

Diploma Thesis on Mobile Computing

Manfred Steinkellner

April 30, 2002

Contents

1	Introduction	1
2	Overview	3
2.1	Feasibility Study	3
2.2	Analysis	3
2.3	Design	4
2.4	Implementation	4
2.5	Goals vs. Effective Done	4
2.6	Conclusion	4
3	Feasibility Study	5
3.1	Steps of Organization	5
3.2	Criteria catalogue	5
3.2.1	Integrated Development Environment (IDE)	6
3.2.2	Hardware Requirements	6
3.2.3	Documentation and Samples	6
3.2.4	Stability	7
3.3	Evaluation	7
3.3.1	Integrated Development Environment	7
3.3.2	Hardware Requirements	7
3.3.3	Documentation and Samples	8
3.3.4	Stability	8
3.4	Product Notes	8
3.4.1	Forte for Java (Sun Microsystems)	8
3.4.2	Codewarrior for Java (Metrowerks)	9
3.4.3	JBuilder Borland	10
3.4.4	Other products	11
3.5	Product Evaluation	11
3.6	Conclusion	11
4	Analysis	13
4.1	Commercial Version	13
4.1.1	The Employer	13
4.1.2	Status Quo	13

4.1.3	System Requirements	14
4.2	Non-Commercial Version	15
4.2.1	Status Quo	15
4.2.2	System Requirements	15
4.3	Use Cases	15
4.4	Conceptual Model	17
5	Design	19
5.1	IT Landscape	19
5.1.1	Commercial Version	19
5.1.2	Non-commercial version	20
5.2	Database Model	21
5.3	Communication	22
5.3.1	Synchronization	23
5.4	Reusability	24
6	Implementation	29
6.1	From Scratch	29
6.2	Palm OS C-API	29
6.2.1	Application Startup and Stop	30
6.2.2	User Interface	30
6.2.3	Memory	31
6.2.4	Storage System	32
6.2.5	Network	32
6.2.6	Class Framework	32
6.3	PDA Client	33
6.4	C++ vs. Java as Server Programming Language	34
6.4.1	Introduction	34
6.4.2	Environment	34
7	Goals vs. Effective Done	37
7.1	Commercial Version	37
7.1.1	Goals	37
7.1.2	Effective Done	38
7.1.3	Conclusion	39
7.2	Non-commercial Version	40
7.2.1	Goals	40
7.2.2	Effective Done	40
7.2.3	Conclusion	41
A	Glossary	43

List of Tables

7.1 Project Milestones	40
----------------------------------	----

List of Figures

4.1	Client Use Cases	17
4.2	Use cases: Synchronization	17
4.3	Entity Relationship Diagram	18
5.1	Database Model	23
5.2	Class Diagram: SOAP Communication	24
5.3	Sequence Diagram: Update Task	25
5.4	Sequence Diagram: Archive Tasks	26
6.1	Palm OS: Heap Structure	32

Chapter 1

Introduction

Nowadays Mobile Computing provides a lot of applications. In opposite to the beginning of the digital age when computers filled whole rooms they fit into suit pockets now. Although the little personal digital assistants are working with low performance CPUs they can be used for small but important applications e.g. time scheduling. Complementing technologies like wireless communication protocols provide the basis for communicational facilities. Based upon those protocols we are able to interchange data between mobile clients or, what is at least that important, to transfer data from the mobile client to a server listening to specific ports. The mobile clients only act as presentation layer and as a temporary storage facility. During synchronization the data is transported to a computer in the local area network. A server within the network is dedicated for mobile device services and reacts upon the received data. With this communication facility we have the ability to start sophisticated transactions on a database only by pressing a button on a mobile device. Imagine you're communicating with a middle tier server that relays your database queries to a database server after it has verified your account data. Hundreds of clients may communicate with the middle tier and nevertheless only compact and optimized queries are forwarded to the database server due an optimizer in the middle layer server. These are possible scenarios suggested from big database vendors like Oracle. Big companies may be interested in those applications but there are still smaller ones which don't have the money or which don't want to spend their money on a Oracle database with a Oracle middle tier server and maybe even an Oracle Lightweight database for the mobile device. You say there must be some other ways, and you are right. Not everybody needs killer applications and therefore not everybody has to use killer tools. Creating applications for mobile devices gets easier and easier. Create a little program for a mobile device, create a nice and cute server and provide some communication facility is what we want to achieve. This College Diploma Thesis is based upon a time registration system. This system was coded in C/C++ for CenterPoint - Connective Software Engineering GmbH. and it was coded in another way a second time in Java. Throughout the thesis we will examine the little and big differences between

those two languages and we will experience the abstraction level in both languages due to existing libraries and frameworks.

Chapter 2

Overview

My Diploma Thesis is based upon my work with CenterPoint - Connective Software Engineering GmbH, where I implemented a time registration system. The following thesis represents insight into the process of developing applications on Palm OS with both C/C++ and Java. While the first chapters concentrate feasibility study, analysis and design the later one's try to give the reader more knowledge on the development and its pitfalls.

2.1 Feasibility Study

This section describes the process of tool analysis and selection. First of all it was necessary to wade through the tons of online documentation available on every vendor's homepage. After I decided to test three specific tools I tried some ways to organize the software. Depending on the vendor the work varied from downloading the software and just testing it to requesting the software (which was finally sent in a package) and writing emails to receive the codes for the software (Rational). During my test phase I made a criteria catalog to define the important criteria for the tools I will work with. The tests emphasized on the criteria defined in the criteria catalog and tried to cover all aspects of the programs that seemed important to me. Each criterion was set up with an accurate description to provide the reader of this thesis with more details on how I made my decision on the tools. The chapter contains simple tables showing each tool and the points/percentage it got for a specific criterion.

2.2 Analysis

After the feasibility study I began to design the system the users would like to work with. This section contains both project requirement specifications, from CenterPoint and for school.

2.3 Design

After the starting analysis of the project goals I investigate the problems in a more detailed level. This chapter contains a database model which is the basis for both solutions. It also contains sequence diagrams on several use cases and a description on the communication model via SOAP is provided.

2.4 Implementation

This chapter is likely the most exciting for mobile programming rookies. Here you can get the plain facts about challenges during my implementations - whether with the C API or the Java framework. I explain all programming challenges I faced and all the problems I solved in this chapter. You may also find some instructions and tips to avoid pitfalls. Finally it provides a comparison between the C/C++ and the Java framework.

2.5 Goals vs. Effective Done

Sometimes what we want to achieve and we complete are two different things. In this chapter I describe the objectives for this project at CenterPoint and at school. While the goals at CenterPoint were redirected during my work according to their needs there was no big adaption of the targets at school. Here we can examine the ever-changing world of business leading to projects which get never finished in their original way and to new, creative solutions. We take a look at the plans and afterwards we look in detail on their fate.

2.6 Conclusion

After all the troubles and sleepless nights there's the resume on the whole project. Suggestions on further projects in the area of Mobile Computing as well as facilities you have to evaluate before starting new projects in this area because they just recently appeared.

Chapter 3

Feasibility Study

The following description provides some organizational information concerning the college diploma thesis and a tool analysis.

The tool analysis for my College Diploma Thesis comprises very different IDEs. One of the main aspects is to find an IDE supporting the development of Java applications for Palm OS efficiently. There is no need of an IDE able to create state of the art Java programs.

3.1 Steps of Organization

Most IDE vendors provide trial editions of their products. To get access to these trial editions you have to register at the vendor's homepage and then you may download the program. Alternatively you can order the trial edition on CD saving you downloading the file. It may occur that some vendors don't offer some editions on CD or that they don't offer them to download. Rational software doesn't support downloading for example.

Many vendors also have special student editions you can use for several months up to a year. In this case most vendors require an application form containing information about the school or university you attend and the approach to use the product.

3.2 Criteria catalogue

We use a criteria catalog to choose our product. The catalog describes seven criteria and their according weighting. Criteria with a higher weighting have more influence on the selection as others. KO means that the tool must have this attribute otherwise it's out of the competition.

Generally all examined tools have to produce pure Java code that is 100% compatible with the Sun Wireless Toolkit. The code must be compilable and executable with the means of the standard toolset included in the Java Wireless Toolkit.

For each criterion we give 10 points at maximum, equal to 100%. The minimum for a criterion is 0 points, equal to 0%.

3.2.1 Integrated Development Environment (IDE)

Every development tool examined has to be an integrated development environment. Furthermore the IDEs have to support the Java Wireless Toolkit from Sun. New Java technologies like Java Beans or Enterprise Java Beans don't have to be covered with wizards by the IDE. The support of basic IDE functionality like embedding the compiler, setting the paths and managing project directories is enough for this project.

