Brno University of Technology

Faculty of Information Technology

Master Thesis

Budiik – Personal Information Management System

15th May 2004

Martin Marek

Assignment

1. Study personal data management theory, methods, techniques and available tools. Concentrate especially to time management issue.
2. On the basis of the tools analysis, design a draft of own PIM System.
3. Discuss possibilities of system Graphic User Interface and implement the selected items.
4. Make a detailed design of the system. Focus on possibilities of the system in multi-user environment.
5. Implement the system and check its functionality in real environment.
6. Strike a balance and discuss further possible system development.
7. Consult design matters with professor Jouni Lampinen and Hanna-Kaisa Lammi, both from Department of Information Technology, Lappeenranta University of Technology.

Abstract

Nowadays, there are many existing both hardware and software facilities for personal information management in Home & Office area, including the well-known classic electronic diaries and organizers, various software systems and also modern mobile devices. Users from this area are not infrequently computer beginners and laymen, who require simple, usable and inexpensive facilities. The objective of this thesis is to offer such a suitable solution for this target group of users.

The thesis investigates various aspects of computer-aided personal information management aimed at Home & Office area, concerning mainly time-oriented and user-oriented information, user interface and possibilities of data sharing and synchronization. It also analyses various facilities for personal information management and reports on some existing software tools. On the basis of acquired knowledge, it attempts to create a novel system for personal information management.

The results of the analysis include the basic requirements and design principles of functional and user-friendly systems for personal information management. Finally, the thesis proposes a design of Budiik, a new PIM software system intended for single users and small groups from Home & Office area. Its partial prototype implementation presents the main principles and ideas of the future system.

Keywords

Time Management, Personal Information Management, PIM, Graphic User Interface, Groupware, Personal Information System.

Declaration

I declare that I disposed this Master Thesis independently under leadership of MSc. Bohuslav Křena and Prof. Jouni Lampinen and I named all sources and publications that I had drawn information from.

Lappeenranta, 15th May 2004 .....

Thanks

I would like to thank to Mr Bohuslav Křena for the individual assignment offer and to Mr Jouni Lampinen for useful advice and comments during the project.

Abstrakt

V dnešní době existuje poměrně široká nabídka technických i programových prostředků pro správu osobních dat určených pro kancelář a domácnost. Kromě dobře známých klasických elektronických diářů a organizérů jsou to nejrůznější programové systémy a též moderní mobilní zařízení. Uživatelé z tohoto prostředí jsou nezřídka laici a počítačoví začátečníci, kteří vyžadují jednoduché, lehce ovladatelné a levné nástroje. Cílem této diplomové práce je nabídnout této skupině uživatelů přijatelné řešení.

Tato diplomová práce zkoumá nejdůležitější aspekty počítačem podporované správy osobních informací v prostředí malé kanceláře a domácnosti, týkající se zejména vztahu informací vzhledem k času a uživateli, uživatelského rozhraní a možností sdílení a synchronizace dat. Dále analyzuje nejrůznější prostředky pro správu osobních informací a popisuje některé existující programové nástroje. Na základě získaných znalostí se pokouší vytvořit vlastní nový systém pro správu osobních dat.

Výsledky analýzy zahrnují základní požadavky a principy návrhu funkčních a uživatelsky přívětivých systémů pro správu osobních dat. Tato práce dále představuje návrh nového programového systému pro správu osobních dat nazvaného *Budiik*, který je určen pro samostatné uživatele a uživatelské skupiny v domácím a kancelářském prostředí. Jeho částečná prototypová implementace prezentuje hlavní myšlenky a principy budoucího systému.

Klíčová slova

Time management, Personal Information Management, PIM, správa osobních informací, grafické uživatelské rozhraní, groupware, osobní informační systém.

Table of contents

Symbols and abbreviations

C/S	**C**lient/**S**erver
CLX	Borland **C**omponent **L**ibrary for **Cross**-Platform
CRM	**C**ustomer **R**elationship **M**anagement
CSCW	**C**omputer-**S**upported **C**ooperative **W**ork
CSV	**C**omma **S**eparated **V**alues
GMT	**G**reenwich **M**ean **T**ime
GUI	**G**raphic **U**ser **I**nterface
HCI	**H**uman **C**omputer **I**nterface
HTML	**H**yper**T**ext **M**arkup **L**anguage
IMAP	**I**nternet **M**essage **A**ccess **P**rotocol
LAN	**L**ocal **A**rea **N**etwork
LDAP	**L**ightweight **D**irectory **A**ccess **P**rotocol
MAPI	**M**essaging **A**pplication **P**rogramming **I**nterface
MDA	**M**obile **D**igital **A**ssistant
MDI	**M**ultiple **D**ocument **I**nterface
MFC	**M**icrosoft **F**oundation **C**lasses
MIME	**M**ultipurpose **I**nternet **M**ail **E**xtensions
NTP	**N**etwork **T**ime **P**rotocol
OLE	**O**bject **L**inking and **E**mbedding
OS	**O**perating **S**ystem
P2P	**P**eer **to** **P**eer
PDA	**P**ersonal **D**igital **A**ssistant
PIM	**P**ersonal **I**nformation **M**anagement
POP3	**P**ost **O**ffice **P**rotocol (v.3)
RTC	**R**eal-**T**ime **C**lock
RTF	**R**ich **T**ext **F**ormat
SDI	**S**ingle **D**ocument **I**nterface
SNTP	**S**imple **N**etwork **T**ime **P**rotocol
SQL	**S**tructured **Q**uery **L**anguage
TCP/IP	**T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol
USB	**U**niversal **S**erial **B**us
VCL	Borland **V**isual **C**omponent **L**ibrary
WAP	**W**ireless **A**pplication **P**rotocol
WIFI	**Wi**reless **Fi**delity
XML	E**x**tensible **M**arkup **L**anguage

Chapter 1

Introduction

This project has a quite long history. At high school I was interested in personal data management and I created **Budík**, a simple PIM system for Windows. The project was surprisingly successful. It was awarded a prize in the Czech national high schools competition (SOČ) in June 1999 and presented at the Czech fairs Invex '99 and Mladex '99. After it, I published the program on the Internet, so I gave it to the community.

Several improvements and patches came after that, but due to a lack of time I was not able to continue in any further development. Therefore I willingly accepted the offer from Mr Křena from Faculty of Information Technology at Brno University of Technology (DITS FIT BUT) to initialise the development of a new program generation within the School-year Project (its client part, particularly). This project dealt with the area of personal information management, available tools and facilities. Later on I proceeded with the project within the Term Project and here I was mainly discussing possibilities and aspects of graphic user interface of personal data management systems. And finally, I decided to continue with this project within my Master Thesis.

1.1 Objectives and Motivation

At the beginning I have to say, why this area is so interesting for me, and what my motivation is.

Nowadays, modern computer technologies spread into all branches of human activity in order to automate and accelerate human work and increase its efficiency in general. Earlier, these facilities were exploited primarily by experts in the IT segment, but with the gradual dissemination especially in the area of consumer electronics, even laymen and common unschooled users such as office staff, pensioners and children familiarize themselves with these tools and equipment. These users do not want to use complex and complicated systems with high power, but low user neighbourliness. Especially for the elders without any previous experience with computers the transition to these systems can be rather painful. That is why there is a strong emphasis on ergonomics, simple control and user friendliness of newly developed equipment and tools. People are made to use these facilities more intensively and more often, claims to enforcement and results grow, but the primary idea still remains – minimal time loading.

Well, there is an endeavour to use these facilities for time saving, but on the other hand, it is necessary to take into account the time, which people need to familiarize themselves with these facilities. Many questions arise in connection with these questions: Is it possible to find new ways of computer-aided time saving and possibilities of handling the human-related data and

information? How to improve existing methods, techniques and tools? And where are the limits of human-computer (and also human-human) interaction and communication?

This thesis deals with all these mentioned facts, aspects and questions. It attempts to explore, analyse and describe the area of personal information management and existing facilities and tools.

The main objective of the thesis is more practical: to design and implement an own system for personal information management intended for Small Office & Home Office area, with regard to requirements and needs of common users. Of course, it is impossible to make a complete and commercially usable system in a short time. So, this thesis contains just a functional prototype of a future system in order to present main principles and ideas of usable and functional PIM systems.

1.2 Contents of the Thesis

In fact, the first chapters of the thesis summarise the results and both theoretical and practical knowledge acquired from prior project work ([Mar03]) and they form it into a comprehensive and homogeneous whole. The second chapter deals with theory of Personal Information Management, Chapter 3 describes particular available systems and tools and Chapter 4 concerns with graphic user interface of these facilities, especially with user interface design.

Chapter 5 discusses possibilities and aspects of data sharing, synchronisation and teamwork within multi-user environment, which is common mainly in business area, but it is also becoming popular for single users and small user groups.

Next chapters deal with my own-designed PIM system called *Budiik*. Chapter 6 describes the conception and basic ideas of the system, Chapter 7 contains a more detailed description of the client application (*Budiik Client*) and Chapter 8 shows possibilities of teamwork, data sharing and synchronization within user groups (*Budiik Server*). Beside the implemented parts, these two chapters also deal with all designed and planned features of particular applications.

Furthermore, the enclosed CD-ROM contains a prototype implementation of Budiik System consisted of executable applications, complete source codes and technical documentation. Contents of the CD-ROM are quoted in Appendix B.

Chapter 2

Personal Information Management Fundamentals

2.1 Time Management

2.1.1 Principles

Let us try to draw up a keynote of time management and its time methods, because they form the backbone of personal information systems we are going to deal with. We will see that time and personal data are linked and we have to take them into account together. The ideas shown here are assumed from [Hay01] and even though they do not provide any formal definition of time management, they take up the fundamentals of this dynamical and elaborative branch. So, let us mention six basic ideas of time management:

Time is a one-shot resource. We have to keep in mind that no instant of time can reiterate – of course if we put aside all physical theories about the spatio-temporal curvature.

Constant quantity of time is pertaining to everybody. This hypothesis is important especially for distribution of labour. Executive staff should keep in mind when performing tasks delegation that every labourer has a certain time (often accurately allotted) for working on certain activities.

Time cannot be stored. It follows from the thought about time as a one-shot source. *Panta rhei*, every instant of time is thus unique, it is impossible to return to it.

Time cannot be switched on / off. Time is absolutely independent of any human activity, it cannot be stopped, and it flows at constant speed – at least in the conditions of the Earth.

Time cannot be substituted. Here, the physical proportion of time is meant. People are totally free of any manipulation with the time. Nevertheless, we can look at work time and free time from another point of view. *"Do not put off till tomorrow what you can do today"*, the classic saying goes.

Time must be consumed by speed of 60 seconds per minute. The point here is utilization of time, because we can grasp it as an untouchable resource that is being consumed at constant speed.

2.1.2 Methods

From the above-mentioned ideas we can extract contents of the time-management methods ([Cau00]). All these methods deal with time and activity analysis and scheduling of these activities. Basically we can talk about time analysis and activities division, in accordance with influences and priorities by means of some of the widely used methods. Nevertheless it is always necessary to take the human factor into account.

Mathematic ergonometry can be regarded as a single line. It is concerned in the application of time management in business environment.

2.1.3 Time representation

The basic requirement is doubtless the possibility to manage time on modern devices and tools, especially time setting, measurement and representation. This is the cornerstone for personal data management in the time area.

In order to manage time using facilities of electronics and computer technology, a time representation had to be chosen. Nowadays, time representation in compliance with **ISO 8601 standard** [I8601] is in use (at least in the western world). It originates from the tradition of the Christian calendar, that uses *seconds* as basic time units followed by derived units like minutes, hours, days, months and years. The standard defines formats of writing date and time and further more it determines, for example, weekday-order or weeks numbering. In the Czech republic, the equivalent to the standard is ČSN EN ISO 28601. Naturally, it is necessary to respect local customs and traditions, as an example we can mention Easter. Similar *"specialities"* can be found in almost all commonly used solar as well as lunar calendars (see [Tøn03]).

2.1.4 Time and computers

And what does the physical (hardware) date and time representation look like?

In electronic devices, there are generally used specialized circuits called **RTC** (Real-Time Clock). Date and time items are stored separately, e.g. in registers. This way is quite logical, so it is often used in software. Nevertheless some programming languages define an own format for physical time storage, for instance a floating-point number (`TDateTime` in Object Pascal), where the integer part of the number means the number of days from accurately defined date (e.g. 1899-12-31, so it can be negative, too) and the decimal fraction means a fragment of the actual day (value 0.5 means noon here). Time represented this way allows easier storage and handling in the program, however, the representation for the user is more exacting.

2.2 Personal Data

2.2.1 Basic definition

Although the *"personal data"* term (or also *"human-related information"*) is not defined formally, in general it covers all the information and data associated with human activity and human time,

working or personal ([Gru92]). There are many divisions and specifications and they always depend on a concrete environment or situation. We can discuss these items one by one.

2.2.2 Kinds of personal data

Human activity can be divided into particular **activities** and they can be more closely specified according to their type, start time, duration, importance, context and so on. We can classify all human activities as *time events* (actions, meetings, anniversaries) and *tasks*.

Although *written correspondence* prevails, recently it has been partially, step by step, replaced by *electronical correspondence*. In both cases, we can term all the **correspondence** relevant to a certain person (sender or recipient) their personal data.

Especially in employment, one comes frequently into contact with many other people. It means s/he is forced to keep at least basic **contact information** about them – such as name, address, phone, email and other relevant data. This information is usually stored in form of *card index* or *contact database*.

Notes mostly represent a temporary storage of some more or less important information, e.g. writing down a certain idea, a meeting, task reminding or a shopping list.

Unlike notes, **documents** are often more voluminous and formal, their informative value is thus usually higher. Various technical reports, forms, contracts, specifications, directives and papers are good examples.

It is no doubt useful to store up some personal data, because we never know, when they can be useful again – it can be an important letter, a contract for a realty bargain, a business card or a cheque. According to [DCC+03], up to 80% of seen and processed information is re-used. That is why bookmarks, catalogues and data archives are used. An **archive** for previously used personal data can look like a cabinet with files or racks for particular archival document types.

2.2.3 Data representation

In section 2.1.3 we have discussed the time representation, which can be processed by a mobile phone or by a computer. But a time representation comprehensible to the user is more important.

Formal representation of date and time values mostly derives from a standard (see [I8601]), nevertheless, it can and sometimes it certainly does clash with local customs. For instance, the introduced norm determines date in the format YYYY-MM-DD, whereas in the Czech republic the German format DD.MM.YYYY is traditionally used. Furthermore, time can be represented by two distinct manners: in 24-hourly or in 12-hourly (AM/PM) cycle.

Naturally, graphic representation of time is usually more comfortable for the user, because a simple time chart can replace several lines of a huge text. There are many ways of time illustration, such as graphic symbols, timeline and time schedule (day, week, month, year), activity diagrams, network charts, Gantt's charts or PERT diagrams.

Beside time-oriented information, we have to represent somehow all the other kinds of user data. Usually, text-based views like *lists* and *tables* are sufficient for both time data and common

personal data. For example to-do lists, contact lists, structured notes or timetables can be presented this way.

For larger data volumes especially in multi-user environment, usage of *database systems* is suitable. Today's relational and post-relational database systems can store all kinds of personal data including pictures, sounds and videos. Data representation itself depends on the client application such as a database client or a web browser.

2.3 Personal Information Systems

2.3.1 Requirements

Basic requirements for personal information systems and generally for personal data management systems (PIMs) are mainly the possibility of interactive inserting, management, archiving, analysis and presentation of personal data related to users and user groups. Those are the data types that have already been discussed: time activities and tasks, correspondence, contacts, notes, documents and all further information directly or indirectly associated with a given person.

2.3.2 Paper versus electronic form

In principle, there are three basic possibilities of using personal information systems: utilization of classic *"paper"* systems, electronic systems or their combinations ([BBN03]). The classic way is primarily utilized by seniors, conservative people and people without wider access to modern technologies. However, more and more people are proceeding to program systems nowadays, especially due to massive spread of computer technology and information systems in the business sphere and public administration. These systems are advantageous especially for the staff working on computer for a considerable part of their working hours. Managers, businessmen and generally intensively travelling men can appreciate ever-rising possibilities of the mobile devices. The main advantages of modern tools are their performance and efficiency.

The combined approach seems to be the optimal (and is recommend by experts from the area of time management) way indeed – utilization only of those parts that are effective for the given person and situation. It is useless to convert all written or printed documents into electronic form, but on the other hand, e.g. the benefits of electronic mail frequently outweigh possible disadvantages ([Gru92]). Duplication and resultant data non-consistence can be a problem of this approach – especially electronic data can change very frequently.

Chapter 3

Facilities for Personal Information Management

3.1 Classic Facilities

These conventional facilities are sometimes called "*pencil & paper*" systems because we can include all kinds of labels, notebooks, diaries, calendars, planners, card indexes or paper documents here. It is impossible to determine a border, what still is and what is not any more a personal data management system. We would also have to think about an ordinary tinny alarm clock when making the exact selection.

Nowadays, many complex pre-printed personal information systems are available (in the price up to hundred dollars) and they usually contain a calendar, a notebook, all sorts of more or less useful information (phone prefixes, tax and financial terms, travelling info...) and space for visiting cards and other bits and pieces.

3.2 Program Facilities

3.2.1 Classification

The term **program facilities** as used in this thesis covers all programs, utilities, applications and platforms of personal computers, which provide some kind of personal information management – from primitive desktop labels and alarms to complex business solutions.

Following pages contain a classification of different PIM-oriented program facilities and several examples of applications for every category. Although Microsoft Windows operating systems are dominating on in Home & Office area nowadays, I also tried to show some examples from the other platforms. Almost all mentioned apps are freeware or they are available in trial versions on web portals, such as Studna (http://www.studna.cz), Stahuj (http://www.stahuj.cz), Slunečnice (http://www.slunecnice.cz) or Tucows (http://www.tucows.com), with the exception of complex business systems like SAP. Prices of all commercial systems are given in USD. For local Czech programs I used following exchange rate: 1 USD ≅ 26 CZK (February 2004).

3.2.2 Notepads and other utilities

Various substitutes for Notepad, desktop sticky notes and similar utilities are included in this software category that is considered to be a part of Home & Office area. These applications are usually very simple, with a primitive control and limited possibility of data insertion management, but for ordinary notes and comments storing they are fully sufficient.

Title	Platform	Type	Description
AM-Notebook (www.aignes.com)	Windows	Freeware commercial (19$)	Simple notebook with sorting and searching
iOrganize (www.brunoblondeau.com)	MacOS	Shareware (20$)	Writing pad
Knotes (pim.kde.org)	Linux	Open source	Desktop sticky notes (part of KDE)
LightNote (www.lightheadsw.com)	MacOS	Freeware	Simple notebook
StickIt (www.singerscreations.com)	Windows	Freeware	Desktop sticky notes
TurboNote+ (www.turbonote.com)	Windows	Shareware (25$)	Desktop sticky notes with additional features (scheduler, alarm)

Table 1: Overview of examined notepads and sticky notes.

