Using Handheld Devices in Synchronous Collaborative Scenarios

Jörg Roth and Claus Unger

University of Hagen
Department for Computer Science, 58084 Hagen, Germany
{Joerg.Roth, Claus.Unger}@Fernuni-hagen.de

Abstract. In this paper we present a platform specially designed for groupware applications running on handheld devices. Common groupware platforms request desktop computers as underlying hardware platforms. The fundamental different nature of handheld devices has a great impact on the platform, e.g. resource limitations have to be considered, the network is slow and unstable. Often, personal data are stored on handheld devices, thus mechanisms have to ensure privacy. These considerations lead to the QuickStep platform. Sample applications developed with QuickStep demonstrate the strengths of the QuickStep environment.

1 Introduction

Collaborative applications help a group to, e.g., collaboratively create documents, write agendas or schedule appointments. A common taxonomy [3] classifies collaborative applications by time and space, with 'same place' and 'different places' attributes on the space axis and 'same time' (*synchronous*) and 'different time' (*asynchronous*) ones on the time axis. Synchronous groupware supports real-time group activities, i.e. events are distributed to group members without considerable delay. In contrast, asynchronous activities may happen at different times.

To develop groupware, especially synchronous groupware, is a difficult and time-consuming task. Usually, groupware is not developed 'from-scratch', but with the help of a groupware toolkit. In this paper, we introduce a groupware platform for synchronous collaborative applications with handheld devices. Our approach assumes that group members operate in a close neighbourhood, i.e. long distance connections between users are not supported.

Although several groupware toolkits are available already, they can hardly be adapted to handheld devices. Straight-forward approaches, i.e. simply cross-compiling existing applications, fail because of the specific properties of handheld devices and the connecting network:

- Handheld devices have low computational power, small memory and usually no mass storage devices (e.g. hard disks).
- Handheld operating systems (e.g. PalmOS [1], Windows CE [2], EPOC [11]) do not offer the same variety of services as desktop operating systems. PalmOS, e.g.,

does not support threads or processes for background tasks, a common technique for desktop computer applications.
- Handheld applications follow a different usage paradigm: they are designed for a small display, have to provide short start-up and response times and are developed for gathering and presenting small pieces of information rather than processing large amounts of data.
- Network connections to handhelds have low bandwidths and are considerably unstable.

The notions of *handheld device*, *palmtop*, *PDA* and *organizer* are often interpreted in different ways. One (older) interpretation distinguishes between *pen-based devices* and *palmtops*, where the latter have keyboards. In contrast, Microsoft divides Windows CE devices into *handheld PCs* (H/PC) with a keyboard, *palmsize PCs* (P/PC) which are controlled by a pen and *handheld PC Pro* devices, which are subnotebooks [2]. In the following, we understand by *handheld devices* pen-based devices with small displays (e.g. 160x160 pixels). Popular examples for such devices are 3Com's Palm III and Casio's Cassiopeia.

2 Collaboration in Mobile Environments

Collaborative applications significantly differ from single-user applications. Many users provide input (often simultaneously), output has to be processed for many users and shared data have to be kept consistent. Groupware applications have to provide a kind of 'group feeling', called *collaboration awareness*: users have to be aware of other users involved in the collaborative task. Collaboration awareness is provided by elements inside the application called *awareness widgets*.

A similar concept applies to the mobility aspect of handheld devices. Mobile devices can be connected to a network at different places. Depending on the location, different information is available. Users should be aware of their current location, including the geographic location as well as the location in the network, e.g. the actual domain. This kind of awareness is called *context awareness*.

We call an application *aware* of something, if it explicitly takes care of a special situation, otherwise we call it *transparent*. *Collaboration aware* applications are especially designed to support a group, i.e. they contain special code for group functions. *Collaboration transparent* applications are original single-user applications, which, with the help of a group toolkit, can be used by many users simultaneously. Collaboration transparent applications do not offer awareness widgets. A similar notion can be applied to the mobility aspect: *mobility aware* applications contain code to handle mobility, e.g. react on unstable network connections and changing network locations. *Mobility transparent* applications cannot handle such problems explicitly, but rely on an underlying platform.

The QuickStep platform is designed to develop both, collaboration aware and mobility aware applications. It provides awareness widgets for collaboration awareness as well as for context awareness. Before we describe the QuickStep approach, related work is presented.

