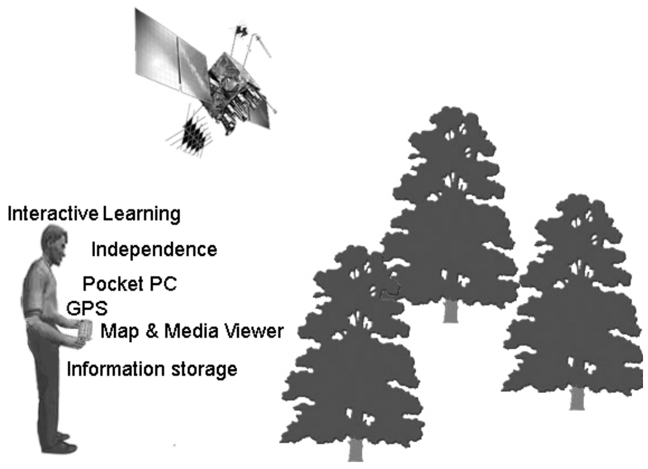
Umeå University, Sweden
Department of Computing Science
Master Thesis Project, Vt -2004
Supervisor: Thomas Johansson

2004-10-15

Using augmented reality as a tool to enhance forest excursions



Author: Jan Svensson

Dedication

I wish to dedicate this report to the memory of my close friend Per-Olof Sehlstedt, founder of MittCon AB Sweden, who passed away in lung cancer at the middle of the project. I want to acknowledge and thank him for his great heart and endless support and whom played a significant role throughout my graduate studies. He consistently encouraged and inspired me. I owe him a special debt of gratitude and thanks for his extraordinary and never failing support and presence. My sorrows are everlasting.

Abstract

The master thesis project "Using augmented reality as a tool to enhance forest excursions" is about the process of developing a technical teaching aid, primarily based on Pocket PC and GPS. The project is an integration project where new code extends an existing application, a map viewer, called LocBrowser. The main challenge in the project was to investigate if it was possible to develop a system, based on augmented reality, a system to be used in forest excursion and that enables media viewing and interactive learning. If realizable, finding appropriate technologies and developing a prototype or parts of the system was the objective.
The result of the project is as follows:
• The project was not fully implementable with the given prerequisites. Lack of access to complete source code and debug version of code were the major reason of why the project was not fully implementable. The code access restraints referred to intellectual property rights in connection to Gil Müllers (founder of LocBrowser) future market- and business plans.
• Thorough research of different technologies ensured appropriate technologies were selected and integrated into the system.
• The main parts of the system on the PC side were implemented.
• The main parts of the system on the Pocket PC side were implemented.

Key Words

Computing Science, Electronics, Teaching Aid, Pocket PC, GPS, C, SQL, HTML, ASP, Pocket Access DB, XML, ADO, ADOCE, ODBC, OLE, Embedded systems, Hardware programming, Embedded Visual Studio, PC programming, Windows programming, Database, Computer Communication, Graphical User Interface, Administration Systems, Optimization, Code density

Acknowledgements

In the writing of this thesis, I have been greatly assisted by a number of people in various ways. I would like to acknowledge and thank them all. In particular, I wish to thank my supervisor Thomas Johansson, Department of Computing Science, Umeå University, for his guidance, advice and dedication throughout the writing of the thesis and for reading and providing me with useful comments regarding various parts of the thesis.

I also want to thank Dr. Mikael Sjöberg CUT/SLU (Swedish Agricultural University) for his flexibility, giving me free hands solving the problems, and for letting me remotely manage the Project from Härnösand, and also for bringing me an exciting and motivating project based on his great idea.

The staff of Institution of Computing Science and the Institution of Applied Physics and Electronics, Umeå University, deserves a special mention for their support, their presence and encouragement throughout my studies at the Institute. I am greatly indebted to all of them and thank them individually and severally for their support and encouragement.

I wish also to acknowledge and thank Researcher Gil Müller, founder of Loc-Browser, for his hints and for answering many of my questions, and Dr. David P. Stern for a thoroughly explanation of the geographical coordinate system and GPS navigation.

I wish also to acknowledge and thank my friends; Göran Harning and Olof Burman, good listeners with lots of empathy when struggling through the hard parts of the project, Thomas Lundström for his optimism and great social ability and for supporting me with a brand new Keyboard, making me write the report even faster. I am greatly indebted to them and wish them good luck in the future.

I finally thank my girlfriend Christine, my parents and my family for their untiring support and encouragement, without whom this thesis would not have been possible.
However, while thanking all the individuals mentioned above, full responsibility for the views expressed in this thesis rests with me.

Table of content

1 List of figures

2 Abbreviations

(ADO)	ActiveX Data Object
(ADOCE)	ActiveX Data Objects for win CE
(API)	Application Programming Interface
(AR)	Augmented Reality
(BLOB)	Binary Large Object
(BT)	BlueTooth
(COM)	Component Object Model
(CS)	Computing Science
(CUT)	Center of Education Technology
(DBMS)	DataBase Management System
(FSO)	File System Object
(GPS)	Global Positioning System (receiver)
(IR)	Infra Red
(ISAPI)	Internet Server API
(ISR)	Interupt Service Routine
(MTP)	Master Thesis Project
(ODBC)	Open Database Connectivity
(OLE)	Object Linking and Embedding
(PC)	Personal Computer
(PPC)	Pocket Personal Computer
(PDA)	Personal Digital Assistant
(SIP)	Supplementary Input Panel
(SLU)	Swedish Agricultural University
(SLUEA)	SLU Excursion Assistant
(SLUEAM)	SLU Excursion Assistant Manager
(SLUEIS)	SLU Excursion Information System
(SLUEDB)	SLU Excursion DataBase
(SLUIDB)	SLU Information DataBase
(SLUUDB)	SLU User DataBase
(SQL)	Structured Query Language
(URL)	Uniform Resource Locator
(USB)	Universal Serial Bus
(VB)	Visual Basic
(XML)	eXtensible Markup Language

3 Introduction

This work proposal has its origin in cooperation between SLU (Swedish University of Agricultural Sciences) and CUT (Center of Education Technology) and is initiated and supervised by Dr. Mikael Sjöberg. The Master Thesis Project is performed at D-level, 20p and includes an in depth study of about 3p with the subject Optimizing code for embedded systems. The internal supervisor is Mr. Thomas Johansson, Lecturer, Department of Computing Science, Umeå University.

4 Background

4.1 A brief historical overview

Forest excursions have almost been performed the same way for decades. The teacher takes the students to a place where the studies are about to be performed. The teacher talk about the actual subject and hopefully all students pay attention. Not much technical teaching aid has been used before to support this difficult teaching area.
The Global Positioning System (GPS) is a worldwide radio navigation system formed from a constellation of 24 satellites and their ground stations.
GPS uses these 24 satellites as reference points to calculate positions accurate to a matter of meters.
These days GPS is finding its way into cars, boats, planes, construction equipment, movie making gear, farm machinery, even laptop computers.
Handheld computers such as Palm Pilot and Pocket PC:s are becoming more and more powerful. These are now powerful tools for professionals, whether in or out of the office. Opportunities to create and edit documents, email, surf the web, display images, play videos, view and store information, makes the hand-held computers very usable in a variety of situations.
Could these two systems maybe facilitate and be applied to teaching and forest excursions? [1,2,24]

4.2 Problem

As described above, forest excursions could be difficult to perform in an effective manner. When the teacher talked about the actual subject he could never be sure that the students paid attention or even heard what the teacher said. There was also no easy way of mixing theoretical and practical information and the lack of interaction made excursions less exciting. Another disadvantage was that the current teaching method did not have any control mechanism of student's knowledge.

Dr. Mikael Sjöberg realized the problem and was eager to solve it, enhancing forest excursions and by that taking the teaching to a higher level. He believed that the problem could be solved by using modern technology such as GPS and handheld computers packed with intelligent software, so he stated a project specification which included a description of the problem and an initial starting point. That's the point were the responsibilities were handed over from SLU/CUT to the author.

4.3 Challenges

The challenges and main objectives in this MT project was the following:
• Finding out whether or not the project specification was realizable and implementable and by that solving the problems anticipated by Dr. Mikael Sjöberg (described above).
• Finding appropriate technologies (databases, transfer technologies, programming and scripting languages etc) compatible with each other in a way that the dataflow between the different platforms could be guaranteed.
• Creating a demo system or a part of the system.
• An indirect, informal challenge was to find a framework or project model that supports one person manage a large software project by his own, from start to end, including risk management, requirement analysis, analysis and design, implementation, testing etc, as well as the integration work.

4.4 Benefits if the problems are solved

If the system could be implemented the following benefits would be achieved:

- Enabling of new media formats as images, videos, voice-clips etc, earlier the media was restricted to paper formats.
- Enabling of control systems, controlling user's knowledge. As the user data from the excursion is stored on the handheld, it easily can be transferred back to the PC where teachers have the opportunity to control the stored information.
- Enabling of reusable excursion data, by using a database.
- Enabling interactivity, which makes the excursion more exciting.
- Enabling independence; the student is not as dependent as before regarding the point in time when the excursion is performed. The user can perform it by himself whenever he have got time. The same yields for the teacher, the teacher can gather information and build the excursion route whenever he wants.
- Gives the student some additional learning occasions. The occasions are the following: When viewing the information on the handheld, when interactively answering questions about the actual subject by filling in html forms, and finally when making reviews at the school after the excursion.

5 Prerequisites

The equipment available when starting the project was a PC with USB connector, a handheld computer, a GPS receiver and a web camera. Everything but the PC was supplied by SLU. My current knowledge connected to the MTP was relatively good, the knowledge about navigation and GPS was on a basic level, as well as the knowledge of handhelds, GUI development and administration systems. I had great knowledge and experience in working with object oriented -analysis, -design and -programming. I also had experience in implementing database solutions and developing software for embedded systems. I had basic experience in project management.

The things to my disadvantage in this MTP were the following:

- New hardware – I have never before programmed a handheld computer, especially not such an intelligent hardware as the Siemens Pocket Loox 600.
- New Development Environment – Normally in school, we are programming in a non Windows environment, which is much simpler than this case, an embedded version of Visual Studio.
- New Operating System – It was the first time ever programming for Windows CE 3.0, Pocket PC 2002 version.
- New Software – The limited hardware on handheld computers affect all inherent applications such as image handling programs, transfer programs and viewing programs. The limited hardware often forces the application makers to only include a subset of the functionality and API:s, consequently it require some extra attention when using it.
- Lack of Internal Knowledge - Students making their MTP on a company often have the company's personnel as a great source of knowledge related to their subject. By doing my computing science MTP in a non computing science area (SLU), I could not demand or expect such support (their main area is forestry).
- Absence of Social network – By moving from Umeå to Härnösand my natural Social Network was a bit disabled. It was hard to get in contact with students, teachers, internal and external supervisors.
- The founder of the application LocBrowser was localized in Germany. It could be hard to establish and maintain an effective correspondence.

6 Restrictions and Limitations

This MTP have been limited to only comprise research and implementation conducting to a prototype with limited functionality. No focus has been put on peripheral systems such as help and support systems. The subsystems are also only certified and tested on the following specific hardware: The (SLUEAM) SLU Excursion Assistant Manager on a PC with Windows XP and MS Internet Information Services 5.1. The (SLUEA) SLU Excursion Assistant on the handheld side using the following Hardware; Pocket Loox 600 from Fujitsu Siemens (ARM CPU). No guarantee is given that the system works under other circumstances.

7 Theory - Important concepts

7.1 General

Before continuing let have some concepts clarified, concepts that simplifies the further reading if familiar with.