3.2.3 Alarms, reminders and calendars

From the standpoint of simplicity and usability, there is a similar category, which consists of programs and utilities for time handling, representation and also for view, organization and reminding the time-related user data. Users expect from this kind of software that they will get a track of their personal time with minimum effort. These applications thus must be simple, intuitive and user-friendly to be usable for laymen in the Home & Office environment.

Software calendars and **diaries** are intended for time-flow representation, commonly in a graphic form (time layout within the day, week, month and year followed by holidays, anniversaries, moon phases, horoscopes, biorhythms and other relevant information). They only work with user's time-related data, mostly with classic activities, journals, tasks or processes. They usually allow a presentation of these data in a specified time period – e.g. day views, week views, month views or lists with selection and sorting options.

Software alarms and **reminders** are utilities for reminding the user of certain facts, like an activity initiation or completion, a task deadline or an important milestone of a process.

Of course, there are many calendar systems for both time data handling and reminding, nevertheless, this combination is more common in complex PIM systems as stated below.

Complexity and functionality of available calendars and reminders is very miscellaneous, there are many simple utilities for time-setting only as well as complex programs that allow to define custom time views, text formatting, task running or time synchronization.

Title	Platform	Type	Description
Alarm Master (www.brigsoft.com)	Windows	Shareware (10$)	Timer and reminder with the possibility of tasks running
KAlarm (pim.kde.org)	Linux	Open source	Alarm-clock, reminder
MindIT! (my-mindit.com)	Windows	Shareware (20$)	Alarm-clock with to-do lists and time synchronization
My Calendar Lite (www.dabsoftware.com)	MacOS	Shareware (35$)	Calendar with adjustable views
Nezmeškej (www.nezmeskej.cz)	Windows	Shareware (8$)	Widely adjustable alarm-clock, calendar and reminder
Task Plus (Contact Plus Software)	Windows	Freeware	Calendar and reminder with the possibility of creating own calendars
Wcal (www.neosystem.com)	Linux	Open source	Calendar and planner with the possibility of data sharing in work groups

Table 2: Overview of examined software alarm clocks, reminders and calendars.

3.2.4 Email clients and card indexes

Although at a glance, there is big difference between mail clients and contact databases, both of these application types have one similar attribute: they collect information about other persons (or from other persons) than the user. Postal programs hold up messages received from or posted to others, card indexes contain personal data about persons themselves – name, place, occupation, phone, email and other relevant information. Any program connection of these data storage facilities increases their usability and makes the work with both data types more transparent. Thus, email can be connected to all kinds of data, like tasks or events ([BDH+03]).

Stand-alone **email clients** are expected to provide a high degree of functionality, which is the main reason of system-integrated mailers replacement (like Outlook Express in Microsoft Windows). These applications are used because the system-integrated programs do not provide required features. Basic requirements are mail sending and receiving, messages sorting and

configuration of the network connection followed by advanced functions such as text formatting, filters, searching, multi-account management and further possibilities of encoding and encryption.

Stand-alone **card indexes** are not very common. They usually are a part in mail clients. Then, users can keep needed information about mail senders and recipients, look for messages associated with a certain person and so on. Advanced systems also allow handling hierarchies of contacts and mail messages.

Title	Platform	Type	Description
Contact Professional (www.databox.cz)	Windows	Commercial (230$)	Client/server tool for contact management
EMU Webmail (www.emumail.com)	Linux	Commercial (750$)	Corporate email system including web-based interface and IM support
Eudora (www.eudora.com)	Windows OS X	Freeware	Email client with many functions (IMAP4, LDAP, filters, templates...)
IncrediMail XE (www.incredimail.com)	Windows	Freeware	Multimedia-oriented email client (messages with sounds and animations)
KEmail (pim.kde.org)	Linux	Open source	Email client with simple control and many functions (part of KDE)
Mozilla Thunderbird (www.mozilla.org)	Windows Linux OS X	Open source	Advanced multi-platform email and newsgroup client with many functions
The Bat! (www.ritlabs.com)	Windows	Shareware (35$)	Email client with many functions including S/MIME, MAPI, templates, filters, plugins and multi-account support
WPeople (www.webmin.com)	Linux	Open source	Personal card index with web-based interface

Table 3: Overview of examined email clients and card indexes.

3.2.5 PIM systems

Unlike the previous categories, the functionality fundamentals of personal data management systems (**Personal Information Management Systems**, PIM, also called *"personal organizers"*; defined as a product category by Lotus in 1987) are administration, organization and presentation of all kinds of user-related or group-related data. The main requirements are handling of activities, contacts and notes followed by email support, document management and the possibility of synchronization with mobile devices, especially with electronic diaries and pocket PCs.

Title	Platform	Type	Description
Data Info Memory (www.top.cz/vhbsoft)	Windows	Freeware Commercial (20$)	Database-oriented system for documents and information handling
iOrganizer (www.channel8software.com)	MacOS OS X	Shareware (25$)	Tool for handling of events, contacts, notes and internet links
KOrganizer (pim.kde.org)	Linux	Open source	Calendar, to-dos, reminding, work groups, web-based interface
Lotus Organizer (www.lotus.com)	Windows	Commercial (75$)	One of the first classic Personal Information Managers, specific design (like a paper diary)
MS Outlook 2003 (www.microsoft.com)	Windows	Commercial (109$)	Part of MS Office (stand-alone, too), high degree of integration into Windows
Organizér (www.firelukesw.webz.cz)	Windows	Freeware	Simple calendar, card index and notepad, partial email support
Phoenix Student Assistant (www.danteproductions.com)	Windows	Shareware (20$)	Personal organizer destined especially for students (tasks, plans, notes...)
Plan (www.bitrot.de)	Unix Linux	Freeware	Calendar, planner, to-dos, notes; data sharing and synchronization
RedBox Organizer (www.inklineglobal.net)	Windows	Shareware (40$)	Time management, expenses, contacts, reminders, report generator and link manager
Time & Chaos (www.isbister.com)	Windows	Shareware (45$)	Calendar, to-dos and contacts with “one screen” interface; user groups, wide possibilities of printing
WinOrganizer (www.tgslabs.com)	Windows	Shareware (40$)	Calendar, to-dos, contacts and notes; specific look, intuitive control
WinPIM (www.winpim.com)	Windows	Shareware (40$)	Contacts, appointments, tasks, diaries and notes with nice look and a possibility of user groups within LANs
Ximian Evolution (www.ximian.com)	Linux	Open source	Outlook-like system for Linux; strong support of work groups in network

Table 4: Summary of the examined PIM systems.

There is the basic idea of PIMs: the integration of various profoundly different, but still closely related components (calendar, to-do list, email client, card index, notebook and file manager) into a one application. So what are the pros and cons of the concept "*everything under the same roof*"?

Unlike stand-alone specialized applications, **integrated PIM systems** cannot focus just on a certain part of the application or data type, because we assume users will use all the components of the system. So we can become aware of some generalization of the user-data view and information. In the active part of the application's window there are shown selected views, the remaining area of the window (toolbars, menus) is the same for all parts of the application. That is why some more advanced PIM systems allow to view the data in the same unified way: each record or item is handled in the same way, just the presentation and interpretation of the contents in the application depends on the item type. This can be related to the idea of data sharing: all the data are accessible to users and applications in the same way and each item type can be converted to any other type. That means that if we for example copy a contact from the card index to the clipboard and paste in the calendar, a new meeting related to given person will be created. We also can send him/her a message, attach a task, share some part of our public data or browse their shared public data and so on.

Another advantage of PIM systems is the wide possibility of data sharing within or outside the work groups as well as import and export of all kinds of personal data. Nowadays, there are many multi-platform commercial systems (e.g. MS Outlook for Windows and MS Pocket Outlook for WindowsCE), so the possibility of data transmission, sharing and synchronization is very desirable.

3.2.6 Complex and distributed systems

For completeness' sake, it is necessary to mention one more category of information systems, which is quite different. It is the category of complex enterprise solutions *"made-to-measure"* by major system integrators, e.g. IBM, Logica, Lotus or SAP. **Enterprise solution** means the summation of all technical, software, personal, knowledge-based and other resources, which provide the complete information background of the company. Under software resources related to personal data management systems come – among others – information systems for handling the information about the company staff, eventually about company surroundings. The major systems' requirements are the possibilities of group scheduling and a complex solution of the email system.

3.3 Mobile Devices

3.3.1 Magic of the mobility

Usage of software facilities becomes useless when people leave their workroom or office. Nevertheless in time of PC's unavailability these people want to keep a track of their time, activities, friends and so on. That is why various mobile devices are getting to the forefront of public interest. They allow handling user-related data and information anywhere and any time – battery life and signal availability are the only limitations.

But "*quid pro quo*" still governs. These facilities have a very specific (an often less intuitive) control, their displays are small and not very fit for graphics or video and also the other technical and efficiency parameters are not extremely astonishing in comparison with the PC platform. It is logical because in fact, these facilities mostly come under the consumer electronics and the price for the customer is critical here. The main requirements are standby time, mechanical ruggedness and possibility of connections with a PC or another device. Nowadays, wireless communication technologies like WiFi or BlueTooth with linkage to mobile devices are very popular.

3.3.2 Electronic diaries and data banks

Electronic **diaries** and **organizers** might be the oldest substitutes for the classic approach to personal data management. They usually look like more complicated scientific calculators but they have alphabetic keyboard and beside calculators they also contain a calendar, to-do lists, a card index or clock alarms. Electronic **data banks** allow easier data handling and they have larger memory for these data, as well. Primitive data banks can be a part of another device, e.g. a wristwatch. For example, Casio is one of the traditional manufactures of these devices.

3.3.3 Pocket computers

This area of computer technology is extremely varied, and also the devices themselves are called in many ways, e.g. handhelds, palm computers, palmtops or palmPCs. Generally, they are similar to electronic diaries, but with higher performance, more user-friendly control (e.g. touch screens and character recognition) and with wide possibilities of data synchronization with another devices, especially with PCs. The user environment usually consists of the operating system and assorted applications designed for the specified platform, including personal data management tools. That is why today's pocket computers are replacing standard diaries and data banks.

At present, we can get in touch with two basic groups of these devices: classical pocket computers called **PDA** (Personal Digital Assistant), which can communicate with the environment via physical connection (serial port, USB) or using wireless transmission (infrared port, Blue Tooth), and new-generation pocket computers called **MDA** (Mobile Digital Assistant, XDA), which represent a powerful combination of classic PDA device and an integrated module for both voice and data mobile communication. The best-known manufactures of pocket computers are Psion, Palm, Sony, Compaq and Handspring and the most widely used operating systems are PalmOS, WindowsCE, EPOC and Symbian (an EPOC version for mobile phones). Siemens and Texas Instruments are the traditional producers of mobile processors and toolkits.

3.3.4 Mobile phones

Usable tools for personal data management within the mobile phones have appeared with the WAP and Java2 Micro Edition technologies coming. This Java clone offers the MIDP API for *midlet application* development for mobile phones. The topic of the midlet design (e.g. creating a simple calendar or notepad) is discussed more closely in [Mah02].

Chapter 4

User Interface of Personal Information Management Systems

4.1 Human-Computer Interaction

In the most recent ten years, computers have become a very ordinary part of human life. First computers appeared already in 1950s, but it took a quite long time until they became acceptable for the public. The reason is clear – former computers had no screen, no mouse, no keyboard, neither windows nor icons. In other words, they had no user interface. Only computer specialists and engineers were able to work with these machines, to control and maintain them. So how is it possible that computers have become so widely spread nowadays? The answer is very simple – thanks to user-friendly interface ([Lau90]).

4.1.1 User-centred software development

As we have already mentioned, a good software program was one that worked and produced some results in early days of computing. Computer programmers were designing programs for use by other computer professionals without regard to easy and comfortable control. This approach is called *problem-centred* software development. Therefore, when non-programmers and laymen started using computers, printed reports and complicated control panels were not enough, so programmers had to adjust this software for normal users. Program-centred software became *user-centred*. Unlike program-centred programming, which focuses mainly on the task, user-centred designers begin by observing how users confront various methods and techniques, how they behave when solving tasks, what are the goals of their tasks and which mental models they create and use. On the basis of these findings, a suitable interface is created. Nevertheless, the final software cannot be released until the user testing and refinement is done properly.

The following definitions and principles are fundamentals to the architecture, design and implementation of effective user interfaces with emphasis on area of Personal Information Management Systems, which take up a very important part of today's commonly used software.

4.1.2 User Interface

User Interface (UI) is the system, which helps the user communicate with the computer system and/or the application system. From the point of view of the whole system, if functionality means what a computer actually does, then the interface is the way the user interacts with the computer and perceives how the computer does its work.

There are various more or less abstract explanations; for instance in [Lau90], Laurel defines the interface as a "*contact surface that reflects the physical properties of the interactors, the functions to be performed, and the balance of power and control*" (p. xiii).

The user interface consists of two parts: a presentation language (*Look*), which is the human-to-computer part of the transaction, and an action language (*Feel*), that characterizes the human-to-computer portion.

4.1.3 Communication channels

Every interaction between user and computer is realized through one or more communication channels. Human communication channels are senses, mainly sight, hearing and touch. Computers use for communication their *peripheral devices*, which provide user input and output.

When talking about computers in Home & Office area, the most classic input device here is surely **keyboard**, which produces character codes of pressed keys. Nowadays, all the computer keyboards are up to standard VT-100, thus they provide writing letters, numbers and special chars and also cursor positioning. This is important, because office users usually use touch-typing; so the use of cursor keys is much faster than using the mouse.

Mouse is another widely used input device. In office systems, mouse is used mainly for cursor/caret positioning, selecting, focus changing, items dragging and another possibilities of direct manipulation. On laptops, mouse is usually replaced by a touch pad or track point. However, every system should offer both keyboard and mouse control.

The primary computer output device is **display**. Colour CRT monitors and LCD displays are widely used nowadays. **Sound output** can be helpful (e.g. warning signals) as well, but it must not disturb the user. Unlike former times, when it was the printer that provided the primary computer output, nowadays, **printer** is only used for more permanent storage of screen output (reports, documents, images).

Nowadays, various **mobile devices** are getting to the forefront of public interest. They allow handling user data and information anywhere and any time, but it costs something – these devices must be small and light enough to be really "mobile". That is why manufacturers have to adjust interface of these devices. Their displays must be small, but clear and contrast to be able to show all the information needed. Keyboards are often simplified (e.g. combined keys for numbers and letters in mobile phones) or replaced by another control facility (touch screen, mini-joystick, roller). For instance, PDA devices often enable drawing text and images directly on the touch screen as user input.

Generally, peripheral devices are only the *hardware part* of the computer's interface. They provide user interaction with the *software part* of computer's interface (command-line, menu,

desktop environment, application). The output of this software part is usually shown on the computer's display. Software developers utilize peripherals (hardware part) for input and output of their program systems (software part). In the following text, we are going to discuss the software part of user interfaces. These interfaces utilize for communication with the user all above-mentioned facilities and devices.

4.1.4 Types of User Interface

With the time passing, various user systems and interfaces have been developed. We can categorize them according to the type of interaction between the system and the user as follows:

Natural-language interfaces permit users to interact with the computer in their everyday-spoken or "natural" language. Unfortunately, any natural language is very ambiguous and difficult to recognize and interpret. In addition, speech synthesis is complicated. That is why there are hardly any existing natural-language interfaces.

Question-and-answer interfaces are based on the very simple principle that the computer displays a question for the user on the screen, the user enters the answer and the computer reacts to this input information in a defined way. This type of interface is most comfortable mainly for new users; therefore it is used for software guides (*Wizards*).

Menu interfaces are represented by an onscreen list of available selections. They can also contain another nested menu (*submenu*). The hierarchy of menus allows the user to move quickly through the program and eliminates menu items, which are of no interest for the user.

Form-fill interfaces are represented by onscreen *forms* consisted of fields for particular data items or parameters. Then, the user can enter or change these entries. The program should help the user by filling fields with already known data or with default values. These interfaces are common mainly in database-oriented systems and also on the Web.

Command-language interfaces come from formerly used text-based terminals and they allow the user to control the system by a series of keystrokes, commands, phrases or text sequences. They are flexible and powerful for expert users, but difficult for new users to learn.

Graphic user interfaces provide a strong metaphor of the system. Any item from the real world can be represented as a graphic object (window, menu, icon). These interfaces allow direct as well as indirect manipulation of the graphical representation on the screen and they also provide a feedback all the time. They can consist of a graphic menu, a dialog or a whole environment (*desktop*). Most of present computer operating systems include a complete graphic environment. These interfaces are discussed in detail in the following section.

4.2 Graphic User Interface

Graphic User Interface (GUI) is a program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical interfaces can free the user from learning complex command languages. On the other hand, many experienced users find out that they can work more effectively with a command-driven interface, especially if they already know the command language.

4.2.1 Characteristics

In graphic user interfaces, the user interacts with a collection of elements and objects that are always visible to him/her and that are used to perform tasks. The user performs actions on these objects such as accessing and modifying them by pointing, selecting and manipulating. The primary interaction mechanism here is a pointing device.

Onscreen-placed objects providing interaction for the user are called *controls*. Modern graphic user interfaces and environments consist of many various controls, both standard and advanced.

Objects can be either *active* (user can manipulate them to perform an action) or *passive* (without a direct reaction). The way a control *looks* depends on its status and its *feel* is given by predefined responses. E.g. a pushbutton can be either pressed or released (look) and it is pressed if a mouse button is pressed and mouse cursor is over the pushbutton (feel).

Active objects can be manipulated either directly or indirectly. *Direct manipulation* means that the user selects objects before manipulation, whereas *indirect manipulation* says that objects can be only selected after an action has been chosen. Direct manipulation is preferred by common users.

As we already know, graphic user interfaces use *metaphors* for the presentation of any information. Metaphors are virtual objects, which have an analogy to real world. They usually have a defined shape (symbol, icon, picture), affordance (physical and perceived features) and given constraints, conventions, manipulations and responses. Metaphors can be text-based (typewriter), graphical (desktop, folders, documents, trash) or VR-based (landscape, flying). All these metaphors and analogies together form the whole user interface, its type and characteristics such as type of manipulation and responses.

Of course, any exploitation of user's knowledge is welcomed and it speeds up familiarization with user interface. For ordinary user, it is always better and more pleasant to work with a system, which has some resemblance to real world. For instance, dragging an icon and putting it into trash is very clear and intuitive, because we can do that operation analogously with real paper document and real trash.

4.2.2 Levels of Graphic User Interface

Operating System level

Mostly all modern operating systems offer uniform object-based graphic user interface. The interface is presented by *desktop* (or *shell*, *workspace*), which can contain windows, menus, dialogs and icons. Desktop and all the windows and controls are usually handled by Window Manager. This system application ensures consistent Look & Feel of the whole system. Many examples of graphic-oriented operating systems can be found at [Lin03].

At the level of the operating system, we cannot usually speak about any direct support of PIM systems. Nevertheless, there are several OS features, which can be utilized in these systems.

Pointer is a graphic symbol that appears on the screen and can be controlled by a pointing device, usually by the mouse. It is a basic element of direct manipulation and provides for actions such as selecting items, changing the focus or dragging the items.