3 Related Work

Collaborative as well as mobile applications have to keep data consistent. Applications which are collaborative and mobile at the same time double the problem of data consistency. Collaborative applications have to synchronise concurrent data manipulations, mobile applications have to keep data consistent when devices are moved inside the network or are disconnected from the network.

Several toolkits have been developed to address the problem of data distribution in mobile environments. Coda [5] provides a distributed file system similar to NFS, but allows disconnected operations. Applications based on Coda are fully mobility transparent, i.e. run inside a mobile environment without any modification. Disconnected mobile nodes have access to remote files via a cache. Operations on files are logged and automatically applied to the server when the client reconnects. Coda applications can either define themselves mechanisms for detecting and resolving conflicts or ask the user in case of conflicts. A follow-on platform, Odyssey [9], extends data distribution to multimedia data such as video or audio data. To support real-time data, bandwidths and available resources have to be monitored. Odyssey applications are mobility aware.

Rover [4] supports mobility transparent as well as mobility aware applications. To run without modification, network-based applications such as Web browsers and News readers can use network proxies. The development of mobility aware applications is supported by two mechanisms: *relocated dynamic objects* (*RDOs*) and *queued remote procedure calls* (*QRPC*). RDOs contain mobile code and data and can reside on a server as well as on a mobile node. During disconnection, QRPCs are applied to cached RDOs. As in Coda, operations are logged and applied to server data after reconnecting.

Bayou [12] provides data distribution with the help of a number of servers, thus segmented networks can be handled. In contrast to Coda, replicated records are still accessible, even when conflicts have been detected but not resolved. Bayou applications have to provide a conflict detection and resolution mechanism, thus no user intervention is necessary. Bayou is not designed to support real-time applications.

Sync [7] allows asynchronous collaboration between mobile users. Sync provides a collaboration based on shared objects which can be derived from a Java library. As in Bayou, data conflicts are handled by the application. Sync applications have to provide a *merge matrix*, which contains a resulting operation for each pair of possible conflicting operations. With the help of the merge matrix, conflicts can be resolved automatically.

Lotus Notes [6] has not primarily been designed for mobile computers, but allows replicated data management in heterogeneous networks. Nodes can be disconnected and merge their data after reconnection. Data in Lotus Notes have a record structure. Fields may contain arbitrary data which are transparent to Notes. Records can be read or changed on different nodes simultaneously. When reconnecting, conflicting updates are resolved by users.

With the help of handheld devices, Pebbles [8] allows to remotely control applications running on a server. It follows a collaboration and mobility transparent concept. Instead of using the mouse and keyboard directly, input is taken from the handheld device's touchscreen and handwriting area. From the application's view, input comes directly from the server's keyboard and mouse respectively. In turn, the server win-

dow output is transferred to handheld devices. With these mechanisms it is possible to remotely control off-the-shelve applications (e.g. MS Word) with handheld devices.

Most of the toolkits above request their mobile clients to be notebook computers with, e.g., hard disks. Only Pebbles is designed for handhelds. The focus of the other platforms is to maintain data consistency in a weakly connected environment. Problems related to handheld devices, such as small memory and reduced computational power, are not handled satisfactorily. Automatic conflict detection and resolution need a considerable amount of resources on the handheld devices. We believe that such mechanisms are (currently) not suitable for handheld scenarios.

Concepts, such as the Rover toolkit, which require mobile code and marshalling/unmarshalling mechanisms currently cannot be adapted to handheld devices, since these are significantly different from their servers. The concept of mobile code requires platform independent code and identical runtime libraries on both platforms. Even though languages such as Java are running on many platforms, handheld portings will provide other runtime libraries, thus mobile code mechanisms will fail.

Looking at typical data types stored in handheld devices we mostly find well structured and textual data. Real-time data such as video or audio data provided by the Odyssey system are currently inadequate due to small network bandwidths, low computational power and reduced peripheral equipment of handheld devices. Even graphical data such as freehand sketches or diagrams are difficult to handle because of inaccurate and inconvenient screen devices.