7.2 Augmented Reality

Augmented Reality (AR) is a growing area in virtual reality research. The world environment around us provides a wealth of information that is difficult to duplicate in a computer. An augmented reality system generates a composite view for the user. It is a combination of the real scene viewed by the user and a virtual scene generated by the computer that augments the scene with additional information. In all AR applications the augmented reality presented to the user enhances that person's performance in and perception of the world. [24]

7.3 LocBrowser Application

The LocBrowser Basic Edition is an application for Pocket PC developed by Gil Müller, Germany. The application is a map viewer suited for outdoor activities, an application that assists people in finding their route. It supports digital maps and supports real time navigation when combined with a GPS receiver. It supports GPS receivers with NMEA 0183 output and is available for Windows CE 3.0 (Pocket PC and Pocket PC 2002). LocBrowser Basic Edition is the application that in this MTP have been; extended with more functionality, renamed to SLUEA, and constitutes an important part of the SLUEI System. [21]

7.4 ActiveSync

Microsoft ActiveSync is a synchronization software for Windows mobile based Pocket PCs and Smartphones. ActiveSync allows the user to create a partnership between the mobile device and desktop computer using a cable, cradle, or infrared. After the partnership is created, the user can use his existing computer to connect to other resources through ActiveSync. ActiveSync helps to keep information up to date on both handheld and PC. If you make a change one of them, the next time you synchronize, the change is automatically made to the corresponding information on the other one. [19]

7.5 MS Device HTTPD Server:

MS Device HTTPD Server is a Web server that supports a subset of the HTTP/1.0 protocol, including limited functionality, status-code subset, and fewer header-field definitions. The Microsoft HTTPD Web Server is an optional operating system component and Windows CE running on current Pocket PC 3.0 devices does not include this component in ROM.

The Web servers Head and Get functions are supported directly. ISAPI filters, ISAPI extensions, and ASP pages all have access to data that can be read from the POST header and body. The Web server does not use data from the POST header and body directly. All other methods are ignored. [19]

7.6 .NET Compact Framework

The Microsoft .NET Compact Framework is a development framework for intelligent devices that brings managed code and XML Web services to devices. The Compact Framework is a rich subset of the .NET Framework, thus providing the same benefits as the .NET Framework; but it is designed specifically for resource-constrained devices, such as PDAs and smart mobile phones. The .NET Compact Framework has two main components: the common language runtime and the .NET Compact Framework class directory. The Compact Framework simplifies the process of creating and deploying applications to mobile devices while still allowing the developer to take advantage of the capabilities of the device. [19]

7.7 ADO, ADOCE, ODBC, OLE DB

7.7.1 ADO

Microsoft ActiveX Data Objects is a high-level interface to all types of data. ADO provides consistent, high-performance access to data, whether creating a front-end database client or a middle tier business object using an application, tool, language, or Internet browser. ADO is part of the Microsoft Universal Data Access model, along with OLE DB and open database connectivity (ODBC). ADO is the object model built over OLE DB. [19,23]

7.7.2 ADOCE

The ActiveX Data Objects control for the Microsoft Windows CE operating system provides a subset of ADO for Windows CE. ADOCE enables access to databases stored locally on a device instead of databases stored on other devices. For example, a docked Windows CE-based device cannot use ADOCE to access an SQL Server running on a network. ADOCE includes its own internal database provider. [19]

7.7.3 ODBC

Open DataBase Connectivity is a standard database access method developed by the SQL Access group in 1992. The goal of ODBC is to make it possible to access any data from any application, regardless of which database management system is handling the data. ODBC manages this by inserting a middle layer, called a database driver, between an application and the DBMS. The purpose of this layer is to translate the application's data queries into commands that the DBMS understands. For this to work, both the application and the DBMS must be ODBC-compliant -- that is, the application must be capable of issuing ODBC commands and the DBMS must be capable of responding to them. [23]

7.7.4 OLE DB

OLE DB is Microsoft's low-level application program interface (API) for access to different data sources. OLE DB includes not only the Structured Query Language (SQL) capabilities of the Microsoft-sponsored standard data interface Open Database Connectivity (ODBC) but also includes access to data other than SQL data. OLE DB is a set of methods for reading and writing data. The objects in

OLE DB consist mainly of a data source object, a session object, a command object, and a rowset object. [19]

7.8 GPS

The Global Positioning System consists of 24 satellites, circulating in orbits, and their ground stations. The satellites continuously send positioning signals that GPS receivers on earth can receive. The signal receives through the GPS receiver's antenna and then continues through an RF-Chain. The analogue signal is then converted to a digital signal via the DAC and after some further stages it finally calculates the user position. Please refer to figure 1. [25,26,27]

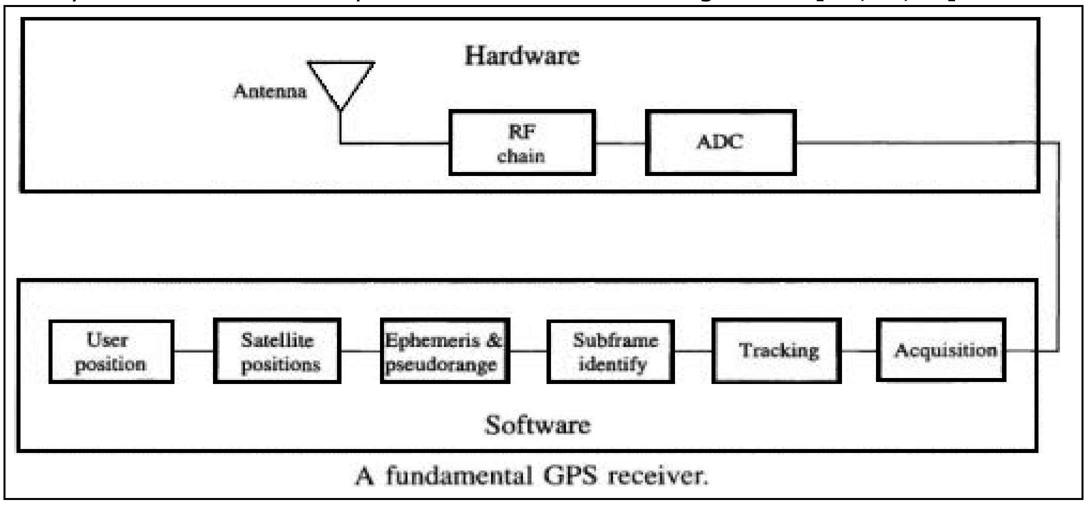


Fig 1: *A fundamental GPS receiver.*

7.9 Win CE, Pocket PC, Pocket PC 2002

7.9.1 Win CE

The distinctions between what's Win CE, Pocket PC and Pocket PC 2002 can be hard to distinguish. Windows CE is a portable, real-time, modular operating system that features popular MS programming interfaces and that is supported by tools that enable rapid development of embedded and dedicated systems. Windows CE is a complete new operating system and not as many believes a ported and optimized Windows. Supported CPUs: Arm720T, DEC SA1100, Hitachi SH-3,Intel i486 & Pentium, AMD Elan SC400, Motorola MPC821. [2,5,19]

7.9.2 Pocket PC

Pocket PC is an operating system produced by Microsoft for PDA's. Pocket PC is based on Windows CE 3 and is a mobile operating system that features built in MS Office Pocket products as well as Media Player and Internet Explorer. Pocket PC is a standard configuration using Win CE 3.0, and Pocket PC does not have a keyboard, instead it supports written character input using SIP and MS Transcriber. It has a cleaner 2D look than previous Win CE versions and is greatly enhanced by both software and hardware. [2,3]

7.9.3 Pocket PC 2002

Pocket PC 2002 is a revised edition of Pocket PC. PPC 2002 only supports ARM based processors, but comes with many new upgrades and features. [19]

8 Method

8.1 General

To manage such a massive and complex project as this MTP, an important task was to find an appropriate Project model or framework that support one person manage the software project by his own, from start to end, including all associated tasks. The Method section is organized the following way. It follows the project time plan, so all the tasks and problems are ordered chronologically just as they appeared in reality. The reader can follow the project time plan throughout this section and make analogies as he continues reading. Note that the method section only specifies how the work was accomplished, and tells nothing about the system.

8.2 Selecting Project Model

The first step was to select a project model, a project model that supports one person manage a huge software project by his own from start to end. The project managing models considered were all models found in project management literature:

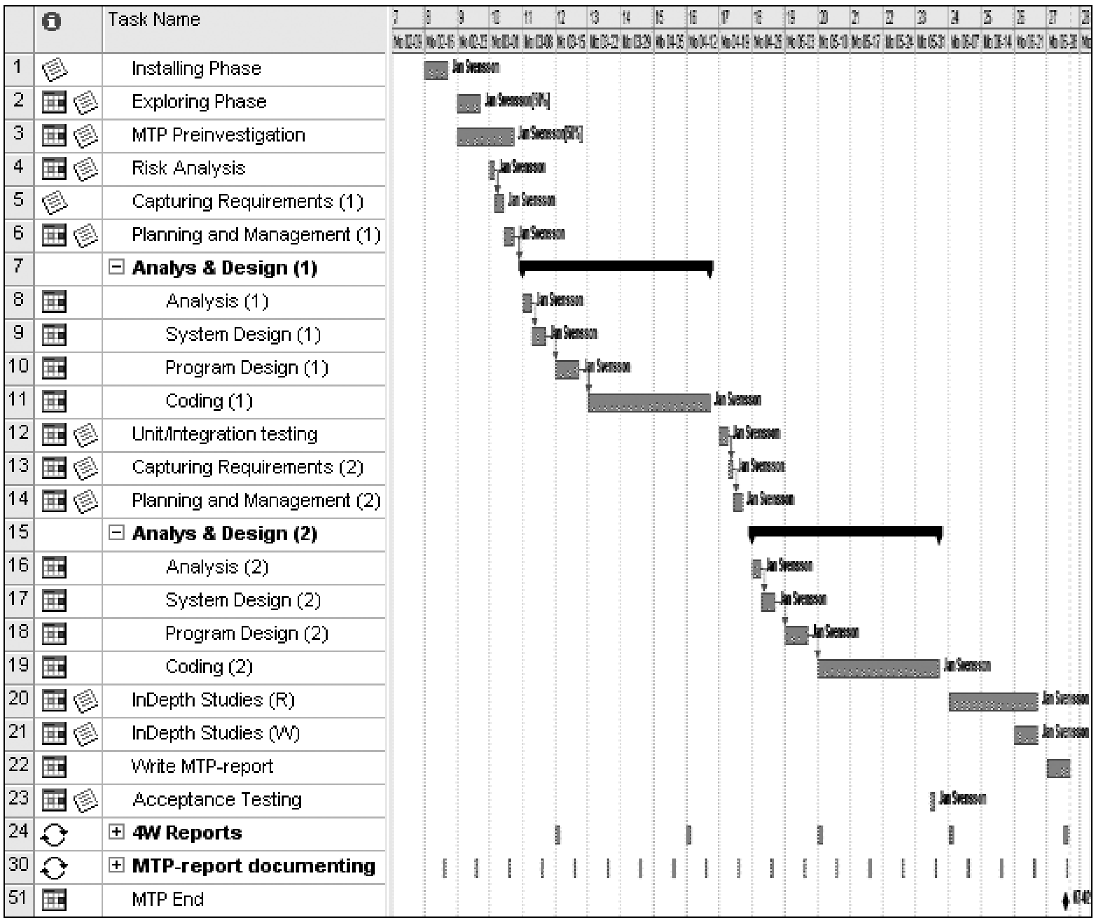
- Extreme Programming project model
- Waterfall model
- V-model
- Spiral model

The Extreme Programming project model seemed not to be a good choice because it required a close and frequent relation to the customer. This would have been hard to achieve as the customer was localized in a different city. The Extreme Programming model also advised pair programming, which is obsolete in a one person project.