Drag & Drop is another useful feature. It allows the user to drag objects to specific locations on the screen and to perform actions on them. This can be a very easy way how to change the date of a calendar event or how to move a contact item into another folder.

Graphic operating systems also make it easier to move data from one application to another via system *clipboard*, which is a special file or memory area where data is stored temporarily before being copied to another location. Of course the clipboard works within the frame of one app.

Application level

Applications form a layer above the system level. Application developers need not worry about their *low-level architecture* (memory management, task switching), because it is fully handled by the operating system, and they can utilize all system services and features when designing *high-level architecture* of these applications, mainly their user interface. Nowadays, mostly all the classic (monolithic) applications are 90-100% user interface.

Component level

Every GUI-based application consists of components. In the speech of programmers, *components* (or *widgets*) are persistent objects, which can own and manage other components. Object-oriented GUIs handle components directly; for example in Microsoft Windows, every windowed component has its own *Handle* (window identification). Classic non-objected environments can utilize a framework to emulate the object-oriented approach (Motif).

Further, *controls* are components, which can be visible and which can interact. All controls have *properties*, *methods*, and *events* that describe aspects of their appearance, such as the position of the control, cursor or hint associated with the control, methods for painting or moving the control and events that respond to user actions. Common applications use pre-defined components and controls and their code contains mainly object responses to user manipulation.

Sets of components usually make up component libraries and toolkits like COM, MFC (Microsoft), VCL, CLX, JavaBeans (Borland) or various third-party component packages. Some operating systems (typically UNIX-based) are not graphical in the right sense, but they can utilize GUI *toolkits* and their *frameworks* (e.g. KDE/Qt, Gnome/gtk+, Motif). The frameworks can also provide for application portability to different platforms (.Net, Java 2 Enterprise Edition, Tcl/Tk).

4.2.3 Advantages and disadvantages

Advantages of graphic user interfaces unwind from the possibility of using graphics. One of the most important things is that we can utilize *symbols*; they are easy to recognize, fast to learn and they can aid in problem solving. They also save space on the screen and they are more universal and language-independent. Graphic symbols in combination with *colours* are very strong metaphors of real objects.

Another advantage is *typing reduction*. Controlling the interface by a *pointing device* can be much easier and more intuitive than typing commands, especially for new users. Using menus and choices also reduces a possibly incorrect user input. For example, graphic organizers usually provide date selection by a small month calendar that can be much faster than typing date directly.

Graphical user interface is not only more attractive than classic text-based interface but it also provides immediate *visual feedback*, thus encouraging the user to interact and explore the features of the environment.

A true graphic interface includes *standard formats* for representing text and graphics. Because the formats are well defined, different programs that run under a common GUI can share data. This makes it possible, for example, to copy a graph created by a spreadsheet program into a document created by a word processor without any information loss (e.g. OLE and DDE techniques).

However, graphic interfaces have some **disadvantages** as well. They are heavy on system resources (speed, memory, disk space) and they need adequate hardware peripheral equipment to function effectively. And last but not least, it is still true that a visually pleasant and fine-spun software system may not always be practical and functioning (ICQ is a deterrent example).

4.3 Graphic-oriented Personal Information Management Systems

In most cases, user interface of applications is given by hosting environment, generally by the operating system. That is why the first graphic-oriented PIM systems appeared simultaneously with graphic operating systems. Of course, they were very simple at the beginning. They were using graphic interface of the system, but they were not able to utilize its power.

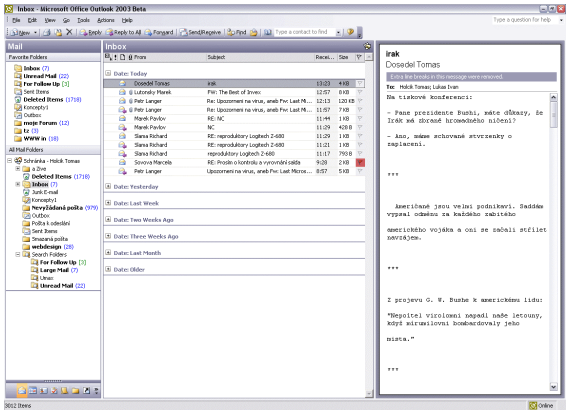
It was not until 1989 that *Lotus Notes* brought about the revolution in this area. In that time, it had a very nice look and quite usable control interface. *Release 3.0* in 1993 was another big step in graphic user interface development. And in addition, it was available for most of used platforms at that time, including MS Windows, IBM OS/2 and Mac OS. Simultaneously, a single-user version called *Lotus Organizer* was on offer. Especially the calendar interface was very sophisticated and user-friendly.

However, just a few years later, Microsoft with its *MS Outlook* became a big player in this field. At the beginning, it was a stand-alone application; nowadays it is fully integrated into MS Office. Naturally, many advanced PIM systems use very similar user interface in order to attract users that are used to MS Outlook; Ximian Evolution for Linux is a nice example (see Figure 1).

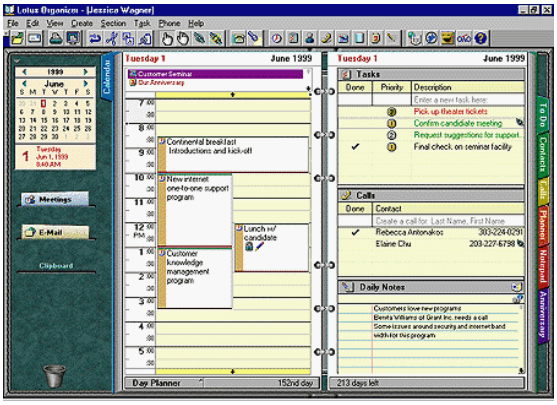
4.3.1 User environment

When talking about the user environment of systems for personal information management, we are mainly thinking about their front-end applications. It can be either a whole program consisted of one or more executables (in case of simple desktop apps and utilities), or a client application of some more complex client/server system. This application provides the interaction between the system and the user and, in general, it offers the entire environment for managing user information – so it is a program the user works with.

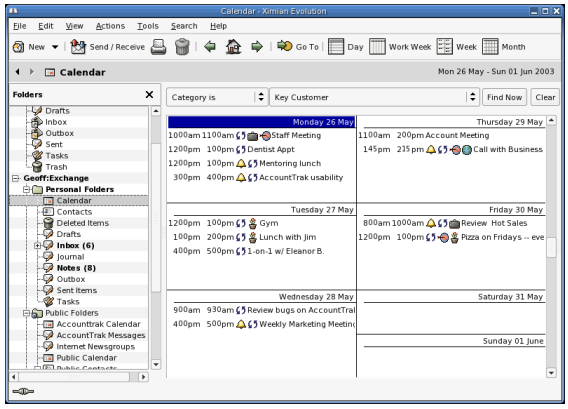
The basic parts and components of these front-end applications, mainly time views, controls, item manipulation and printing components, have gradually acquired their typical appearance. Their control has been adapted to the needs of common users, too. Of course particular systems offer various enhancements and improvements, such as advanced views (time lines, quarter planners), specialized controls (graphical buttons, menus, toolbars) or visual themes.



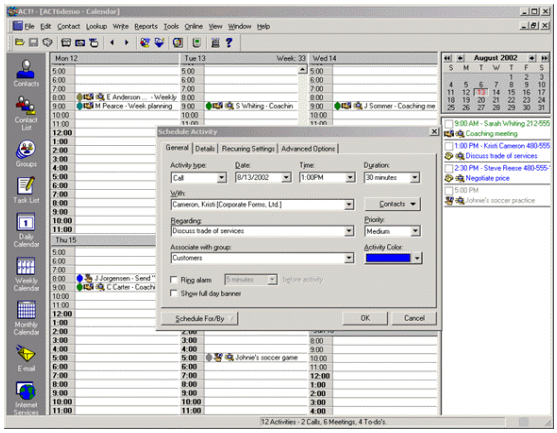
(a) Microsoft Outlook 2003



(b) Lotus Organizer 5



(c) Ximian Evolution



(d) Best Software Act! 6

Figure 1: Examples of graphic-oriented systems for Personal Information Management.

Every application has one or more windows, in which the interaction between the user and the application takes place. The user can manipulate windows like objects on the worktable (position and size changing, layering, closing). Windows contain various controls. Windows in **MDI** (Multiple Document Interface) applications can contain other windows, as well. However, common applications usually offer the standard **SDI** interface (Single Document Interface), which consists of a main menu, a toolbar, a status bar and a user-defined area. This is the case of simple PIM systems like organizers, notepads or email clients. Main window shows data and handles all required interaction and particular dialogs provide additional features.

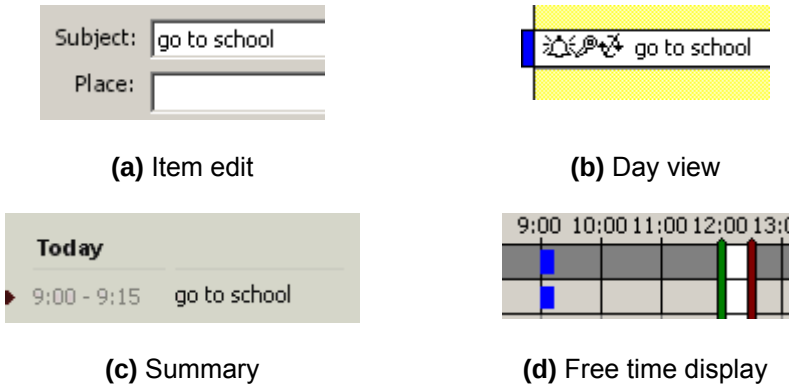
User-defined area of the main window serves for the actual view, consisted of particular panes and components. There are several possibilities to sketch out their arrangement. Simple utilities use for displaying information only *one pane*, which occupies the whole window area. That is suitable for non-categorized items and documents. *Two-pane view* is often used for categorized items. One pane contains a folder list or a tree and the second one displays the content of a selected folder. If

items represent some more complex information then we can use the *three-pane view* where the third pane displays the content of the selected item. For example, MS Outlook uses this scheme for displaying emails. Other ways of displaying the application layout are to split the view to several windows (like in MDI app), or to display all the information at once – this is called *"one screen"* interface. For example, Time&Chaos offers such a user interface.

4.3.2 Data items

The most important purpose of PIM systems is managing user's data and information. The user stores his/her personal data in these systems in order to keep a good track of it.

The basic question is how to represent the data. All the **user-related information** can be represented in the text form. However, graphic symbols are more suitable, for instance, for logical values (graphical switch buttons) or information about item type (symbols, icons). Figure 2 shows various possibilities for representation the same data on selected example of a single time event.



(a) Item edit **(b)** Day view

(c) Summary **(d)** Free time display

Figure 2: Examples of data representation (Microsoft Outlook).

There is one related problem, which is specific just for PIM systems. We need to display and edit **time-related information**, concretely time and date values. All these systems enable date and time entries using appropriate controls. Some systems offer various versions of time-oriented edit boxes with in-place date picker in the shape of small month calendar. Lotus Organizer5 uses a special resizable timeline, which can be also useful, but it requires a certain level of handiness. The following Figure shows several examples of controls intended for date and time entries.

Generally, we can recognize read-only controls, which only display the information, and components providing both input and output. Controls from the second group are more difficult to implement, because they have to handle possible incorrect user input such as invalid date and time values or out-of-range numeric values. That is why they often replace direct text-based entry by graphic-oriented controls. For example, numeric edit boxes often include small spin buttons for increasing and decreasing the value.

(a) WinPIM **(b)** Cute Organizer **(c)** Pimento **(d)** Microsoft Outlook

Figure 3: Examples of controls providing date and time entries.

Lot of applications split input and output controls, because information views without input controls are more compact and transparent. Then, the user can look at this summary information and eventually change it by switching into the edit mode or invoking a corresponding dialog.

Table 5 presents the most typical kinds of user-related information and controls, which are suitable for displaying and editing such information. For the controls, their common names from window-oriented programming languages are used.

Information	Examples	Suitable controls
Text value	Title, name, address	Edit box, memo (multi-line)
Numeric value	Day, month, year, priority	Restricted edit box (mask edit), spin, tracker, slider
Logical value	Privacy, Recurrence	Check box, option box, group button, switcher
Date and time	Begin, end, current time	Advanced edit box, time line, calendar, planner
Time range	Event duration, time view	Advanced edit box, time line, calendar, planner
Enumeration	Item type, occurrence list	List box, set of buttons, option boxes or check boxes

Table 5: Visual controls for displaying and editing simple user-related information.

Furthermore, we can group these simple data values into a bigger unit. Then we get *structured information* (record). PIM systems usually present such records as particular **data items**. Also components providing their views and edit features are structured. They can be either made of several simple controls (e.g. forms or dialogs), or they can stay as stand-alone complex controls.

Next table presents the most used kinds of structured data items and their suitable views.

Information	Examples	Suitable views
Time event	Action, task, remind	Calendar, item list, table, summary
Message	Email, instant message	Item list, tree list, table, preview control
Contact	Person, group	Item list, tree list, table, preview control
Note	Text, images, objects	Item list, tree list, table, embedded preview/editor
Document	Text document, sheet	Item list, tree list, table, embedded preview/editor

Table 6: Views for displaying and editing structured user-related information.

4.3.3 Views

Naturally, all the views mentioned above in Table 6 can also show more than one item at one time. That is why we are not speaking about particular controls anymore, but about complete (multi-item) views and panes. **Item views** usually consist of several more or less complex controls and they offer wide possibilities of displaying and editing large numbers of data items.

Perhaps the most problematic part is the calendar, because calendar views are very specific. Basic event lists can be created from standard controls and creating the time views can be very simple, too; nevertheless, if we want to ensure required interaction such as inserting and editing items then we have to use or create special controls.

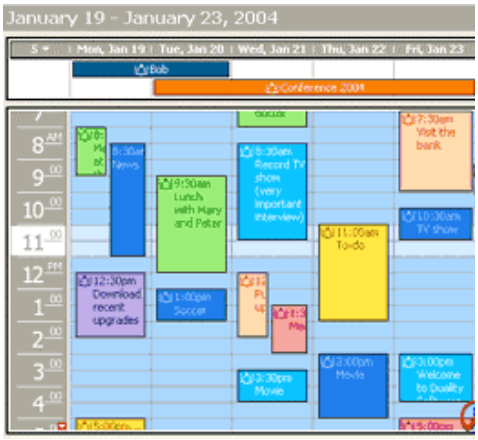
The basic time-oriented view is certainly **Day view**. Here, the time line is usually represented by rows, which can contain inserted events. More sophisticated calendars can display time length of the items in various ways (item frames, time bars), they provide interaction (creating, selecting, editing and deleting items) and they can also show more days at the same time (see Figure 4).

Week view can be represented either in the same way as the day view or it can look like a real weekly calendar with big cells for particular days.

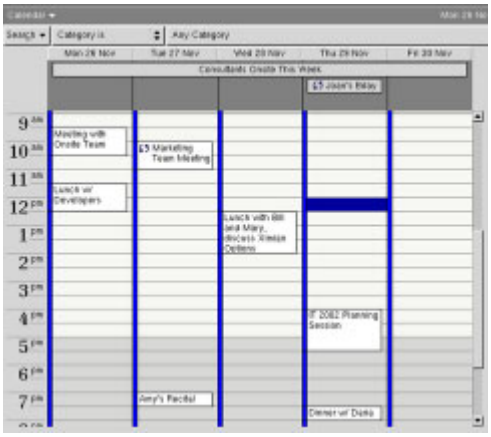
Month view is always represented by a grid with 7 columns (week days) and 6 rows (weeks). Only some calendars enable further interaction such as item editing in this view. Special kind of this view is a small monthly calendar without items that can be used for date selection. Thus, all days, which contain items, can be at least marked using different colours or fonts.

Year view usually consists of twelve monthly calendars without items and enables smooth transition between them.

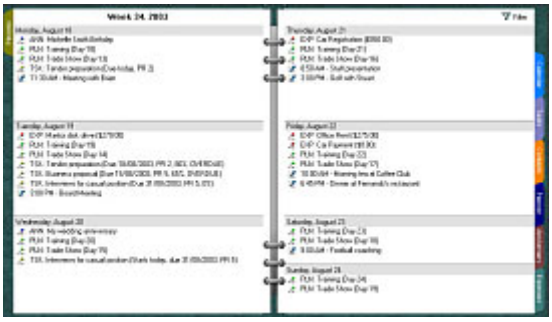
Another approach how to represent time are **time planners**. Here, time is shown as a line of days, time within days is omitted. Such plans are suitable for all-day events. This approach is also known already from Lotus Notes and Organizer. Figure 5 shows the year planner from Pimento, an organizer with very similar interface like Lotus Organizer.



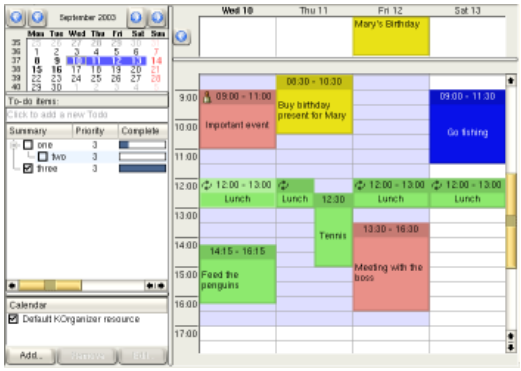
(a) Calendarscope



(b) Ximian Evolution



(c) Pimento



(d) Korganizer

Figure 4: Examples of multi-day view.

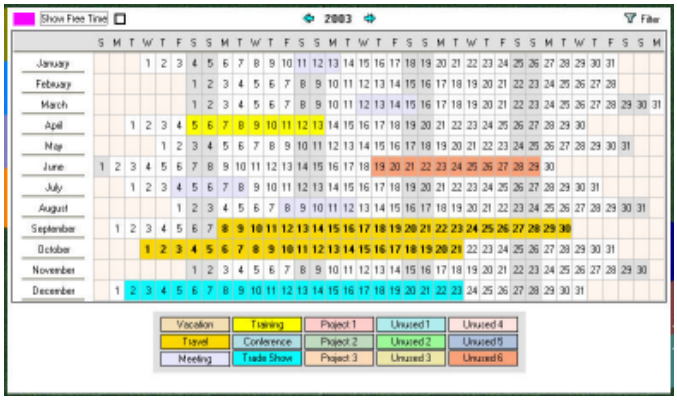


Figure 5: Example of year planner (Pimento).

Task lists (or **to-do lists**) are usually represented using standard controls such as tables or list boxes, which can – beside the basic item view – offer additional features, for example direct editing of items in rows or graphical representation of some values (task progress, priority symbols; see Figure 6). In some applications, task lists are placed on panels or together with time views.