Another critical issue, concerning the selection of an IDE, are the hardware requirements, separately dealt with in the next section.

3.2.2 Hardware Requirements

Due to the fact that not every software development firm has the most powerful state-of-the-art computers it is necessary to examine the hardware requirements of a product. The practical part of the College Diploma Thesis is developed on my old home PC and therefore the hardware requirements must be very low.

3.2.3 Documentation and Samples

How good is the documentation that comes along with the product? We also look at the sections covered by the documentation. To get a good mark it is necessary that there are a couple of documents describing the IDE itself as well as a reasonable amount of documentation for the available libraries. It is absolutely necessary that the product ships with a documentation for all the classes included in its framework or library.

There are different kinds of help systems

Loose HTML files Some products just provide a HTML version as help leading to an awkward information search for the programmer.

PDF version Those files are little in size and provide basic search and navigation ability within the file. They are not much better than HTML files.

Combined HTML help system Such as it is provided from Microsoft in their Visual Studio. It supports the user in searching specific words or phrases and it eases finding the right information.

Combined HTML help systems are the most convenient way to search for information. Using HTML or PDF files means doesn't provide a facility to search all files at once in contrast to the combined help system.

This main criterion describes whether there are any sample applications or not, too; as well as the quality of existing samples. Sample applications decrease the time an unexperienced developer needs to get a clue of the tool and the available technologies. Advanced developers may use sample programs to refer to when they have a problem in their current task, to find out what's wrong in their programs.

3.2.4 Stability

The program I am looking for must be stable. Neither any developer nor any customer is interested in an unstable product leading to program crashes. In some cases such crashes may lead to a loss of already written code and thus frustrates the programmer. There's no need of a tool that lowers the productivity of an encouraged person.

3.3 Evaluation

The evaluation is based on my computer system at home. This is a PII 300 MHz with 256 MB RAM and a 30 GB harddisk. Thus the hardware requirements have to be very low to collect many points. The following tables show the division of the main criteria. Each main criterion is split up into sub criteria, varying in their influence on the tool selection, depending on the assigned points. The result a tool gets on a main criterion can be calculated by accumulating the points for all sub criteria. Each tool is examined using these guidelines.

3.3.1 Integrated Development Environment

Every tested product fulfills this criterion.

3.3.2 Hardware Requirements

CPU (40%)	Result
$\dot{}$ PII 300 MHz	100%
PII 300 MHz- PIII 450 MHz	75%
PIII 450 MHz - PIII 600 MHz	50%
PIII 600 MHz - PIII 750 MHz	25%
$\dot{}$ 750 MHz	0%

Memory (40%)	Result
$\dot{}$ 128 Mbyte	100%
128 - 192 Mbyte	50%
192 - 256 Mbyte	25%
$\dot{}$ 256 Mbyte	0%

Harddisk (20%)	Result
ı 200 MB	100%
200 - 300 MB	50%
ı 300 MB	0%

3.3.3 Documentation and Samples

Documentation - Medium (20%)	Result
Combined HTML help system	65%
Loose HTML or PDF	35%

Covered Topics (40%)	Result
Java SDK	25%
Java ME	30%
Add-on products	10%
IDE	10%
Proprietary libraries	25%

Samples - Topics (40%)	Result
Graphical User Interface	30%
Network	15%
Database	20%
Conduit	15%
Web Clipping Applications	20%

Documentation and Samples (45%) Express for Java	Division Forte for Java Borland Handheld			
	Codewarrior			
Medium	20 %	7%	7%	7%
Covered Topics	40 %	22%	18%	30%
Sample	40 %	26%	20%	12%
Overall	100 %	55%	45%	49%

3.3.4 Stability

The criterion is based upon the number of crashes that occurred during evaluation. Every program got 100% in this section because no program crashed during the tests.

3.4 Product Notes

In the following we provide some detailed notes on the tested products.

3.4.1 Forte for Java (Sun Microsystems)

Suns Forte for Java is a mighty tool that comes with wizards supporting the creation of Enterprise Java Beans, WebServices and the iPoint WebServer as

add-on, only to name a few. As an enterprise development product the IDE supports the development of Java NetBeans as well. The product is designed to be as flexible as possible, so you can change almost everything concerning the layout and the behavior of the IDE. Therefore one of its problems is that you need a lot of time to get used to the system and to know where you have to change a detail to get the effect wanted.

The documentation of the Forte for Java 3.0 Community Edition only concerns NetBeans, iPlanet WebServer and an ORDBMS from PointBase. In opposite to other IDEs the documentation is in HTML format. This has some disadvantages when searching for keywords in opposite to a help system (e.g. MS Visual Studio help system). Examples for Palm OS are only available after the installation of the Sun Micro Edition Wireless Toolkit (to download for free at www.sun.com).

The IDE is written in Java and thus requires a lot of hardware resources.

Configuration	CPU	RAM	Paging File Size
Minimum Configuration	PII 350 MHz	128 MB	128 MB
Recommended Configuration	PIII 450 MHz	256 MB	256 MB
Test Configuration	PII 300 MHz	256 MB	380 MB

The minimum configuration according to the documentation is far too less to work in an efficient way with the tool. The test configuration only works well if Forte for Java is one of few applications that are open. Some users may have many programs open and will not be pleased with 256 MB RAM, therefore 512 MB should be available.

Another negative detail concerns the sample applications that are included in the Java Wireless Toolkit provided from Sun as well. When installing this toolkit it provides the option to be integrated into the Forte for Java IDE. Nevertheless the sample projects from the toolkit can't be compiled without an amendment to the project settings in the IDE.

Notice There was a Syn-Con competition on the best IDE won by IBMs Visual Age, followed from Inprises Jbuilder and WebGains VisualCafe. Suns Forte made the fourth rank. The Sys-Con Competition for Java was divided into several sections e.g. books, applications, tutorials and so on.

(See <http://www.sys-con.com/2001/PR/code.cfm?page=09242001a>)

3.4.2 Codewarrior for Java (Metrowerks)

The installation procedure has problems if the user doesn't obey everything requested. Although I chose Jdk 1.2.2 in the custom installation dialog I tried to skip it. This led to a cut in the installation; it completed but hasn't configured everything all right.

The product is an IDE too and obviously faster than Sun's Forte. The installation needs 167 MB on the harddisk what is more than the double of Sun's Forte (76 MB) and comes along with additionally provided documents and products.

Metrowerks and Sun add the iPlanet WebServer from PointBase to their package.

The documentation includes Sun's Java SDK 1.3, documentation on Voyager (an object request broker) and on the IDE itself. Documentation is provided in HTML and PDF versions but they are on different topics. The IDE utilizes a combined help system to provide user context dependent information but the content is very poor and not very helpful.

Configuration	CPU	RAM	Paging File Size
Minimum Configuration	Pentium MHz	32 MB	-
Recommended Configuration	Pentium MHz	64 MB	-
Test Configuration	PII 300 MHz	256 MB	380 MB

There are no performance problems on the test system with this IDE.

The sample applications added to the CodeWarrior for Java are determined to the Standard SDK. Only few samples are available for mobile devices. The samples can be compiled without any problems or further configurations.

While snooping around on Metrowerks' homepage (www.metrowerks.com) I found out that Metrowerks offers online courses free of charge at their homepage. This runs like this: After you have signed up to their community you may enroll in a course provided. A course consists of chapters you have to work through and assignments you have to solve at the end of the chapter. These assignments are posted on a message board accessible to every student as well as the mentor. There is even a section that suggests course material for those who prefer books instead of online sources and would like to concentrate more on the topic. After a student has solved the assignments the homework is posted on the message board and can be discussed between the pupils and their mentor.

3.4.3 JBuilder Borland

The hardware requirements are acceptable. The program runs comfortably on my system. A lot of documentation is available lacking only the Java Standard SDK documentation. Sample applications for mobile devices are added. The Borland JBuilder is the only tested tool including a code versioning system (CVS). This system can be utilized when working in a team to develop an application. This is a must if the files are edited from more than one developer so you can see what has changed in the file and decide whether you apply the new changes or abandon them. An already included CVS also saves money because you don't have to buy another expensive software that enables you to efficiently develop software in a team.

3.4.4 Other products

RationalRose is used for analysis and design. The software is provided from Rational for the duration of my College Diploma Thesis. The only restriction I must accept is the request for a new product key every month. The product is used at CenterPoint GmbH as well and so it is easy to create documents available and usable for both - the school and the employer of the commercial version.

