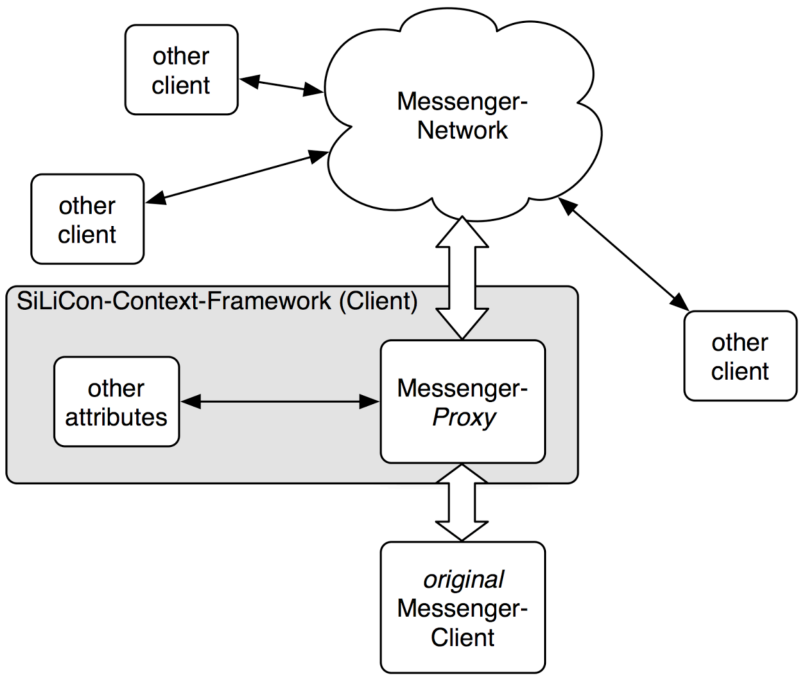


Figure 22: Interface using an original messenger-client

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 23.

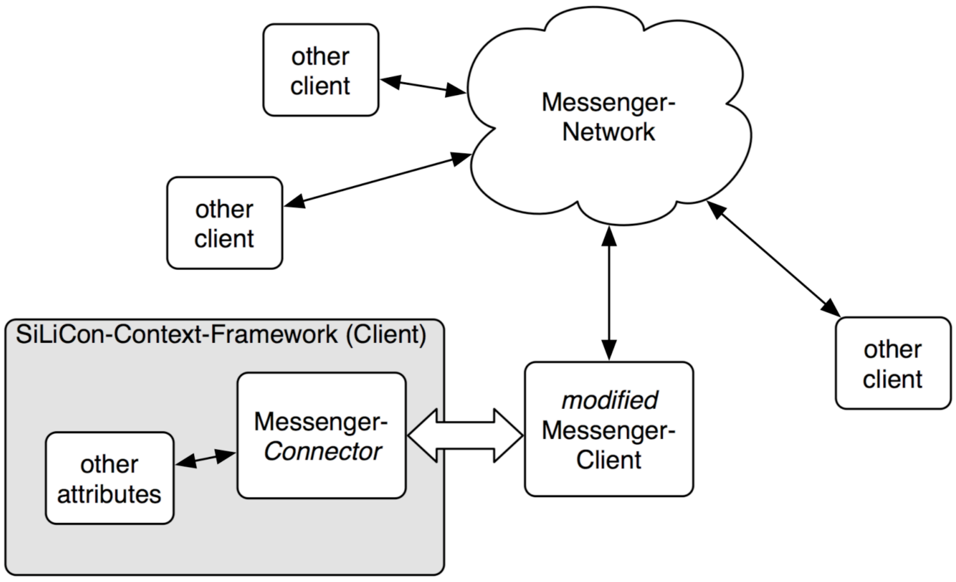


Figure 23: Interface using a modified messenger-client

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*.

5.4.3 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 6.1.2). 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 6.1.2). A few minor changes that were needed to implement the services of *GISS* are described in chapter 6.2.

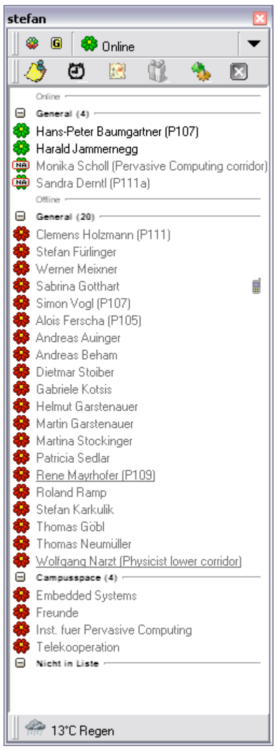

stefan
Online
Online
General (4)
Hans-Peter Baumgartner (P107)
Harald Jammernegg
Monika Scholl (Pervasive Computing corridor)
Sandra Derntl (P111a)
Offline
General (20)
Clemens Holzmann (P111)
Stefan Fürlinger
Werner Meixner
Sabrina Gotthart
Simon Vogl (P107)
Alois Ferscha (P105)
Andreas Auinger
Andreas Beham
Dietmar Stoiber
Gabriele Kotsis
Helmut Garstenauer
Martin Garstenauer
Martina Stockinger
Patricia Sedlar
Rene Mayrhofer (P109)
Roland Ramp
Stefan Karkulik
Thomas Göbl
Thomas Neumüller
Wolfgang Narzt (Physicist lower corridor)
Campusspace (4)
Embedded Systems
Freunde
Inst. fuer Pervasive Computing
Telekooperation
Nicht in Liste
13°C Regen

Figure 24: User-interface of SIM (with GISS-toolbar)

As can be seen in Figure 24, 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.

5.5 *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 2.3.4) are identified.

5.5.1 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	

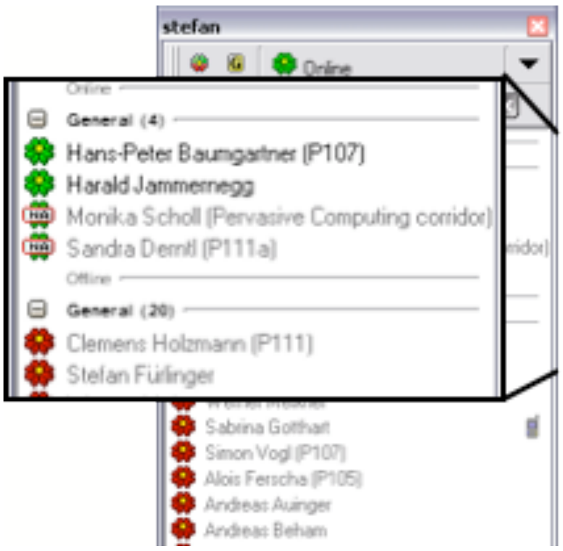
In the following chapters, the services listed above are described in detail from a conceptual point-of-view. The details of implementation are presented in chapter 6.2.