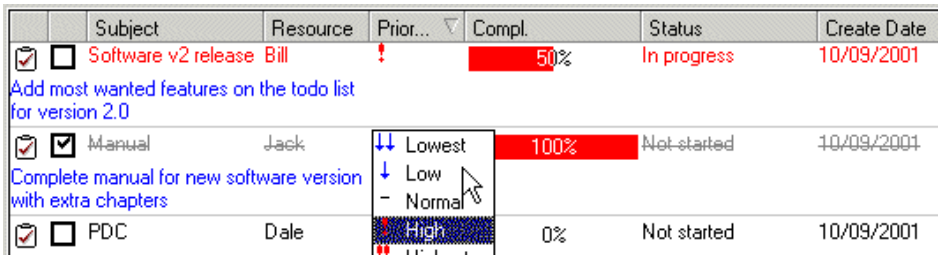


Figure 6: Example of to-do list with additional features (`TTodoList` component from TMS).

From the point of view of using controls, **contact books** are easy to implement. Just in case of using groups or another kind of multi-level hierarchy of contacts, we have to represent this hierarchy by corresponding controls. *List views* and *tree views* known from Windows Explorer are mostly used. These lists enable displaying sets of items as big or small icons, list or detailed table, where every item is shown together with its properties in particular columns. Then, all the items can be sorted according to any selected column.

The same applies for **notes**. Simple utilities rather provide note lists without any hierarchy, more sophisticated applications allow the user to use folders or categories. To look more natural, some programs display items like real stickers or diaries, which can be tacked anywhere to the desktop. For this purpose, they utilize improved edit controls, which are independent of the main window and allow the user to insert texts or graphical notes. For an example see Figure 7.

Figure 7: Example of sticking notes (StickIt!).

A very important requirement is the possibility of information **filtering** and **searching** in the item views. Every user sometimes wants to see just some certain kinds of items, for instance all events within a date range or contacts related to some company. Unlike filtering, which ensures the

display of the requested information implicitly, searching represents an explicit way to get the data that we are looking for. Another way of sorting and matching the items is using **categories**.

Most of today's used PIM systems include advanced memos, so-called **rich-text edits**, which provide formatted text including justification, different fonts, colours, numeration, inserting objects (mainly pictures) and links. They are usually based on HTML or RTF standard.

These controls can be used for displaying needed information in many ways, mainly in form of various previews, summaries or print reports. The following Figure shows an example of formatted contact book:

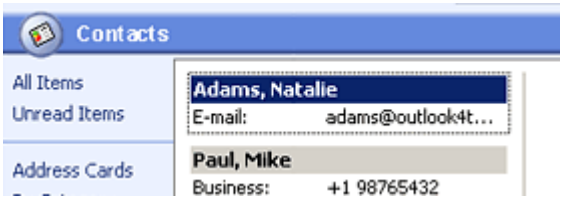


Figure 8: Example of a formatted text (4Team for MS Outlook).

4.3.4 Additional features

PIM systems naturally utilize all GUI components and controls provided by parent OS in order to ensure various features and functions. Besides basic user-data management including creating, editing, transformation and deleting items, they provide additional features such as sorting, searching, clipboard operations, history (Undo/Redo) or context-sensitive help. From view of user interface, all these functions must be easily accessible and their progress must be transparent. In other words, the user should immediately recognize, which part of the program is currently active and which function is being processed.

Figure 9: Example of using tray icons.

Last but not least there is the *global scope* of the PIM system. All the desktop applications are more or less integrated into parent environment (OS). They are usually accessible via desktop icons or via some system menu and the operating system can also take care about their uninstallation. Some applications register their data file formats during the installation, which means for the user that s/he can execute the application by opening a data file in the OS environment (e.g. file manager).

The workspace of the application does not need to be bounded by the main window and dialogs. The whole desktop space is available. Especially programs and utilities for notes handling use the desktop area for placing “stickers” (see above). Dynamically generated desktop wallpaper can also display useful user-defined information, e.g. a week schedule or a to-do list.

Most of today used desktop environments include a special desktop window, which usually contains a set of controls for doing common tasks, task bar, status area and some kind of main menu. For instance, Main Panel in Microsoft Windows, KDE Panel in Linux and CDE Front Panel in Solaris are well known. Status area of this window offers easy access to running programs via so-called *tray icons* and it can also inform about their current status, as you can see on Figure 9.

4.3.5 Mobile devices with Graphic User Interfaces

And what about **mobile devices**? In fact, simple graphic interfaces were already common in electronic diaries and data banks in early 1990s, but it was not until 1996 that Palm introduced the first handheld computer, *PalmPilot*, enabling real scheduling, a to-do list, a phone book and notes.

Personal Information Management is the basis of PDA devices. As we have mentioned, there are three widely used platforms on the market nowadays: PalmOS, Epoc/Symbian and PocketPC standard with WinCE. They can be seen in newer mobile phones (so-called *smart phones*), too. Their graphic environments are shown on Figure 10. All these mobile systems offer generous possibilities of managing time-related and user-related data.

(a) DateBook+ (PalmOS) **(b)** Handy Day (Symbian) **(c)** Pocket Informant (WinCE)

Figure 10: Examples of Personal Information Management Systems for mobile devices.

4.4 User Interface Design

4.4.1 Basic design principles

The main idea of user interface design is to make a powerful but complicated system simple and intuitive for the user. The control of such a system should be easy to use and the output of the system should be visually transparent and well structured. *Simplicity* is paramount; if a system is user-friendly, attractive and uncomplicated, it is a perceived success.

In [Tog92], Tognazzini enumerates lots of objectives, which should be met in order to design a good user interface. Here are the basic and most important ones:

- **Productivity** – the main goal of every system is to increase user's work productivity
- **Effectiveness** – the user may be able to access the system in a way that is congruent with his/her individual needs
- **Efficiency** – the system should help the user to speed up data entry and reduce errors
- **Robustness** – the system must provide for the user's data in case of any error
- **Simplicity** – the control must be easy and intuitive and the output must be transparent; the user may understand what the computer says and vice versa
- **Feedback** – the user should always be informed about the current status of the system
- **Consistency** – the look, control and responses of the interface should be consistent. We know three kinds of consistency: *internal* (within the application), *external* (with other interfaces) and *consistency with the real world*. The consistency should be upheld where necessary. If a system is consistent with some common standard then it is easier for the user to learn it.

4.4.2 Graphic User Interface Design

Every usable GUI must fulfil a set of basic principles. We have already mentioned some of them above; the others are specific just for graphic interfaces.

Graphic user interface should provide so-called *visual stimulation*. It means that the appearance and control of the interface should be natural and pleasing to the user. Visual interface design includes graphic layout, component design and usage of colours, images and multimedia.

Controls must be designed so that the user is able to anticipate their behaviour from their visual properties as well as by using knowledge gained from other systems. For instance, a control for applying changes looks like a button. If user clicks the button, it becomes "pressed" and a particular action is executed. The system provides adequate feedback to the user by showing the button as really pressed.

Users like *customisable environment*. They prefer certain colours, layout, windows size and position, fonts, background pattern or keyboard accelerators. The system should preserve individual interface settings for every single user to offer stable perceptual cues for a sense of

"home". However, the graphic interface should be only an instrument of interaction between the user and the computer, not a centre of their work with the computer.

Interface designers sometimes forget about older and weak-eyed people and they use small fonts, petty controls or low-contrasted colours, in spite of basic *recognition rules*.

It can sound a little bit strange, but the system and its interface are responsible for *keeping the user occupied*. If the interface is untransparent, the user becomes confused and not able to work effectively. Controls should be well arranged and messages, texts and labels should be short, clear and consistent in order not to hinder the user.

No system must force users to perform tasks in a specific order by using *modal behaviours*. However, if modal behaviours are consciously and thoughtfully applied they can be used to advantage (e.g. dialog boxes). It is closely involved with reversibility of made changes; the system should always provide the user with a possibility to cancel previous actions (*Undo*).

Users don't like *waiting*. If it is necessary to perform some long-time action, they should be informed (by a message window or hourglass cursor) and should be offered to cancel the action.

And finally, users expect *help* when they are not able to solve a problem themselves. Thus, every good graphic user environment should provide a usable Help system.

4.4.3 Methods, techniques and tools

We know that we need effective user interfaces to fulfil their potential. Designing such an interface is partly a *discipline* (following the conventions and good design principles), partly a *science* (usability testing) and partly an *art* (creating screen layouts that are informative, intuitive and they look nice).

Analysis

Thorough analysis comes before every design. We can recognize and analyse various aspects concerning dialogs between the human and the computer, namely *human-related* properties (senses, workload, consideration, knowledge), *computer-related* properties (speed, performance, cost) or *environment-related* properties (market, customers, cultural and social aspects).

For the analysis we can use some standard methods, for example some of empirical, cognitive, anthropomorphic methods or modelling. We should pay attention especially to human-related aspects when doing user-centred design. Users must not be overloaded by computer's output because otherwise they cannot concentrate on all the information and their working efficiency starts decreasing.

Design

Analogously, there are several common approaches for *interface design* itself. One of widely used design methods is Conceptual Design. It utilizes a guide definition (*concept*) of the user interface and its look, functionality, navigation, architectural scheme and data flow.

All relevant rules and prescriptions are usually encapsulated into a design guideline, so-called *Style Guide*. Nowadays, every manufacturer of a graphic interface keeps and improves an own Style Guide. The most-known Style Guides include IBM CUA Guide, Macintosh Human Interface

Guidelines, Common Desktop Environment Style Guide, OSF/Motif Style Guide or Microsoft Guide for Window Interface ([Tro95]).

In bigger projects, using a Style Guide is the only way to preserve the interface consistent. If a software project is so specific or closely specialised that there is no suitable guideline available, then the developers must design an own corresponding interface. For instance they can use standard Waterfall Model (sequent development) or Prototyping. Developers using prototyping produce prototypes of future user interface and on the basis of precedent analysis they iteratively improve its functionality and capability in quest of final shape. The emerging interface must be tested by end users continuously in order to ensure desired features.

Implementation

Aspects of implementation are not visible to the user, however, they are important for the designer and developer of the user interface. For a certain type of user interfaces, it is very suitable to use corresponding programming approach. E.g. for classic command-language interfaces, procedural programming is sufficient, whereas objects and items of graphic user interface can be handled more easily by using benefits of object-oriented and event-driven programming.

Common implementation problems and tasks include managing both static and dynamic controls, user input and error handling, redrawing or creating user-defined components.

The selection of programming approach also determines, what tools (here programming languages and environments) can be used for the implementation. Today's widely used platforms and programming languages are C++, Java, Delphi, Visual Basic, PHP or .NET platform.

Testing

It is quite difficult to say, how good and usable a user interface is, because there are many aspects affecting the system in general, such as speed, performance, error rate or time to learn. We are able to measure and evaluate some of these specific quantities and the acquired results can help us assess the user interface as a whole. Users can help us by answering questions related to their experience with the user interface. Here are some examples of useful questions:

- *"How much time does it take to learn?"*
- *"How much time does it take to issue instructions?"*
- *"How often do you need help?"*
- *"How many errors do you make?"*
- *"Do you improve quickly?"*
- *"Do you remember how to use it over time?"*
- *"How satisfied are you?"*
- *"Is it intuitive?"*
- *"Are the results as you were expecting?"*

We can assess these characteristics as well, but we can find it quite difficult to evaluate them. Especially the answers and classifications of ordinary users, novices and laymen can be very subjective and ambiguous. Nevertheless, a very subjective opinion from a user of some personal information system (*"I like it. / I don't like it because..."*) can be very important.

Formerly performed tests and evaluations of existing user interfaces have confirmed some aspects of good and usable user interfaces (see [Rau93]):

- Standard and common look, control and behaviour
- Easy communication and interaction
- Items collected into logical units
- Minimal user action
- Default actions, values and responses
- Clear and understandable error or warning messages
- Ergonomics
- Help and support

As we can see, User Interface Design is a lengthy and sophisticated process – and expensive, too. Anyhow, a well-done and usable user interface can save lot of money on documentation, training and support calls ([Mar02]).

Chapter 5

Multi-user Environment

5.1 Data Sharing

5.1.1 Private, public and group data

Till now, we have only discussed the user's personal data and we have just occasionally mentioned the term *"data sharing"*. However, in practice it is necessary to share data with others, whether we like it or not ([Pal98, pp. 83-108]). Let us classify user-related data like this:

- **Private data** – user's confidential data, invisible to the others
- **Public data** – user's data, visible to the others
- **Group data** – data and information related to the whole user group

Private data can be for example access passwords or private notes, *public data* are especially the information about working activities and contact information. The information and documents related to the whole group (e.g. concerning some collective project) are *group data*. Of course, users meticulously protect their private data. Only public data and group data can be shared.

5.1.2 Working in groups

What are our possibilities, when we join a group?

The main feature of group-oriented work is doubtless *data sharing*. It means that every particular user can allow the others to see his/her schedule, tasks, contacts, notes, documents and objects. The entire group can also own collective data, which are accessible for all joined users. In this case, the group is the only owner of this data and users can work with them according to their access rights. It is very important to ensure sufficient privacy and security, because mainly in corporate enviroment the information can be very confidental and valuable.

Another important part of teamwork is *communication*. It can be very difficult or impossible for the users within a group, team or company to meet each other at the same place and at the same time. That is why communication platforms and applications like ICQ, MSN or AIM are so popular – they allow users to communicate anywhere and anytime. Other widely spread possibilities of computer-aided communication are email, forums, Internet telephony or videoconferences.

The combination of data sharing and communication within groups provides many features. Users can discuss problems related to their common work, they can organize live meetings or use boards and forums, and they can work on collective tasks and inform each other about current work status. According to [GMN02], we can say that all the shared data fall prey to socio-technical evolution within the group.

Figure 11 shows basic dimensions of multi-user environment and possibilities of collective computer-aided work and communication. This is also basic premise for groupware systems.

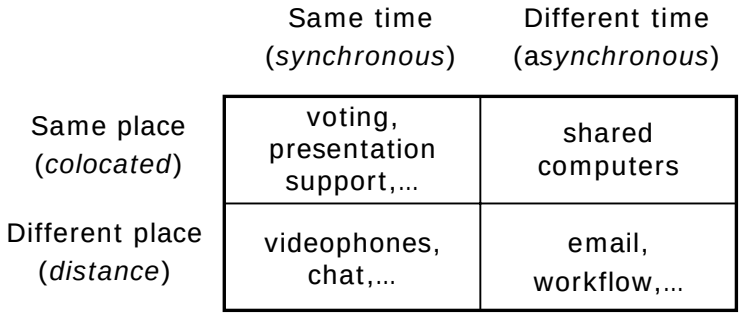


Figure 11: Basic dimensions of group-oriented work.

5.1.3 Groupware

In general, *groupware* is a technology designed to facilitate the work of groups. In scope of personal information management it is a specific class of technologies, which rely on modern computer networks and which are used to communicate, cooperate, coordinate, solve problems, compete or negotiate. That is why this field is also called *computer-supported cooperative work* (CSCW). The key issues of CSCW are group awareness, multi-user interfaces, concurrency control, communication and coordination within the group, shared information space and the support of a heterogeneous, open environment which integrates existing single-user applications (see [MPB92]).

Groupware is also the name for software systems, which provide these features. They allow users to share data and communicate within the group; so it means they offer the most essential functions from areas of time management (especially time scheduling at the group level), project management, CRM (Customer Relationship Management) and workflow management. In this case, personal data take a part in the spectrum of group data, documents, messages and information that are created, stored and shared within the entire group or its subset.

When making design of a groupware system, we have to keep in mind some important aspects of this area. At first: unlike single-user applications, multi-user systems must be very flexible in order to reflect all possible changes. Groups and roles are dynamical and they change very quickly.

Organizing and scheduling for groups is also more difficult than for individuals, especially in large organizations.

Furthermore, such systems should be able to manage multi-group environment. For instance, particular users can change the parent group or they can become members of more groups. This can be really hard to solve, especially if the groups are hierarchically organized, as in case of a big company with offices, buildings, divisions, departments and branches.

Various aspects of groupware systems and computer supported cooperative work in general are well described in [Bae92].

5.1.4 Technical solutions

We know two basic types of groupware systems according to their usage of network interface:

- **Asynchronous** systems – email, newsgroups, mailing lists, boards and forums, workflow systems, collaborative work systems
- **Synchronous** and **real-time** systems – instant messaging (IM) and chat systems, shared whiteboards, videoconferences, decision support systems, multi-player games

There used to be very poor possibilities of network communication in former applications, because there were hardly any sufficient network resources. First groupware systems were working just within local networks (LAN). It sounds a little bit strange in the time, when the Internet is almost taken for granted. The Internet (together with local and intranet networks) is the main medium for groupware systems. The network architecture is usually based on some of following schemes:

- **Direct communication** (synchronous and real-time systems)
  - **Peer-to-peer** (P2P) – all clients communicate with one another directly via network interface, without any master or arbiter. This is not suitable for bigger groups, because the required N2N communication becomes very heavy on network load. However, this communication can be very effective within small groups on local networks.
  - **Client-server** (C/S) – all clients are connected to the server, that takes care about the entire system including user management, data management, distribution of shared information, security and access policy. This scheme is advantageous for any group type. The only serious drawback here is that in case of any server failure the entire system stops working.
- **Indirect communication** (asynchronous systems)
  - **Electronic mail** – beside normal correspondence, clients can use email for sharing any information. All the shared data are transformed into text form and sent via network as ordinary email. The other recipients get this information as soon as they check their mail. For example, Net Folders technology from Microsoft uses this scheme.

- **Web interface** – the client interface is web-based and platform-independent, which means that the users can use the system from any place connected to the Internet. In spite of its slow speed and higher network load it is very popular nowadays.
- **Common space** – all the shared information is stored at certain pre-arranged place, usually in a database or a network directory with shared access. All clients have to take care about the data management and related things (synchronizing, exclusive access). This solution is suitable mainly for small local networks without Internet.
- **Import and export** – clients exchange the information by passing the data stored in a temporary file from one client to another. This approach is more suitable for data exchange between different systems, which understand the same exchange format.

5.1.5 Standards for data sharing

Common technologies defined for Personal Data Interchange (PDI) include vCalendar and vCard formats, currently handled by Internet Mail Consortium. They both are text-oriented, which means they are also platform-independent (like XML) and available in almost all PIM systems.

The **vCalendar** format is designed for calendar and scheduling data exchange in a simple, automatized and consistent way. Every system meeting the standard should be able to recognize this format and present the data stored in this format in the same way.

Likewise the **vCard** (a visit-card) format is designed for the personal contact data exchange. Various applications use this format – PIM systems, mailers, fax programs or smart cards. The format description can be found in [HSD98].

A vCard item can look like this:

```
BEGIN:VCARD
VERSION:3.0
N:Moskowitz;Robert
FN:Robert Moskowitz
EMAIL;TYPE=INTERNET:rgm-ietf@htt-consult.com
END:VCARD
```

Unquestionably the Internet is a very suitable medium for the disclosure of personal data from anywhere. Nowadays many web-oriented calendars, schedulers, databases and information systems exist; the best-known systems are Amphora, iOffice 2000, Teamware Office or PHPGroupware.

Obviously that is why the **iCalendar** protocol standard has come to the scene. It is designed for calendar and scheduling data exchange within the Internet, mainly for transferring information about activities, tasks, free time, reminding, time zones, location etc. In principle, it is only a network extension of vCalendar. Basic specification is stated in [DS98] and related documents.