Data stored inside a handheld device are usually viewed as *private*. Even more than desktop computers, such devices are viewed as personal ones [10]. Personal data, e.g. telephone numbers, birthdays and leisure-time activities are stored inside such a device. If a handheld device is connected to an untrusted network, a platform has to offer mechanisms to guarantee privacy of individual data. None of the platforms above contains such mechanisms.

4 The QuickStep Approach

The QuickStep platform supports developers of collaboration and mobility aware handheld applications. They can use communication and collaboration primitives provided by the platform and can concentrate on application-specific details. A set of predefined awareness widgets can be integrated into an application with a few lines of code.

The QuickStep approach can be described as follows:

- QuickStep supports applications with well-structured, record oriented data, as being used by built-in software for handheld devices (e.g. for to-do lists, memos, telephone lists). QuickStep has explicitly not been designed for supporting multimedia data, graphical oriented applications or continuous data streams.
- QuickStep is mainly designed for supporting synchronous collaboration.
- QuickStep provides awareness widgets for collaboration awareness as well as context awareness.
- QuickStep comes along with a generic server application which allows to support arbitrary client applications without modifying or reconfiguring the server.
- The QuickStep architecture ensures privacy of individual data.

Before we describe the QuickStep platform itself, we present a sample application developed with QuickStep.

4.1 A Sample Application

Consider a scenario in which members of a meeting want to schedule appointments for future meetings. Each member owns a handheld device, which already contains a list of appointments as well as entries indicating the time one is unavailable because of, e.g., vacations or travels. Figure 1 presents an application that can help to find a date, when all members are available.

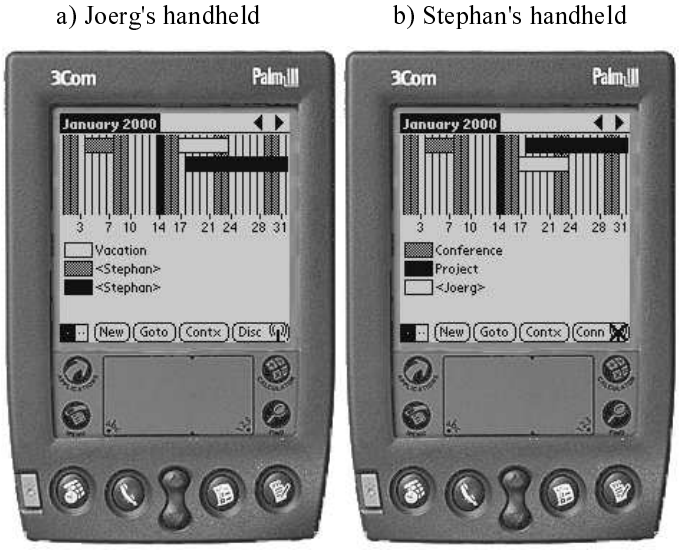

Fig. 1. A collaborative calendar tool

The figure shows two users' views on their personal handheld devices. The upper half of the windows displays the days of a month. Each range of dates when someone is unavailable is indicated by a bar. To get a better overview, the view can be switched to a two-months display. The lower half of the window is the legend for the upper half.

Both users, Joerg and Stephan, can see their own and the foreign bar, the latter being labelled with the user name rather than the local label. With regard to the foreign bar, only the date range is of interest, not the reason why someone is unavailable. Each user can make new entries which are distributed to the other user in real-time. With the help of this application it is very easy to find dates, where all members are available.

To develop such an application 'from-scratch', a developer has to implement many tasks, communication protocols, e.g., have to be integrated, shared data have to be managed. The application should offer awareness widgets. All these services have to

be developed in addition to the main task, the calendar function. This might overwhelm a developer.

QuickStep helps a developer to concentrate on the application-specific details; communication and data primitives as well as predefined awareness widgets can be used from the platform. In the following chapters, we present the platform in more detail.

4.2 The QuickStep Communication Infrastructure

The sample application above requires a communication link between the handhelds. In principle, the devices could be connected directly. Unfortunately, the computational power of handheld devices is currently too low to handle communication in the background. Often, handheld operating systems (e.g. PalmOS) are generally unable to run background tasks, a prerequisite for handling incoming communication requests. Thus, we need an additional computer, which acts as a communication relay between handhelds. This computer, the *QuickStep server*, contains a generic server application which is able to serve arbitrary QuickStep applications.