5.5.2 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 25. 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 18 (page 54). 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.


stefan

Online

Online

General (4)

Hans-Peter Baumgartner (P107)

Harald Jammernegg

Monika Scholl (Pervasive Computing corridor)

Sandra Derntl (P111a)

Offline

General (20)

Clemens Holzmann (P111)

Stefan Fürlinger

Sabrina Gotthart

Simon Vogl (P107)

Alois Ferscha (P105)

Andreas Auinger

Andreas Beham

Figure 25: 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 26). 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.

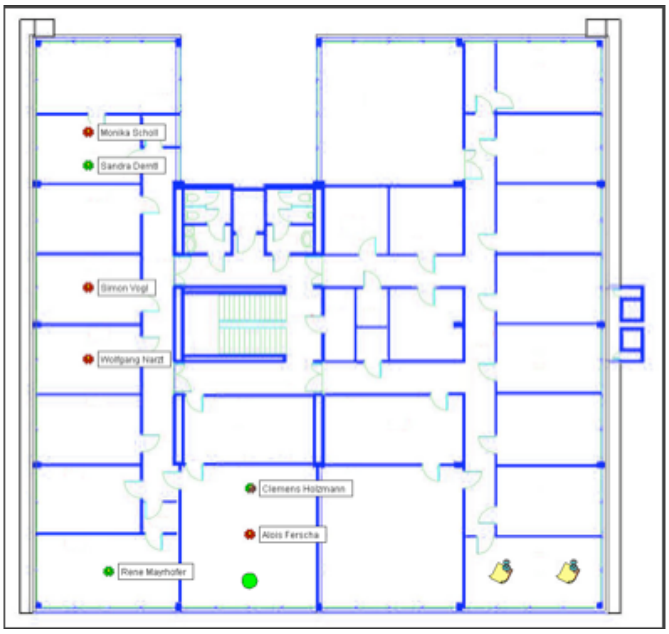


Figure 26: Location awareness via graphical viewer

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 26. 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.

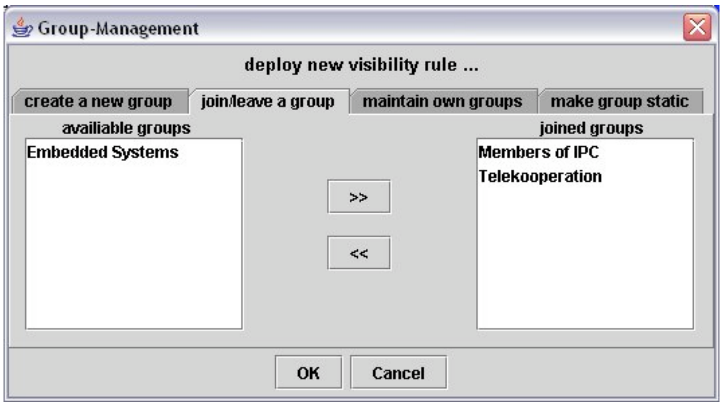
5.5.3 Forming and Managing Groups

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*, everybody 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.

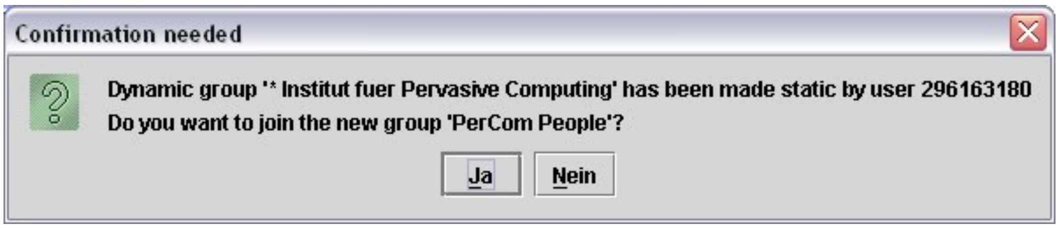


Embedded Systems

Members of IPC
Telekooperation

Figure 27: 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 27). 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 28). 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.



Dynamic group '* Institut fuer Pervasive Computing' has been made static by user 296163180
Do you want to join the new group 'PerCom People'?

Figure 28: 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.

5.5.4 Synchronous Group Communication

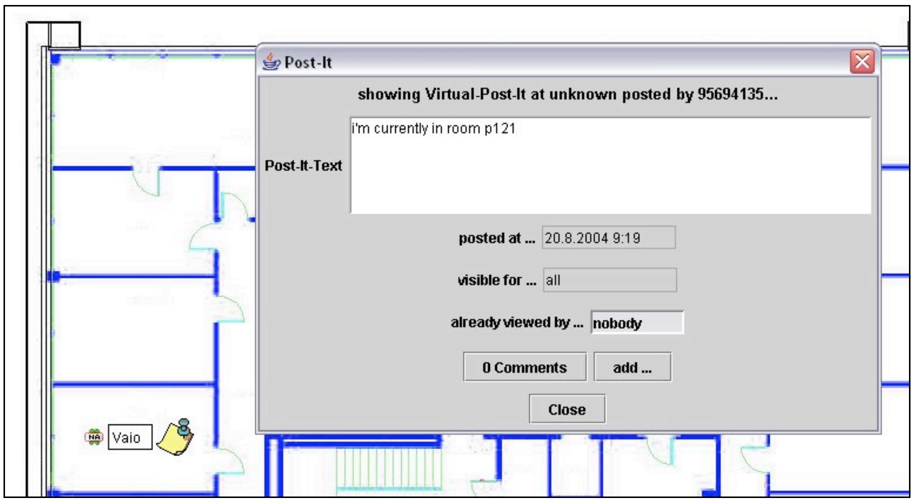
Means of synchronous communication as defined in chapter 2.4.2 are mainly intended for spontaneous, informal interaction among people. Today, instant messengers are widely used for this purpose, at least in direct person-to-person communication. Most of today's messengers lack the possibility to easily 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).

5.5.5 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 29).


Post-It

showing Virtual-Post-It at unknown posted by 95694135...

Post-It-Text: i'm currently in room p121

posted at ... 20.8.2004 9:19

visible for ... all

already viewed by ... nobody

Figure 29: 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 31). 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 5.5.6), 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 30.