The waterfall model was excluded cause of its shortcomings in treating software as a problem solving process and the insensitiveness of changing in requirements. The V – model appeared to be better than the Waterfall model but was excluded at the end because of the better and more advantageous Spiral Model. [9]

The Spiral model, with two iterations was selected because of the following reasons. As the project plan was started, basic ideas of how it would look like already existed, and the spiral model seemed to be the appropriate tool. The most programming projects perform at a two or more iteration programming cycle, and I had already performed some projects this way with success. The first iteration should catch and implement the main requirements and served as an estimation of time to the second iteration. A preparing initial stage was also natural, which included risk management and requirement capturing. The final stage included different levels of testing. The in-depth studies and theoretical parts were placed at the end of the project and writing on the MT Report was scheduled as a recurrent task distributed all over project time. Please check the project time plan in figure 2.

After selecting a project model, a project time plan was constructed. The tasks in the time plan followed the process model cycle, adapted to the projects special requirements and prerequisites.



	Task Name
1	Installing Phase
2	Exploring Phase
3	MTP Preinvestigation
4	Risk Analysis
5	Capturing Requirements (1)
6	Planning and Management (1)
7	⊟ **Analys & Design (1)**
8	Analysis (1)
9	System Design (1)
10	Program Design (1)
11	Coding (1)
12	Unit/Integration testing
13	Capturing Requirements (2)
14	Planning and Management (2)
15	⊟ **Analys & Design (2)**
16	Analysis (2)
17	System Design (2)
18	Program Design (2)
19	Coding (2)
20	InDepth Studies (R)
21	InDepth Studies (W)
22	Write MTP-report
23	Acceptance Testing
24	⊞ **4W Reports**
30	⊞ **MTP-report documenting**
51	MTP End

Fig 2: *Project Time Plan*

8.3 Installation phase Setting up Dev.Environment

8.3.1 General

The aim of this phase was to successfully install and set up the development environment, establish an information channel so that people outside the project could watch the current status of the project, and finally succeed compiling the source code. It did not turned out as planned. The start of the project was disastrous. The PC system: (Intel P4 2 GHz, Win XP Pro, 512 Mb Ram, 2*80Gb Raid 0 Array system) did not seemed to be compatible with the Embedded Visual C++. Blue screen of death, repeatedly unexpected restarts, hardware failures, were all unwanted effects that occurred. Problems never experienced before. The raid array in conjunction with the complex developing environment (including HW emulators) was not a good combination. Unmounting the raid system, reinstalling Win XP and the Development Environment on normal IDE-drives, solved the problem.

Another frustrating problem referred to the installation of Embedded Visual Studio was to find out what caused the error messages: "TrueVector engine:

File "C:WINDOWSInternet LogsSTORA.ldb" was corrupt and has been copied to "C:WINDOWSInternet LogsxDB2.tmp". By analyzing the system logs enough information were assembled to identify the problem. A conflict between the Software Firewall and MS Embedded Visual Studio, a disabled firewall solved the problem.

8.3.2 Establishing Information Channel

Developing a homepage for the MT-Project was highly prioritized, that is to ensure the supervisors having insight to the project.
The site was developed using MS Frontpage and the site supported PHP and mySQL which gave me the opportunity to implement a comfortable tool for managing and updating information continuously. Information was published frequently until the day the project bumped in some major problems and all the time was dedicated to solve these problems.

8.3.3 Compiling and building

The first attempts to compile and build LocBrowser ended up with a heavy crash of the computer. Problem with Memory Access to restricted areas was the caused by the emulator. By reconfiguring the environment, not using the emulator, setting the compile target to be the real Pocket PC, the problem could be solved. After that, all the subprojects compiled without error and warnings. The files automatically transferred to the Pocket PC after every build, by ActiveSync.

8.4 Exploring phase

When the initial phase was finished, when the development environment was stable and when the compiling the source was successful the project really started. That’s the point were the huge information search started. Information search about hardware, LocBrowser and developing environment, appropriate tools, correct drivers, Manuals etc. In this phase a lot of software and related manuals were downloaded and tested.

8.5 MTP-Preinvestigation

The cause of the MTP preinvestigation was to serve as a tool to check whether the MTP was realizable or not, to find possible dead ends and black holes. If no dead ends or black holes were found, another important assignment was to examine what kind of capabilities and opportunities the system hardware and software offered, what was already provided and what got to be obtained. Small different tests were performed at this phase: Supported and available transfer methods, database methods, programming and scripting technologies, readability of code and documentation etc.
An excerpt from a test result when investigating how transfers and conversions affect the original data when transferring from PC to PPC is presented below.

- Converting and transferring Word-files eng - success
- Converting and transferring Word-files swe - fail
- Converting and transferring Access-files – fail
- …

In this example, using Swedish characters appeared to be a problem.

8.6 Risk analysis and risk assessment

8.6.1 General

According to L. Pfleeger, a systematic risk analysis is crucial in big projects. The purpose of the risk analysis and risk assessment is to serve as a tool handling risks in the project. It should help identifying threats to the project and protect the project from being exposed to unnecessary risks. It should also help minimizing the potential damages associated to the risks. Vast efforts were done to:

1. Anticipate the most common risks related to the project.
2. Estimate the potential degree of damage of each risk.
3. Estimate each risks probability to occur.
4. Finding solutions to minimize the damage or avoid if possible. [9]

8.6.2 Table of Risks

A table of risk was created according to Pfleegers recommendation, where every risk got a unique assigned "Risk ID". The risk table showed the potential danger of the risks and in the table each risk (id) matched a certain "impact" and "probability" for the risk to occur. By multiplying these two numbers the risks risk number was obtained. The higher the more troublesome and higher effort was needed to handle the risk.

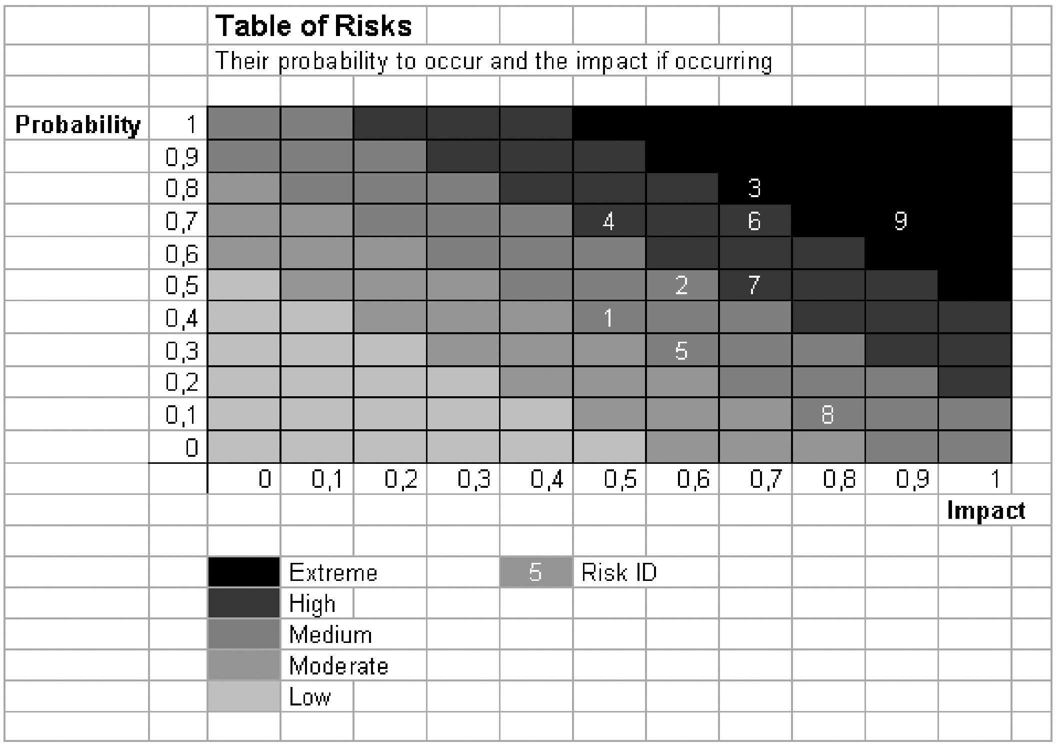


Table of Risks
Their probability to occur and the impact if occurring

Probability	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
1											
0,9											
0,8								3			
0,7						4		6		9	
0,6											
0,5							2	7			
0,4						1					
0,3							5				
0,2											
0,1									8		
0											

Impact

■	Extreme
▩	High
▦	Medium
▤	Moderate
□	Low
5	Risk ID

Fig 3: *Risk table, showing impact of risks in the project. The higher risk the darker colors. High risks must be minimized or eliminated.*

The following risks were identified

1) Indistinct, incomplete, inconsistent or unrealistic requirements
2) Possible communication problem.
3) Insufficient time to complete the project.
4) Project highly dependent on a key person.

5) No support from the environment (social network)
6) Existing code difficult to use and or extend
7) Hardware problem – HW-faults, Crashes, Reinstallations…
8) Absence of manpower - Sick leave
9) Pre study not accurate enough

When the analysis and estimation of the project risks were done the result showed that it existed several severe risks that potentially could damage the project. One of them was that the LocBrowser source code maybe was not usable, because of the following reason:
1) Only parts of the source code were released and available to public.
2) The released code was only partly documented.
3) The code was badly maintained.
4) The readability of the code was bad due to a strange system implementation.
Other high risks were about time pressure and the crucial Pre Study. Please refer to the Risk Management Document for details.

8.7 Capturing requirements

When the risk management was finished the next step was to capture requirements. The requirements were derived trough systematical analysis of the MTP-Specification and notes made at the meeting with contractor Dr. Mikael Sjöberg at SLU 2004-01-20. A requirement document was created and an excerpt of the most important requirements is listed below.

8.7.1 Requirements (General)
•Design should be adapted to teachers IT experience.
•It should be easy to add information to the database.
•It should be easy to transfer information from PC to handheld (PPC)
•Overall functionality and user interface should be as intuitive and usable as possible.

8.7.2 Requirements (PC Side)
•Hardware: System should be able to execute on PC (x86 architecture). and should not require additional system hardware.
•Operating system: The program must be able to execute on Windows XP.
•Transfer Technology: Optional, not explicitly expressed.
•Database and database engine: Optional, not explicitly expressed.
•User Interface: Aim for high usability and user friendliness, avoid technical intervention, adapt to teachers IT experience.

8.7.3 Requirements (Handheld Side)
•Hardware: System should execute on; at least (Strong Arm) Fujitsu Siemens Pocket Loox 600 and should not require additional hardware. It should also make efficient use of hardware, because of the limited system resources on a Pocket PC.
•Operating system: The program must be able to execute on WinCE 3.0, Pocket PC edition and above.

•Implementation Language: Optional, not explicitly expressed. The system may be implemented in an arbitrary programming language or script technology, for example: C, C++, Java, ASP, JSP, PHP, CGI.
•Other technologies – Optional, no explicit expressed technology required.
•User Interface: Aim for high usability and user friendliness, avoid technical intervention
•Media support: Support of the following media format (file extensions) is desirable.

- •Image format: gif, tif, jpg, bmp
- •Video format; mpg (mpeg)
- •Sound format; mp3, wav
- •Text format: txt, htm
- •Other media format: Macromedia Flash

•Interactivity: Interactivity is desirable. A user should be able to interact with the application by answering questions related to the explored excursion object. The answers could preferable be stored in a database on the handheld and after completed excursion be retransferred to PC. The purpose of the stored information is to serve as statistical material and to give the students another occasion to reflect over the accomplished excursion.

8.8 Planning and management

When both risk- and requirement-analysis were completed a foundation to draw the outlines of the rest of the project existed. In this phase the outlines was set, primary for the first analysis- and design-phase. The time needed for every task to be completed was estimated. The order in which everything should be performed was decided. Every task was given a certain priority.