5.2 Synchronization

5.2.1 Keeping data consistent

The reason for data synchronization is that user data are very unstable. One can create, change or delete them practically whenever. Synchronization ensures, that the other users can always (or mostly always) see the current data status. There are two different ways how to synchronize:

- **Implicit synchronization** – after every data change is current version automatically sent to all subscribers. This way ensures permanent consistency of the information on all places, but it loads the sender application and the network.
- **Explicit synchronization** – the subscribers get actual version of the data in the moment, when they ask for it. That can be done either manually or automatically by the target client in certain time intervals. The synchronization times should be balanced with consideration – in order to ensure both permanent consistency and low consumption of system resources.

5.2.2 Time synchronization

Data synchronization works in that way two different versions of the same information are compared according to their time and the older information is replaced by the newer one. To be able to do that, we must have the same time on both sides. That is why we also want to synchronize time itself.

There are many sources of exact time (so-called *atom time*) available. They are provided by organisations such NASA or German Physikalisch-Technischen Bundesanstalt (it runs the DCF77 transceiver, too). A few network protocols have been designed for getting the exact time: **Time** protocol, **NTP** (Network Time Protocol) and **SNTP** (Simple Network Time Protocol). They come up to accuracy $\pm 1 \div 500$ ms (according to operating system).

Sources of exact time usually work with global **UTC** time (Universal Coordinated Time). This time is also well known as GMT or "London's Winter Time". It means the local time related to the zero-meridian and its notation is *hhmm UTC* (or *hhmmz*, where the letter *z* means "Zulu" thus world time – obviously from historical reasons). The entire world is currently (since April 2003) divided into 37 time zones. For example, the Czech republic and other Central-European states use CET zone (Central European Time, +0100) for standard time and CEST zone (Central European Summer Time, +0200) for daylight saving time.

Thus, when designing the time synchronization, we have to keep in mind that particular users can be placed in different time zones anywhere in the world.

Chapter 6

Budiik System

6.1 Philosophy

Budiik is a simple software system for personal information management (PIM), intended for single users and small groups from Small Office/Home Office area. It is currently available for the MS Windows platform and it is based on a former application called *Budík*. The name (Budiik) is derived from the Czech word "budík", which means "alarm clock".

Budiik provides personal information management including calendar, tasks, contacts and notes. Its main purpose is to ensure that its users will keep a good track of their schedule, work and tasks, calls, contacts, ideas, documents and generally all user-related data. The system attempts to offer very simple and intuitive user interface to ensure fast and effective usage, without needless and complicated features and details.

Budiik is intended mainly for single users, however it also provides basic possibilities of collaborative work, data sharing and communication within small user groups (up to several dozen users) connected via the Internet or local network. Technically it is based on client/server architecture, where particular client applications connect to central server.

Budiik attempts to fill a gap between simple single-user PIM applications and utilities with insufficient functionality, and complex corporate solutions, which are usually very complicated and expensive. It is also a cheap alternative to similar commercial products. Unlike many other apps, Budiik is multi-lingual. It is very simple to translate into new languages.

And last but not least, Budiik continues in tracks of its successful ancestor Budík with a wide user community.

6.2 Requirements

6.2.1 Requirements analysis

During the work on my School-year Project ([Mar03]), I made a public enquiry of the Budík's users in order to get their experience with the current system and also their suggestions, comments and requirements for further improvements. I wanted to get necessary information and a basis for the analysis and system design, consequently. Above 160 informants sent their answers.

Following matters have emerged from the inquiry results:

- Budík is used evenly in office and household. Users mostly work on computers in the office and they usually hear about Budík there.
- Calendar and to-do lists, eventually notes are used daily by majority of users.
- Most of users have got the program from a computer magazine, although the Internet is the most widely used medium for communication and getting new versions.
- Users prefer a stand-alone mailer rather than a PIM system with integrated mail client.
- Data sharing (in smaller local networks) and synchronization (especially with electronic diaries and pocket computers) seems to be attractive for many people.
- The current form of the system is preferred: a tiny free application available on the Internet, without complex documentation and technical support (nevertheless a part of users would accept some basic technical support for adequate payment).

6.2.2 Functional requirements

The inquiry results named above together with underlying ideas and principles form up basic requirements for the new system:

- Tiny and fast client application with low system requirements
- Availability for the most used operating systems (Windows98, 2000 and XP)
- Simple and intuitive graphic user interface
- Basic personal information management (calendar, tasks, contacts and notes)
- Sufficient item handling (creating, editing, copying/moving, deleting, categories)
- Data presentation: scalable views, item preview and summary, rich-text support
- Particular data interconnection (meetings with people from contacts, birthdays, day notes)
- Additional item-related functions: clipboard, searching, sorting and printing
- Advanced time-oriented information (day timeline, moon phases, name days, feast)
- File management: extensible data format, encryption, data backup and restoration
- Possibilities of time and data synchronization, import/export features
- Possibilities of data sharing and communication within small groups
- Teamwork: organized meetings, collective tasks, shared board
- Privacy and security: private/public data, passwords, encrypted communication
- National Language Support: localized client interface, national name days and feast
- Integration into environment (desktop and Start Menu shortcuts, access via tray icon)
- Easy installation, update and uninstallation, without any additional libraries or DB engines
- Context-sensitive help

6.3 Architecture

Although Budiik is intended mainly for single users, technically it is split into the client part (Budiik Client) and the server part (Budiik Server), as Figure 12 shows. Budiik Client is a classical standalone application, which provides complete interface for managing diaries of particular users. Budiik Server is a small multi-threaded program, which serves for data sharing and synchronization either among several places belonging to one user (home and office, PC, laptop, mobile device) or among more users within a user group. The server is managed by an administrator, who is responsible for the server settings including the file storage, user management and security policy.

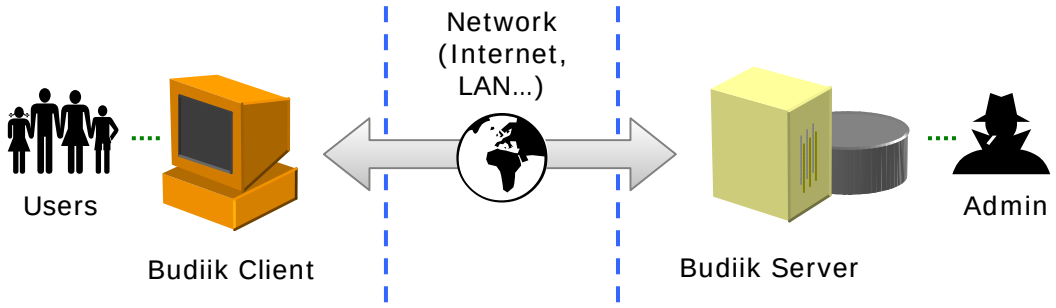


Figure 12: Budiik System architecture.

The client and server part are connected via network, typically via the Internet or within a local area network. Figure 13 shows the network architecture, with various possibilities of connecting particular users, places, devices and applications of the Budiik System.

There are several "Use Cases" how the system and network organization can look like:

- The most typical situation is that there is only one computer with the client application, which is used just by one user, without any data sharing. Data can be exchanged with another applications via import/export.
- There can be more users that use the application on one computer, for example members of a family. All users work with their diary and only one diary can be open at one time. Public data of particular users can be exchanged via import/export.
- Another typical scenario is that a person works on a computer at different places, usually at home and in the office or somewhere on a laptop. In order to work every time with the same personal diary, s/he synchronizes the local diary with data stored on the Budiik Server, that is running somewhere in the network, for example at the home PC or on the company's server.
- However, data files can also be transferred between different places on a physical medium (floppy disk, USB memory, CD-ROM) or directly via network (email, FTP, file send).

- Besides synchronizing with the server, any user can synchronize data with a mobile device like PDA or smart phone, either directly (not yet in Budiik) or using data import/export.
- The main purpose of Budiik Server is data sharing, communication and teamwork within a small group, for example a small company (up to dozens of employees), an office crew or a bunch of friends living at different places. Then, particular users are connected to the Budiik Server, which is running on a common server somewhere in the LAN or in the Internet.
- In case there is no dedicated server (like in P2P networks), Budiik Server can also run at some other easy-accessible place – usually on somebody's computer. It follows that both the client and the server application can run on one computer at once. Of course, this is the purpose of a convention between all the group members (e.g. who will be the admin).

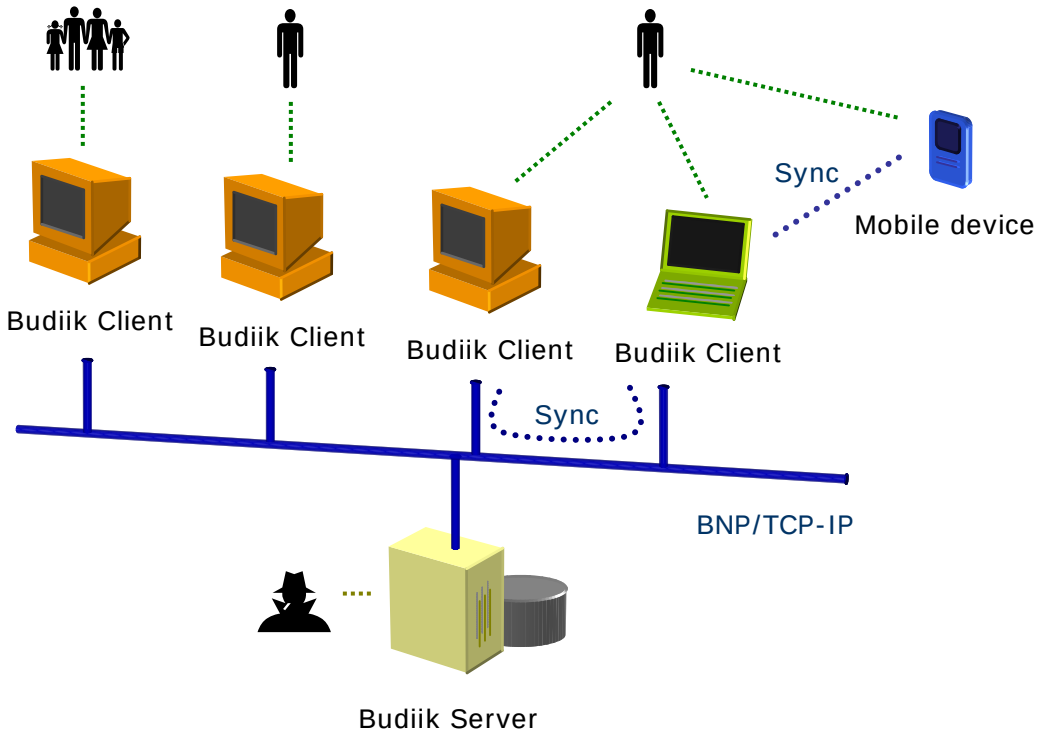


Figure 13: Budiik – Network architecture.

Both client and server application consist of three logical tiers:

- Client Tier – graphic user interface providing interaction with the user
- Application Tier – common functions, network interface, diary management etc.
- Data Storage Tier – file management and low-level routines

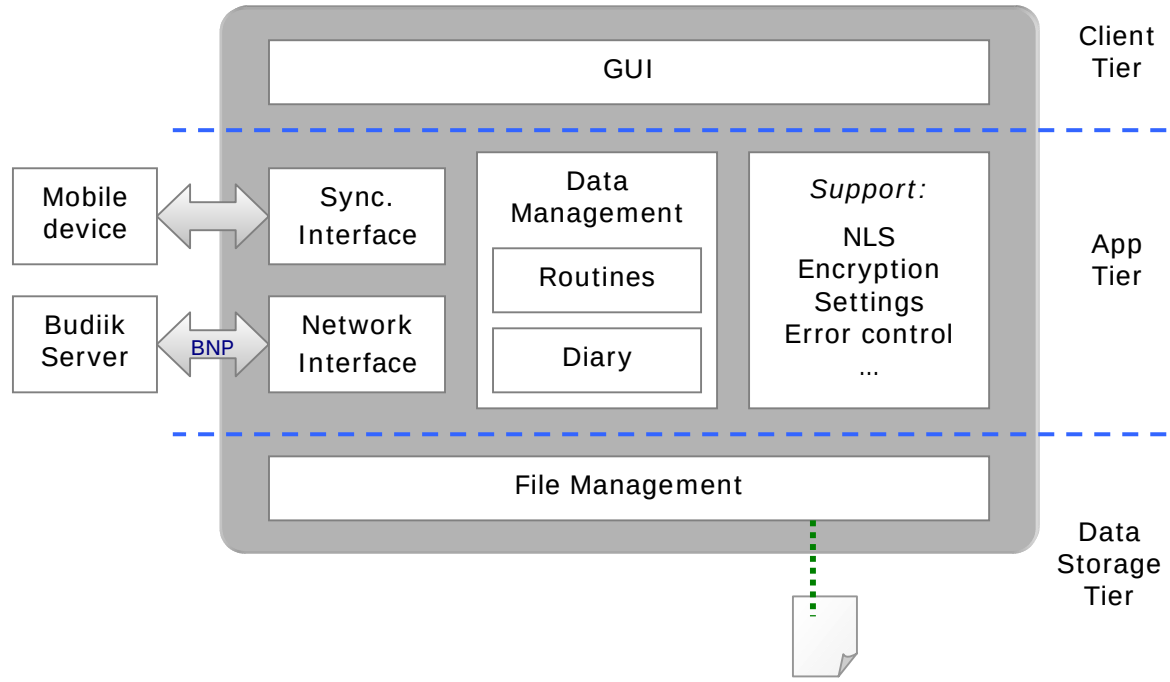


Figure 14: Budiik Client architecture.

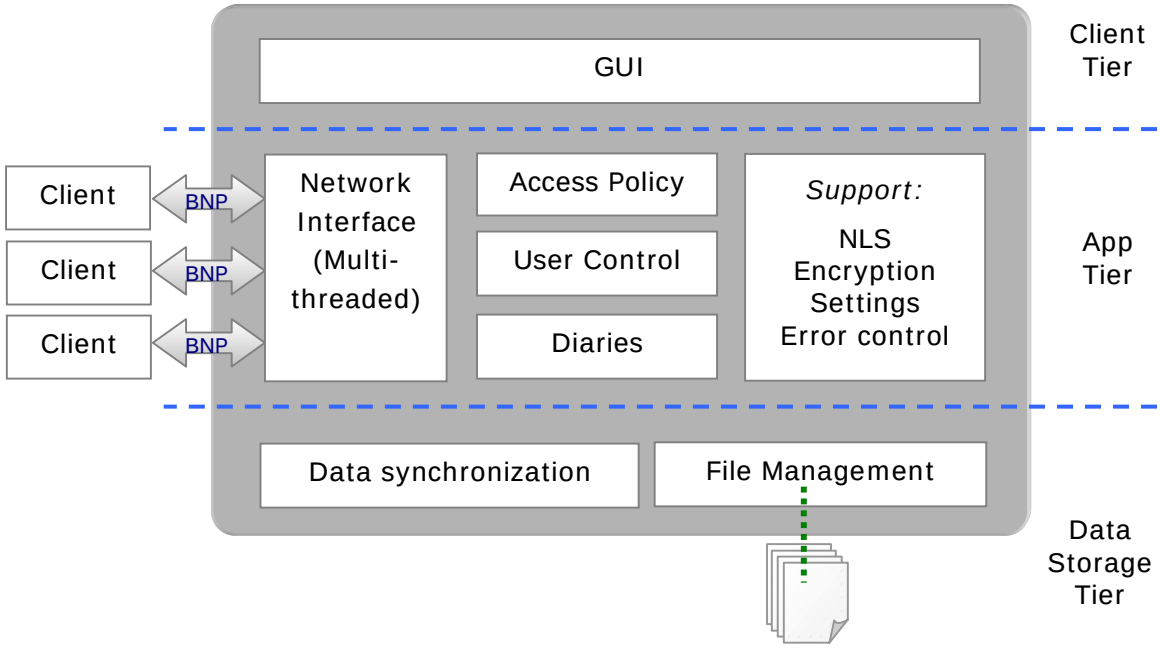


Figure 15: Budiik Server architecture.

Figure 14 shows the architecture of the client application. Here, top-level client tier is the most important part of the application because it provides interaction between the user and the system and it determines how the system will be user-friendly and usable for the user. Application tier takes care about diary data, system routines and algorithms (mainly encryption, application settings, multi-language support and error control), synchronization, communication with the server and advanced features like searching, sorting, printing, data import/export and background activities (Sleeping Mode). And finally the lowest tier handles data storage of user diaries.

Conceptual design of the server application is similar (see Figure 15). Administrator can control the server via simple graphic user interface, which is encapsulated in the client tier. Besides common system routines, the application tier contains functions for user control, security policy and multi-user data management. The multi-threaded network interface consists of sockets, which communicate with particular clients. The low-level core encapsulated in the data storage tier works in the same way as in the client application, but it controls multi-user diary management and data synchronization.

6.3.1 Modules

Both applications consist of several modules (units). Most of the modules are common for client and server, possibly they are adjusted for particular application or they contain client as well as server components.

All designed modules are listed in Table 7. Their usage in applications is mentioned in columns *C* (client) and *S* (server). For more details about the implemented modules and their content, see the Budiik's technical documentation on the enclosed CD-ROM.

Module	*C*	*S*	Description
bCrypt	✓	✓	Data encryption and decryption (Blowfish cipher with keys 128b)
bDateEdit	✓	×	Visual controls for date and time edit
bDiary	✓	✓	Core for diary management and data storage
bNet	✓	✓	Network interface (client and server sockets)
bNLS	✓	✓	National Language Support (language localization, feasts, error control)
bServer	×	✓	Server core for controlling the multi-user diary management
bSettings	✓	✓	Application settings stored in Windows Registry
bSync	✓	×	Synchronization with another data sources (diaries, mobile devices)
bView	✓	×	Visual controls for data view (calendar views, lists, tables)

Table 7: Budiik – List of modules.

6.3.2 Data structures

Data management is provided by module **bDiary**. This module defines a class called `TbDiary`, which encapsulates all user data and information and related operations.

The basic data unit in Budiik System is the **data item**, represented by the class `TbItem`. This is only a virtual class for particular data types and it defines basic properties like identification, time stamp, item type, privacy, priority, owner and category. There are several descendants: `TbIActivity` (calendar item), `TbIContact` (contact), `TbINote` (note), `TbIFolder` (folder of contacts or notes) and `TbIMessage` (board message). All items can be collected in **containers** represented by class `TbCollection`, which acts as an item list with sequential or random access. Every diary contains collections of activities, contacts, notes, messages and users (Budiik Server).