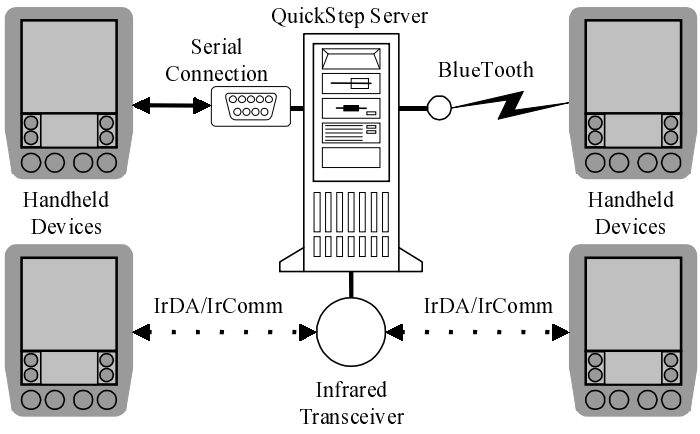

Fig. 2. A QuickStep communication scenario

Figure 2 shows a typical QuickStep communication scenario. In various scenarios the QuickStep approach can be used without setting up an application dependent server. The QuickStep server can be viewed as 'inventory' of a specific environment, e.g. of a meeting room or of 'public' locations like trains or public halls. Once installed, it normally has not to be reconfigured or administered. The server runs without an operator and does not need a user interface, thus can work 'invisibly' behind a panel.

The network connection works either in a wireless way (e.g. via an infrared or radio connection) or via serial cables. Wireless communication protocols are, e.g., BlueTooth and IrDA/IrComm. Typical handheld devices already provide a serial port and a built-in infrared transceiver. In addition, a TCP/IP communication stack is integrated into most handheld operating systems.

4.3 Group Management

Groups of collaborating users are not defined explicitly in QuickStep. All users connected to a specific QuickStep server at the same time and using the same QuickStep application form a collaborative session. This concept allows to run a server without defining groups centrally. It is possible for a user to join a group without having explicit permission from existing users. Since a mechanism for anonymizing data is integrated into the platform, a user cannot spy out private data (see below).

QuickStep is mainly created to support synchronous collaboration. In contrast to desktop computers, handhelds are not permanently switched on. During collaboration, the handheld may be switched off because of the auto-power-off mechanism. In addition, handheld network connections tend to be unstable, thus unwanted disconnections are possible. Following the strict definition of synchronous collaboration, anytime a member is disconnected, she or he would automatically leave a running session. To overcome this problem, we introduce the notion of *relaxed synchronous collaboration* for group members who collaborate synchronously, but may infrequently be disconnected from the network for short periods of time. Relaxed synchronous collaboration is placed between (strict) synchronous and asynchronous collaboration. As in asynchronous scenarios, shared data have to be stored during disconnections, but data manipulations are happening much more frequently.

QuickStep does not provide services for leaving a session. When a user disconnects, the server first assumes a temporary disconnection. Only if a user is disconnected for a longer time (e.g. an hour), the server removes that user from the session. The period of time, a user has to be disconnected until a leave operation is performed, is defined by the corresponding application.

4.4 Managing Data

Usually, data inside handheld devices are well-structured and record oriented. Common operating systems for handhelds have built-in services to store and retrieve data records. PalmOS supports an entity called *database* [1] (not to be confused with the classical database). A palm database is a persistent collection of records. Each record has a unique identifier, which allows its identification, but its content is opaque to the operating system. Constructing and interpreting records solely depends on the corresponding application.

The database is a common programming abstraction in handheld applications, thus the ideal abstraction for collaborative applications as well. QuickStep follows the same paradigm when collecting and distributing data. The QuickStep application programming interface (API) has similar database functions as the handheld operating systems. An application developer can use well-known services to handle application specific data. Data stored in QuickStep databases are automatically distributed among a session by the QuickStep platform. Similar to native database services, the actual content of records is not of interest for the distribution mechanism and can only be interpreted by the application. Especially, the QuickStep server does not know the record structure.