8.9 Analysis and design 1

8.9.1 Analysis

According to Pfleeger L. a large problem can be described as a collection of small problems and their interrelationships, and by breaking the problem in small pieces the problems are now hopefully easier to solve. When all the small problems have been solved, then even the original problem is solved.
This fundamental method to tackle problems was used in the analysis phase. In the analysis phase system requirements and user needs were systematically analyzed. A lot of different software constellations were considered and internal correlations. Please refer to the implementation section if interested in system specific details. [9]

8.9.2 System Design

In the system deign phase the design goals of the project were defined and the system were decomposited into smaller subsystems that were easier to realize.
The aim was to produce clear descriptions of strategies, subsystem and other components. The system design originated from the requirements and from the user description (How the system was supposed to be used). One of the things that were done was to transform the abstract user description to a concrete, implementable version. Most of the ideas during this phase were sketched by

pen and paper. Please refer to the implementation section if interested in system specific details.

8.9.3 Program Design

In the program design phase, often called object design phase, a lot of sketches were made, using pen and paper; data access models, flow charts, communication models, component structures and classes. All ideas and models from the system design phase were adapted to the next phase, the implement-tation phase.

8.9.4 Coding

The mission in this phase was to implement the program designed in the previous phase.
The main coding focused on the following tasks at the PPC side:

- Menus and Event handling.
- ShellExecuteEx, taking URL:s as Arguments (Executes Explorer).
- File handling, opening and reading files (Map of: coordinates - html url:s).
- Reimplementation and fix of the virtual serial transfers using BlueTooth.
- Work with viewing of information icons.

The main coding focused on the following tasks at the PC side:

- Creating databases
- Implementing file handling systems
- Navigation systems
- Page generation system.

8.10 Unit and Integration Testing

This was the phase dedicated for unit and integration tests, a phase designated to assure every modules correctness before integrating them into the system. This was not the way it was carried out in reality. The real case looked like this: At the end of every implementation moment, for an example; implementing a database module or a transfer module, the module was tested right after implementation. The same applies to the rest of the implementations. This behavior diverged from the project time plan but as it felt natural and was fairly effective, the deviation from project time was permitted.

8.11 Iteration 2

After the unit and integration test, iteration nr 2 was to come, theoretically the same as iteration 1 but now trying to complete the entire system. The moments in iteration 2 included the following moments:

- Capturing requirements
- Planning and management
- Analysis and design
- -Analysis
- -System Design
- -Program Design
- -Coding

One of the most important reflections in iteration 2 was that all attempts of intervention with locBrowser had led to a dead end, mostly caused by void pointers to hidden functions. This was the point when a new core strategy developed. The aim of the strategy was to avoid as much intervention with LocBrowser as possible and isolate the new application from LocBrowser with a well defined API. This was also the point were the information on the MTP homepage stopped update.

8.12 Acceptance testing

Acceptance testing is a test procedure that leads to formal acceptance of new or changed systems. This phase is a critical phase of any systems project and requires significant participation by the "End Users". To be of real use, an Acceptance Test Plan should be developed in order to plan precisely, and in detail, the means by which 'Acceptance' will be achieved. This test has not been performed when the writing of this report still were in progress.

8.13 Recurring tasks

The final tasks on the project time plan included the writing on the MTP-report and the "every 4-Week" –reports. These were set as recurring tasks, only to ensure continuous writing and updating of information. The idea was good, not waiting with all writing to the end of the project, but it was truly hard to accomplish in reality.

9 Implementation and realization

9.1 Hardware

The system hardware consists of a wireless BlueTooth GPS Receiver, a handheld Pocket PC (PDA) from Fujitsu Siemens and an ordinary PC System.

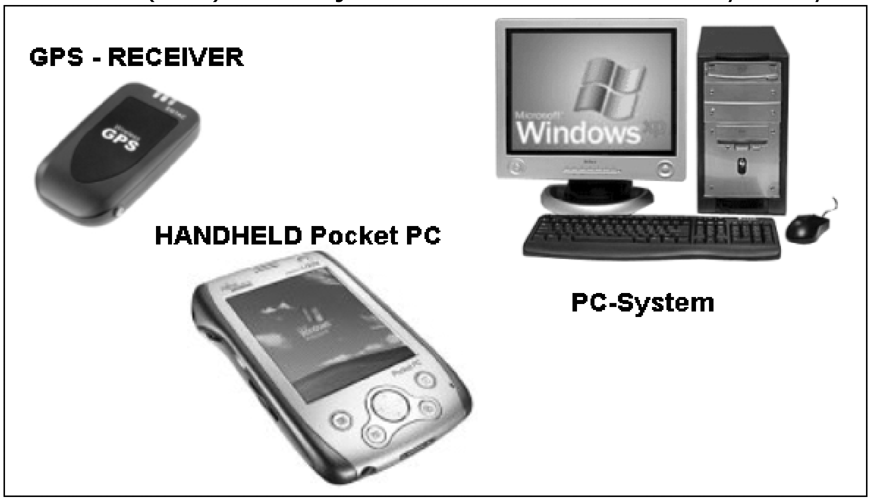


Fig 4: *System hardware. The system hardware consists of a PDA, a Personal Computer (PC) and a GPS-reciever.*

9.2 A brief system sketch

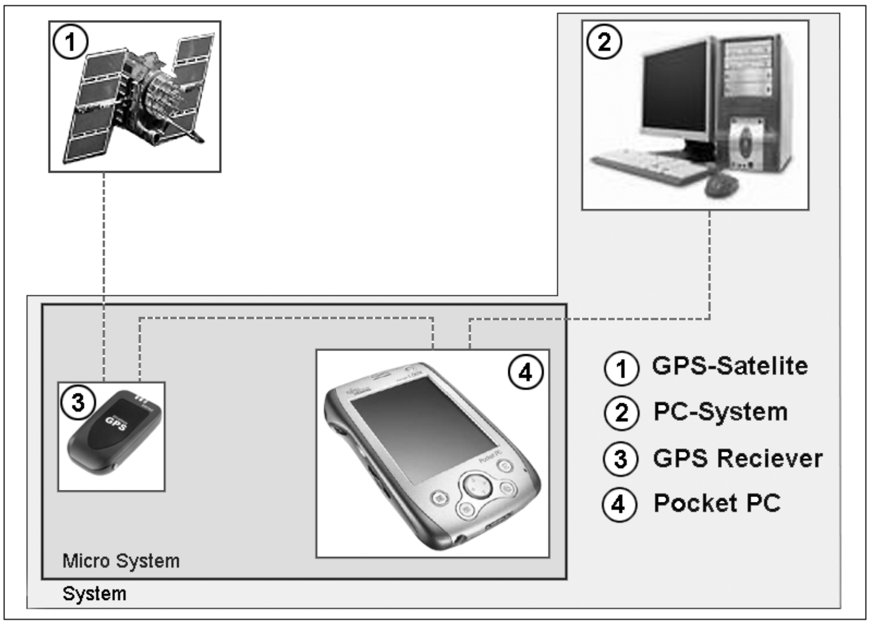


Fig 5: *The first brief system sketch including all hardware. Parts colored light gray belongs to the system. Part 3-4 colored dark gray constitutes a micro-system within the system.*

The first brief system sketch done by pen and paper looked something like this. A PC system (2) to administer information media on the large SLU Information Database and to administer data transfers between PC and the handheld PPC

(4). A GPS-receiver (3) that receives positioning information from GPS Satellites (1) and retransfers it to the handheld PPC. Please refer to figure 5. Parts colored light gray belongs to the system and dotted lines between components illustrate some sort of data exchange between the components.

9.3 Concretization of the excursion route

The following concretization of the excursion route was made: A Route is the path the user is following during an excursion occasion. The Route may contain one or several TargetPoints. A TargetPoint is a specific excursion object to be examined, for example a region of mature pine forest or a region with very bad growth conditions. A TargetPoint may include one or several InfoItems. An InfoItem is a media item containing information. It could be of different formats like images, videos, sound-clips or plain text. See figure 6 below.

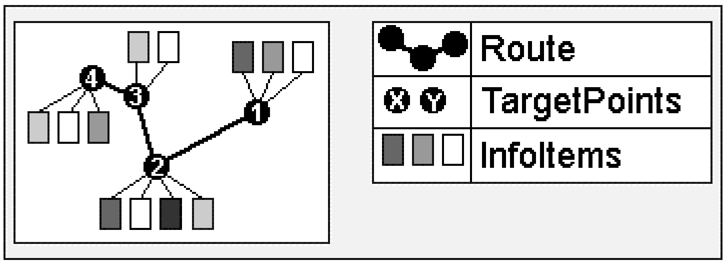


Fig 6: *Concretization of the excursion route (stage 1). Different colors on the InfoItems represent different media.*

The new more concrete version directly could be implemented as Database Tables. See figure 7 below.

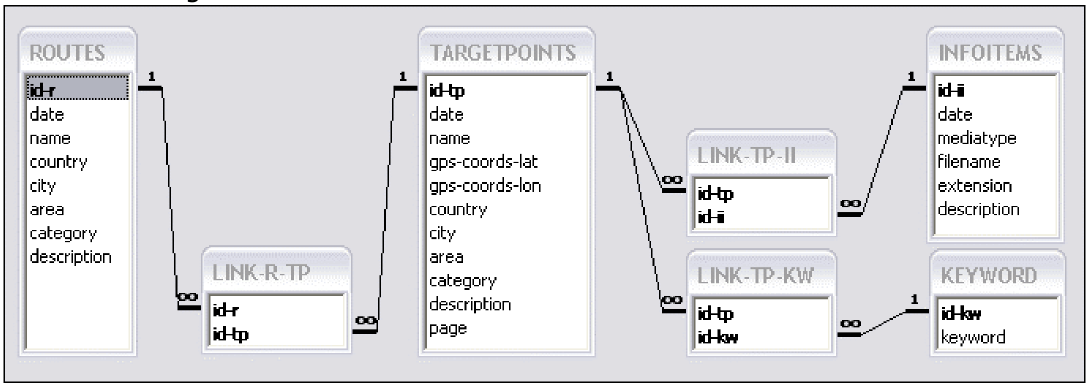
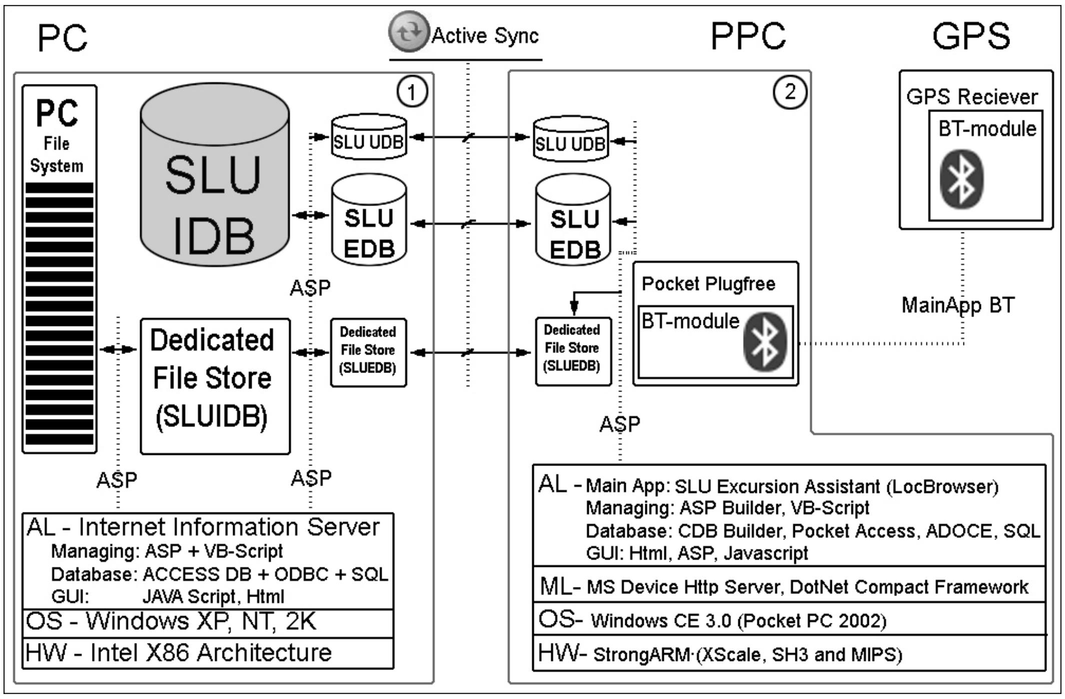


Fig 7: *Concretization of the excursion route (stage 2). Every object were implemented as own tables with appropriate attributes.*

9.4 System Overview

A lot of system parts forced me to make some naming conventions and the most of the abbreviations are explained below. After these explanations a more coherent description of the system is made.
The SLUEIS (SLU Excursion Information System) is the name of the entire information system implemented in this MTP, including all software on the PC side, as well as on the handheld PPC side. The SLUEIS is comprised of all databases, script and modules, it includes the PC side application SLUEAM (SLU Excursion Assistant Manager) and the PPC side SLUEA (SLU Excursion

Assistant). The system supplies the user with a tool to easy manage a large information database SLUIDB (SLU Information DataBase), were he can add, edit and delete information and media. The SLUEIS is equipped with an ASP-engine that generates and transfers dynamic data as Html/ASP pages. The generated ASP pages can then easily be transferred from PC to PPC, including all media files.