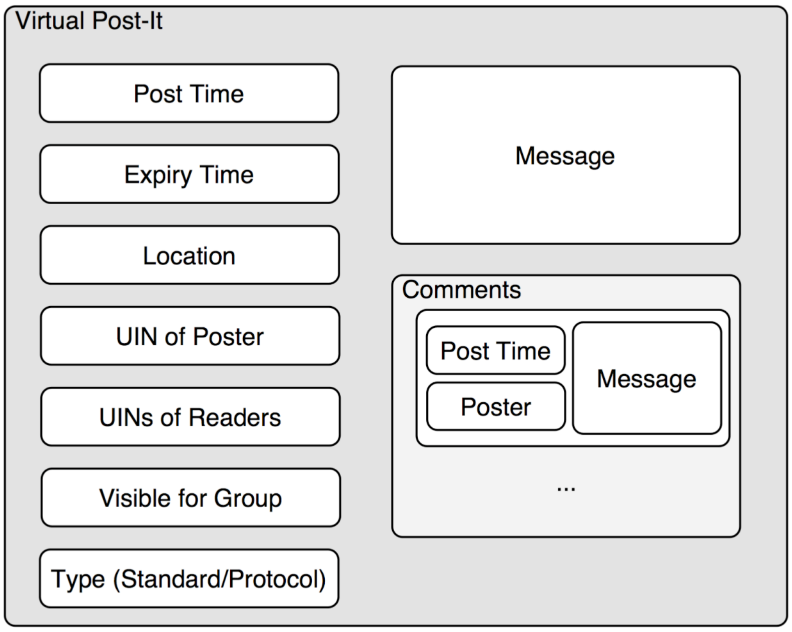
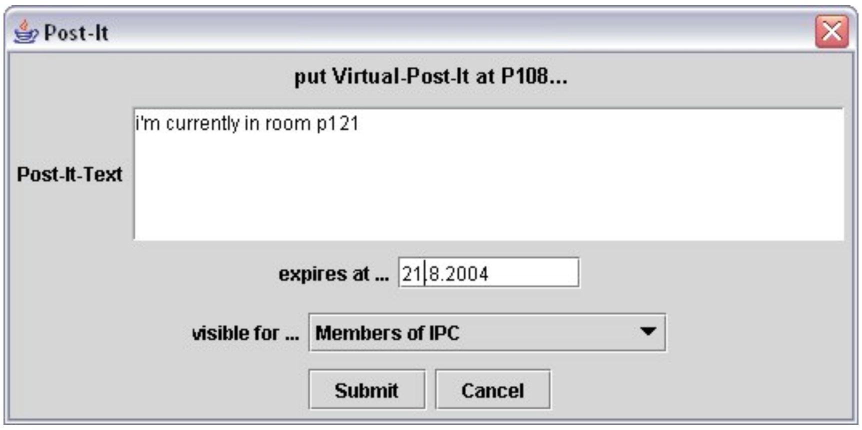


Figure 30: Structure of a Virtual Post-It

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.



i'm currently in room p121

expires at ... 21.8.2004

visible for ... Members of IPC

Figure 31: Interface for entering virtual post-its

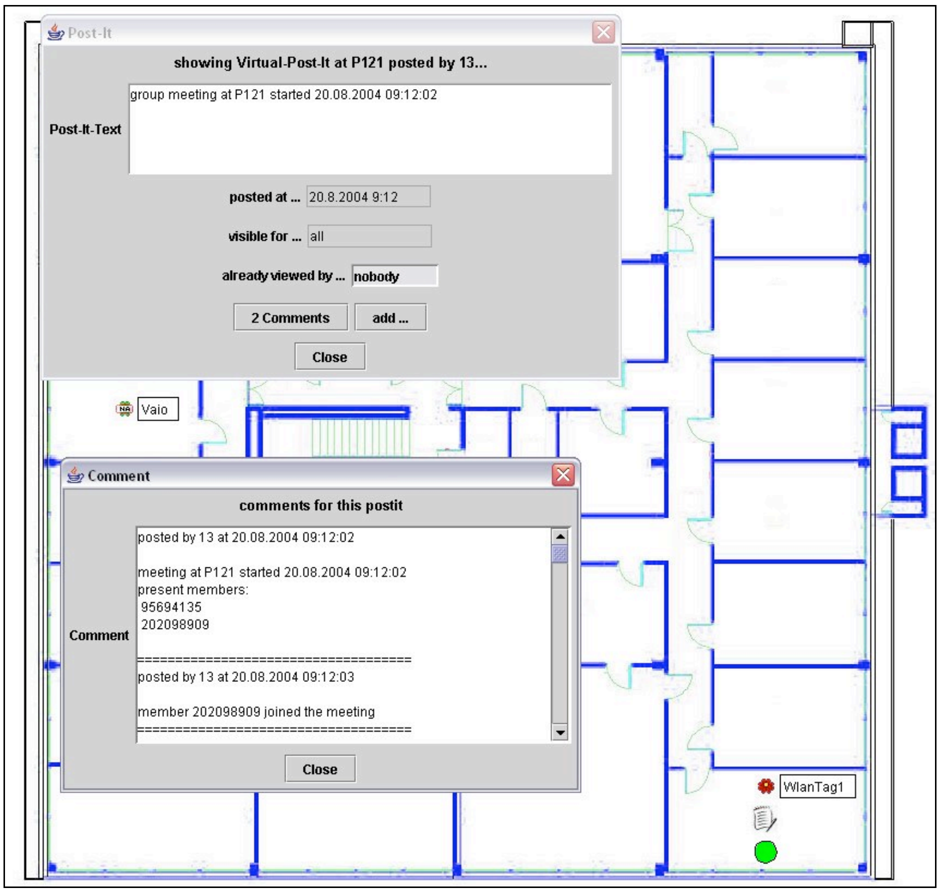
Virtual post-its can also be opened via the 2D-viewer-interface where they are depicted as blue post-its. When they have already been read, their colour changes to yellow. Via this interface, the user is offered the possibility to access also already read post-its.

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.

5.5.6 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 32). 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. Possible improvements of the recognition algorithm are discussed in chapter 8.3.



showing Virtual-Post-It at P121 posted by 13...

Post-It-Text: group meeting at P121 started 20.08.2004 09:12:02

posted at ... 20.8.2004 9:12

visible for ... all

already viewed by ... nobody

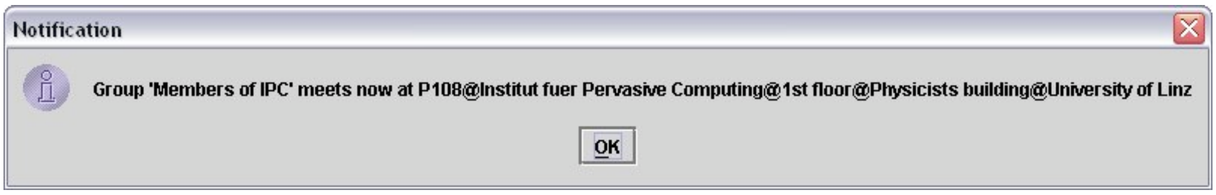
comments for this postit

Comment:

```
posted by 13 at 20.08.2004 09:12:02

meeting at P121 started 20.08.2004 09:12:02
present members:
 95694135
 202098909

====================================
posted by 13 at 20.08.2004 09:12:03

member 202098909 joined the meeting
====================================
```

Figure 32: Screenshot for gathering recognized in room in the lower right corner

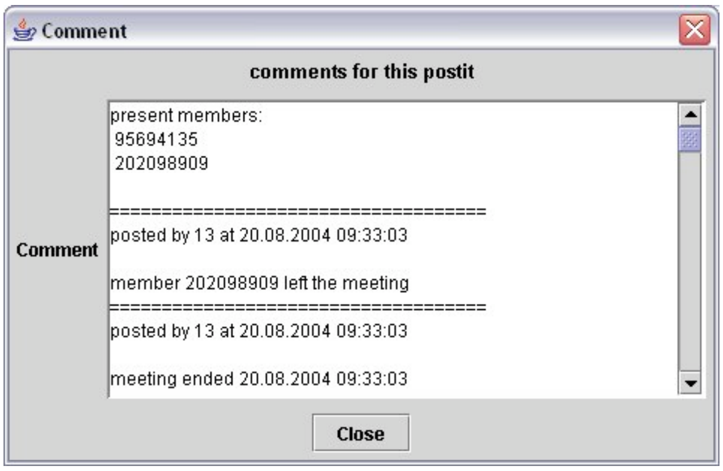
Whenever a gathering is recognized, several actions are triggered to support the group by carrying out routine work automatically. First, all absent (and not *busy*) group members are sent a message to notify them of the gathering and where it takes place (cf. Figure 33). This allows them to join, if they are nearby or at least contribute by synchronous communication via instant group messaging.