Tomcat - the reference implementation for state of the art Java possibilities - is used as web server.

3.5 Product Evaluation

Rank	Product	Result
1	Codewarrior for Java	77,05%
2	Forte	62,25%
3	JBuilder	59,75%

3.6 Conclusion

Although it seems that I have tested three high quality products the winner has slightly better performed in the categories hardware and documentation. If hardware requirements would not be that important to me the choice would maybe have fallen on another product.

Chapter 4

Analysis

This chapter shows the experience gained from the analysis phase while working on the Time Registration System.

4.1 Commercial Version

In the following paragraphs we present the current situation at CenterPoint. Then a specification of the system requirements follows.

4.1.1 The Employer

CenterPoint GmbH. is a little software company. They employ 10 people and 3 people who are working on their College Diploma Thesis. The company develops software for the specific needs of the customer. Their foundation was in January 2000. The software produced so far was created for companies like Infineon or Carinthian Tech Research.

Due the size of the company they develop software according to the eXtreme Programming approach. There are even several products like the Base Library - including formatter classes, logger classes, string classes, etc - which are implemented and partly tested from a single person.

4.1.2 Status Quo

CenterPoint deploys an MS-Access database to store working time records. Every employee has his/her own database to log his/her records. At the end of the month everybody must extract all records of the current month and send them per email to the administration. Somebody has to copy the data from every extract into the central database. Further analysis is based on the existing data in the central database.

This system implies several disadvantages. At first everybody has to extract the data at the end of a month; this action could be automated. Some people might forget to extract the data thus the data analysis is delayed. Maybe people

have even to be reminded to send their extracts. These steps delay the start of evaluation tasks.

Another essential weakness is updating all databases e.g. when new projects are added. Because every user has its own database this leads to a multiple effort if an update has to be applied. Updates come along at least once a month and include new projects, customers or workcodes (project phases such as design, analysis, test,). So every colleague has to apply it. If anyone forgets to update his/her database he isn't up to date and most recent projects aren't available in the database. It even occurs that updates may contain errors and so the administration must send another update to the employees and they all have to run the update again.

Although the current solution contains little problems and some overhead it provides all necessary functionality - but only in the office. When an employee is abroad there is no facility to log the working times. Of course that's no big problems when being at the customer for only one afternoon but imagine a colleague is abroad in Germany for a week. The MS-Access database can't be taken abroad easily and so there is no facility to log the working times. We need a program that allows storing the working times if an employee is not in the office.

4.1.3 System Requirements

The goal was to create a system enabling the employee to log his/her working times over a day, independent of the current location. The employee must have a facility to log the working times at the customer as well as in the office. The program must provide a user interface supporting the selection of a specific project and workcode. Furthermore the employee writes down a short note describing the work and selects the start and stop time. The employee may check whether he worked at the customer or not. Looking at the term 'independent of current location' the choice fell on mobile devices powered by Palm OS.

The system consists of a server and a client. The server is utilized to authenticate the client, to access the repository as well as synchronize data with the client. It supports the creation of working time records and furthermore editing, searching and deleting those working time records. It might support the query for existing projects and workcodes, too. Searching for already existing records must be available for a specific record as well as for a time frame.

The already existing MS-Access database for recording working times will be used for data storage.

To hit the requirements of a three tier architecture an application server is used to transport messages from the Palm device to the server. This application server communicates by the means of SOAP protocol with its clients. Thus the communication between the mobile device and the application server is based on SOAP protocol [SOAP]. In order to use the application server I have to implement the server for the time registration program as a Dll.

4.2 Non-Commercial Version

This part of the diploma thesis serves as reflection. After I have implemented the commercial version in C++ this version will be implemented in Java. Working with two programming languages enables me to compare various aspects of them concerning Palm OS. In #section I will provide details on the existing framework in Java and in C/C++. Another essential aspect experienced in my work was GUI creation and data storage on mobile devices run by Palm OS.

4.2.1 Status Quo

I don't have to pay respect to any existing infrastructure or anything else. My work also won't be integrated into that environment giving me free decision in communication protocols.

4.2.2 System Requirements

Generally I have to admit that the non-commercial version doesn't provide all the functionality of the commercial version. The goal in building the non-commercial version is to get a functional time registration system that allows the user to log his/her working times on the PDA and to synchronize them with the server. That's the same functionality requested of the commercial version. The only restriction I'm bound to is the database system which must be a MS-SQL server or a Oracle system.

4.3 Use Cases

After the system requirements are specified we start analysing the system. For this stage we refer to the project requirement specification from CenterPoint. The non-commercial version equals in large parts its predecessor but there are some features that are not implemented. So we take only necessary use cases for the non-commercial version.

First we identify several use cases which are the basis for the class and sequence diagrams. Generally the system consists of two parts. The first part is the client program that facilitates the user to insert the data. This program provides a little GUI where the user can select the actual project he's working on, the workcode, the start and stop time, date and a short note on the work that was done. Figure 4.1 lists the basic use cases for the client program. According to the project requirement specification the program has to support creation, deletion, search and edition for all tasks. The implementation of this use cases will require a local data storage facility on the mobile device. If the data would be stored elsewhere there would be an actor in the diagram indicating this fact. The only dependence, as you can see from the figure, consists between *delete task* and *update task* and *search task*.

The second part of the use case diagram concerns the synchronization between

the mobile device and the database server. There's a little increase in the complexity of the diagram compared to the client use case diagram. Lets examine the diagram in detail to check about potential problems and dependencies between the use cases.

Authentication Is a very important use case. The client must authenticate on the server before any operation may be executed. This is an essential security feature of the system and should be treated carefully. The best way to guarantee high level security will be to check the user's permission every time he wants to execute an operation on the server.

Update Workcodes This use case will cover the update of any workcodes. Workcodes may be updated if a new one is added in the central database. New workcodes can be added on demand e.g. if the time spent on a specific activity increases so that a new workcode for this activity becomes necessary.

Update Projects In opposite to workcodes new projects are added quite often. The functionality is the same as in *Update Workcodes*.

Archive Tasks This is one essential part of the synchronization process. The client transfers all its tasks to the server where they must be stored. Thus a concept must be developed to guarantee that all tasks received from the client are stored in the central database. Database transaction model will be used here to ensure the required functionality.

Synchronize This use case comprises the complete process of synchronization. The synchronization consists of updating projects and workcodes on one hand and moving all tasks from the client to server on the other hand. We must ensure that that in case of abortion the system doesn't remain in an inconsistent state. Imagine the client already copied 3 tasks to the server and they were stored in the database. Suddenly a communication problem occurs and the tasks aren't deleted on the client. Without an undo mechanism the 3 tasks will be copied into the central database the next time again.

Database This isn't a use case but another actor in the diagram. We use the symbol of an actor because the server in our system will communicate with a database server (Oracle) which is then responsible for storing and retrieving the data in the database.

Note Although we stick to UML in this project, due to the project size we don't utilize all various diagrams provided by UML standard.