When planning data distribution, many contradicting requirements have to be taken into consideration. Many platforms described above have complex mechanisms to detect and resolve conflicts caused by concurrent data manipulations. In our opinion,

such mechanisms cannot be used inside handheld devices. Our concept for solving conflicts is simply to avoid them: it is not possible to concurrently manipulate data. For this, each record of data can only be changed by the handheld device which originally created the record. Copies residing on other handheld devices can only be viewed. To modify data which were created by another user, one has to make a private copy, which is treated as a new record.

4.5 Mirroring and Caching

The computation power of handhelds and network bandwidths are considerably low compared with desktop environments. The transfer of processing tasks to a server would relieve handheld devices of heavy computation. On the other hand, it is not possible to transfer large sets of data between handhelds and a server. To reduce network traffic and to perform as many computations as possible on a server, we developed a combined mirroring and caching mechanism. Figure 3 shows the architecture.

The main entities are the following:

- Each handheld has its own *local database* which contains the application's records. Only the owner can add, change or remove local records.
- The QuickStep server has a copy of each local database, the *mirror database*. The mirror database is incrementally updated each time a handheld device is connected to the server.
- To allow viewing data during a disconnection, a local copy of other users' mirror database entries exists on the handhelds, called the *cache database*. Since the amount of data of all mirror databases might be too big for the handheld, a *selector* set by the application reduces the number of cache entries.
- An application accesses the local database and the cache database via the *database proxy*. The proxy provides a similar interface as a conventional database.

The *anonymizer* and the *lifetime supervisors* are related to privacy mechanisms, which we will describe later.

The selector can, e.g., specify the range of days the calendar tool currently displays or a category label in a memo tool. With the help of the selector, only records are loaded and updated which are currently displayed. This approach results in a dilemma: on one hand the set of records which match a specific selector should be computed by the server, not by the handhelds, on the other hand, the selector may be highly application dependent, i.e. hardly to be handled by a generic server.

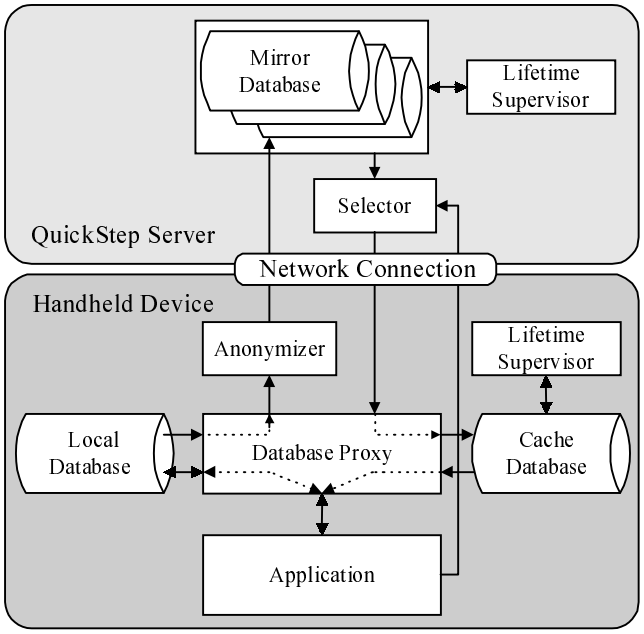

Fig. 3. The QuickStep architecture

To address this problem, we identified two selector types which match most applications: the *number range selector* and the *string match selector*. Both selector types are predefined in the QuickStep server. The number range selector can be used for all kind of records which are ordered by a numeric value. A good example are date entries in the calendar tool presented above. To each entry, a number can be assigned, e.g. the day index. An entry matches a specific number range selector, if the corresponding index is within the selected range. In the calendar application, the current view is defined by a range (*from date...to date*). This range is sent to the server, which in turn only sends records which are within this range. The string match selector is used to match records which have a string value attached. This value has not to be unique inside the database. The string match selector can be used to select all memos of a specific category.

To maintain consistency of the mirror database, a logging mechanism is included into the platform. Every local record in the database has a unique identifier which can be used to identify records in the server's mirror database. When the application removes, adds or changes a record, the corresponding change is logged locally. Whenever the handheld is reconnected to the QuickStep server, the changes are transferred to the server, which updates the corresponding mirror database. Applying changes incrementally ensures that the amount of transferred data is low, even when the handheld database has many entries.

4.6 Private Data

For users, privacy is an important requirement. We decided not to transfer any private data across the network. For this, every record can be marked as private (the default value). Private records reside only in the handheld and will not be transferred under any circumstances.