Group 'Members of IPC' meets now at P108@Institut fuer Pervasive Computing@1st floor@Physicists building@University of Linz

Figure 33: 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 34).



```
present members:
 95694135
 202098909

==================================
posted by 13 at 20.08.2004 09:33:03

member 202098909 left the meeting
==================================
posted by 13 at 20.08.2004 09:33:03

meeting ended 20.08.2004 09:33:03
```

Figure 34: 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.

5.5.7 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 35, the interface for entering a reminder is shown in Figure 36.

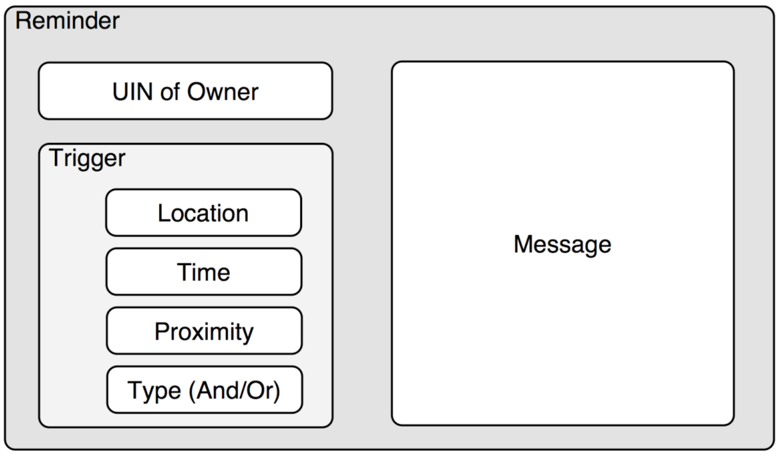
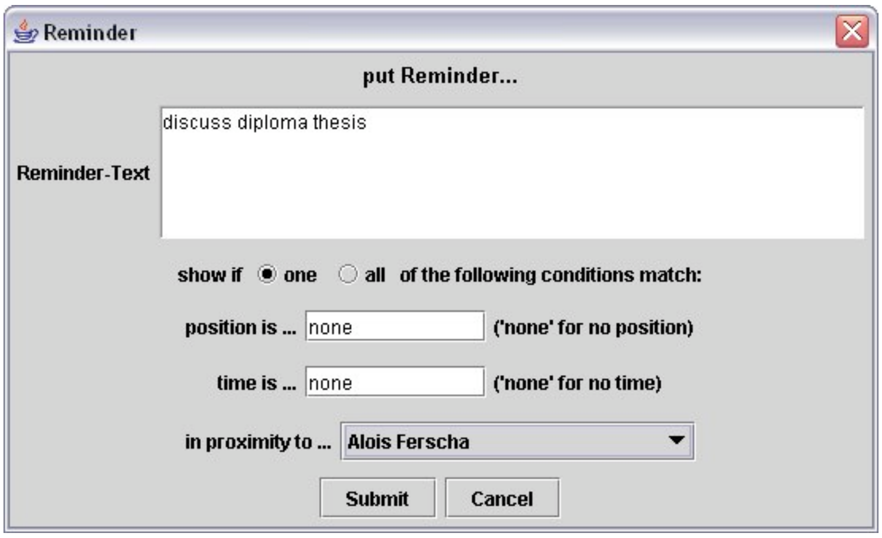


Figure 35: Structure of Reminder

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.



Reminder-Text: discuss diploma thesis

show if [✓ one] all of the following conditions match:

position is ... none ('none' for no position)

time is ... none ('none' for no time)

in proximity to ... Alois Ferscha

Figure 36: 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.

5.5.8 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 37.

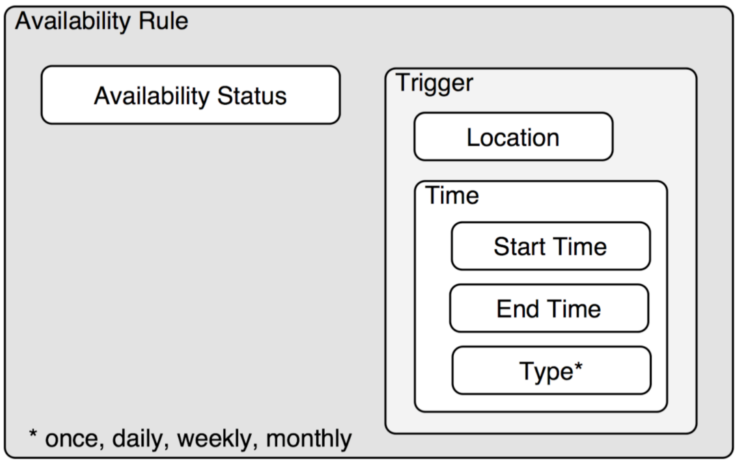


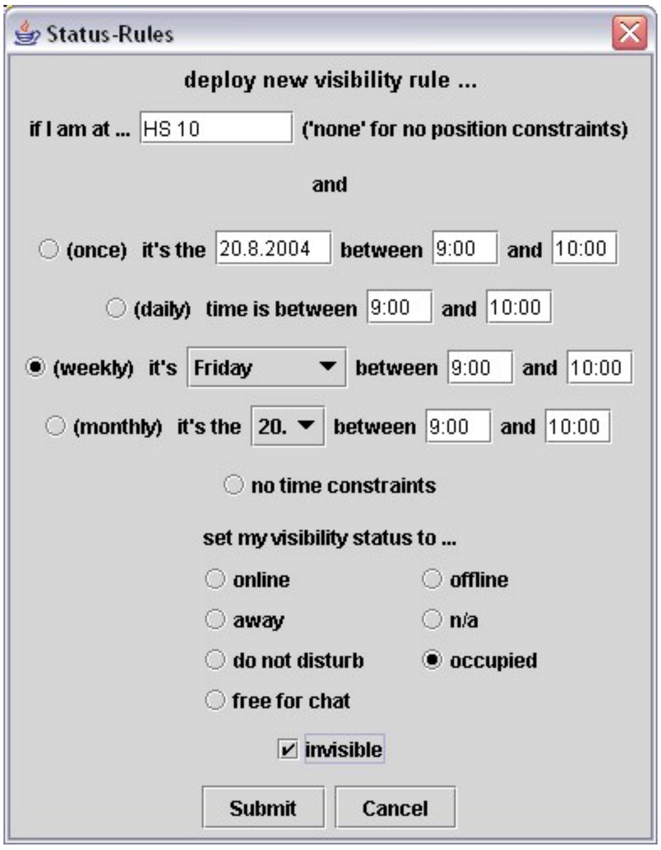
Figure 37: Structure of Availability Rule

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 38).