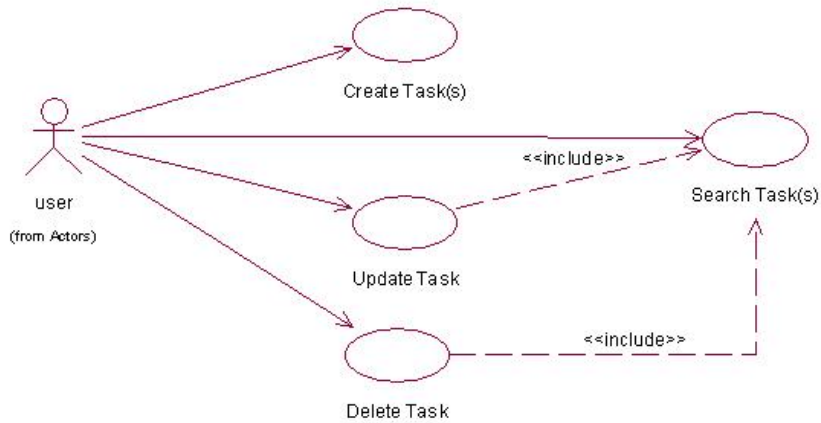


Figure 4.1: Client Use Cases

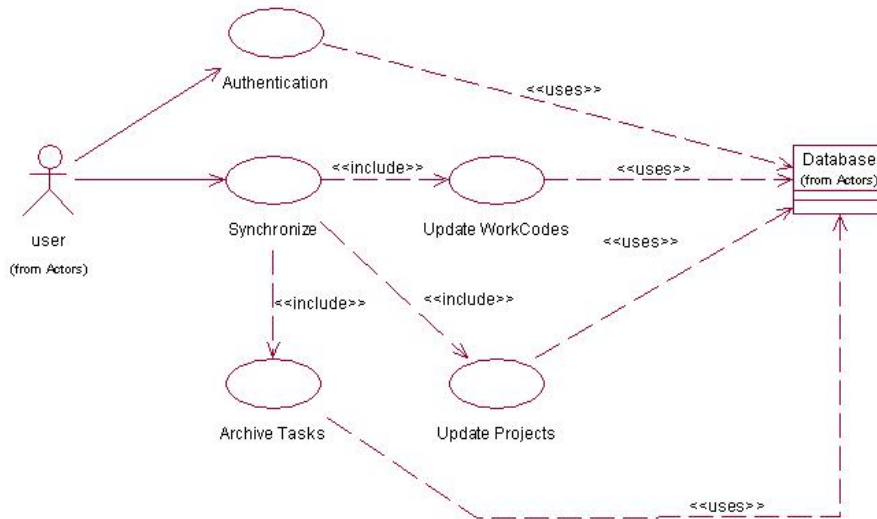


Figure 4.2: Use cases: Synchronization

4.4 Conceptual Model

After the use cases have been identified we will take a closer look on the data required for this system. One way to identify the attributes used in the database is to underline all nouns in the project requirement specification. Afterwards you

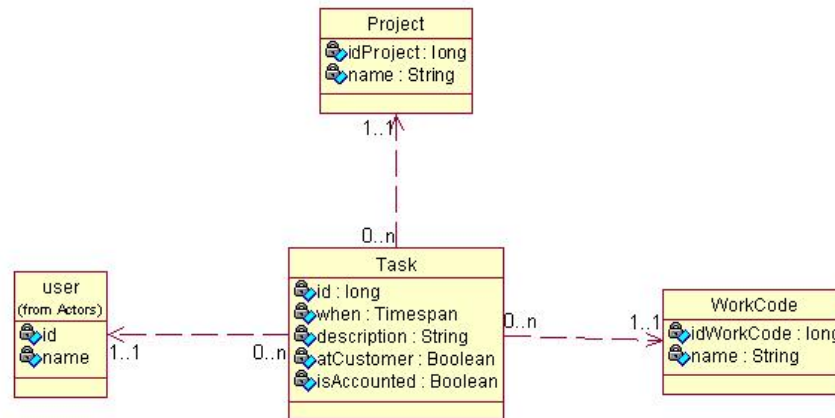


Figure 4.3: Entity Relationship Diagram

create the entities containing the underlined attributes. Speaking of the time registration system there are just a few entities. Figure 4.3 shows all of them and how they are related to each other. Here you can recognize the most important entity in this system - *Task*. The system will be able to create and delete, search and update tasks so it's natural that all is based on tasks. *Workcode* and *Project* are attributes of a task. Each task is assigned a specific project and was done in one of the available project phases e.g. analysis, implementation, support and so on. The user entity is used to store the different users. Like *Project* and *Workcode* also *User* and *Task* have a n:1 relationship. All tasks that exist must refer to one entry in the other entities.

In the commercial version there is also an entity storing the account information which consists of username and password. The entity *Authorization* has a 1:1 relationship with user because there must be precisely one entry for each user. This entity is important when the user wants to synchronize the data. First he has to authenticate. During this process the server verifies the given information from the user using the data from this entity. If the user-password-pairs don't match the login attempt from the user is rejected.

Chapter 5

Design

In this chapter our knowledge of the system's functionality will increase. So we can improve our data model used in the database and we will recognize some difficulties we have to face.

To increase our knowledge of the system we utilize sequence diagrams as defined in the UML standard to represent the required communication processes comprising the classes. On the other side we will create a class diagram covering all real world objects. Each class represents an object that exists in the system. We will take a closer look at the system requirement specification to determine the attributes for each class and its operations.

5.1 IT Landscape

When it comes to designing new software you also have to inspect already existing software your's will work with. You have to locate potential problems and try to avoid them in your design approach. Define interfaces to the other systems or if there are already existing interfaces you have to obey these.

5.1.1 Commercial Version

In the case of CenterPoint there are several systems influencing our design decisions. First there is an already existing MS-Access central database. This database is in use now. It contains all project data comprising all tasks and all workcodes available. It contains a list of user, too. Of course there are other tables like some for customers and other stuff too but we aren't paying respect to these because they don't influence the application. The only organizational instance accessing the database is the management. As already described in the project requirement specification in section 4.1 they distribute the updates to each user and the user has to apply them on the local copy of the database on his machine. At the end of the month the users have to create a data extract for the recent month and send this extract via mail to the management where

they copy the data from all extracts into the central database.

In the system we design there is one very important difference. In future there will only be one central database and all data will be stored in this database without maintaining a database replica on each workstation. So we have to write a piece of software that is responsible for storing the data in the MS-Access database.

Now it is necessary to introduce another important part of the software infrastructure at CenterPoint. It is the CenterPoint Application Server (AS). We use the AS as a middle-tier layer. The AS passes the messages received from network and encoded in XML to the running TRS server. By passing we mean that it calls the according method with the given data. By using the AS the TRS server implementation doesn't have to bother on the communication protocol used for data transport. It also ensures that we can swap the frontend or backend layer without a lot of work. On one hand if we would create a new backend we would only have to ensure the implementation of at least the same methods as the previous server had. On the other hand we would have to ensure that a new frontend would stick to the SOAP implementation used by the AS. Another variant of project adaption would be to develop any other client supporting the same backend. As you can see in section 7.1.2 there was a fix idea in the management of CenterPoint to develop a LAN-based TRS which would extensively use the offered facilities of the AS. This LAN-based TRS would use another important functionality of the AS. Its support for CenterPoint proprietary server pages (in short .csp). This server pages may be compared to Java Server Pages and provide the same functionality by using CenterPoint RMI/Script. This is a script language invented for the purpose of creating server pages with the ability to utilize all existing Dlls on the server. CenterPoint RMI/Script enables you to create an object that is registered at the AS and then use it. This feature enables you to create very sophisticated server pages by utilizing the state of the art programs located and registered on the AS.

After discussing the state of our middle layer we have to mention an already existing SOAP implementation for Palm OS we will use in our implementation. The library is full functional and contains several classes to ease the use of SOAP communication. You can find more on this in section 5.2.

5.1.2 Non-commercial version

In opposite to the commercial version there is not so much software we have to obey in the non-commercial version. The products we use here is (a) Oracle 8i database and (b) Tomcat web server. Since we use JDK 1.3 there are no problems in connecting to the database using JDBC. The only adaption I had to make was to install the latest Oracle JDBC driver.

In theory the only things we have to inspect concerning Tomcat are some details in deploying servlets. But in reality there was a great deal of additional time required to fix several problems concerning the path configuration, for more details please see section # further tomcat trouble details. Tomcat uses HTTP to communicate with a client. Thus we would have to implement our mobile

client using HTTP instead of SOAP. Although since the Java framework provides some comfortable classes covering this topic we can compare this task with the one we faced in the commercial version - from the view angle of the programmer.

5.2 Database Model

At work I created class and sequence diagrams parallel. Due the constraints of a written document we will first create the class diagrams and afterwards we will analyse the use cases and draw some sequence diagrams.

Based on the ERD (figure 4.3) we work out a class diagram. All entities in the ERD become a class because they are the business objects in this application. The whole time registration system focuses on task creation and manipulation. In this stage of development we have to staff the classes with according methods. To implement the software with the given Palm OS C API it is necessary to create some generic wrapper classes based upon the basic Palm API written in C. In the following we present some enumerations describing the listed classes in figure 5.1 and their attributes and methods.

Database A generic class *Database* will handle the database access, see figure 5.1. This class provides the basic methods for data manipulation.

Create - creates a database object with the given name *dbName* on the specified platform. This can vary from implementation to implementation very strongly. Speaking of a *SystemDatabase* on Palm OS this means only to call a single function whereas a *SOAPDatabase* means to create some objects of the underlying SOAP library and to initialize them.

Open - opens the specified database.

Close - closes this database.

Insert - inserts the given record into the database.

Remove - searches for the given record and removes it from the database if the record is found.

DeleteAll - deletes all records in the database.

Find - returns the index of the record within the database.

GetAt - returns the record at the given index.

GetLast

GetAll - returns a vector of records that were found in the database.