The following figure shows a simplified class diagram of the bDiary module with all significant classes:

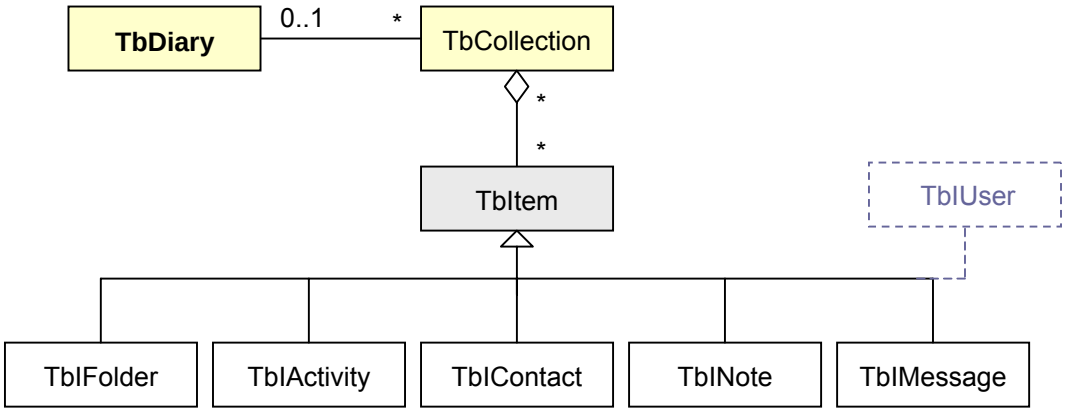


Figure 16: Module bDiary – simplified class diagram.

Beside these fundamental classes, there are a few additional classes for storing various lists of indexes, categories or table columns. Instances of all these classes are also included in `TbDiary`.

6.3.3 Data storage

The main data class `TbDiary` contains all needed algorithms and functions for data storage. Every diary is stored as a set of binary data files in some predetermined directory on the hard drive. The list of all stored data items is represented by one instance of `TbIndexList` class, which is also a member of the diary class.

All the information included in the user diary is divided into index file, data file and settings file, which are placed in the same directory with full (read and write) access. The basic scheme is shown in Figure 17:

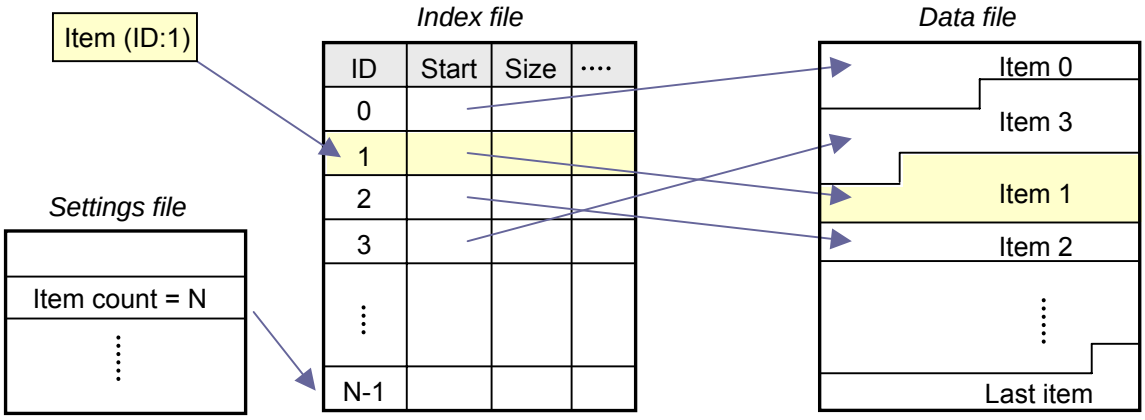


Figure 17: Data storage of user diary.

Index file (*client_diary*.budi, *server_diary*.bsdi)

Every stored item is described by a related index, which determines, where the data are placed (position a size of data blob in the data file) and if they are valid. All indexes have a fixed size and they are stored in the index file, which allows random read/write access – as well as the data file.

Data file (*client_diary*.budd, *server_diary*.bsdd)

This file contains the data items themselves. Because of the effective access, deleted items are not physically removed from the file, but they are only marked as free space. New items are either added to the end of the file or they can be placed into some sufficient free space inside the file.

The format of the data file is very variable and it allows various supplements and extensions, mainly adding new data types and properties. Every property consists of property ID and the value itself. String properties also contain information about string size, as following example shows:

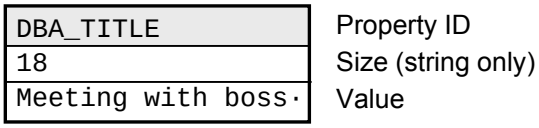


Figure 18: Example of a property (DBA_TITLE) stored in data file.

All the data are still available in the program memory, except large strings, which are loaded on demand – e.g. when the user opens a dialog with the corresponding item. Particular items consist of an item header and a freely ordered sequence of properties. During the reading a data item, all found known properties are processed, unknown properties are skipped and missing properties are initialised by default values. That ensures compatibility between different versions of the data format.

Settings file (*client_diary*.buds, *server_diary*.bsds)

This file stores user settings – window sizes, positions and alignment, colours, fonts, locales, start settings, default values for new items, data summary and things like that. Particular properties are stored in the same way as in the data file, but the difference is that the whole file is read or stored at once, so there is no need of auxiliary indexes.

Every diary is principally multi-user to the intent that it allows to store items from different users. Every item includes information about its owner and original identification in home diary. That is why Budiik Server uses the same data core. Data of all group members (like their synchronized copies) are stored in one diary, which is managed by the server. Unlike many existing client/server systems, this system has a big advantage: it does not need any additional database engine, framework or driver. Of course this approach is not as effective as a database, but in small groups (up to several dozens of members) there is no appreciable speed decrease and no extremely fast and expensive machine is needed for the server application.

Although the shared server-side diary and single user diaries are managed in the same way, there are some differences. The server stores all user data and also own settings (user access rights, activity, start settings etc.) in detached data files in order to set the shared diary and all user diaries apart. Information about users and their settings is stored in form of special data items that are instances of `TbIUser` class (descendants of `TbItem`).

6.3.4 Network communication

Network communication always takes part between the client and the server application. A specialized protocol called **Budiik Network Protocol** (BNP) has been designed for this purpose.

BNP is a binary protocol working over TCP/IP, which is intermediated by system sockets (WinSocks library in MS Windows or Libc in Linux). Synapse, a freeware synchronous TCP/IP library is used for easier work with low-level sockets. The protocol layering is defined as follows:

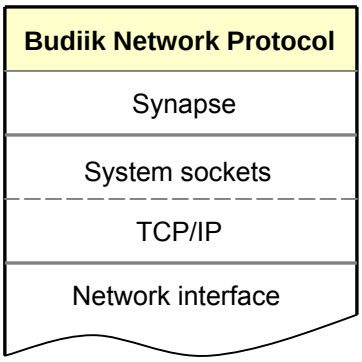


Figure 19: Budiik Network Protocol.

Communication is carried out using binary BNP packets, which contain information about the protocol version, user ID and command, which the receiver has to process (see Figure 20).

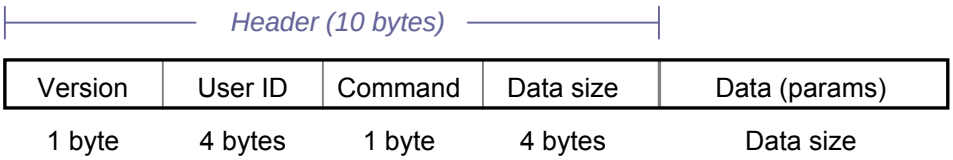


Figure 20: Structure of BNP packet.

Client and server must agree on a protocol version, which is understandable by both sides. User identification is important mainly for the server. In most cases, commands (messages for the receiver) are followed by additional information of fixed or variable size, for instance by data items. Except packets for registration and user login, all data following the header are encrypted.

The protocol contains a complete set of commands for following kinds of communication:

- User management (registration, session control)
- Diary synchronization with the server (general updates, changes)
- Data sharing (distribution of shared data to subscribers)
- Messaging (message board, meetings and group tasks)
- Settings and status information (server info, user lists, access rights, offers of shared data)

The protocol implementation is put in the module **bNet**, which encapsulates the network interface. Appendix A contains more detailed description of the protocol including BNP packet format, list of commands and typical communication sequences between the client and the server.

6.3.5 Security

Despite Budik is designed mainly for single users in home environment, the security issue is very important, because every diary can contain very confidential personal information concerning business, family, friends or health problems. This question is even more essential in the multi-user environment, where particular persons communicate and hand data over unsecured network.

That is why Budiik uses data encryption for both data storage and data transfer via the network. All data encryption and decryption is carried out in the **bCrypt** module, which contains an implementation of synchronous Blowfish algorithm with key size 128b. The algorithm always processes 8-byte data blocks in the ECB (Electronic Code Book) mode without block chaining, encrypted blocks are thus mutually independent and they can be handled separately when working with particular items.

No doubt the first security issue is to protect data files that can become an object of somebody else's interest. Budiik protects all data (except index file, which contain no profitable information)

using the above-mentioned encryption; the key is the user's password. However, it is not stored in data files because of security reasons. The application always attempts to decrypt the data using the entered password and if the decrypted data header is understandable, the password is correct. If any password is set, it is required for diary opening and usually for switching on from the Sleep Mode.

Packets transferred between the client and the server applications via network are encrypted in the same way, but the difference is that a special communication key known by both sides is used.

The biggest security risk bears on client registration. This communication is not encrypted until the client gets the communication key. Client's registration query contains a random secret key, which is used by the server for protection of the real communication key that has to be sent to the client. The problem is that somebody, who is tapping the line, can get the random secret and decrypt the real key, which is used for all further communication. That is why paranoiac clients can obtain the key by another way, for instance via email or physically from the server administrator.

6.3.6 National Language Support

National Language Support (NLS) functions included in the **bNLS** module support the different language-specific and location-specific needs of Budiik users. Language-depended resources are stored in a plain-text form that means Budiik can be easily translated into any language supported by operating system. The default language version is English, which is also selected in case when the system uses an unsupported language. Of course, the Czech version is being supplemented simultaneously. Furthermore, the client application can show various feasts related to certain country or language. Feasts include name days, public holidays, local red-letter days and fetes.

Date and time local settings are used in accordance with the system settings and they include date and time presentation, week order, working days and so on.

6.4 Configuration

Nowadays, Budiik is being developed in the Borland C++ Builder 6.0 IDE and it is being written in the C++ programming language. Following freeware libraries and components are utilized:

Name	Author	Description
FlatBtn	Vít Kovalčík	"Flat button" visual controls
Moon	A. Hörstemeier	Library for computing the moon phases
RXLib	SGB Software	Huge library of visual and non-visual components for common use
Synapse	Lukáš Gebauer	Synchronous TCP/IP library
VirtualTree	Mike Lischke	Visual components for displaying trees, lists and tables

Table 8: List of libraries and components used in Budiik.

The system includes client application `BudiikClient.exe` and server `BudiikServer.exe`, which are stored in the main directory of the system. The hierarchy of directories is defined as follows:

```
Budiik        … main directory with executables
├ doc         … documentation and help
├ nls         … localization
├ reports     … printing reports
└ users       … data files
```

The `users` directory is used for both user diaries and server-managed shared diary. NLS files with localized texts (`lang.*`) and feasts (`feasts.*`) are located in the `nls` directory. App settings are stored in Windows Registry in key `HKEY_CURRENT_USER\Software\76house\Budiik\`.

The last table describes the version handling of the Budiik system with important milestones in future development. In fact, the first partially functional version 1.90a is the objective of this thesis.

Version	Description
1.90a	First alpha-version, 50-60% of declared functionality
1.90b	Beta-version, up to 70% of declared functionality
1.90	First running version, over 70% of declared functionality
2.0a	Alpha-version, at least 90% of declared functionality
2.0b	Beta-version, at least 95% of declared functionality
2.0	Final version, up to 100% of declared functionality

Table 9: Version handling of the Budiik system.

Thus, this listing just presents a basic concept of the project's configuration management. Detailed description of the configuration can be found in the separate file `Budiik.conf.txt` on the enclosed CD-ROM, as well as the Doxygen-generated technical documentation (HTML), which is stored in the `doc` directory (see Appendix B).

Chapter 7

Budiik Client

7.1 Overview

The client application called Budiik Client is principally based on the original program Budík and designed as a compact and easy-to-use application for personal data management. It is intended for all common versions of Microsoft Windows, mainly for Windows 98 and Windows XP, which are the most popular desktop environments nowadays. Figure 21 shows its typical appearance.

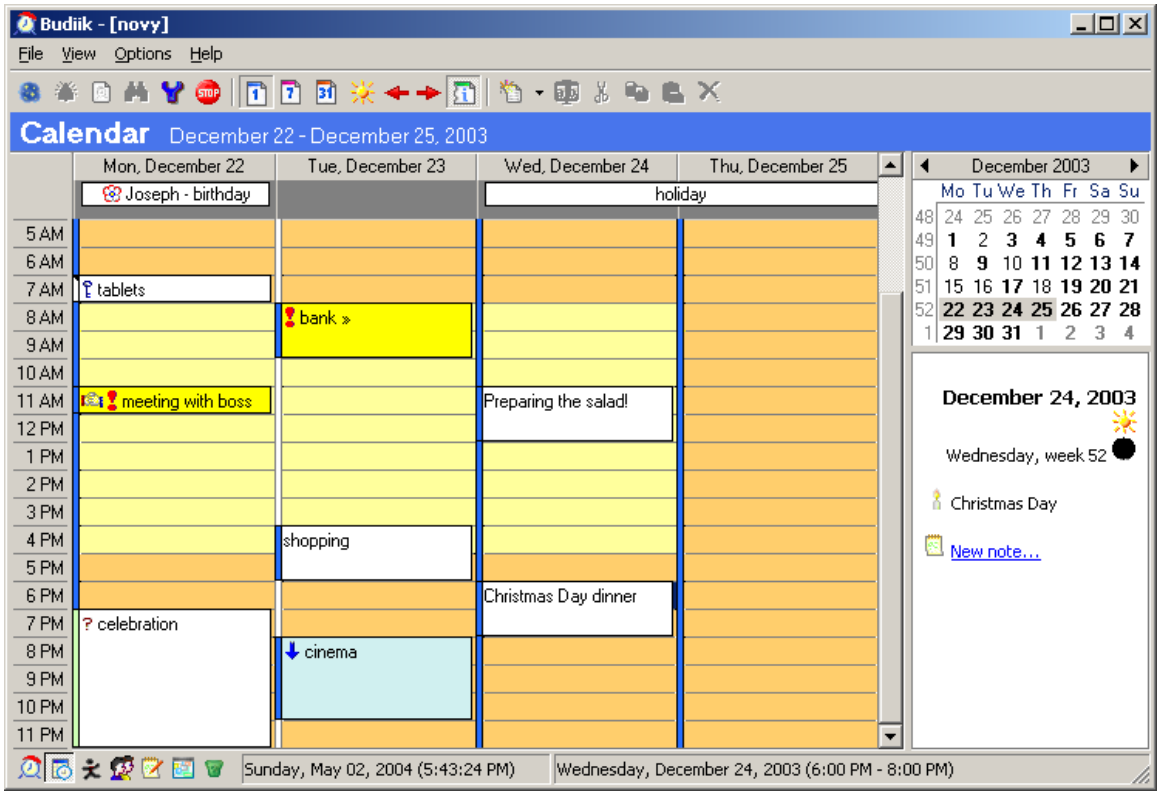


Figure 21: Budiik Client – main window.

From the view of user interface design, Budiik Client is an SDI-based application represented by main window, which contains all toolboxes and views, and by stand-alone user dialogs handling some advanced functions. This common *"all-in-one"* approach is useful mainly for novices that might be confused by lots of tool-windows, pop-ups and messages. The application utilizes both standard components from VCL library and advanced components including several own-designed controls in order to offer transparent, usable and easily configurable graphic user interface.

When a diary has been opened, the main window of the application appears. The top part of the window contains *main menu* and *toolbar*. On the toolbar, there are several buttons for common operations like program exit, settings, searching, reminding or diary selection and further set of page-depended buttons for view settings and item operations. The *status bar*, which is placed in the bottom part of the window, displays various status information and context help. Page selection can be done via the main menu or using *Budiik Bar* – a tensile tool bar containing buttons for particular pages. Budiik Bar can be either full-sized or minimized into the left part of status bar.

In *client area*, there is always a selected view displayed: Budiik Today, Calendar, Activities, Contacts, Notes, Board or Recycle Bin. Every page consists of several composite controls (panes) resizable by splitters. Some panes are optional, so the user can hide them via menu `View`.

Re-usage of the same controls in particular views ensures the application's "*look & feel*" consistency. Information about current window layout including component sizes, positions, alignment, fonts and colours is kept in diary files together with all user settings.

Following sections discuss particular views and functions of the application.

7.2 Budiik Today

Inspired by concurrent systems, this view contains a summary about all currently relevant user data, especially events and activities in near future, reminders, to-dos in progress and overdue items. It is planned as highly customisable, with possibilities of setting the view layout and kinds of showed information.

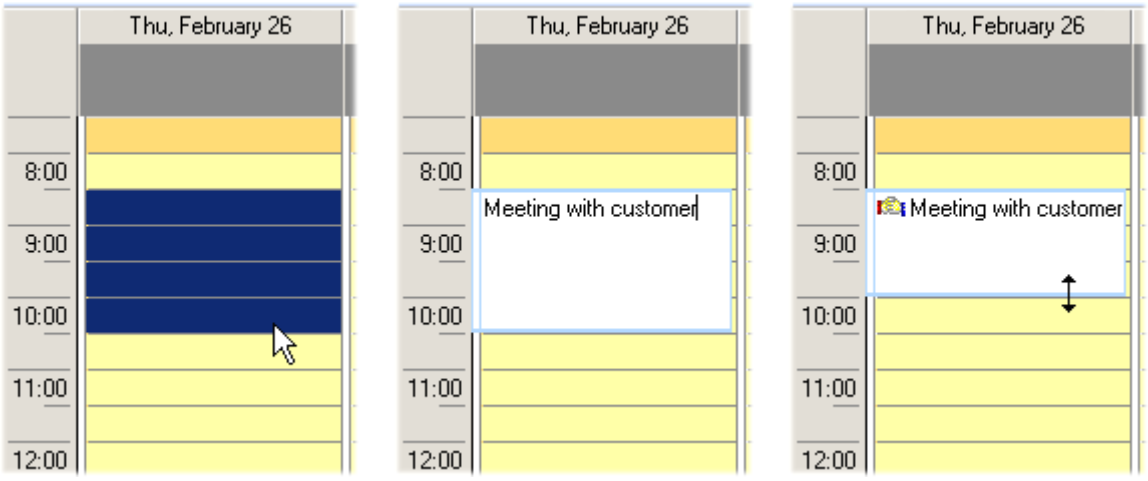
The main goal of this view is to give the user a clue about all current things and matters s/he should not miss of forget (tasks, terms, deadlines, meetings). That is why it is implicitly shown at the start of the application. Particular items work as shortcuts to correspondent places, where the information itself is stored. For example, the user can locate or open an item by clicking on the relevant link name in the view.

7.3 Calendar

Calendar is the most important and the most extensive part of the client application. It offers very specific user interface with wide possibilities of data presentation and user control. For this reason, the prototype application contains a nearly complete calendar implementation in order to present main ideas and principles of usability, which the system is based on.