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



Status-Rules

deploy new visibility rule ...

if I am at ... HS 10 ('none' for no position constraints)

and

(once) it's the 20.8.2004 between 9:00 and 10:00

(daily) time is between 9:00 and 10:00

[✓ (weekly) it's Friday between 9:00 and 10:00]

(monthly) it's the 20. between 9:00 and 10:00

no time constraints

set my visibility status to ...

online / offline / away / n/a / do not disturb / [✓ occupied] / free for chat

[✓ invisible]

Submit Cancel

Figure 38: 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.

6 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 5.5 are realized.

6.1 *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.

6.1.1 Communication between Entities

As already stated in chapter 5.2.2, 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 5.2.2), 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.

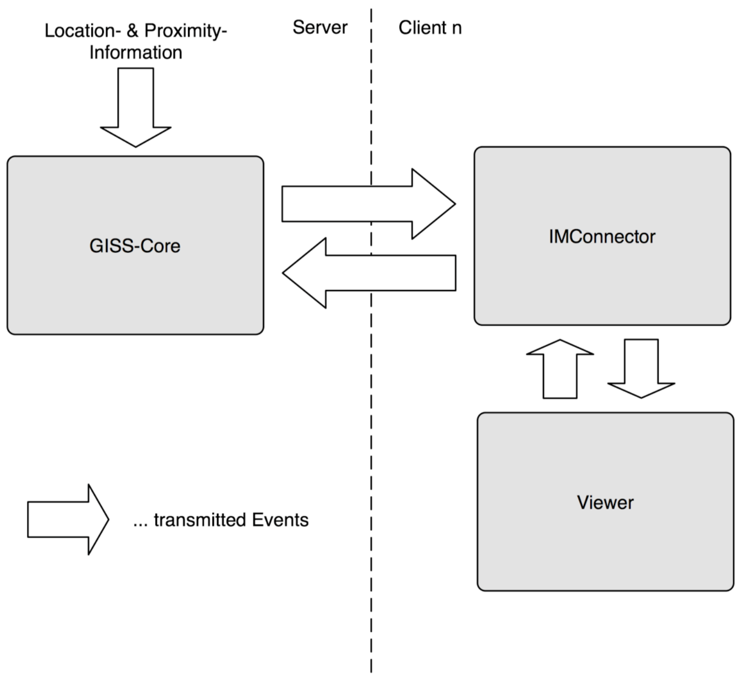


Figure 39: Flow of Events between Server and Clients

The flow of events between the attributes is depicted in Figure 39. 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.

6.1.2 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 40). 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.

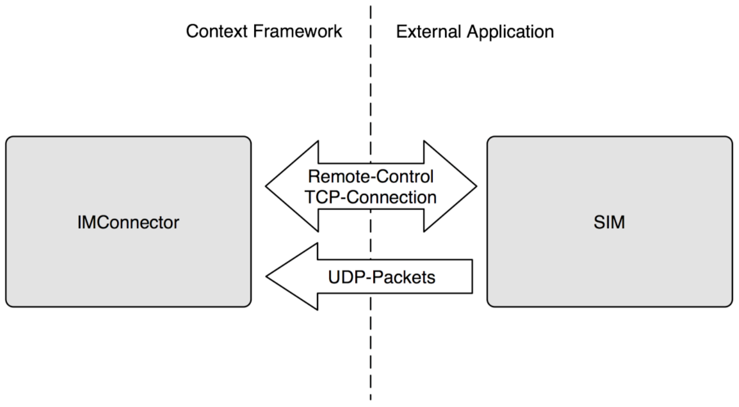


Figure 40: Flow of communication between Context-Framework and *SIM*

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 6.2.2)

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 6.1.3); all of these external communication channels are opened only locally, as said before, the complete network communication is realized via the SiLiCon Context Framework.

6.1.3 Database

The application data on server side is stored persistent in a database (*MySQL*) according to the data-model presented in Figure 41. 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.