GetRecordCount - returns the number of records in the database.

Every class supporting persistence has to implement the *Record* interface. It contains comparing and compressing/decompressing methods.

The generic class *Database* may be used for communication purposes, too. Looking at Figure 2. you will notice that *SOAPDatabase* implements the class *Database*. Every *SOAPDatabase* object needs a hostname and a port where it expects a SOAP-server responding to its requests. A timeout may be set according to the communication facilities. The class *SOAPDatabase* overrides all methods provided in the base class *Database*. Additionally this object transports tasks over the network to the server where they are finally stored in the central database. Based on this generic architecture you may support whatever protocol you want by adding a new subclass of *Database*. The software developer working on the client doesn't have to worry about the communication protocol or whether a network is utilized for communication or if the data is stored on the device. This design leads to a class framework supporting a lot of backends. If there is a new communication protocol the class framework wants to support, only a new subclass of *Database*, which communicates by the means of the specified protocol, must be implemented.

The generic class *Database* may be used for communication purposes, too. Looking at #Figure 2. you will notice that *SOAPDatabase* implements the class *Database*. Every *SOAPDatabase* object needs a hostname and a port where it expects a SOAP-server responding to its requests. A timeout may be set according to the communication facilities. The class *SOAPDatabase* overrides all methods provided in the base class *Database*. Additionally this object transports tasks over the network to the server where they are finally stored in the central database. Based on this generic architecture you may support whatever protocol you want by adding a new subclass of *Database*. The software developer working on the client doesn't have to worry about the communication protocol or whether a network is utilized for communication or if the data is stored on the device. This design leads to a class framework supporting a lot of backends. If there is a new communication protocol the class framework wants to support, only a new subclass of *Database*, which communicates by the means of the specified protocol, must be implemented.

This architecture is highly flexible in supporting new database backends.

5.3 Communication

SOAP was chosen as transport protocol because of its flexibility and its wide spreading and the amount of servers already supporting SOAP. It is used as communication protocol between the mobile device and the server.

The communication with the server is achieved by means of an already existing SOAP implementation for Palm OS - which is part of the class framework for Palm OS and was written by Christian Gaal, software developer at CenterPoint. This implementation provides the classes *SOAPClient*, *SOAPRequest* and *SOAPResponse*. To start communication with the SOAP-server you have

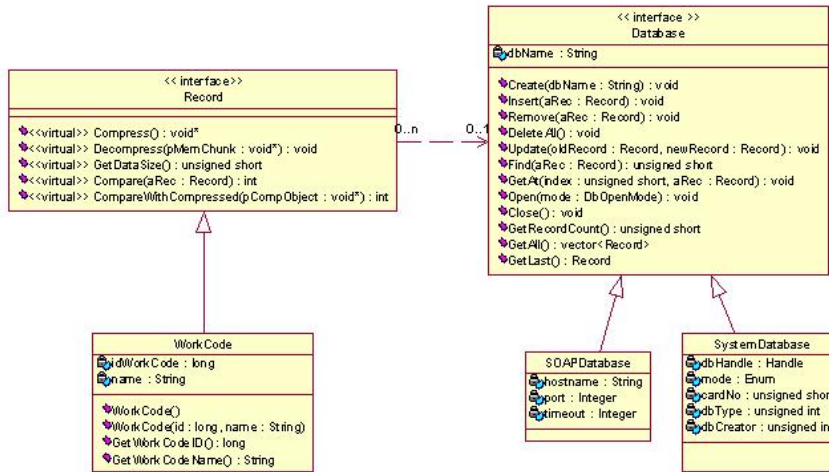


Figure 5.1: Database Model

to create an *SOAPClient* object. It provides methods for configuration settings like server name, port, timeout and sending method (*get* or *post*).

In order to set up a communication with the server the class *SOAPRequest* must be instantiated. An *SOAPRequest* object specifies the method the client wants to call. Use the method *AddHeaderProperties* to add new properties (parameters). After the request is sent, the server returns an *SOAPMessage* object. In case of success the returned object is an *SOAPResponse* object otherwise it is a *SOAPFault* object. If a *SOAPFault* object is returned the methods *GetFaultString* and *GetFaultCode* provide information about the problem. To execute the method on the server you have to call the *Invoke* method from *SOAPClient*, passing the *SOAPRequest* object as parameter. This is a synchronous remote method call returning the result immediately. The client program receives an *SOAPMessage* object as answer. Depending on the success of the method call the object is an *SOAPFault* or an *SOAPResponse* object.

5.3.1 Synchronization

To support full flexibility to all employees the system has to synchronize the data as often as possible. The user has to start the synchronization. By putting the Palm device in the cradle and starting the synchronization via the client program. Synchronizing comprises saving all existing tasks to the server and deleting them on the device as well as updating projects and workcodes.

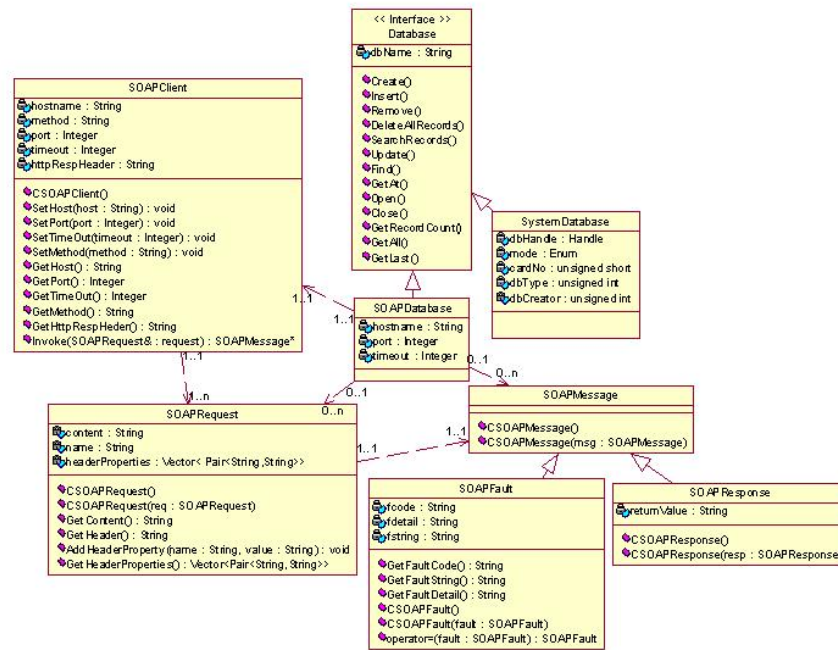


Figure 5.2: Class Diagram: SOAP Communication

Figure 5.4 shows the sequence diagram for synchronizing the tasks between mobile device and server. The synchronization may only be interrupted if there are duplicate record IDs. Otherwise no user interaction is necessary.

5.4 Reusability

The commercial version is designed as a three tier architecture. The design is very flexible when it comes to swapping a layer. As already shown in Figure 5.1 all existing database backend classes implement the class *Database*. A further backend class doesn't require any adoptions in the business layer.

An existing class framework from CenterPoint was brought to Palm OS. This class framework contains a lot of reusable code like formatter, collections, parser, exceptions and so on. If you are utilizing an object oriented programming language it is very important to build a class framework during the evolution of

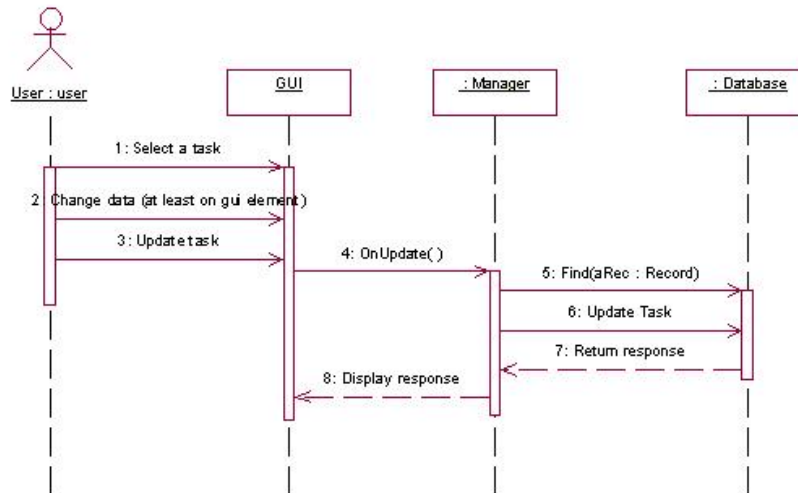


Figure 5.3: Sequence Diagram: Update Task

your software products. Don't always try to reinvent the wheel - it's too cost intensive and making everything anew means also creating new bugs.