Budiik Client uses a very classic way of event representation in form of a timetable, which can contain time records. They are represented by floating rectangles occupying a certain place in the

view and they can be created, deleted, selected and directly modified by dragging. This way of item managing is very pleasant for users – creating a new event just by selecting a certain area and putting down some text as the event's title is fast and easy, as following figure shows:



(a) Time selection **(b)** Entering the title **(c)** Adjusting the time

Figure 22: Budiik Client – creating a new event in calendar.

The user can create and manage several basic types of time items:

- Action – classic simple or repeated event with fixed begin and end
- Meeting – action with assigned people, with possibilities of group organization (invitation)
- Anniversary – annually repeated all-day event (feast, birthday)
- Remind – very simple time event without duration, intended just for reminding
- Task – computer-related event for batch execution at certain time (backup, anti-virus)
- To-do – working activity with or without a deadline, with possibility of teamwork

Beside basic properties like begin, end, title, notes, priority and privacy, all time events include common properties concerning reminding, recurrence and additional properties depending on the item type like a list of invited people (meeting) or working progress (to-do).

Calendar also offers various possibilities of time representation in form of day view, week view and month view. All the views enable both direct and indirect item manipulation and an easy switch to the same time moment in another view.

Day view

This basic view displays a certain number (1 to 7) of consecutive days. Every day is represented as a stripe split up to rows, where the first row creates a space for all-day events and all the remaining

rows represent time intervals within the day. Working and non-working time is distinguished by background colour. The user can select one or more rows using mouse or cursor keys to define a certain time for a new event or to smoothly change the selected day. Floating rectangles, which contain the time events, can be selected and dragged by the mouse.

Week view

Week calendar is displayed in the same way but the difference is that the view is set to seven days with fixed day order. That means the first column is also the first day of the week – typically Monday (according to ISO standard [I8601]) or Sunday in Anglophone countries.

Month view

In order to preserve the GUI's consistence, month calendar offers very similar look and control. Here, the entire view consists of a table with dimensions `Number_Of_Weeks` (2 to 6 rows) x `Days_In_Week` (7 columns), where particular days are represented by single cells. All-day activities are displayed together with time-defined events and the user can manipulate them in the same way as in the day view. The user can also go smoothly to previous or following weeks.

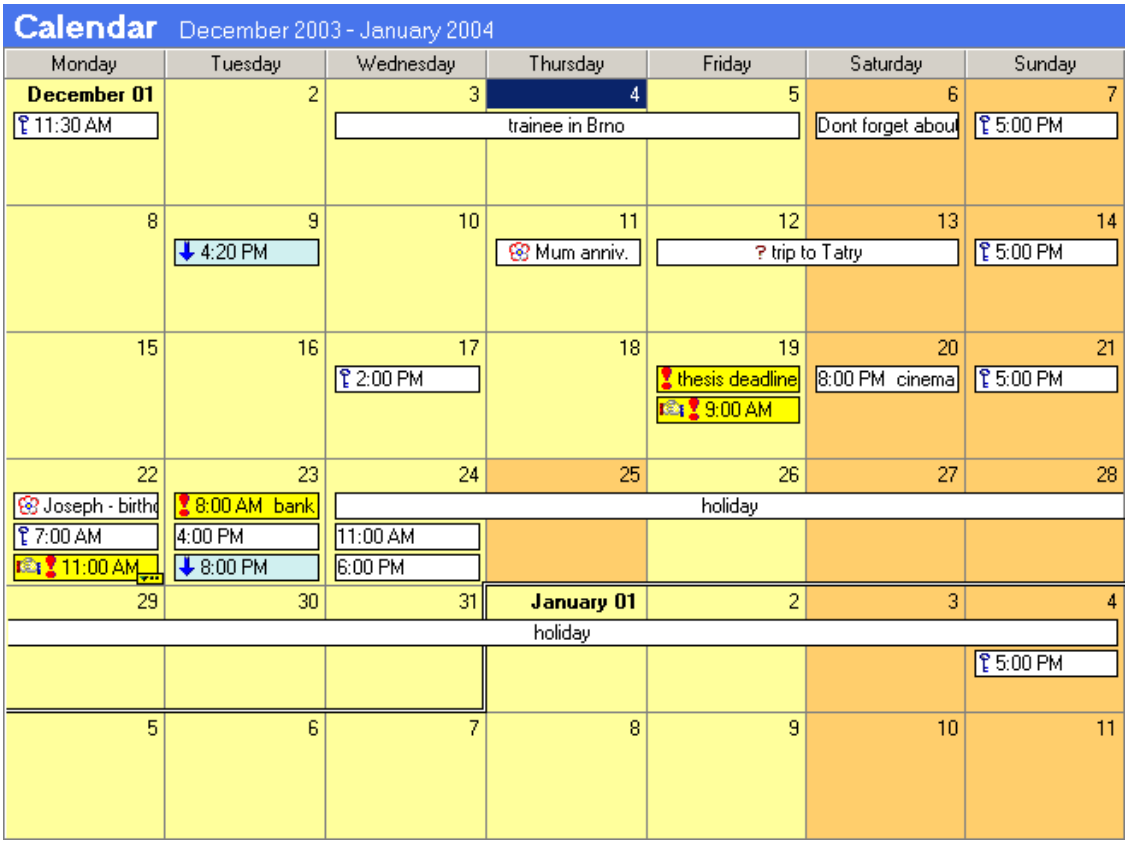


Figure 23: Budiik Client – Month view.

Furthermore, all calendar views include a resizable **right-side panel**, which displays a small month calendar and a bottom-placed information area.

This month calendar consists of one or more (up to 12) months and it allows the user to change the view (days – weeks – months) very quickly just by selecting an appropriate time interval. That is why I decided not to include a stand-alone year view into Budiik Client, because this is fully represented by this side-panel calendar.

Information area displays a short summary of the currently selected day like the date itself, day name in week, week number, moon phase, day notes and selected list of feasts and current to-dos. To prevent any obstructions during quick date-selection changes, drawing of the day information is a little bit delayed, as well as emphasis of days containing items in the month calendar. It is worth mentioning here that the client application implements a simple mechanism of event-lists caching, because otherwise the constant computation of event lists for every day of the view would be too time-consuming, especially in case of big amount of recurrent events.

All the language-specific and location-specific time information is represented in accordance with current system settings, that means all date and time values are displayed in the same way as the operating system displays.

7.4 Activities

Technically, both actions and to-dos are time events with many common aspects and properties. The problem is: how to represent to-dos in the calendar? We can mark deadlines and milestones, but we cannot afford to overfill the calendar sheet by all current to-dos and long-term duties, especially if they have no specified time span.

In practice, all today's PIM systems offer a special view, which is intended only for to-dos; usually it is a table or an item list. Then, it can be distilled into the calendar view in form of various side panels in order to keep all the time-oriented data together. Such a table/list view offers wide possibilities of searching, sorting and categorization, so it can be very useful for instance when the user is looking for some older items or items with similar properties (date, keyword, category…).

Budiik Client enhances this run-in concept for all calendar items to provide as comfortable time-oriented data management as possible. A view called *"Activities"* serves for this purpose. Here, all calendar items are shown as an item list and they can be sorted and filtered according to various aspects such as item type (action, to-do, remind…), priority, privacy, date, occurrence, category, contained text and so on.

Working activities (to-dos) are also characterized by their present status: they can be "current" (all to-dos in progress or without a specified time span), future, overdue or completed. They can also get additional status within a working group (assigned, accepted, refused, partially completed).

In order to save space in the memory and in the diary file, every series of recurrent events is handled as one item with undifferentiated properties for all occurrences. In this form it is also shown in the item list. Any changed occurrence is excluded from the series, but it stays connected to the mother item thanks to a bi-directional reference. It follows that the user can revert all made changes without any problems. In this case, all excluded items will be just removed.

7.5 Contacts

Budiik Client uses the well-tried concept of contact management from the original Budík, where all items are placed in folders. Here, the folders can be structured in a multi-level hierarchy; this provides (together with categories) wide possibilities of item sorting and classification.

Every contact contains a set of required properties for basic identification followed by various optional properties concerning residence, occupation, contact information, personal information and possible user-defined properties. The most common properties are summarized in Table 10.

Category	Common properties
Identification	Type (person/group), first name, surname/group name, nickname, titles
Residence	Address (street, municipality, zip code, country, state)
Occupation	Profession, company, branch, office, role, address, contact
Contact information	Phone, cellular, fax, email, web, IM contact… (Personal/business)
Personal information	Sex, birthday, relationship, relatives
User-defined	Assigned meetings, notes and various user-defined properties

Table 10: Common properties of contact items.

Page called *"Contacts"* uses a three-pane view for contact-database management. The left pane contains the folder tree and the right pane shows the content of a selected folder. Here, the items of the folder can be displayed either as a collection of icons, or as an item list. An optional right-bottom pane acts as a preview of a selected item. The view also allows showing all items at once as an item list without folders; this is called *"flat view"*.

The user can create new items or edit existing items in a dialog window. The other operations like dragging the item to another folder, deleting and clipboard operations can be performed directly in the basic view.

Of course, the contact database would be of no avail without essential functions like printing, sorting and searching. Another features relate to meetings and to-dos. Particular people or groups can be tagged as members of a meeting or they can take part in some team-oriented working activity. Then, the user can easily get the list of all meetings and to-dos related to a certain contact. Furthermore, birthdays can be inserted into the calendar as regular anniversaries. The program also watches the links between calendar items and contact items and warns about their disconnection.

Every diary implicitly contains one contact item with information about the owner of the diary. This contact is accessible via the dialog window *File – Properties – Personal information* and is useful mainly for storage of the user's personal information like addresses, accounts and phone numbers. This information is important mainly within work groups, because the server utilizes this information for the user's identification.

7.6 Notes

The concept of the folders and items hierarchy is used for notes as well. The left part of the view displays the folder tree, the right part contains the content of the selected folder and an optional third pane shows a preview of the currently selected item. Items can be created and edited in a dialog window; all other operations can be done directly in the view. Unlike contact items, notes include only basic properties like priority, privacy, title and naturally the *"notes"* property itself. This property can contain either plain text or formatted text (RTF) with a possibility of loading and saving to a file.

On the basis of many users' requests, now it is possible to add so-called "day notes" to any date in the calendar. This allows users keeping some kind of journal. These items are stored in a special folder (*Day notes*) and they are implicitly sorted according to the date they are assigned to.

7.7 Board

This page is functional only if the user has joined a user group. It serves as a message board for all users within the group. Particular users can add, modify and delete their own messages and read the messages from other users, in accordance with server's access policy settings.

Messages are very similar to notes, but the difference is they have a property *"receivers"*; this means they are intended either for certain selected users or for users of the group. The view itself is organized as an item list with an optional bottom pane for the preview of the selected item. Because board messages are handled and synchronized the same way as all other items, they are always accessible in a local diary, without regard to availability of network connection. Also the entire item management is consistent with the above-discussed views.

7.8 Recycle Bin

There are two common ways to remove items: physical deletion or transfer to some specific place. For this purpose, the client application offers a special folder called *"Recycle Bin"*, which works in the same way as the ordinary Recycle Bin in MS Windows. Items that have been placed into this place can be either restored (moved to original place), or physically deleted from the diary.

The view itself consists of a trashed items list and it provides basic item management.

7.9 Functions

Any PIM software would not be successful without a set of essential functions that significantly extend its usability. These include clipboard utilization, searching, sorting, import/export, printing, user-customized program settings and understandable help.

Sleep Mode is one of the most important Budiik's functions. It allows minimizing the application to stand-by mode. All unnecessary memory resources are released before switching. In this mode, the application is hidden and quietly waiting in the background. It is only accessible via

the *tray icon* in the system area of the Main panel. Then, the user can restore the application directly by a click on the icon or using an appropriate pop-up menu. This is shown in Figure 24.

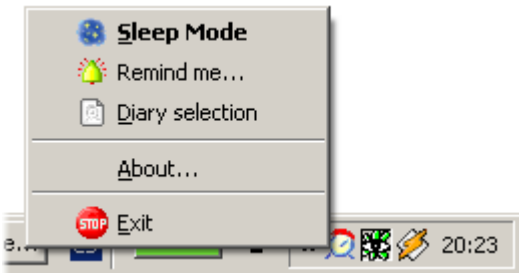


Figure 24: Budiik Client – tray icon.

Furthermore, developmental plan of Budiik takes these following functions and enhancements into account:

- **Clipboard operations** (cut, copy & paste)
- **Sorting, filtering** and **searching** item views according to the given criteria (type, keyword, date, contained text, owner, category)
- **Reminding** calendar events and deadlines
- **Import** and **export** for various data exchange formats (vCalendar, vCard, CVS, rich text)
- **Printing** selected items or entire views (calendars, to-do lists, address labels, summaries)
- Easy-accessible **program settings** (item handling, defaults, fonts, colours, app behaviour)
- **Bookmarks** for frequently used items
- Context **help**

Naturally, this *"list of features"* is only a concept and groundwork for final implementation. New features might be added or some stated ones might be considered useless – as well as email support, which was on this list for a long time until I came to the conclusion it was more reasonable to concentrate on other, more important parts of the system.

Chapter 8

Budiik Server

8.1 Overview

Because the server application has not been implemented yet, this chapter contains only a rough design of this part of the system and furthermore it discusses the basic ideas and principles of multi-user diary management and work within groups provided by Budiik System.

It is necessary to mention that this client-server system has been designed just for synchronous communication between the client application (Budiik Client) and the server program (Budiik Server) in order to ensure mainly synchronization of user-oriented data and their intermediation within a user group. That is why this system does not support direct peer-to-peer connection or any kind of instant messaging.

Budiik Server is supposed to be a simple application running on the background and waiting for incoming client connections. For this purpose, it uses a server socket listening on a certain TCP port. Furthermore, it keeps its own server-side diary open. Beside all synchronized data of client users, this diary contains the user list and server settings (access policy, network configuration).

As we have already mentioned, both the client and the server use BNP for bi-directional communication (also see Appendix A). When a client application attempts to connect to the server, at first it has to prove its identification. If the server has accepted the client's connection request, it creates a new thread for further communication with the client. The most common situation is that a client asks for data synchronization. In this case, the server evaluates differences between both sides and determines next steps for exchanging newer versions of the data. The client usually sends its own data and receives shared data from the others. It can either close the connection (in case of dial-up connection), or it can stay connected by repeated sending KEEP_ALIVE messages. In local networks or in case of permanent connection, the client stays connected for the entire program runtime and it calls for synchronization in regular time intervals. However, this is upon the client, because it always starts and finishes the communication. It follows that the server application should be multi-threaded to ensure simultaneous user connections.

There are several technical issues left to consider. One of them is the design of Budiik Server as an NT service. Such a service application can serve well in modern multi-user systems (Windows NT/2000/XP and newer), where it can run independently of switching between the user accounts, but unfortunately it does not work in older Windows 95/98, which are still used by many users.

8.2 Server Administration

Budiik Server offers a very simple user interface for administration purposes. Nevertheless, the interface should be robust and easy-to-use enough, because mainly in home environment it may be administrated by inexperienced users, as well.

Like in the client application, the main window is accessible via a tray icon placed on the Main panel. It consists of main menu, tool bar with the most used commands, status bar and client area for currently selected view. For now, the application is supposed to offer following views:

Summary

This view displays general information about the group (logged users, new users and awaiting for authorization), current status of the server and log file containing recent connections, activities and errors. Here, the administrator can see if everything is working well or not.

Access policy

If we have two or three computers connected to a home network, we do not have to worry about any strict privacy settings (if our children are not very curious…). However, in every bigger group it is necessary to tell the rules. That is why the server application makes it possible to set different levels of privacy restrictions.

Thus, the server can be either *"open"* for anybody, or *"restricted"* to a certain group of people. Permanently disturbing users can be moved to Black list, so they may be no more allowed to create new items in others' diaries or to send messages to the message board, or they can be even removed from the group without any possibility of further registration.

Users

According to the access policy settings, users can either sign in automatically, or they have to ask for authorization by the administrator. The second approach is advised mainly if the server is accessible from the Internet. Here, the administrator can manage incoming authorization requests. The administrator can also add new users manually and/or deliver them information about their registration (server address, login, communication key) by email or personally. This can facilitate application of the system in corporate network.

Configuration

The server configuration includes following settings:

- Server behaviour – server start settings, activity monitoring, turning on/off the service
- Diary management – server-side diary storage, backup and restore
- Network settings – port number, proxy settings, timeouts

All server settings are stored in the server-side diary. General information about diary location and start-up settings are stored in Windows Registry.

8.3 Working in Groups

8.3.1 User groups

Every Budiik Server can store data of one or more signed clients. Generally, the users may not know one another, especially it the server is public and accessible to anybody. We can assume these users use the server only for storage and synchronization of their own data. However, some users or all of them can be relatives, friends or co-workers. They usually want to interact with the others and to have access to the others' data. In this case we can talk about real user groups.

In fact, the server application can manage only one group at a time. All users can theoretically see the others' data or they can have access to the same message board, which is common for the entire group. However, everybody can individually restrict the access to his/her data for particular users of the group, which means there might exist some pre-arranged subgroups within the entire big group.

8.3.2 Data sharing and synchronization

The data-sharing mechanism is based on following principle, which is also displayed at Figure 25:

1) User **X** stores selected part of own diary data to the server (synchronization).
2) Then, s/he sets a certain part of this synchronized data as *"shared"* for the others.
3) The other users are informed by the server about this offer of shared data **X**.
4) These users can subscribe for some parts of this offer.
5) Appropriate data are stored in subscribers' diaries during the synchronizations of the diary.

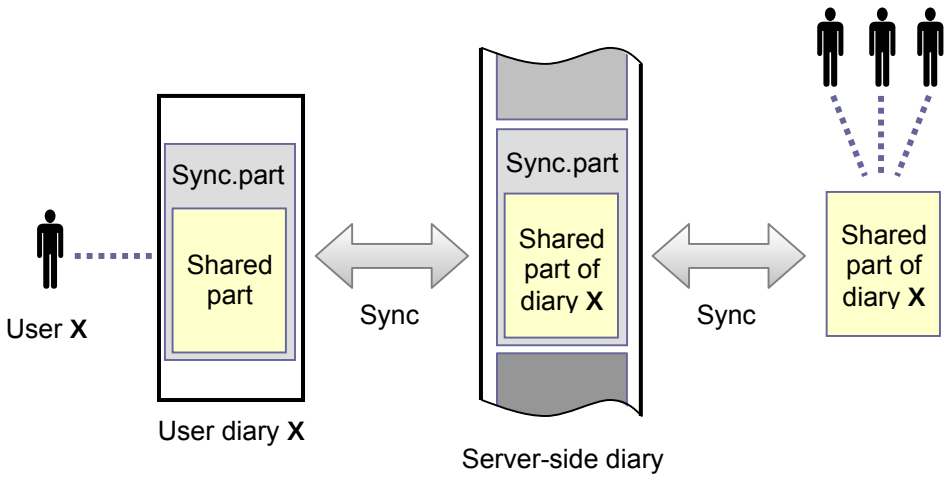


Figure 25: Data sharing and synchronization.

In fact, the server is responsible for extracting all shared data from user diaries and for the distribution of these data to appropriate places in accordance with available information about data offers and subscriptions. It follows that the server determines who is allowed to get the data.