PC
Active Sync
PPC
GPS
1
2
PC File System
SLU IDB
SLU UDB
SLU EDB
ASP
Dedicated File Store (SLUIDB)
Dedicated File Store (SLUEDB)
AL - Internet Information Server
Managing: ASP + VB-Script
Database: ACCESS DB + ODBC + SQL
GUI: JAVA Script, Html
OS - Windows XP, NT, 2K
HW - Intel X86 Architecture
Pocket Plugfree
BT-module
GPS Reciever
MainApp BT
AL - Main App: SLU Excursion Assistant (LocBrowser)
Managing: ASP Builder, VB-Script
Database: CDB Builder, Pocket Access, ADOCE, SQL
GUI: Html, ASP, Javascript
ML- MS Device Http Server, DotNet Compact Framework
OS- Windows CE 3.0 (Pocket PC 2002)
HW- StrongARM·(XScale, SH3 and MIPS)
Fig 8: *The SLUEI System, including subsystems, databases and components.*

SLUEIS - SLU Excursion Information System
SLUEIS is the name of the entire information system, including all databases, the PC side Excursion Assistant Manager (1) and the PPC side Excursion Assistant (2). SLUEIS is the name for the complete information system.

SLUEAM - SLU Excursion Assistant Manager (1)
SLUEAM is the PC side administration system, managing transfers between PC and PPC and enabling adding, editing and deleting information from the big media database SLUIDB. It includes an ASP-engine that generates and transfers dynamic data in the form of Html/ASP pages.

SLUEA - SLU Excursion Assistant (2)
SLUEA is the PPC application name of the former, extended and modified LocBrowser. SLUEA is a map and information viewer that makes use of SLUEDB and manages SLUUDB. SLUEA is hosted on the handheld (PPC).

SLUIDB - SLU Information DataBase
SLUIDB is the big storage database, storing all information historically. It holds routes, TargetPoints, and InfoItems and all associated media files.

SLUEDB - SLU Excursion DataBase

SLUEDB is the database storing information associated to current excursion. It holds current route, TargetPoint and InfoItems including references to the actual media files.

SLUUDB - SLU User DataBase
SLUUDB is the database storing user data. Data submitted during the current excursion. Example of such data is all the answers the user submits by filling in the application html forms associated with the current excursion.

9.5 System Description

The SLUEIS consists of two subsystems. The PC side SLU Excursion Assistant Manager (SLUEAM) and the PPC side SLU Excursion Assistant (SLUEA).

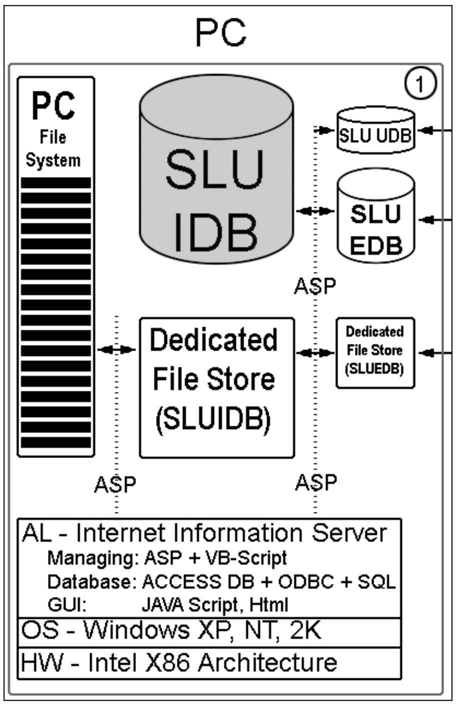

PC
1
PC File System
SLU IDB
SLU UDB
SLU EDB
ASP
Dedicated File Store (SLUIDB)
Dedicated File Store (SLUEDB)
ASP
ASP
AL - Internet Information Server
Managing: ASP + VB-Script
Database: ACCESS DB + ODBC + SQL
GUI: JAVA Script, Html
OS - Windows XP, NT, 2K
HW - Intel X86 Architecture

Fig 9: *SLUEAM including all components*

The PC side SLUEAM application is a bundle of ASP pages that executes VB script commands and by that manages all other components. It is running on the hardware platform: Intel X86 Architecture, Operating System: Windows 9X, Me, NT, 2K, XP. On the application Layer, Internet Information Services 5 is running. This system is used by the teacher to administer excursion data. The teacher can add, edit and delete media files, connect them to routes and TargetPoints. The media is stored in a large database SLUIDB that is supposed to grow every year as the teachers continue to fill it with information. Actually the media files are stored on a dedicated file store and only the references are stored in the database. When an excursion is about to be performed the teacher select appropriate excursion objects "TargetPoints" from the database and compose an excursion route. The route is then easily transferred to the handheld PPC. The transfer works this way: The SLUEAM copies files and databases to a directory that is synchronized with the PPC. As soon as the PPC is synchronized the files from the PC is transferred. Please refer to figure 9.

The SLUEAM make frequent use of the FSO component (FileSystemObject). When adding Information to SLUIDB the SLUEAM simply copies the media files from the PC file system and stores it in a directory named __SystemPC__ that is part of a dedicated file system called __System_SLUEAM__. See (1) figure 10 below. The path and references to the media files is then stored in the database

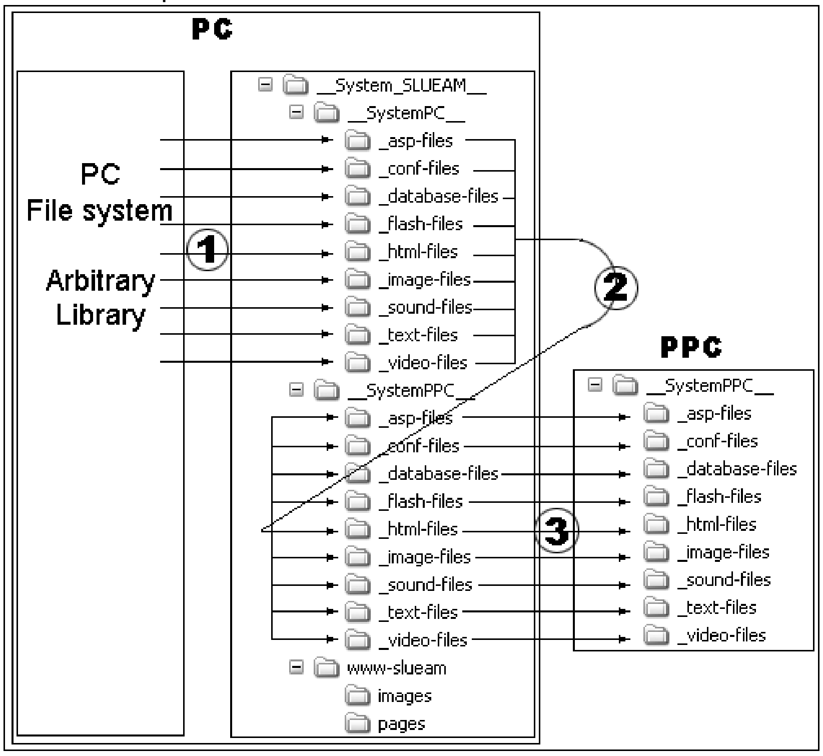


Fig 10: *Information flow PC file system to PPC*

SLUIDB.
When the user is about to transfer a route to the PPC, including media files, he simply select the route to be transferred and then press the transfer button. What actually happens behind the curtains is the following. The SLUEAM makes a Database Request of the selected route and creates a recordset with all associated media files. Then again, the FSO component is used. A copy of all media files associated to the route is performed. The target is the directory called __SystemPPC__ See (2) figure 10. If ActiveSync is activated the files are automatically transferred to a directory at the PPC with the same name as the PC side, namely __SystemPPC__ See (3) figure 10. If ActiveSync is not activated, the files will be transferred at the next occasion the handheld is synchronized. See figure 10. As you can see it may concurrently exists 3 copies of each media file. This is done purposely. The core idea is to consider the PC file system as an unsafe system, a system where files can be modified or deleted any time. The same yields for the __SystemPPC__ directory, a directory that contains files associated with the most current excursion. Consequently, the system is safe and still contains the stored files, even if the user deletes them on the PC:s ordinary file system.

The SLU ExcursionAssistant (SLUEA) is a Win32 application written in C and works as a map and information viewer. It runs on a Pocket PC 2002 device equipped with an ARM CPU, the operating system is Win CE 3.0 Pocket PC 2002.

The application requires .NET Compact framework and MS Device Http server as well as it needs Pocket CDB- and ASP-builder by InetInfoSoft.

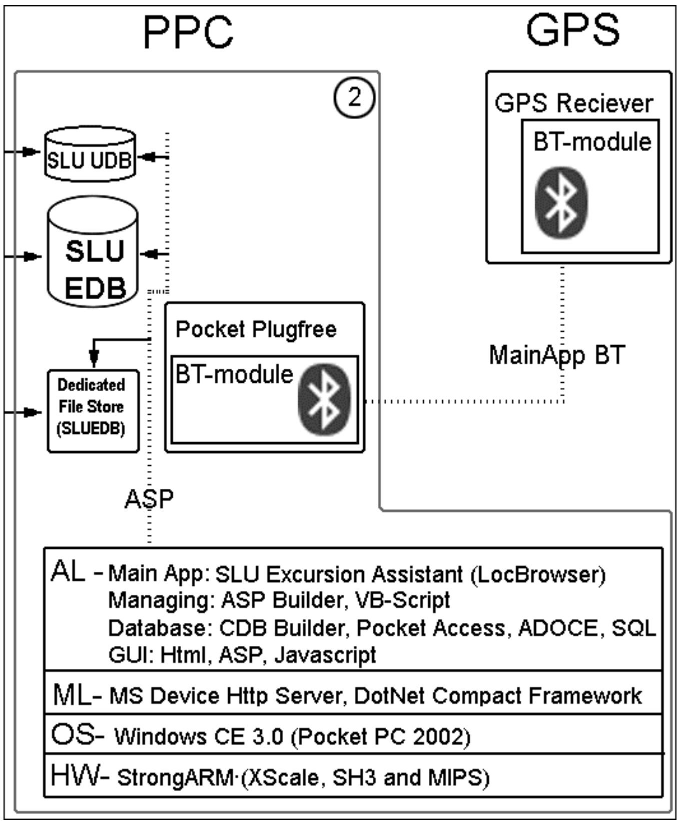

PPC
GPS
2
GPS Reciever
BT-module
SLU UDB
SLU EDB
Pocket Plugfree
BT-module
MainApp BT
Dedicated File Store (SLUEDB)
ASP
AL - Main App: SLU Excursion Assistant (LocBrowser)
Managing: ASP Builder, VB-Script
Database: CDB Builder, Pocket Access, ADOCE, SQL
GUI: Html, ASP, Javascript
ML- MS Device Http Server, DotNet Compact Framework
OS- Windows CE 3.0 (Pocket PC 2002)
HW- StrongARM·(XScale, SH3 and MIPS)

Fig 11: *SLUEA and all components*

The SLU ExcursionAssistant (figure 11) makes use of SLUEDB and SLUUDB and is hosted on the handheld (PPC). On excursions the user navigates in real time using the SLUEA and a GPS receiver. An Avatar shows the users current position on the map. When approaching an excursion object (TargetPoint) and assumed the map is correct configured, a small yellow icon can be viewed (figure 12), an information icon. When clicking on the icon a menu appears, and by selecting "view SLU info" the information connected to the current excursion object is displayed. It could be videos, images or plain text. What's actually happening under the cover is the following: The menu selection (figure 13) is handled and Pocket Internet Explorer is launched equipped with the arguments selecting the appropriate ASP-page, the page containing the information of the excursion object. The correct page is distinguished by a table look up in SLUEDB, which contains only one table, a table whose only job is to map GPS-coordinates to a unique page/url. The page viewed may contain html application forms that the teacher prepared before the excursion, forms that post user data to the SLUUDB. The SLUUDB is a static database that is overwritten every time a synchronization is done and if the PC side is about to make a new transfer.