A very important aspect about software engineering comes to daylight again especially when building class frameworks. It's the quality assurance of the produced software. Of course you have to test all the software you create but speaking of class frameworks, which are extended all the time, there are some tricky things you might consider. Writing software and testing it manually is very time and cost intensive. In the age of computers we could invent a mechanism to automatically test our software thus leaving the community of developers more time for creating new, exciting pieces of software.

One of those tactics is the test suite. Test suites are pieces of software that contain various test scenarios for a given class. A test scenario contains a lot of method calls for the specified class and pre-conditions as well as post-conditions examining the data passed to the method or the object's state. Classes without any bugs succeed all tests contained in a test suite. The intention of the test suite is to detect bugs. It is on you to produce test suites, running without user interaction, which leads to a high acceptance rate in your company. An important fact about this topic is the consciousness of the developers, who should use these test suites and produce new ones for new pieces of software. Emphasize on the time and money you save by using test suites. Test suites will also improve the quality of your software because they will detect bugs in a way human beings aren't able to. Implementing the test suites before implementing the library itself helps in specifying your expectations of the functionality.

Anyway, imagine the extension of an existing class framework. Maybe only new classes are included but eventually the old classes are edited in some way. The

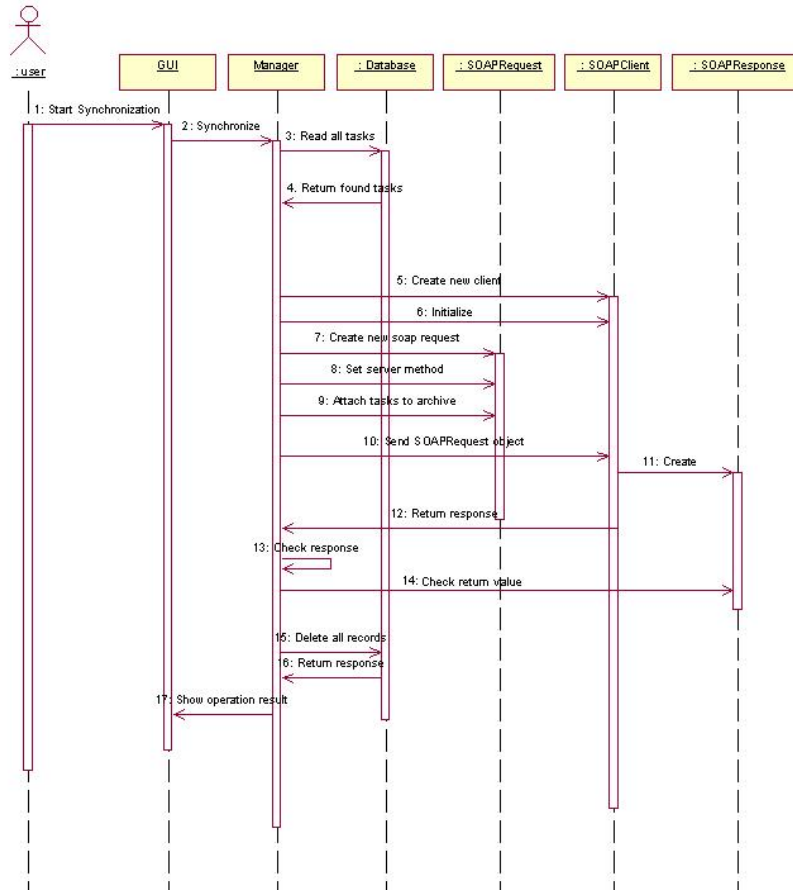


Figure 5.4: Sequence Diagram: Archive Tasks

update of an old class should improve the performance or the aesthetics. Unfortunately the developer didn't recognize a dependency to another existing class and the new code doesn't work correctly. If the developer doesn't have a test suite or any other kind of test mechanism he will recognize the problem any time in the future e.g. when an already existing program built upon the class framework doesn't work any more.

A test suite shows immediate response by indicating that one of the test scenarios has failed. This leads to an instant bug search and the problem is found and fixed in a fast way. The mechanism can be automated and the developers don't have to bother with testing the software manually. If new classes are added to the class framework a new test suite or at least a new section within a test suite

must be written to ensure the quality.

This thought of an automated test environment may lead to an automated shipping system that first checks the most recent stable branch (for a specific software) out of a software configuration management system. Checking out means here to synchronize all sources to a computer with the required operating system (Windows NT, UNIX, Solaris) where a compiler builds the required program or library. But before the compiler comes in action automated test suites are run to ensure a high quality before delivering the software to the customer. Quality and cost saving is yours if you can establish a completely automated shipping process. If certain test suites fail, a report is mailed to the responsible software developer(s) and the project leader. Another important fact that shouldn't be forgotten is the time you win with an automated system.

Chapter 6

Implementation

The following sections provide details on the implementation specific challenges.

6.1 From Scratch

The commercial version is implemented in C/C++. The existing API for Palm OS 3.5 is written in C. Therefore the already existing class framework at Center-Point is brought to Palm OS. This is a good basis to start writing an application. Otherwise you would be stuck with details concerning the Palm OS C API and there would be a lack of convenient classes allowing you everyday work without re-inventing the wheel.

The non-commercial version is built with J2ME 1.0.3. Considering the complexity and the size of this class framework no basis implementations like in C++ have to be done. The Java Micro Edition provides a lot of classes for convenient and very quick application building.

6.2 Palm OS C-API

”Writing applications for handhelds, specifically Palm OS platform devices, is a bit different from writing desktop applications because the Palm OS platform device is designed differently than a desktop computer. Also, users simply interact with the device differently than they do desktop computers.”

Palm OS Programmer’s Companion

The existing API for Palm OS 3.5 is very rudimentary. The API is written in C and it doesn’t provide any classes yet. All functions are comprised within Managers, which are collections of functions for a specific area. The following descriptions of various sections in the Palm OS API are based on Palm OS Companion [PalmComp] and Palm OS SDK Reference [PalmRef].

6.2.1 Application Startup and Stop

Launching programs on common PC systems means to double click on an icon or - if we speak of shells - enter the program name and press 'Enter'. Applications on Palm OS can be launched in several ways. First of all the user can touch the application icon with the pen or he can press one of the four buttons on the Palm device which results in the launch of the associated program. The last possibility of starting a Palm OS program is by another program. To launch another piece of software you only have to specify the launch code of the desired program. Launch codes are a means of communication between the Palm OS and the application (or between two applications). There are several different launch codes to indicate different startup modes. The calling program can be any program or the global search facility. This is a mechanism enabling the user to search all program databases on containing the provided keyword. Global search uses specific launch parameters indicating that the programs don't need to fire up their GUI but only have to search their database and then disappear in the memory again.

6.2.2 User Interface

The Palm user interface and the input devices differ enormous from a PC interface and its peripherals. Palm users don't have a mouse enabling to click with one of two or three buttons or to combine mouse clicks with keyboard strokes. Forget the known input devices. Input on Palm OS is produced by the pen, which is coming along with a Palm appliance.

The Palm OS API provides a lot of user interface elements to encourage the programmer community in building highly sophisticated applications enabling the end user to perform complex tasks. The variety of interface elements begins with simple text elements, moves on to lists and menus and even enables the developer to create table elements. Examining a lot of applications on Palm OS you will recognize that the most of them only use the basic elements to achieve the functionality.

Use form functions to access the user interface elements. Forms contain all user interface elements and thus allow access to them. Elements are retrieved by passing the element id or by passing the element name.

GUI elements for Palm OS work fine if you just stick to common widgets. Problems arise when more complex user interface elements like drop down lists should be displayed. But first you need to separate static drop down lists from dynamic drop down lists. As the name already suggests the static drop down list keeps the same data from the application start on. On the other hand dynamic drop down list need to be filled after the program has started. In lack of an object-oriented approach there is no object for the drop down list that could take care of the represented data. Instead the software developer has to ensure that the data displayed is in a locked memory chunk. Detailed documentation about the memory management follows in the next section. This can lead to enormous problems because there is no simple mechanism to

ensure if a memory chunk is already locked or if it is still unprotected.

Object-oriented systems use a reference counting mechanism to avoid such potential problems.

Frequent system crashes can be the result of lacking memory locks. Those crashes may occur when the user touches a GUI element with the pen and the data contained doesn't reside in a globally locked memory area. When the program wants to access the data the program crashes.