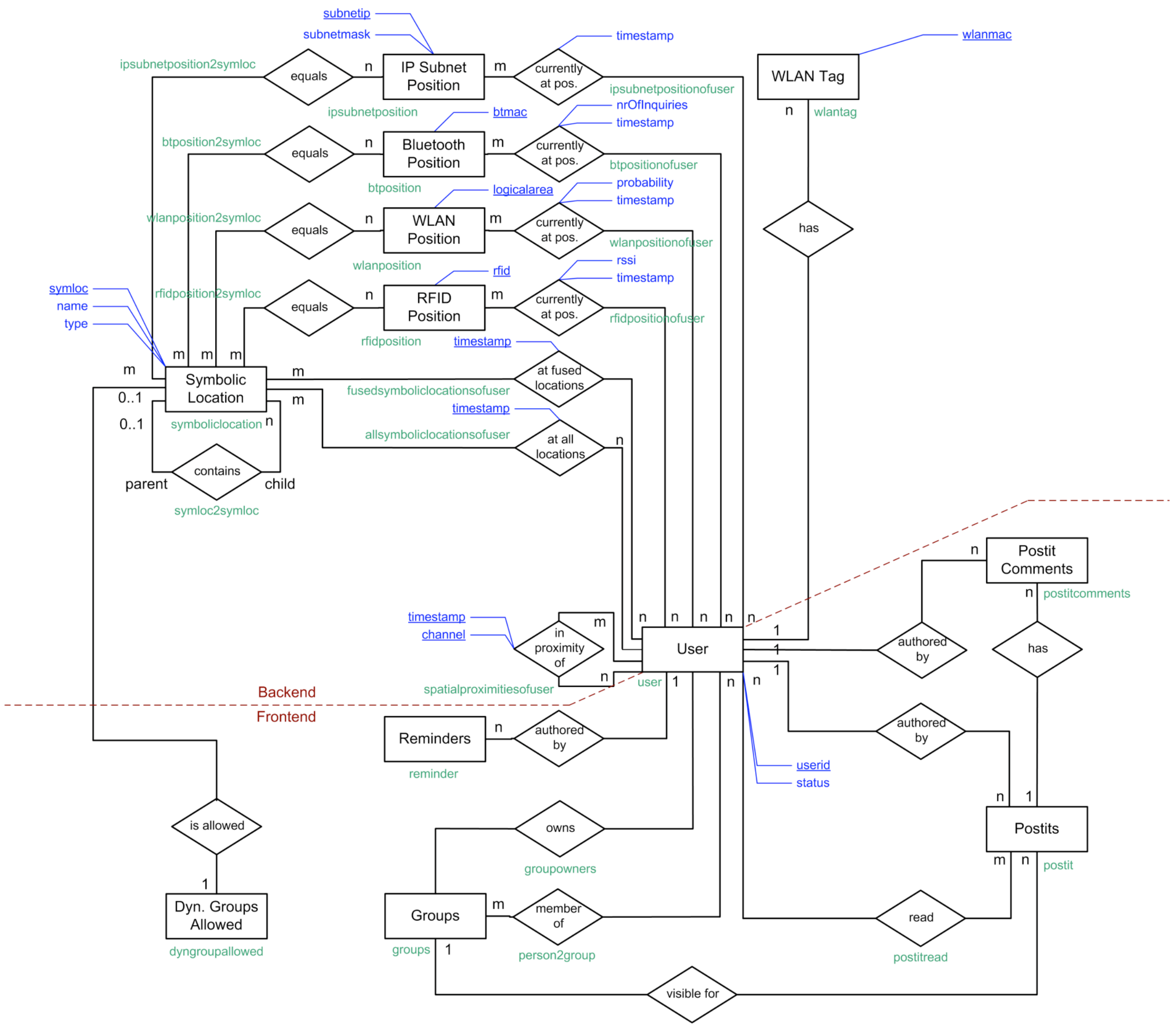


Figure 41: ER-Model of *GISS*-Data-Structures

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.

6.2 *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.

6.2.1 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.

6.2.2 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 (cf. chapter 8.3).

6.2.3 Triggering of Reminders

As presented in chapter 5.5.7, 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 6.2). 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.

6.2.4 Formation of Dynamic Groups

In chapter 5.5.3, 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 service, 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 42. 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.

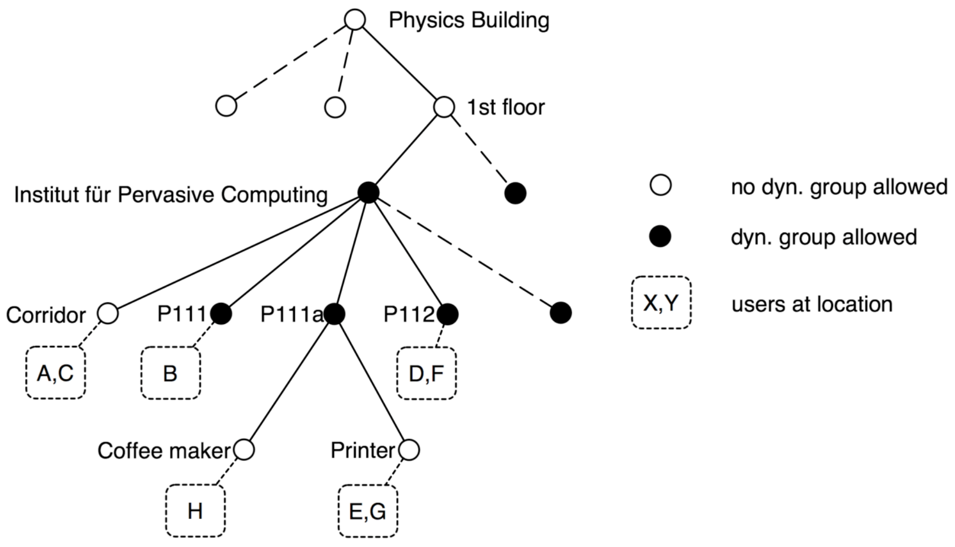


Figure 42: Formation of dynamic groups (example)

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.

6.2.5 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 5.3.2) 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 (cf. chapter 8.3 for possible improvements).

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.

6.2.6 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.

7 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.

7.1 *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.

7.2 *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.

7.3 *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.

8 Conclusion

In this chapter, the results of this work are summarized. After presenting the achieved goals, the fields where open issues have shown up are discussed. In the last section, possible approaches for solving these open issues in future are given.

8.1 *Achieved goals*

The goals defined in chapter 1.2 have been reached. The system described in this work is in a stable version and has been tested with up to four clients. The implemented services provide a good overview of what is possible with context information used to support interaction in groups.

Not repeating the results given in chapter 1.3 here, this chapter should provide a rather informal description of what has been achieved. The *integration of location information* in the instant messenger user-interface is considered the maybe most useful feature of *GISS*. By providing an undistracting but nevertheless always visible representation of one's friends location, a real surplus value is provided to conventional instant messaging by enhancing the knowledge about the others' context and thus improving the group members' feeling of belonging together.

With *virtual post-its* and *instant group messaging*, the users are provided with means of communication that go beyond the possibilities of email or forums on the asynchronous side or conventional instant messengers on the synchronous side. Especially instant group messaging is considered useful for spontaneous communication with a larger amount of people. Using virtual post-its to realise a *virtual pin board* is considered an important use case for this feature, because of the enhanced features a virtual-post-it provides in comparison with conventional bulletin boards (commenting notes, awareness of who has already read the note, automatic removing on expiration).

The *recognition of group gatherings* out of the group members' context has been implemented prototypical and needs refinement. Nevertheless the services triggered whenever a meeting is recognized (notification of absent members, support in taking meeting minutes) have been proven to be really useful during testing.

Reminders and *availability rules* do not directly support group interaction but have shown to be valuable extension in the daily work. Especially reminders, that show up on the proximity of specified people are considered a feature with high practical relevance. As availability has chosen to be globally uniform and not being able to be specified for each group for usability reasons, this feature has lost much of its potential. But with the decision to build upon an existing system that already comes with an availability status, the introduction of another availability concept (with an individual status for each group one is member in) would have caused confusion on user side.

The implemented system has completely fulfilled the requirements of modularity and extensibility by being built upon the SiLiCon Context Framework. However, the use of this framework has caused the application to behave more indolently and to need more resources than it would have been the case when implementing a monolithic, stand-alone application. As the need for resources stays within limits (constantly about 10% of CPU and network utilisation each approximately on recent notebook hardware), this has no impact on the practical usability of the system.

To come to an conclusion, *GISS* has fulfilled the specified requirements but still remains a proof-of-concept, as some of the features need to be refined to better meet the requirements of everyday use. A further open question is, if users are not overwhelmed with the manifold of different services provided by *GISS*. Although nobody is forced to use all of the services, the pure availability could possibly cause a lack of acceptance. This will have to be evaluated in field tests, which are subject of future work.

8.2 *Open issues*

There are several open issues in different services of *GISS* that need to be implemented or refined to improve the practical usability. The following issues have been identified during testing and discussing the system as it is implemented now.

The *graphical viewer* needs to be extended to properly display larger numbers of users and post-its in a room. In the now chosen implementation, a tiled view is provided but at larger numbers of artefacts (depending on the size of the room), the graphical representations overlap and become more and more unreadable. This could be rather easily solved by grouping them if a certain amount is exceeded. Another open point concerns the discernability of virtual post-its. As for now, post-its are only represented by their icon, which changes colour, when the post-it has been read. When a larger amount of post-its is placed in one room, it becomes hard to impossible, to distinguish them in terms of remembering, which note lays behind them. This could be solved by always displaying only one icon, when post-its are available in a room, which leads to a list of the post-its (to be distinguished by the poster, the posting date and the first few words of the message), where unread ones are displayed uppermost. Last but not least, for demonstration purposes only one floor has been adopted yet, the rest of the campus is subject of future work.