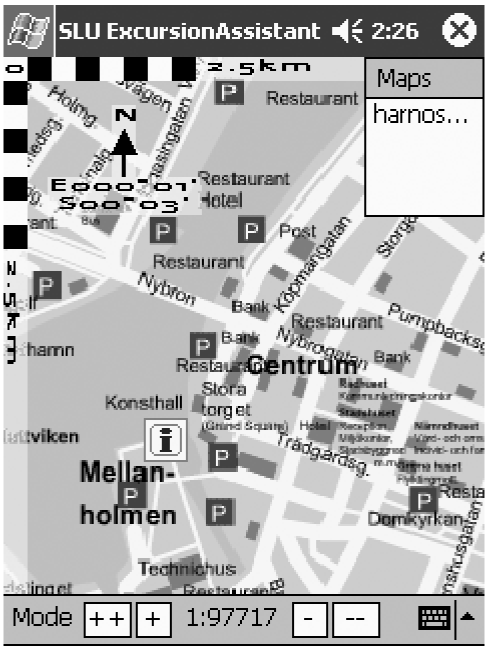


Fig 12: *SLU Excursion Assistant*

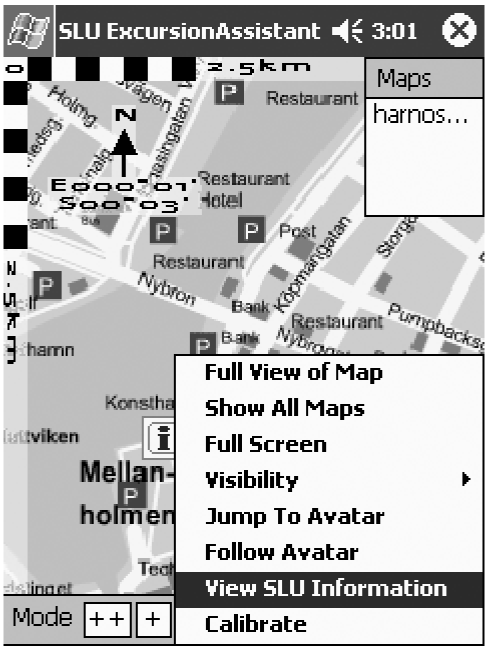


Fig 13: *SLUEA Menu selection*

GPS

The Global Positioning System satellites continuously send positioning signals that GPS receivers on earth can receive. SLUEA requires the GPS output signal to be encoded using the NMEA 0183 format. The GPS-Receiver uses a virtual serialport to retransfer the signals to the SLUEA. The serial communication data traffic is transferred wireless by an embedded BlueTooth module. On the PPC you must do the following procedure to establish a connection between the PPC and GPS-Receiver:

A short description of the order to make GPS-PPC communicate:

```
==========================================
Switching GPS on (BT inquiry runs)
Activating BT unit on PPC (Switch)
Starting Pocket PlugFree (Wireless BT Transmissions)
Configuring Pocket Plug Free (Selecting services)
Starting Pocket PlugFree
Pocket PlugFree detects other BT-Devices (GPS)
Connection GPS - PPC established
==========================================
```

The configuration file LocBrowser.txt must also be modified before doing the procedure above. This procedure is only required one time.
A successful configuration is achieved with these settings:

```
==========================================
Baudrate: 38.400
ComPort: 5
Databits: 8
Parity: 0
Startbits: 0
```

```
Stopbits: 0
FlowControl: 0
========================================
```

Before the maps are correctly viewed the user must calibrate the maps. The maps must be calibrated every time they are imported to the SLUEA, but as long as they are not unloaded, SLUEA remembers the calibration. The maps are calibrated this way:

```
Calibrating MAP by GPS Input
========================================
1 Select sample 1 by marking the map where you are positioned.
2 Get the corresponding GPS coordinates by selecting "Get by GPS".
3 Move to a position as far away from the first sample point as you can.
4 Select sample 2 by marking the map where you are positioned.
5 Get the corresponding GPS coordinates by selecting "Get by GPS".
6 Press ok -> Completed
```

9.6 Database storage solutions

9.6.1 Selecting Database Engine

Since the LocBrowsers only offer rudimentary database support and not in any way was compatible with MS Access databases, other databases must be considered. MS have stopped developing Pocket access. As a result the pocket access was no longer supported. Instead there existed some third party software. The goal was to find a free of charge database with MS Access Database compatibility, a database with a well defined and complete API compatible with the rest of the system. The reason to find an Access compatible database was that SLU in the Initial phase stated that they have the Office Suite on most of their machines. An alternative to find an Access compatible database was to find a vendor supplying a system where the database engine was free of charge to both the PC and PPC platform. For an example, SQL server 2000 for WinCE were great on almost any evaluation point, except for that it required the user to buy the PC side version of SQL server for about 20.000 SEK, which could not be accepted. Primary 9 different databases were evaluated:

1. DDH Software HanDBase
2. BioHazard Software Data-on-the-Run
3. KaioneSoft SprintDB
4. Pocket Innovations Pocket Database
5. PocketSoft abcDB
6. Software 4 CE HandyDB
7. Syware Visual CE
8. Pocket CDB Builder InetInfoSoftSys
9. SQL Server 2000 win CE

To make a long story shorter, the major drawbacks of the examined databases was one or several of the following:

- Lack of support, if problem arised no support was available.
- Not open or available API, creating unnecessary dependencies.
- Not a free of charge technology, should be avoided if possible.
- Not all required functionality, missing important functionality.
- Badly documented, time consuming to learn managing the database.
- Used technology not compatible with the new system.

The only one that fulfilled the requirements stated at the beginning was InetInfoSoftSys CDB Builder. It was free of charge, had a well defined API, it supported ADOCE and was compatible with MS Access databases.

9.6.2 Selecting data access architecture

The picture below illustrates the data access architecture used in SLUEIS.

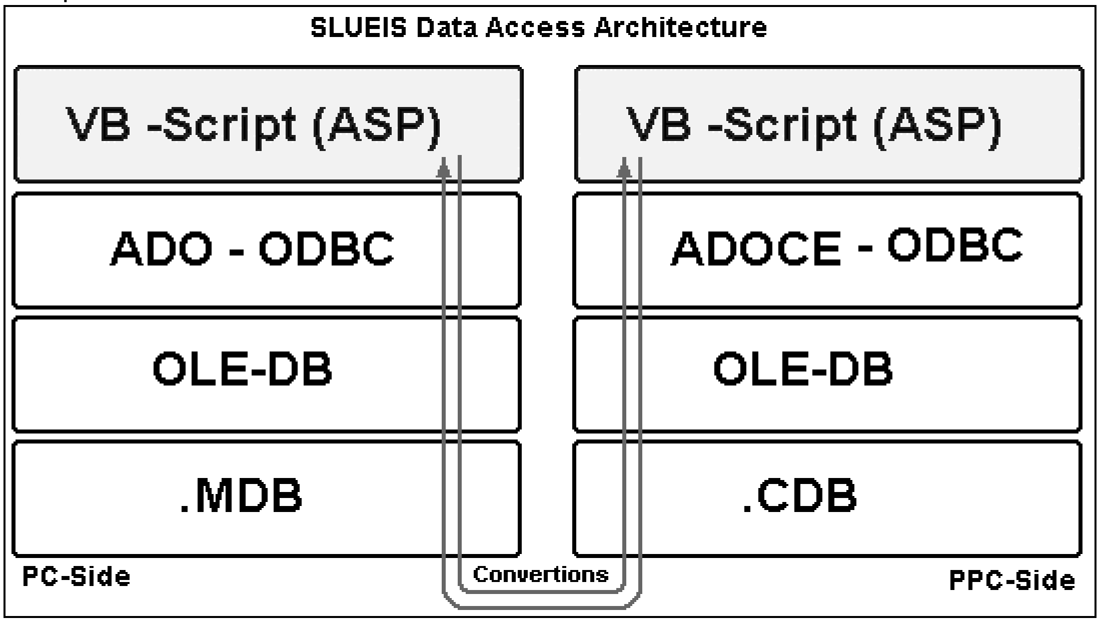


Fig 14: *SLUEIS data access architecture*

ADO can be viewed as a set of wrapper classes that are built on top of OLE DB interfaces and were designed to be the object interface to OLE DB. It is not possible to program straight to the OLE DB interface from VB, as it is when using the ODBC API. The OLE DB interface is much more sophisticated and complex and VB can't really deal with pointer arithmetic and some of OLE DB's structures. ADO and the corresponding ADOCE are designed specifically for VBScript. ADO is an application programmer's interface that provides developers with an easy way to access the underlying OLE DB data access interface and is especially suited to use along with Active Server Pages. This is also the reason why the SLUEIS data access architecture is designed the way it is. At the application layer ASP-applications are running and uses VB-script. VB-scripts that by ODBC and ADO make database requests through the low level interface OLE DB as finally make the read and write operations on the .CDB and .MDB Databases.

9.6.3 Implementing databases

Below, all of the databases implemented in the system can be viewed. The SLUEDB, seen in figure 15, is a database consisting of one single table, and the only functionality of simply connect an URL to the corresponding GPS coordinates. The entries consist of GPS data using the NMEA 0183 format. The data is divided into smallest pieces possible, to facilitate SQL queries. The SLUUDB, seen in figure 16, is a database consisting of one single table and with the functionality of storing user data during an excursion. It stores entries like user name, course, teacher, date of excursion as well as user answers associated to all excursion objects. The SLUEDB, seen in figure 17, consists of six tables, of which three are help tables enabling many-many relations. When implementing the SLUEDB, the concretization of the excursion route in figure 6 was used. The database most important implementation was the implementation of routes, TargetPoints and InfoItems, which is the most central elements in the whole system.