Non-private records are not transferred until an anonymizing process relieves them from personal fields. Since the record structure is opaque to the underlying system, the anonymizing function has to be provided by the application. In the calendar application, e.g., the anonymizing function blanks out the labels of appointments and transfers the dates only.

As an additional concept, each record has a 'time to live' entry, after which a record is deleted automatically from the QuickStep server and other handheld devices. This approach guarantees a user that her or his data are not available for ever on other computers (even in anonymized form). The time to live entry can be either *session*, *min*, *hour*, *day*, or *forever*. If the value is *session*, the corresponding record will be immediately removed from the server and handheld caches when the corresponding handheld is disconnected. The other values indicate the time, a record will reside after disconnection. The lifetime is controlled by lifetime supervisors (see Figure 3) which exist on the handheld devices and on the QuickStep server.

4.7 Context and Group Awareness

A user who collaborates with the help of the QuickStep platform wants to know about the context she or he is currently working in. For this, a QuickStep application can integrate a 'context' button with opens a frame.

The context frame is the central instance for all context-related information:

- What is the current connection state (connected or disconnected)?
- To which server is the handheld currently connected (server name, organisation)?
- Where is the server located?
- Who can be called in case of problems (e.g. network failures)?
- Which users currently form the session and what are their connection states?

Except for the user list, the context information is fixed and has to be configured once when setting up a QuickStep server. The user list is automatically computed and constantly updated.

The context information is important when a user enters an unknown location. Consider a scenario where a huge building is equipped with a number of QuickStep servers (e.g. one per floor). Each QuickStep server provides information about the current location and thus can be used as a beacon for navigating inside the building.

For collaborating users, the connection state is very important. If a user is disconnected, all changes applied to data cannot be viewed by other users. Thus, information about the connection state should be available on the main window of an application. We designed an integrated button and state indicator (see Figure 1, lower right button). This widget allows to connect and disconnect to a QuickStep server and indicates the current state with the help of a small icon.

States can be:

- Disconnected: the button allows to reconnect.
- Connected: the button allows to disconnect.
- All members of a session are connected: this means that all data which can be viewed are up to date. As in the connection state, the button allows to disconnect.
- Error: the QuickStep platform is in an unexpected state, e.g., because of a corrupted local database. Neither connecting nor disconnecting is allowed.

The button/state indicator as well as the context frame are predefined awareness widgets and can be integrated in an application with the help of the QuickStep library. In addition, an application can retrieve state and context information via the QuickStep API and can react on events (e.g. joining a session or disconnecting from a network), thus helping an application developer to create his or her own awareness widgets.

4.8 Realisation Aspects

When realising QuickStep, we decided to use the programming language Java. The QuickStep platform consists of two parts, the QuickStep server application and the handheld platform. We developed the server application with Sun's Java Development Kit for desktop computers.

For handheld devices, especially for PalmOS, three Java platforms for handhelds are currently available: KVM (http://java.sun.com/products/kvm/), Spotless (http://www.sun.com/research/spotless/) and Waba (http://www.wabasoft.com/). KVM has currently no network APIs and Spotless tends to be unstable. We decided to use Waba. Waba provides all services we need, runs stably, and offers a virtual machine and a runtime library. To compile Waba applications, a common Java compiler from any desktop Java Development Kit can be used. Waba has the great advantage of supporting both, PalmOS devices as well as Windows CE devices.

Although Waba covers a wide area of classes and methods, two kinds of services are still missing. First, Waba does not support threads. As a work-around, Waba offers so-called *timers* which can periodically call a predefined method. Unfortunately a method call is only performed, when no other instruction is being executed. Waba does not support real background operations.

Another missing service is the server socket. Waba only allows to open a socket connection to another server. A handheld device cannot offer a socket service itself. This affects how communication is established between two parties. The handheld device always has to be the initiating part of a communication. As a consequence, it is not possible to connect two handheld devices without having a server in-between.

One further drawback of Waba is that rather than using native widgets, all user interface widgets are re-implemented because of two reasons: there does not exist a common set of widgets which is available on all supported operating system platforms. Thus such a set has to be provided by Waba. Secondly, PalmOS does not allow the dynamic creation of native dialogue widgets, the usual way dialogues are created in Java. Since all widgets are handled by Waba, they cost a considerable amount of valuable object memory. In addition they react slightly differently compared to the native PalmOS widgets.