The algorithm for the *recognition of gatherings* needs refinements. With the currently available context information, a really dependable recognition is nearly impossible. An improvement could possibly be reached by introduction of further meta-data to locations (like markers for meeting rooms or the number of workplaces in a room). However, a dependable recognition will only be possible with the utilisation of further context information, especially in the area of activity context.

The triggers for *reminders* should be extended to arbitrary context sources; an obvious extension would be to allow the triggering on another users availability status. If the triggering sources were extensively extended, there would be need of a different means of entering those triggers, as the approach chosen now quickly becomes confusing. A possible approach would be a textual entering of the triggers like it has been done in *CybreMinder* [Dey00].

Taking into account a *user's history* when analysing his context could significantly improve and extend the possible services. For location context, the data is already saved in this application but not further utilized. The use of context history could open a wide field of extension possibilities, especially in terms of context prediction and proactive triggering of services.

8.3 *Future work*

The most important point of future work is an extensive field test to evaluate acceptance and scalability of the system. As stated before, *GISS* has only been tested with up to four users, but it is not known, if the underlying Context Framework can also handle several hundreds of users. As unknown as the issue stated before is the acceptance of the system in real life settings. Clarification of these points has to be the first step of future work.

To improve practical usability, the interfaces of *GISS* will need a rework to fit the common look and feel of the operation system and the user interface of the instant messenger. Because of using the Java Swing library for building user interface, the final appearance is not always foreseeable but is often worse than expected. Using a different, proprietary library for user interfaces could possibly solve this problem.

Last but not least, the extensions mentioned in the former chapter should be implemented to improve the functionality of *GISS* itself. Especially the extension to maps of the whole campus would be a prerequisite for serious field tests with a larger number of people.

9 References

[AIM04] Website of AOL instant messenger: http://www.aim.com (19.08.04)

[Bark03] Barkhuus L., Dey A.: Is Context-Aware Computing Taking Control Away from the User? Three Levels of Interactivity Examined. Proceedings of the Fifth International Conference on Ubiquitous Computing UbiComp 2003 (Seattle, USA, 2003), Springer, p. 150-156.

[Bark04] Barkhuus L., Dourish P.: Everyday Encounters with Ubiquitous Computing in a Campus Environment. Proceedings of the Sixth International Conference on Ubiquitous Computing UbiComp 2004 (Nottingham, UK, 2004), Springer, to appear.

[Beer03] Beer W., Christian V., Ferscha A., Mehrmann L.: Modeling Context-aware Behavior by Interpreted ECA Rules. Proceedings of the International Conference on Parallel and Distributed Computing EUROPAR'03 (Klagenfurt, Austria, 2003). Springer Verlag, LNCS 2790, p. 1064-1073.

[Beig00] Beigl M.: MemoClip: A Location based Remembrance Appliance. Personal Technologies 4 (2000), p. 230-234.

[Bent97] Bentley R., Appelt W.: Designing a System for Cooperative Work on the World-Wide Web: Experiences with the BSCW System. Proceedings of HICSS'30: The Hawaii International Conference on the System Sciences, (Maui, USA, 1997), IEEE Computer Society Press.

[Brow96] Brown P.J.: The stick-e document: a framework for creating context-aware applications. Electronic Publishing ´96 (1996), p. 259-272.

[Brow97] Brown P.J., Bovey J.D., Chen X.: Context-Aware Applications: From the Laboratory to the Marketplace. IEEE Personal Communications, 4 (1997), p. 58-64.

[Broy04] Broy M: The Cawar-Project – context aware architectures @ Campus TU München-Garching. http://www.cawar.de (01.08.2004)

[BSCW04] Website of BSCW: http://bscw.fit.fraunhofer.de (19.08.04)

[Burr02] Burrell J., Gay G.K.: E-graffiti: evaluation real-world use of a context-aware system. Interaction with Computers 14 (2003), p. 301-312.

[Chen00] Chen G., Kotz D.: A survey of context-aware mobile computing research. Paper TR2000-381, Department of Computer Science, Dartmouth College, November 2000.

[Dey98] Dey A., Abowd G., Wood A.: CyberDesk: A framework for providing self–integrating context–aware services. Knowledge Based Systems 11(1) (September 1998), p. 3-13.

[Dey00] Dey A., Abowd G.: CybreMinder: A Context-Aware System for Supporting Reminders. Proceedings of the 2nd international symposium on Handheld and Ubiquitous Computing (Bristol, UK, 2000), p. 172-186.

[Dey00a] Dey A.: Providing Architectural Support for Building Context-Aware Applications. Ph.D. Thesis, Department of Computer Science, Georgia Institute of Technology, (Atlanta, USA, November 2000)

[Diap93] Diaper D., Sanger C. (eds): CSCW in Practise: and Introduction and Case Studies (1993), Springer.

[Dour92] Dourish P., Bellotti V.: Awareness and Coordination in Shared Workspaces. Proceedings of the 1992 ACM conference on Computer-supported cooperative work (Toronto, Canada, 1992) ACM Press, p. 107-114.

[Drur02] Drury J., Williams M.G.: A Framework for Role-Based Specification and Evaluation of Awareness Support in Synchronous Collaborative Applications. Eleventh IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises WETICE'02 (Pittsburgh, USA, June 2002), IEEE, p. 12-17.

[Espi01] Espinoza F., Persson P., Sandin A., Nyström H., Cacciatore E., Bylund M.: GeoNotes: Social and Navigational Aspects of Location-Based Information Systems. Proceedings of Ubicomp 2001: Ubiquitous Computing, International Conference (Atlanta, USA 2001) Springer, p. 2-17.

[Fers00] Ferscha, A.: Workspace Awareness in Mobile Virtual Teams. Proceedings of the IEEE 9th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'00) (Gaithersburg, USA, 2000). IEEE Computer Society Press, p. 272-277.

[Fert96] Fertig S., Freeman E., Gelernter D.: "Finding and Reminding" Reconsidered. SIGCHI Bulletin, Vol. 28 (1996).

[Grei88] Greif I.: Computer-Supported Cooperative Work - A Book of Readings, Morgan Kaufmann Publishers (1988)

[Gree91] Greenberg S. (ed): Computer-supported Cooperative Work and Groupware, Academic Press Ltd (1991)

[Gris03] Griswold W.G., Shanahan P., Brown S.W., Boyer R., Ratto M., Shapiro R.B., Truong T.M.: ActiveCampus – Experiments in Community-Oriented Ubiquitous Computing. UCSD CSE Technical Report #CS2003-0750.

[Grun94] Grundin J.: Groupware and social dynamics: eight challenges for developers. Communications of the ACM Vol. 37, Issue 1 (January 1994), p. 92-105.