6.2.3 Memory

Memory management is a very important issue on Palm OS looking at the low memory amount on Palm devices. Basically you have to separate between *MemPtr* and *MemHandle*. A *MemHandle* object doesn't directly refer to the data's memory address; instead it is a resource identifier which Palm OS has assigned to a specific memory chunk. #Figure 3.1 is a heap representation on Palm OS. At the beginning of the memory area there is the heap header; each heap has a header with a heap ID, status flags, the heap size, and the master pointer table. The heap header is followed by the master pointer table which is a dynamically-built table of persistent handles which map to the location for each chunk. Now you see that handles aren't referring to the direct memory address but to a table that contains the memory address the handle is mapped to.

This mechanism enables the system to move memory chunks during runtime. The advantage of this mechanism is to avoid memory fragmentation. If the space between the allocated memory spots gets too small no new memory can be dynamically allocated which would lead to a system breakdown.

The first step in requesting new dynamic memory is to create a *MemHandle* object. When you need to write on the memory chunk you have to lock it. When locking a memory chunk the system tries to allocate a memory chunk at the end of the heap memory area and marks it as unmovable as long as the lock exists. Here you notice that it is very important to release the lock after the write operation succeeded. Releasing means that the unmovable flag is removed and the Data Manager tries to move the chunk towards the beginning of the heap memory. You should be aware when working with memory pointers (=locked memory chunks). If the memory address isn't locked anymore the program will show an error message and crash afterwards. This is due to the memory surveillance performed by the Data Manager. It's one of the Data Manager's tasks to avoid access to unlocked memory areas. This is a very strict principle that can lead to a lot of bugs, especially when being not familiar with the memory management idea in Palm OS, which are hard to find.

Figure 6.1: Palm OS: Heap Structure

6.2.4 Storage System

6.2.5 Network

6.2.6 Class Framework

Since the class library already exists it must be brought to Palm OS. Palm OS has several restrictions due to primitive memory management and limited resources; there are a lot of constraints to obey. It doesn't support memory management functions like `memcpy` and `memcp` although existing in C. Instead I have to use specific Palm OS functions that replace those. But the syntax or functional problems have been very few compared to memory management troubles on Palm OS. There is a restriction to the memory size of a program. If it exceeds a memory size of 64 KByte you won't get the program running. The function calls within the program are limited to an area of 32 KByte ahead and behind the actual position - a simple address size problem that needs a lot of reading to be realized.

C++ is a critical issue speaking of Palm OS. There are some special features like virtual function tables which aren't working yet.

To build a library seems to be difficult because not even advanced developers - who are posting in a lot of newsgroups - have experiences with them. Thus programs which are using classes out of the PocketBase - as the class framework for Palm OS is called at CenterPoint - have to add the necessary source and header files to their project and recompile them every time the project is rebuilt.

6.3 PDA Client

The program is split up into a client and a server. As already remarked a class framework was created to support a faster development on the Palm OS platform. The class framework - called PocketBase at CenterPoint - also provides a better chance for software reuse.

Implementing a program on Palm OS can be compared to writing a program using the Win32 API. I have worked as Win32-API programmer some years ago for about a year and it was very basic programming. On the other hand basic programming leads to basic understanding which is very essential for successful software developers. Look at the people in the software industry who have never written Win32 applications; they started their career using the MFC (Microsoft Foundation Classes). As I pointed out before it is very important to use a class framework for rapid development but you also have to know how things work. Especially when you have to hunt down some bugs it is a key issue to know the depth of the MFC and the basic mechanisms that it is built upon.

Palm OS programs also have message queue that contains the latest events which are dispatched to the corresponding message handler. In the same the Win32 API has a lot of function codes indicating different events Palm OS has them too. But as the spiral of development leads on and on the way becomes a path of trial and tribulation. Nasty little bugs arise out of nothing and a lot of sweat is wasted to track them down and find a solution as well as an explanation for the bug.

Memory Management is one of these awkward parts of Palm programming. Due to the restricted resources the programmer needs to take care about the memory. In opposite to status quo of application development on PCs - messing with the resources - the development on the mobile devices leads into another direction. Few and expensive memory must be treated as a treasure and sophisticated mechanisms have to be invented to avoid memory waste e.g. the two different types provided from the Memory Manger: *MemHandle* and *MemPtr*.

The client provides an easy-to-use GUI, as shown in #Figure 3.2, to track the tasks. It supports the creation, edition, deletion and search of tasks. Small databases on the Palm device are maintained to store the tasks as well as the projects and workcodes. In respect to future developments on the WindowsCE platform a three tier architecture was chosen. The program is divided into a presentation layer, a business layer and a backend (data access) layer. In case of an improved database interface - maybe from a third party vendor - only the data access layer is replaced by the new one and minor changes and/or extensions have to be applied in the business layer.

To get the program running on the WindowsCE platform, the data access layer and the presentation layer have to be replaced but the business layer remains the same. This concept guarantees minimized expenditure for further adoptions and extensions of the software.

6.4 C++ vs. Java as Server Programming Language

According to the given infrastructure - most of the implementations and the class framework at CenterPoint are implemented in C++ - the server for the commercial version is written in C++. In opposite to the core C API there is already a very extensive Java framework making it as easy as a children's game to build a program from scratch.

6.4.1 Introduction

A servlet is a Java technology based web component, managed by a container, that generates dynamic content.

Java Servlet Specification v2.3

Commercial Version (CenterPoint)

The server is written in C++. It provides methods for adding, editing, searching and deleting tasks. Furthermore the server has to check if the user has the security permission to communicate with the system. User validation is based upon a table in the database that contains user/password pairs.

Non-Commercial Version (School)

This version is implemented in Java to take a look at the facilities and features of this language concerning mobile devices and to compare it with the commercial implementation in C++.

The system consists of a server and a client (an Java application on a mobile device). The server is implemented as a servlet thus supporting the standard servlet methods. It supports a query for all existing tasks as well as the creation of new tasks. When querying for all tasks the server opens a connection to the database via JDBC and retrieves all tasks from the specified table.

6.4.2 Environment

This section describes other programs used for the project. In opposite to a proprietary application server used in the commercial version an Open Source product - Tomcat - is deployed in the non-commercial version.

Commercial Version (CenterPoint)

Building the server requires some knowledge about the environment. The server is implemented as a Dll that is hosted by CenterPoint Application Server (AS). The AS runs on any machine (various platforms supported) on

the specified port. Connecting to the AS is achieved by typing the URL in the address bar in a browser and pressing the enter button.

The AS is based CenterPoint Server Pages (csp files). This technology is comparable to Java Server Pages. Csp files enable you to use the underlying CenterPoint class framework [CePo02]. When requesting a csp file from the AS it is first parsed and the instructions are executed. The process output is a HTML file that is sent to the browser.

Non-Commercial Version (School)

The reference implementation for Servlets from Sun - Tomcat [CePo02] - is used to host the servlet. The server can be downloaded under [CePo02] and is available for many platforms including Windows, Linux and Solaris 8. As you will recognize on their homepage, Jakarta is an Open Source project with a highly motivated developer community trying to solve problems quickly and extending the program at a fascinating speed.

The installation of Jakarta is very easy. After you have extracted the program into the destination directory you must set the *JAVA_HOME* environment variable to the current installation of the JDK. Then all preconditions are fulfilled and you can start Tomcat by launching the file *startup.bat* in the bin directory.

To deploy a servlet you first have to compile the servlet and then place it into the destination directory. # The destination directory can be any subdirectory of *webapps* named *classes*. A possible directory to store the class file of the servlet is the *jakarta-tomcat-4.0.1\webapps\examples\WEB-INF\classes* directory. If you are using a jar file that contains all necessary class files you have to deploy it in *..\WEB-INF\lib*.

Chapter 7

Goals vs. Effective Done

After introducing the chapters we start with a very exciting chapter concerning projects that always get done in another way than they were meant to. During school all of us heard from projects, their milestones and finally their fulfillment. Did you ever wonder where they got these information? This chapter offers a valuable experience which is the difference of commercial development and work in the non-commercial or scientific area. Look in detail at the changing requirements in the commercial development compared to the fix requirements for the implementation at school.

7.1 Commercial Version

In the following I enumerate some milestones for the development. Due to lacking time schedules for milestones the description does not contain information on missed milestones. Instead I draw a dim picture of the early expectations and the final result. For a detailed project requirement specification please refer to section 4.1.