An important problem of Waba is the memory usage under PalmOS. The memory allocation mechanism of PalmOS restricts the entire object heap to 64k, even if the handheld device has a total of 2MB of RAM. Since Waba makes heavy usage of dynamic memory, it is currently not possible to develop bigger applications with Waba.

4.9 More Examples

In addition to the calendar, tool we developed several applications to verify and improve the QuickStep platform. We now briefly describe two of them.

The brainstorming tool (Figure 4a) allows to add ideas to a collaborative list. An idea is presented by a short description, usually one line of text. All collected ideas are presented in a scrollable list box.

Usually, data of collaborative applications are dynamic. The business card collector (Figure 4b) has a completely different character. A personal business card is typically stored once and never changed. When one enters a public location, e.g., a conference, the application presents a list of all other users who published their business cards. A user can view these cards and collect interesting cards in a persistent area.

a) The brainstorming tool b) The business card collector

Fig. 4. More QuickStep sample applications

5 Conclusion and Future Work

The QuickStep approach allows to develop mobility and collaboration aware applications and has been especially designed for handheld devices. A generic QuickStep server relieves the handheld devices from heavy tasks and stores data during disconnection. The QuickStep server operates without human intervention and can serve

arbitrary QuickStep applications without modification. A server offers contextual information, which can be used by handheld applications.

Data distribution is handled by a caching and mirroring mechanism. Since data are copied across an untrusted network and stored on other handheld devices, mechanisms to ensure privacy are very important. We strongly believe that a platform can only gain acceptance, if users are convinced that their personal data are kept private.

The QuickStep platform is implemented on a Java for handhelds and fully operable on Palm and Windows CE devices. However, more complex application have memory problems, thus we are currently working on a porting for the PalmOS platform in C.

References

1. Bey C., Freeman E., Mulder D., Ostrem J.: Palm OS SDK Reference, 3Com, http://www.palm.com/devzone/index.html, Jan. 2000
2. Boling D.: Programming Windows CE, Microsoft Press, 1998
3. Ellis C. A., Gibbs S. J., Rein G. L.: Groupware - some issues and experiences, Communications of the ACM, Vol. 34, No. 1, Jan. 1991, 39-58
4. Joseph A. D., Tauber J. A., Kaashoek M. F.: Mobile Computing with the Rover Toolkit, IEEE Transactions on Computers, Vol. 46, No. 3, March 1997 337-352
5. Kistler J. J., Satyanarayana M.: Disconnected Operation in the Coda File System, ACM Transaction on Computer Systems, Vol. 10, No. 1, Feb. 1992, 3-25
6. Lotus Development Corporation: Lotus Notes, http://www.lotus.com/home.nsf/welcome/lotusnotes
7. Munson J. P., Dewan P.: Sync: A Java Framework for Mobile Collaborative Applications, special issue on Executable Content in Java, IEEE Computer, 1997, 59-66
8. Myers B. A., Stiel H., Gargiulo R.: Collaboration Using Multiple PDAs Connected to a PC, Proceedings of the ACM 1998 conference on Computer supported cooperative work, 1998, 285-294
9. Noble B., Satyanarayanan M., Narayanan D., Tilton J. E., Flinn J., Walker K.: Agile Application-Aware Adaptation for Mobility, Proceedings of the 16th ACM Symposium on Operating System Principles, Oct. 1997, St. Malo, France
10. Stabell-Kulø T., Dillema F., Fallmyr T.: The Open-End Argument for Private Computing, Proceeding of the First International Symposium on Handheld and Ubiquitous Computing, Karlsruhe, Germany, Sept. 1999, Springer, 124-136
11. Tasker M., Dixon J., Shackman M., Richardson T., Forrest J.: Professional Symbian Programming: Mobile Solutions on the EPOC Platform, Wrox Press, 2000
12. Terry D. B., Theimer M. M., Petersen K., Demers A. J.: Managing Update Conflict in Bayou, a Weakly Connected Replicated Storage System, Proceedings of the fifteenth ACM symposium on Operating systems principles, Copper Mountain, CO USA, Dec. 3-6, 1995, 172-182