[Gutw95] Gutwin C., Stark G., Greenberg S.: Support for Workspace Awareness in Educational Groupware. Conference on Computer Support for Collaborative Learning (Bloomington, USA, October 1995).

[Hers98] Herstad J., Van Thanh D., Audestad J.A.: Human-Human Communication in Context. International Workshop on Interactive Applications of Mobile Computing IMC'98 (1998).

[Holz04] Holzmann C.: Location Sensing for Context-Aware Applications. Diploma-Thesis, Institut fuer Pervasive Computing, University of Linz, to appear.

[ICQ04] Website of ICQ instant messenger: http://www.icq.com (19.08.04)

[Jabb04] Website of the Jabber Foundation: http://www.jabber.org (28.07.04)

[Jang00] Jang C.Y., Steinfeld C., Pfaff B.: Supporting Awareness among Virtual Teams in a Web-Based Collaborative System: The TeamSCOPE System. ACM SIGGROUP Bulletin Vol. 21, Issue 3 (December 2000), p. 28-34.

[John03] Johnson B., Yano D.: Improving Virtual Team Collaboration with Internet2. Presentation at the Internet2 Member Meeting in Fall 2003 (October 2003). http://www.internet2.edu/presentations/fall-03/20031015-Video-Johnson.pdf (02.08.2004).

[Liec00] Liechti O.: Awareness and the WWW: an overview. ACM SIGGROUP Bulletin Vol. 21, Issue 3 (December 2000), p. 8-12.

[Ljun99] Ljungstrand P.: WebWho: Support for Student Awareness and Coordination. Proceedings supplement of ESCSW'99 (Copenhagen, Denmark, 1999).

[McCa99] McCarthy J.F., Meidel E.S.: ActiveMap: A Visualization Tool for Location Awareness to Support Informal Interactions. Proceedings of the International Symposium on Handheld and Ubiquitous Computing (HUC) (Karlsruhe, Germany, 1999), p. 158-170.

[Miya86] Miyata Y., Norman D.A.: Psychological Issues in Support of Multiple Activities. User Centered Design, edited by Norman, D.A., Draper, S.W. Chapter 13 (1986), p. 265-284.

[MSN04] Website of MSN Messenger instant messenger: http://messenger.msn.com (19.08.2004)

[Osca04] Unofficial documentation of the OSCAR-protocol (original © by AOL): http://oilcan.org/oscar/ (23.08.2004)

[Pasc97] Pascoe J.: The Stick-e Note Architecture: Extending the Interface Beyond the User. Proceedings of the 2nd international conference of Intelligent user interfaces (Orlando, USA, 1997), p. 261-264.

[Pasc98] Pascoe J.: Adding generic contextual capabilities to wearable computers. Proceedings of the 2nd IEEE International Symposium on Wearable Computers ISWC'98 (Pittsburgh, USA, 1998), IEEE, pp. 92-99.

[Perv04] CampusSpace project description on the website of the Institute for Pervasive Computing at the University of Linz. http://www.soft.uni-linz.ac.at/Research/Projects/Wireless_Campus/Campus_Space/index.php (29.07.04)

[Ryan98] Ryan N., Pascoe J., Morse D.: Enhanced reality fieldwork: the context-aware archaeological assistant. Computer Applications and Quantitative Methods in Archaeology. V. Gaffney, M. van Leusen and S. Exxon, Editors. Oxford (1998).

[Schi94] Schilit B., Theimer M.: Disseminating active map information to mobile hosts. IEEE Network 8(5) (September/October 1994), p. 22-32.

[Schi94a] Schilit B., Adams N., Want R.: Context-Aware Computing Applications. Proceedings of the 1st International Workshop on Mobile Computing Systems and Applications (1994), p. 85-90.

[Stei99] Steinfield C., Jang C.Y., Pfaff B.: Supporting Virtual Team Collaboration: The TeamSCOPE System. Proceedings of the Group'99 Conference (Phoenix, USA, 1999), ACM Press, p. 81-90.

[Stie97] Stiermerling O., K. H., Wulf V.: How to make software softer - designing tailorable applications. Symposium on Designing Interactive Systems (1997), p. 365–376.

[Tang01] Tang J.C., Yankelovich N., Begole J., Van Kleek M., Li F., Bhalodia J.: ConNexus to awarenex: extending awareness to mobile users. Proceedings of the SIGCHI conference on Human factors in computing systems (Seattle, USA, 2001), p. 221-229.

[Tang03] Tang J.C., Begole J.: Beyond Instant Messaging. ACM Queue Vol. 1 Number 8 (2003), p. 28-37.

[Want92] Want R., Hopper A., Falcao V., Gibbons J.: The Active Badge location system. ACM Transactions on Information Systems 10(1) (January 1992), p. 91-102.

[Yaho04] Website of Yahoo instant messenger: http://messenger.yahoo.com (19.08.2004)

10 List of Tables and Figures

Curriculum Vitae

Personal Data

Name:	Stefan Oppl
Address:	Reithoffergasse 2a/5 A-4400 Steyr
Date of Birth:	10th August 1980
Martial status:	Single
Nationality:	Austrian
Parents:	Walter Oppl, politician Ursula Oppl, teacher

Education

1986 - 1990	Elementary School: Volksschule 2 Ennsleite, Steyr
1990 - 1994	Secondary School: Bundesgymnasium Werndlpark, Steyr
1994 - 1999	Technical High School: Department of Electronics – Technical Computer Science, Steyr
2000 - 2004	Studies in Computer Science: Johannes Kepler University, Linz

Occupation

July 1996	Practical at “BMD Computer Systemhaus”, Steyr
July 1997	Practical at “Ennskraftwerke AG”, Steyr
July 1998 – August 1998	Practical at “SDP Engineering Centre – Department of Acoustics”, Steyr
October 2003 - January 2004	Tutor for the lecture “Telekooperation”, University of Linz
Since March 2003	Research associate at the “Institut für Pervasive Computing”, University of Linz

Military Service

September 1999 - April 2000	Medical Service, Salzburg and Kirchdorf

Publications

Ferscha A., Holzmann C., Oppl S.: *“Team Awareness for Personalized Learning Environments”*, The Proceedings of the 3rd International Conference on Mobile Learning (MLEARN 2004) as a Full Paper. Rome, Italy, 5.-6. July 2004.

Ferscha A., Holzmann C., Oppl S.: *“Context Awareness for Group Interaction Support”*, The Proceedings of the 2nd ACM International Workshop on Mobility Management and Wireless Access (MobiWac 2004) as a Regular Paper. Philadelphia, PA, USA, 26.September-1.October 2004

Technical Reports

Holzmann C., Oppl S. *“Bluetooth in a Nutshell”*, Technical Report (156 pages), Context Framework for Mobile User Applications, Siemens Munich CT-SE 2, December 2003.

Talks

“Bluetooth in a Nutshell”

Workshop about Bluetooth technology, Siemens Munich CT-SE 2, 4. December 2003.

“Team Awareness for Personalized Learning Environments”

3rd International Conference on Mobile Learning (MLEARN 2004), Rome, Italy, 5.-6. July 2004.