7.1.1 Goals

The first expectations of CenterPoint were to create a time registration system enabling the employee to log his/her working times independent to the current location. This implied the usage of mobile devices running programs that offered such a facility. The mobile devices should communicate with an application server within the company. On this proprietary application server the server for the database access should be running. So we have a three-tier-architecture with palm devices as frontend, an application server as middle layer and a server for database access as backend.

Although during the first evaluation talks a new LAN-based time registration system was mentioned no note about it can be found in the system requirement specification. But we will discover that it was a very important

and time-consuming task during my work at CenterPoint.

7.1.2 Effective Done

The first week I used to study the Palm C API. I tried to get some examples up and running and was focussed on how to create a simple GUI with perspective of the challenging time registration system client for Palm OS. In this phase I encountered some GUI related problems that could lead to further troubles in the implementation for the client. So the first week passed over and I had gained a little insight into the Palm GUI programming.

In the second week the lead software architect of CenterPoint and I had a meeting on the design of the system architecture. In the end we created some design documents based on our ideas. Then it was clear that the first step in creating this system was to implement the server. So I started focussing on the server after my first experiences with the palm development.

In the following weeks I focussed on server development. Due to our three-tier-architecture I had to work with the CenterPoint proprietary application server. The time registration server would be a Dll that is hosted from the application server. This relation forced me to investigate the application server with its existing samples. So another week went by where I spent my time setting up the application server and trying to get all things running which were mainly already existing samples and some little scripts I wrote.

Then finally I finished the design phase for the server and began with the implementation. My first thoughts about the implementation phase especially the time frame for the server were scattered during my first experiences with the MFC and its database support. Challenges I expected to be solved quickly laid like a curse on me and consumed a lot of time. So the server development came to an expense of more than three weeks. Although it also included the creation of test suites that were used for automated testing. For details on automated tests please see section 5.4.

Finally the server was up and running and all obstacles were scattered into pieces. But as already mentioned in the beginning the real business world and its requirements are changing fast and so my next assignment was quite out of scope - as programmers would say. Instead of turning my focus to client implementation again I had to work out a database extension solution for a customer. They had a little problem with an existing Fabasoft installation that was running out of space. So my job was it to create a program that would copy the data from a source database into a target database. No problem, if you are working with little MS-Access databases. But they used MS-SQL-Server engines with tons of data. Thus the program had to comprise a sophisticated transaction concept in case anything would fail during the copy process. Anyway, this solution was consuming time I needed for the time registration system and therefore was another reason why the original objectives for the time registration system were missed.

After I succeeded in creating a solution for this problem I was looking forward

to the client implementation on Palm OS. Again there was the ever changing business world and the plans shifted. The fact that the current database needed to be updated quite often from for each employee and that there was avoidable work in the administration when sending mails with requests for the monthly extract and the following work lead to the decision of prioritising the creation of a LAN-based solution in opposite to the solution for mobile devices. So the second half of my time at CenterPoint were used to create a user interface based on CenterPoint Server Pages (This technology is comparable with Java Server Pages). During my implementation of a LAN-based time registration system we discovered some minor bugs in the scripting language we used. This bugs and the resulting sessions with the lead software architect lead to further insight in the development of scripting languages and potential problems but also spent valuable time I would have needed for outstanding implementations. After more than three weeks development the basic functionality of the LAN-based time registration system worked fine and was tested.

Although I had created some quite cool applications that worked fine the original assignment was still unfinished - the implementation of a mobile solution to track working times everywhere. Finally I had time to focus on the implementations on Palm OS. The biggest problems in this section were the rudimentary C API provided from Palm and some tough memory management troubles that resulted from the API. Especially GUI programming was a pain in the chest at that time. Wrapper classes needed to be created and consumed an essential part of the time. But the end of the summer holidays draw nearer and the mobile applications was miles away from being complete. And that was also the state at the end of the holidays: uncompleted mobile application. The mobile application would still need a lot of time because there was still a solution for the data storage to implement and what would even be more complicated was the communication with the application server via SOAP. Another software developer at CenterPoint, Christian Gaal had already created a class library supporting the communication via SOAP. I would still need some time to get used to the framework and afterwards accomplish the communication with the server which would be quite tricky.

7.1.3 Conclusion

The main reasons for the unfinished work lay in out-of-scope assignments as well as further requirements that hadn't been mentioned so far. Speaking of the server implementations there were a lot of problems concerning database access with MFC. This resulted in a lot of experience but cost much more time. Expected time schedules concerning the Palm implementations were partly missed due to tricky details in the Palm memory management and graphics programming.

Nr	Milestone	Deadline	Done
1	Project Requirement Specification + Use Cases	11.11.	18.11.
2	Analysis and Design Phase	23.12.	23.12.
3	Servlet with DB-Access	31.1.	15.2.
4	Server Documentation; Palm Client	3.3	17.3.
5	Server-Client-Communication; End of Implementations	5.4.	26.4.
6	Test Phase End; Project End	30.4.	30.4.

Table 7.1: Project Milestones

7.2 Non-commercial Version

This section covers the time schedules for the non-commercial version and the final outcome. It describes the fulfillment or miss of a milestone and what kind of problems lead to the delay.

7.2.1 Goals

The intention of the non-commercial version was to let me re-implement the system in another language enabling me (a) to improve some decisions I made in the commercial version that proved to be bad and (b) to compare the two programming languages I used.

For detailed system requirement specification see section 4.2. The system's requirements based on the commercial version but were reduced in several aspects.

7.2.2 Effective Done

In the following I present a table that lists the project milestones and the time the work was delivered. The concentrated reader will recognize some differences which are mostly related to technical problems.

Ad 1) The final handout was on the 18.11. but I already worked out the most of the use case diagrams on the given deadline. The system requirement specification was finished on the date, too.

Ad 2) At the given deadline the first design documents were already completed. Due to a shortage of details in the sequence diagram as well as the class diagram I was allowed to finish my work over the Christmas holidays. One week after the Christmas holidays the whole set of documents was delivered.

Ad 3) Due to delays in the previous project phase the servlet implementation couldn't be delivered at the given deadline. But there were also technical problems that bothered me. Tomcat was chosen as servlet-container. Immense path problems occurred due to a mess of batch files used to start and shut down the server. After I solved the path problems I got stuck in accessing the oracle database. The solution to the oracle database was downloading a new driver from the oracle homepage which solved the problem.

Ad 4) As already mentioned above a delay in one project phase leads to follow-up-delays. This was the problem here.

Ad 5) Again several problems with Tomcat server occurred.

7.2.3 Conclusion

In opposite to the commercial version I made a time schedule planning all the milestones and delivery dates.

Appendix A

Glossary

Task The collection of data used to store the working time of the coworker. It contains the the assigned project and the currently project phase (workcode), too.

Workcode The current project phase e.g. implementation, test. The granularity of workcode varies from implementation to support.

SOAP Is a communication procedure. XML is used as protocol and HTTP is used as transfer layer.

XML eXtended Markup Language. It enables you to create a generic protocol that you may map to another protocol if it is necessary with a mapping mechanism (XSLTD).

HTTP Hyper Text Transfer Protocol. It is used for communication in the world wide web.

Mobile Client The device you use e.g. a PalmIIIc

PDA Personal Digital Assistent also known as handheld

TRS Time Registration System

URL Unified Resource Locator

GUI Graphical User Interface

Dll Dynamic Link Library - Enables a programmer to load program code dynamically during the program is running in opposite to statically linked libraries.

J2ME Java Micro Edition

IDE Integrated Development Environment - Is a graphical user interface supporting fast development of software by utilizing an underlying toolkit.

API Application Programming Interface - Set of routines, protocols, and tools for building software applications.

MFC Microsoft Foundation Classes - A C++ class framework provided from Microsoft to enhance and simplify the development of Windows-based applications.

UML Unified Modelling Language - is a standard for object oriented analysis and design. UML is a language for specification, visualization, construction and documentation of software-system models, use cases and other non-software systems.

JDBC Java Database Connectivity - A facility to communicate with a database via .

Bibliography

- [PalmComp] Programmers Companion, Document Number 3004-003,
unknown publisher, ed. 2000
- [PalmRef] Palm OS SDK Reference Document Number 3003-003, unknown
publisher, ed. 2000
- [CePo02] CenterPoint/Base,
<http://www.cpointc.com/html/products-base-en.html>, 11 April 2002
- [Jak02] Jakarta Project, <http://jakarta.apache.org>, 2 March 2002
- [SOAP] Simple Object Access Protocol 1.1, <http://www.w3.org/TR/SOAP>,
15 April 2002