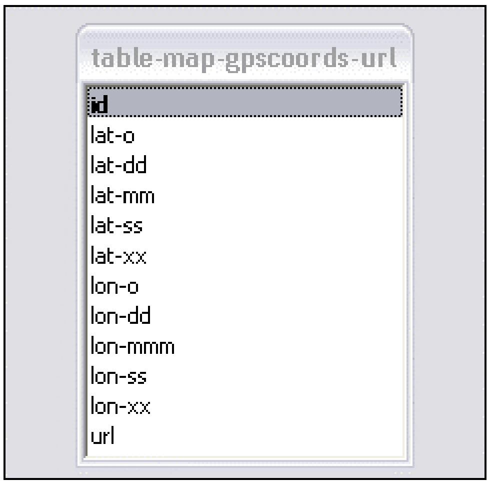


Fig 15: *SLUEDB-Map Url-GPSCoords*

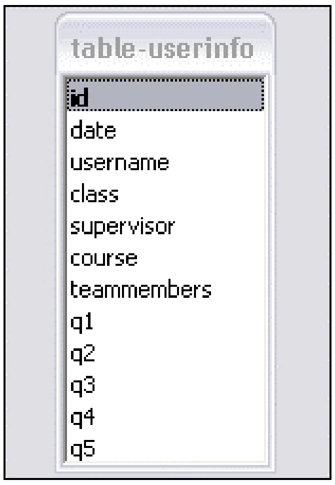


Fig 16: *SLUUDB - storing user data.*

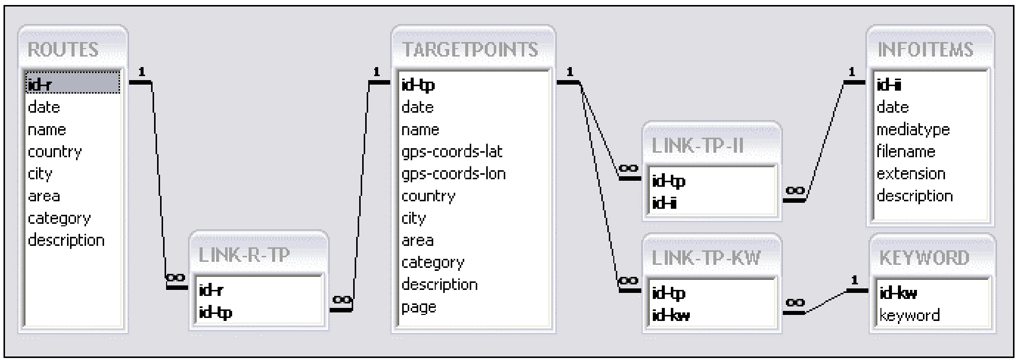
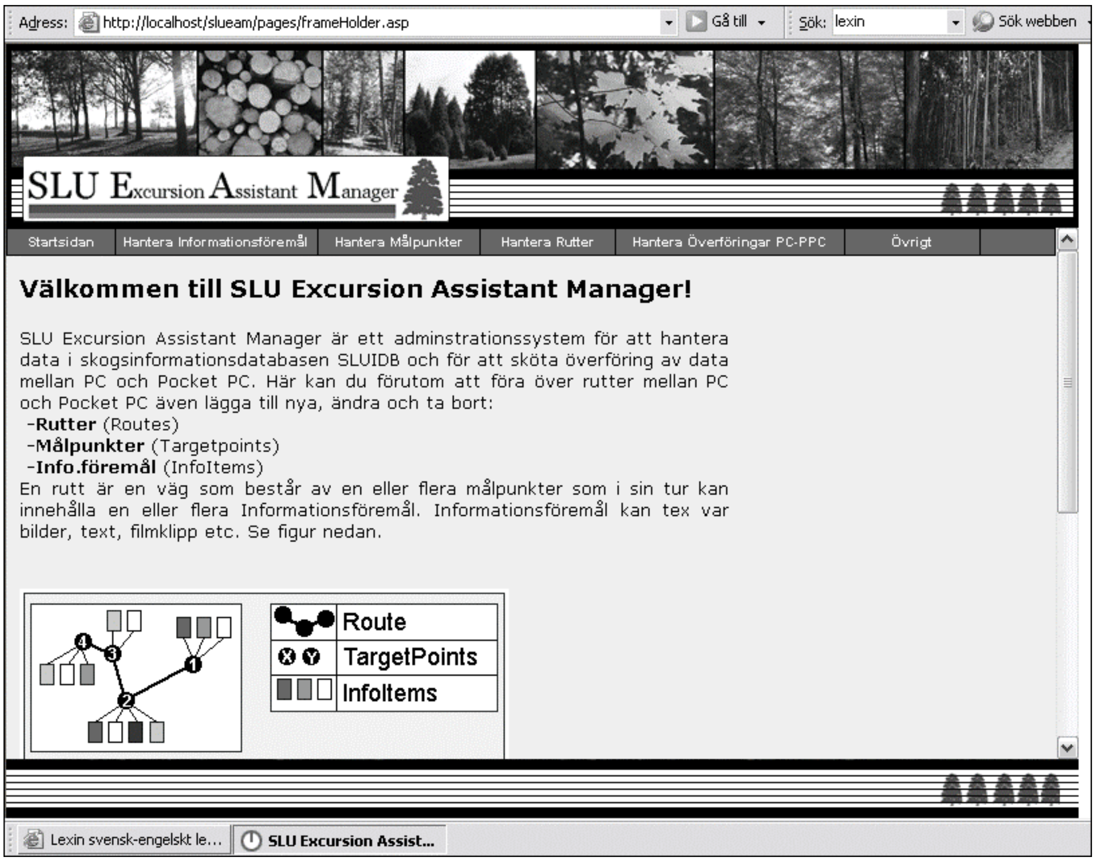


Fig 17: *SLUIDB including the tables: Routes, TargetPoints and InfoItems. Every object were implemented as own tables with appropriate attributes.*

9.7 User Interface

When developing the graphical user interface the following guidelines were used:

- **Using WebInterface** – using a web based user interface for managing the SLUEAM gives a lot of benefits. The web interface is an interface many people is familiar with and consequently not afraid of using.
- **Top & Bottom Banner** – A Uniform top banner is used on all pages to help the user distinguish when navigating within the application and when not.
- **Menu Bar** - A menu bar with drop down menus is used. It is easy and intuitive to use and it follows the current web standard.
- **Contrast ratio** – The text to background contrast ratio used within the application is correct (greater than 1:3).
- **Task analyzed interface** - The user interface is designed after a vast analysis of the underlying functions. Frequently used functions are made most available and placed in the order they are normally used. Left->Right.
- **Language** – Difficult or technical terminology was avoided.


Adress: http://localhost/slueam/pages/frameHolder.asp
Gå till
Sök: lexin
Sök webben
SLU Excursion Assistant Manager
Startsidan
Hantera Informationsföremål
Hantera Målpunkter
Hantera Rutter
Hantera Överföringar PC-PPC
Övrigt
Route
TargetPoints
InfoItems
Lexin svensk-engelskt le...
SLU Excursion Assist...

Välkommen till SLU Excursion Assistant Manager!

SLU Excursion Assistant Manager är ett adminstrationssystem för att hantera data i skogsinformationsdatabasen SLUIDB och för att sköta överföring av data mellan PC och Pocket PC. Här kan du förutom att föra över rutter mellan PC och Pocket PC även lägga till nya, ändra och ta bort:
-**Rutter** (Routes)
-**Målpunkter** (Targetpoints)
-**Info.föremål** (InfoItems)
En rutt är en väg som består av en eller flera målpunkter som i sin tur kan innehålla en eller flera Informationsföremål. Informationsföremål kan tex var bilder, text, filmklipp etc. Se figur nedan.

Fig 18: *Graphical User Interface SLUEAM*

9.8 Transfer methods

When the teacher is done creating a route for next upcoming excursion, the next step is transferring the route from the PC to the PPC. The requirement stated that it should be as easy as possible to do it and technical intervention should be avoided. A lot of different methods of transferring data between the PC and PPC existed and the following techniques were considered:

- ActiveSync via: IR, USB or TP
- BT Pocket Plugfree via: Blue Tooth
- Own transfer protocol via: IR, USB, BT, TP

Combining them gives at least 8 transfer methods.

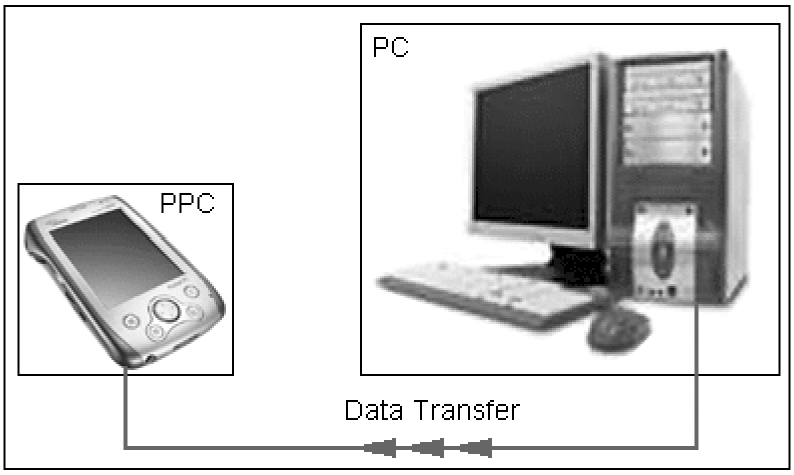


Fig 19: *Transferring data from PC to handheld PPC.*

The following aspects were taken into account when selecting transfer methods: User friendliness, reliability, teacher's earlier knowledge, time to implement.

To implement a new perfectly tailored transfer method was possible, it was the first thought, because the literature had lots of great examples. Sadly the project time did not allow this solution. All transfer methods based on BT were also excluded, because it forces the PC side to be equipped with BT, which creates an unnecessary requirement.
The Transfer method selected was ActiveSync because of the following advantages: Easy user interface, supports both IR, USB and TP. People who have used a handheld have most likely also used ActiveSync. Reliable transfers and already implemented converting routines existed. No new transfer routines are needed to learn. The user just synchronizes his PC and PPC and the needed files is converted and transferred.

9.9 Programming Language and used technologies

At the PC side ASP and VB-script was selected as implementation language and SQL as database communication language.
At the PPC side ANSI C was selected as implementation language. As the LocBrowser were already implemented in C, it was natural to continue using C as implementation language, but VB-script and SQL were also used at the PPC side.

10 Problems encountered

10.1 Initial problems

Several of the problems arised in the initial phase of the project. The problems concerned hardware incompability in conjunction with the development environment. Other related problems were; missing dll:s and configuration files, firewall problems etc.

10.2 Order of building project

When trying to build the LocBrowser for the first times it was impossible. Different error messages were displayed. The founder of LocBrowser were contacted. He told that the existing projects MAKE-routine only was semi automatic, and that each subproject was forced to be built manually before the executable could be built. This issue maybe never had been figured out, if the founder refused to inform. This kind of dependency on external people is not healthy for a project.

10.3 Estimating time for each task

One of the hard things was to estimate how much time each task would take so a correct time plan could be established. One task that was totally misjudged was the time it would take to find a functioning alternative database. It almost took a week to examine all 9 alternatives.

10.4 Using BLOB or not

When handling large files in databases the question was: Is it better to store the whole binary large object (BLOB) in the database or is it better only storing the reference to the object and leave the BLOB beside? A systematic research was done and it showed that the whole world is split in two parts regarding that question. But in my case it was obvious: If storing the whole BLOB in the database the memory required when handling it is twice as much as the BLOB, which is unacceptable when dealing with embedded systems. The solution is therefore to store the BLOB:s separately in a file system and only store references to those objects in the actual DB.

10.5 The flow criteria

One of the criteria's was to get the data flow through the whole system without getting trapped in a dead end. The meaning of that was that it was important to find a red line of compatible techniques. It just could not be decided to use "this" database and "that" scripting language before that the rest of the system was guaranteed to support it. All techniques had to be compatible with each other to achieve that criterion.

11 Results

The results of the MT project have been presented below:
On the PPC side a system called SLU Excursion Assistant was developed. The system is able to view maps, and offers real time navigation. The current implementation supports viewing of the following media format:

- Video format; mpg (mpeg)
- Image format: gif, tif, jpg, bmp
- Sound format; mp3, wav
- Text format: txt, htm
- Flash format: swf

The system support interactivity. The user is now able to answer questions by filling in html forms that is created by the teacher before the excursion. The user data is stored in a database that easily can be transferred back to a PC for further analysis. On the PC side an information management system is developed, a system where the teacher can add, edit and delete information, as well as transfer the information to the handheld.