And how do the shared data appear in the others' client applications?

Beside the message board, which is shared implicitly, there are three other kinds of shared data – calendar events, contact items and notes including folders. In calendar views, the subscriber has to select, whose calendar is to be displayed, because there can be only one calendar displayed at once. For this purpose, there is a special combo-box placed on the tool bar; this control contains a list of available calendars. The same goes for the *"Activities"* view.

On the contrary, all contacts and notes are displayed at once. Respectively, the folder-tree pane in these views contains separated folder trees of all available users. Root items of the trees are named according to the users, owner's root items are called *"Local contacts"* and *"Local notes"*.

8.3.3 Collaborative work

One of the advantages of data sharing within the group is the possibility of teamwork. Because a person, who is organizing a meeting, can see free time information of the others, s/he can adjust the date of the meeting to suit all assigned members. Furthermore, s/he can easily send an invitation to the members and get their responses (*"I will come."* / *"I won't come…"* / *"I don't know yet."*).

Budiik also supports team-oriented working activities. Every to-do can be assigned to one or more members of the group. Then, the author gets a feedback about the progress from particular workers. The to-do is marked as finished when all workers have finished their part.

Conclusion

The first objective of this thesis was to explore the area of Personal Information Management and to analyse all available tools and facilities. During this work I discovered that this area was extremely extensive, rapidly developing and very labyrinthine too. It is not very easy to study and explore it, because there are not many comprehensive handbooks or compendiums, which would discuss this area as a whole. The dissertation work by Leysia Ann Palen [Pal98] concerning the area of multi-user information management is a rare exception.

The key results follow from the performed analysis, which examines and describes the most important aspects of this area, related especially to time management issues, definition and classification of personal data and various ways of personal data management. As Chapter 3 shows, nowadays there are many hardware and software PIM systems available. All of them offer higher or lower level of usability, functionality and user friendliness, and surprisingly – not in all cases the better systems are more expensive. Some systems are really innovative, but they are not always as successful as other ones, which just copy well-tried approaches. Many not very good clones of MS Outlook give a nice example.

Another outcome of the analysis says that only a well-designed and sufficiently functional user interface can ensure the usability of the system, as well as its commercial success in keen competition. That is why significant part of the thesis deals with user interface and its design. This issue has also been seriously considered during the design of the new system.

Nevertheless, the most important finding is that there is a large group of users from Home & Office area, who miss a really suitable solution for managing their personal data. The thesis offers such a solution: it proposes a simple PIM software system called Budiik, which attempts to bridge the gap between tiny utilities with limited functionality and expensive corporate solutions.

Design and prototype implementation of this system was the second objective of the thesis. During the work I was following the basic requirements I had acquired from users of the previous version called Budík. They put strong emphasis especially on sufficient functionality together with simple and intuitive user interface. I learned that sometimes it was difficult to harmonize these two requirements and to find a trade-off, which would be optimal for majority of users.

Because the project time was limited, I found it impossible to finish the implementation of the whole system. So after completion of the low-level layers and modules providing data management and system functions I focused on the calendar part of the client application. This is the most extensive part of the system and it presents the main ideas and principles of usability the system is based on.

Despite the results of the analysis and the presented solution itself are considered to be the most significant contribution of this thesis, in fact they just form up a point of departure for future development. The developmental plan includes completion of the client application and entire implementation of the server application, communication layer and multi-user support.

Reference

[All86] Allen, Jane Elizabeth: *Beyond Time Management: Organizing the Organization*. Reading (MA), Addison-Wesley 1986. ISBN 0201157934.

[Bae92] Baecker, Ronald: *Readings in Groupware and Computer-Supported Cooperative Work: Assisting Human-Human Collaboration*. San Francisco (CA), Morgan Kaufmann 1992. ISBN 1558602410.

[BBN03] Bergman, O., Beyth-Marom, R., Nachmias, R.: The User-Subjective Approach to Personal Information Management Systems. *Journal of the American Society for Information Science and Technology*, 2003, vol. 54, no. 9, pp. 872-878.

[BDH+03] Bellotti, V., Ducheneaut, N., Howard, M., Smith, I.: Taking email to task: the design and evaluation of a task management centered email tool. In *Proceedings of the conference on Human factors in computing systems April 5-10, 2003 Ft. Lauderdale, Florida, USA*. New York, ACM Press 2003. pp. 345-352.

[BW92] Banks, W., Weimer, J.: *Effective Computer Display Design*. Englewood Cliffs, Prentice Hall 1992. ISBN 0-13-401-027-2.

[Cau00] Caunt, John: *Organise Yourself*. London, Kogan Page Ltd. 2000. ISBN 0749432616.

[Cun00] Cuny, Janice: *Time Management and Family Issues*, September 12, 2000. Document is available on URL http://www.cra.org/Activities/craw/projects/mentoring/mentorWrkshp/time_manage.pdf (February 29, 2004).

[CW93] Cox, K., Walker, D.: *User-Interface Design*. New York, Prentice Hall 1993. ISBN 0-13-952888-1.

[DCC+03] Dumais, S., Cutrell, E., Cadiz, J.J., Jancke, G., Sarin, R., Robbins, D.C.: Human interaction: Stuff I've seen: a system for personal information retrieval and re-use. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval July 28-August 1, 2003 Toronto, Canada.* New York, ACM Press 2003. pp. 72-79.

[DS98] Dawson, F., Stenerson, D.: *RFC 2445 - Internet Calendaring and Scheduling Core Object Specification (iCalendar)*, November 1998. Document is available on URL ftp://ftp.isi.edu/in-notes/rfc2445.txt (March 1, 2004).

[Ebe94] Eberts, Ray E.: *User Interface Design*. New York, Prentice Hall 1994. ISBN 0-13-140328-1.

[GMN02] Goecks, J., Mynatt, E.D., Nguyen, D.H.: Augmenting shared personal calendars. In *Proceedings of the 15th annual ACM symposium on User interface software and technology October 27-30, Paris, France.* New York, ACM Press 2002. pp. 11-20.

[Gru92] Gruber, David: *Osobní informační systém (Personal Information System)*. Ostrava, Gruber-TDP 1992. ISBN 80-85624-03-6.

[Hay01] Haynes, Marion E.: *Personal Time Management, 3rd ed*. Menlo Park, CrispLearning 2001. ISBN 1-56052-585-1.

[HSD98] Howes, T., Smith, M., Dawson, F.: *RFC 2425 - A MIME Content-Type for Directory Information*, September 1998. Document is available on URL ftp://ftp.isi.edu/in-notes/rfc2425.txt (March 1, 2004).

[I8601] *ISO 8601: 2000 (E). Data elements and interchange formats – Information interchange – Representation of dates and times*. Geneva, International Organization for Standardization 2000.

[Kar95] Karash, Richard: *Groupware and Organizational Learning*, 1995. Document is available on URL http://world.std.com/~rkarash/GW-OL (March 1, 2004).

[Kuh01] Kuhn, Markus: *A Summary of the International Standard Date and Time Notation (International Standard ISO 8601)*, November 10, 2001. Document is available on URL http://www.cl.cam.ac.uk/~mgk25/iso-time.html (March 1, 2004).

[Lau90] Laurel, Brenda: *The Art of Human-Computer Interface Design*. Boston (MA), Addison-Wesley 1990. ISBN 0-201-51797-3.

[Lin03] Lineback, Nathan: *Graphical User Interface Gallery*, 2003. Document is available on URL http://www.toastytech.com/guis/ (February 28, 2004).

[Mar92] Marcus, Aaron: *Graphic Design for Electronic Documents and User Interfaces.* Reading (MA), Addison-Wesley Publishing Co. 1992. ISBN 0-201-54363-9.

[Mar02] Marcus, Aaron: *Return on Investment for Usable User-Interface Design: Examples and Statistics*, February 28, 2002. Document is available on URL http://www.amanda.com/resources/ROI/AMA_ROIWhitePaper_28Feb02.pdf (February 28, 2004).

[Mar03] Marek, Martin: *Functional part of Personal Information Management System (school-year project)*, May 8, 2003. Document is available on URL http://76house.wz.cz/files/RP.xmarek21.en.pdf (March 1, 2004).

[Mah01] Mahmoud, Qusay: *Learning Wireless Java*. London, O'Reilly 2001. ISBN 0-596-00243-2.

[MPB92] Marca, D., Prasad, B.E., Bock, G.E.: *Groupware: Software for Computer-Supported Cooperative Work (IEEE Computer Society Press Tutorial)*. Los Alamitos, IEEE Computer Society 1992. ISBN 0-8186-2637-2.

[NWI+02] Nardi, B. A., Whittaker, S., Isaacs, E., Creech, M., Johnson, J., Hainsworth, J.: Integrating communication and information through contactmap. *Communications of the ACM*, 2002, vol. 45, no. 4, pp. 89–95.

[Pac94] Pacovský, Petr: *Velká kniha o uspořádání času (Big Book about Time Order)*. Praha, IDG Czechoslovakia 1994. ISBN 80-900872-7-2.

[Pal98] Palen, Leysia Ann: *Calendars on the New Frontier: Challenges of Groupware Technology (dissertation)*, 1998. Document is available on URL http://www.cs.colorado.edu/~palen/dissertation/LPdissertation.pdf (March 6, 2004).

[Per03] Perlman, Garry: *Suggested Readings in Human-Computer Interaction (HCI), User Interface (UI) Development, & Human Factors (HF)*, January 27, 2003. Document is available on URL http://www.hcibib.org/readings.html (March 1, 2004).

[Rat95] Rathbone, Andy: *About Face: Essentials of Window Interface Design*. New York, John Wiley & Sons 1995. ISBN 1568843224.

[Rau93] Rauterberg, M.: *Quantitative Measures for Evaluating Human-Computer Interfaces*, August 8, 1993. Document is available on URL http://www.ipo.tue.nl/homepages/mrauterb/publications/HCI93paper.pdf (March 1, 2004).

[Sal97] Salvendy, Gavriel: *Handbook of Human Factors (2nd edition)*. New York, John Wiley & Sons 1997. ISBN 0-471-11690-4.

[Tog92] Tognazzini, Bruce: *Tog on Interface*. Reading (MA), Addison-Wesley 1992. ISBN 0-201-60842-1.

[Tro95] Trower, Tandy: *The Windows Interface Guidelines for Software Design*. Redmond (WA), Microsoft Press 1995. ISBN 1-55615-679-0.

[Tøn03] Tøndering, Claus: *Frequently Asked Questions about Calendars*, June 24, 2003. Document is available on URL http://www.tondering.dk/claus/cal/calendar26.html (February 28, 2004).

[Ver03] Verbeek, Fons J.: *Human Computer Interaction*, 2003. Document is available on URL http://www.liacs.nl/~fverbeek/courses/hci/hci2003-5.pdf (February 28, 2004).

[Wod93] Von Wodtke, Mark: *Mind Over Media*. New York, McGraw-Hill 1993. ISBN 0-07-067633.

Appendix

Appendix A: Budiik Network Protocol Reference

This appendix contains a rough draft of **Budiik Network Protocol version 1** (BNPv1), mentioned in section 6.3.4. It is a protocol based on TCP/IP and intended for communication between Budiik Client and Budiik Server application within a user group.

Structure of BNP packet

Property	*Size (B)*
Header	10
Data (params)	Data size

Structure of BNP packet Header

Property	*Size (B)*
Version	1
UserID	4
Command	1
Data size	4

Commands from client

Command	*Param name*	*Param size*	*Param description*
REG (01h)	NameLen	4	
Client asks for creating a new account on the server	Name	NameLen	Login (diary name) to registrate
	PwdLen	4	
(not crypted)	Pwd	PwdLen	Random secret for key encryption
	ClientIP	4	
	ClientPort	2	
	EmailLen	4	
	Email	EmailLen	Email for key sending
	MsgLen	4	
	Msg	MsgLen	Authorization message (if needed)

Command	*Param name*	*Param size*	*Param description*
GET_REG_STATUS (02h) Client awaiting authorization asks for current status *(not crypted)*	NameLen Name PwdLen Pwd ClientIP ClientPort	4 NameLen 4 PwdLen 4 2	 Login (diary name) to registrate Random secret for key encryption
KEY_QUERY (03h) Client asks for sending the communication key via email *(not crypted)*	NameLen Name ClientIP ClientPort	4 NameLen 4 2	 Login (diary name) to registrate
UNREG (04h) Client wants to unregistrate from the server and to clear all synchronized data	KeyLen Key	4 KeyLen	 Key for authorisation
LOGIN (05h) Client attempts to login to the server	ClientIP ClientPort KeyLen Key	4 2 4 KeyLen	 Key for authorisation
LOGOUT (06h) Client logs off server	*No params*		
KEEP_ALIVE (07h) Client refreshes the connection to the server	*No params*		
GET_SERV_INFO (08h) Client asks for summary info about the server	*No params*		
GET_SERV_TIME (09h) Client asks for current server time in order to get time offset	*No params*		
GET_USERS_INFO (0ah) Client asks for info about users of the server	*No params*		
OFFER (0bh) Client sends information about shared data	CalAccess ContAccess NotesAccess	1 1 1	Information about calendar sharing Information about contacts sharing Information about notes sharing (None/Partial/Read/Write/Full)

Command	*Param name*	*Param size*	*Param description*
SUBSCRIBE (0ch) Client subscribes for receiving shared data from other users	GroupType UserCount UserID[1] RecCalendar[1] RecContacts[1] RecNotes[1] …	1 4 4 1 1 1	None/ListOfUsers/All 0 for None, 1 for All 0 for All Yes/No Yes/No Yes/No
GET_SYNC_LIST (0dh) Client asks for data synchronization and sends info about its own items and made changes	SyncType ItemCount ItemID[1] LastChange[1] ChangeType[1] …	1 4 4 8 1	All/ChangesOnly List with information about items Item identification Server-adjusted time New/Header/Light/Size/Deleted
SYNC_DATA_C2S (0eh) Data for server synchronization	ItemCount ItemID[1] ChangeType[1] DataSize[1] Data[1] …	4 4 1 4 DataSize[1]	 Item identification New/Header/Light/Size/Deleted Size of sent item data blob Required part of item data blob
SYNC_SETTINGS (0fh) Client queries where newer version of diary settings is	LastChange	8	Time of client's last change
SYNC_SET_C2S (10h) Client sends its settings to synchronize with server	SetSize SetData	4 SetSize	

Commands from the server

Command	*Param name*	*Param size*	*Param description*
REG_REPLY (A1h) Server accepts or rejects the registration of a new user *(not crypted)*	Result KeyLen Key MsgLen Msg	1 4 KeyLen 4 MsgLen	Ok/Await/UserExists/Refused 0 = key was sent via email Key crypted by user's random secret Message from administrator
LOGIN_REPLY (A2h) Server accepts or rejects the user's query	Result	1	Ok/WrongKey/UnknownUser/Error

Command	*Param name*	*Param size*	*Param description*
ACK (A3h) Universal response for client queries	Result	1	Yes/No
SERVER_INFO (A4h) Information about the server	NameLen Name Public MsgLen Msg AliveTimeout ResyncInterval	4 1 4 MsgLen 8 8	 Server diary name Public/dedicated for certain users Additional message for users Timeout for automatical client logout Minimum time between resync
SERVER_TIME (A5h) Actual server time	ServerTime	8	
USERS_INFO (A6h) Information about registered users of the server	UserCount UserID[1] NameLen[1] Name[1] UserStatus[1] CalAccess[1] ContAccess[1] NotesAccess[1] …	4 4 4 NameLen[1] 1 1 1 1	Number of users User identification User login name Online/Offline Information about calendar sharing Information about contacts sharing Information about notes sharing (None/Partial/Read/Write/Full)
SYNC_LIST (A7h) Server replies the client's sync query by sending an IDList of required newer client data	ItemCount ItemID[1] ItemPart[1] …	4 4 1	 Item identification Header/Light/Whole
SYNC_DATA_S2C (A8h) Data for server synchronization	ItemCount ItemID[1] ChangeType[1] DataSize[1] Data[1] …	4 4 1 4 DataSize[1]	 Item identification New/Header/Light/Size/Deleted Size of sent item data blob Required part of item data blob
SYNC_SET_S2C (A9h) Server sends client's settings stored on the server or asks for a newer version	NewOnLocal SetSize SetData	1 4 SetSize	True = client has newer version 0 if NewOnLocal = true No data if NewOnLocal = true

All the communication is synchronous, that means the client always sends a query and the server answers. Here is an overview of typical client-server communication sequences:

Client registration

- Client sends **REG**, the server answers **REG_REPLY**. In case of waiting for authorization, client can ask **GET_REG_STATUS** later and server answers **REG_REPLY** again.
- If needed, client asks for the communication key (**KEY_QUERY**) again; server sends the key to the formerly given email address and replies **ACK**.
- Client can clear the registration by sending **UNREG**, server answers **ACK**.

Client session

- Client sends **LOGIN** and server replies **LOGIN_REPLY** (Result = Ok means that client is logged now).
- Client has to send **KEEP_ALIVE** message in defined intervals in order to keep the present connection with server; otherwise it will be logged off.
- When closing the session, client sends **LOGOUT** and server answers **ACK**.

Status information

- Client sends **GET_SERV_INFO** to get info about server, server sends **SERVER_INFO**.
- Client sends **GET_SERV_TIME** to get time offset between client and server, server sends **SERVER_TIME** containing the actual time.
- Client sends **GET_USERS_INFO** to get the actual user list, server sends **USERS_INFO**.

Settings

- Client offers its own data using **OFFER** message, server accepts it by sending **ACK**.
- Client sends **SUBSCRIBE** to subscribe the shared data provided by other users, server replies by sending **ACK**.

Data sharing and synchronization

- Client asks for synchronization of data items by sending **GET_SYNC_LIST** with changes made in a local diary, server compares this information with its own version of the data, determines how to make the synchronization and sends **SYNC_LIST** asking the client for newer data items. Client sends **SYNC_DATA_C2S** with all newer data items and gets **SYNC_DATA_S2C** with all newer items from the server (swap).
- Client asks for synchronization of diary settings by sending **SYNC_SETTINGS**, server sends **SYNC_SET_S2C** as a reply, which can contain the settings. In case there is older version of diary settings on the server, client sends **SYNC_SET_C2S** with the newer version and server answers **ACK**.

Appendix B: Contents of enclosed CD-ROM

```
\
|
|- budiik ................. Main folder of Budiik System
|    |
|    |- bin ................ Executables (Budiik Client)
|    |    |
|    |    |- nls ............ National language support
|    |    |
|    |    |- users .......... User data folder
|    |
|    |- doc ................ Documentation
|    |    |
|    |    |- technical ....... Code documentation (generated by Doxygen)
|    |
|    |- lib ................ Third-party libraries
|    |    |
|    |    |- _help .......... Help and documentation
|    |    |
|    |    |- _install ........ Install packages
|    |
|    |- src ................ Source code
|         |
|         |- obj ............ Objectives output
|         |    |
|         |    |- bpl ......... Run-time packages
|         |
|         |- res ............ Graphic resources
|
|- thesis ................. Master thesis (technical report)
```