- The project was not fully implementable according to the given prerequisites. Lack of access to the complete source code and debug version of code were the major reason of why the project was not fully implementable. The code access restraints referred to intellectual property rights in connection to Gil Müllers (founder of LocBrowser) future market- and business plans.
- Thorough research of different technologies ensured that appropriate technologies were selected and integrated into the system.
- An appropriate Project model was found and led to a successfully managed project. The model was the Spiral Model with 2 AoD iterations.

12 Conclusions and future work

12.1 General

It has been an exciting period working with this Master Thesis Project, especially the work with PDA and GPS which is a great combination with lots of potential in a number of areas. It has also been a pleasure cooperating with SLU, CUT and the supervisors Mikael Sjöberg and Thomas Johansson and I hope SLU continues to strive for better aided forest excursions using PDA:s and GPS. The MTP have been a mix of problems and pleasure. I remember how lucky I was when I succeeded compiling the source code for the first time, and I also remember how sad I was when I got aware of that not all of the source code was accessible.

12.2 Acquired knowledge and experience

During the development phase of SLUEIS I have learnt a lot of things. I have learnt how important it is to perform the development within a framework. As long as the framework is followed, the progress of the project is relatively safe.
I have also acquired knowledge of working in an object oriented process model and how to apply the theoretical knowledge in real life.
I have learnt how important it is to perform a thorough prestudy and the importance of handling risks.
I have acquired experience in how to work with integration projects, which is a common working task today. I have seen how dependencies on other people can affect a project and I have acquired experience in analysis, design and implementation using C, ASP, SQL, Java script, VB script, as well as how to constructively solve problems.

12.3 Future work

Through the development process of SLUEIS a lot of great ideas of how to improve and further develop the system have come on my mind. The major drawbacks with the current solution are that the SLUEA does not have a layer that can translate the screen taps to GPS-coordinates on the map, and thereby the application is missing an important function.
As the system works now it is not possible to have more than one TargetPoint in each route. The constraint depends exclusively on the inaccessible sourcecode. To fix this should be the main objective, but as long as the founder of LocBrowser does not release the sourcecode it can be hard to solve.
Gil Müller is right now releasing LocBrowser Basic Edition version 1.20. It would be wise to take closer look on that release. Maybe the AIS library is free in that release?
Another important thing is to improve the synchronization and transfer mechanism. As the system works now, the teacher has to make transfers to every PPC one at the time, which is not effective. By implementing a transfer mechanism that enable transfers from one PC to many PPC:s, the effectiveness and usability of the system should increase a lot. There are lots of examples available how one can implement such systems. The GUI of the PC side SLUEAM is also one of the things that can be improved, it could be revised to better and

more intuitively adapt to the functionality. Currently three steps are needed to create a route: Add files to an InfoItem, add InfoItems to a TargetPoint and add TargetPoints to a route. The adequate number of steps is one. The rest of the improvements are left to the reader. Thereby I want to thank the reader for taking the time reading this report. Now follows a theoretical part focusing on how to optimize code for embedded systems.

13 Optimizing code for embedded systems

13.1 Introduction

The emergence of integrated circuits in which both program ROM and the processor are integrated on single die initiates a new epoch of problems for programming language compilers and for programmers working with optimization.

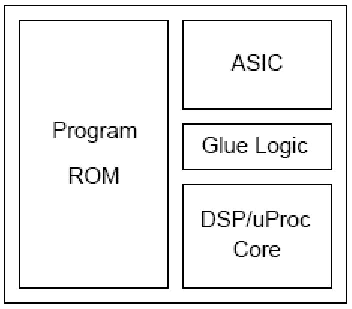


Fig 20: *Overview - system on a chip with both ROM and CPU*

In such a micro architecture, code performance and particularly code density, gain an unprecedented level of importance and new code optimization algorithms will be required to supply the required code quality. The need for low-cost versions of products drives hardware designers to provide just barely enough processing power and memory to get the job done. The embedded systems wide area increases every day and ranges from toasters, printers and cellular phones to advanced control systems, as well as the hardware used in this MTP, the handheld Fujitsu-Siemens Pocket Loox 600 and the Emtac GPS Receiver.[4,13,15]

All modern compilers provide some degree of code optimization, but, most of the optimization techniques that are performed by a compiler involve a tradeoff between execution speed and code size. Programs can be made either smaller or faster, but not both. For example, a subroutine call takes less space than inline code, but is generally slower. Obviously an improvement in one of these areas may have a negative impact on the other and it is up to the programmer to decide which of these improvements is most important. Given that single piece of information, the compilers optimization phase can be appropriately chosen whenever speed versus size tradeoff is encountered. [4, 13]

In this text I have tried my best to compile and summarize the most important aspects and techniques regarding code optimization. I start presenting the most common and classical techniques of manual code optimization, optimization not performed by the compiler. Later on focus are moved to compiler techniques and how compilers must be constructed to generate the densest high quality code possible, if targeted for embedded systems.

13.2 General optimization techniques

13.2.1 Increasing speed

FIRST OF ALL - KNOW THE CODE
Before starting optimizing code it is crucial to "know the code". It is important to identify subroutines and modules that are the most time critical for overall code efficiency as well as the interrupt service routines, high priority tasks and functions that are either compute intensive or frequently called. These are typically candidates for optimization. According to Agner F. 99% of the CPU time is spent in the innermost loop, when dealing with compute intensive applications. Identifying the most critical part of your software is therefore necessary if you want to improve the speed of computation. There is a tool called "profiler", often included in some software development suites that can be used to inspect those routines in which the program spends most of its time. When these routines are identified, one or more of the following techniques can be applied to optimize the code. [4, 28]

INLINE FUNCTIONS AND TABLE LOOKUPS
By adding the keyword inline to any function declaration the compiler replaces all the calls with copies of code that is inside. This eliminates runtime overhead associated with the call. This type of optimization is most effective when the inline function is called frequently and contains a few lines of code.
A switch statement consists of a set of "test and jump". To maximize speed it is important to put the case with highest probability to occur at the top and the last likely to occur at the bottom. This will reduce the average execution time, but will not improve the worst case scenario. If there are lots of instructions to be executed within each case it might be more efficient to replace the entire switch statement with a table of pointers to functions. [4]

BREAKING DEPENDENCE CHAINS
Modern microprocessors can do out-of-order execution. If a piece of software specifies the calculation of A and then B, and the calculation of A is slow, then the microprocessor can begin the calculation of B before the calculation of A is finished, if the value of A is not needed for the calculation of B. To take advantage of that, long dependence chains have to be avoided.
An example calculating sum of 50 numbers (long dependence chain):

```
double list[50], sum = 0.;
for (int i = 0; i < 50; i++) sum += list[i];
```

If an addition takes 5 clock cycles, then this loop will take approximately 250 clock cycles. The performance dramatically can be improved by splitting the dependence chain in two:

```
double list[100], sum1 = 0., sum2 = 0.;
for (int i = 0; i < 100; i += 2) {
sum1 += list[i];
sum2 += list[i+1];}
sum1 += sum2;
```

If the microprocessor is doing an addition to sum1 from time T to T+5, then it can do another addition to sum2 from time T+1 to T+6, and the whole loop will take only 128 clock cycles, twice as fast as the first example.

GLOBAL AND REGISTER VARIABLES
It is more efficient using global variables than passing a parameter to a function. This eliminates the need of pushing and popping the parameter on the stack before the function call and completeness. This optimization method may be considered twice before use because of its negative effect regarding software

modularity and reentrancy, it should be used carefully and only if every speed increment is indispensable.
Analysis of code may also give the information that some local variables are used more than others. These variables preferably declare as register variables using the keyword "register". Declaring variables this way makes the compiler place the variables in a general purpose register rather than on the stack.

POLLING
Interrupt Service Routines ISR are often used to improve program efficiency, but there are some cases in which the overhead associated with the interrupts are causing inefficiency. This happens when the average time between interrupts is the same order of magnitude as the interrupt latency. In such cases it might be better to use polling. As the optimization technique "using global variables" this technique also has a negative effect on modularity and should be used carefully.

FIXED POINT ARITHMETIC
If the target platform does not include a floating point processor, a very large penalty for manipulating float data in your program must be paid. The compiler-supplied floating point directory contains a set of software subroutines that emulate the instruction set of a floating point co-processor and many of these functions take a long time to execute relative to their integer counterparts and also might not be reentrant. If floating point is to be used only for a few calculations, it is recommended to reimplement the calculations themselves using fixed point arithmetic only. It is theoretically possible to perform any floating point calculation with fixed point arithmetic. [4]

ARRAY OF STRUCTURES TO IMPROVE CACHING
Use array of structures rather than structure of arrays. Variables that are used together should preferably be stored near each other in order to improve caching. If having two arrays, a and b, and the elements are accessed in the order:

a[0], b[0], a[1], b[1], …

then you may improve the performance by making an array of a structure which contains one a and one b, so the elements are accessed this order:

array[a0,b0], array [a1,b1], array [a2,b2], array [a3,b3], …

STUDY THE ALGORITHMS
A fundamental thing that many programmers forget, relying on the underlying compiler technology, is to study the algorithm used in the critical part of the code, to check if the algorithm really is the most effective or if it can be improved. Often, optimizing the critical algorithm is the most effective way of gaining more speed, compared to any other optimization method. There are lots of examples in literature were applications running on slow hardware have prevailed over applications running on a much faster hardware just by having a more efficient algorithm. When dealing with time critical systems the only alternative to make code more efficient may be to use hand-coded assembly, but using hand-coded assembly must be considered from time to time, though an average programmer produce less effective machine code than the most C, C++ compilers.

13.2.2 Decreasing Code Size

In the text above the optimization techniques concerned execution speed, now focus is centered on how to decrease code size, how to produce code using smallest amount of memory possible. First of all: Try to trust your compiler. Only if the code still is too large to fit the embedded system memory, the following essential procedures is accessible for high density code producing.

AVOID STANDARD DIRECTORY ROUTINES
Avoid using large standard directory routines. Many of them are expensive only because they try to handle all possible cases. If your program only uses a subset of a routine functionality it might be better to only implement the functionality that is needed. A good example of a routine with a huge code size is the standard C directory's sprintf. Much of the overhead is connected to the routines floating point manipulation. But if you don not need that floating point manipulation, it is better write your own integer version. This way you save several kilobytes. [4]

NATIVE WORD SIZE
Every CPU is built in such way that it has a native word size. ANSi C and C++ standards state that data type int must map to that size. Manipulation of larger or smaller datatypes may require use of additional machine language instructions. By consistently using int whenever possible, several hundreds of bytes will be saved and code size will be kept by at minimum.

GOTO STATEMENTS
Good software engineering practice dictates against the use of GO TO statements just as well as use of global variables. But if really hunting for the last bytes to save it might be appropriate to use it, because of its ability to remove complicated control structures and sharing blocks of repeated code. Use with care.

OTHER CODE DECREASING OPTIMIZATIONS
Other useful techniques to decrease code size are the techniques used to increase speed, described earlier, especially hand-coded assembly, table look-ups register- and global variables.

Be aware of the effect of an optimization. A program that is optimized is not the same program as before optimization. That's completely naturally; code has actually been taken away. Tests must be carefully performed after an optimization to assure that the program behaves the same as before the optimization.
In some cases, it is RAM that is the limiting factor for your application rather than ROM. In these cases you want to reduce your dependence on global data, the stack and the heap. These are all optimizations better made by the programmer than the compiler.

13.2.3 Reducing memory usage

DECLARE CONSTANT DATA WITH KEYWORD CONST
Because Rom is usually cheaper than Ram, one strategy for reducing the amount of global data might be to move constant data into Rom. This can be done automatically by the compiler if you declare all of your constant data with the keyword const. There exists a special data segment that is recognizable to

the locator as the Rom-able where most C, C++ compilers place all of the constant global data they encounter into. This technique is valuable if there are lots of strings or table-oriented data that does not change at runtime.

REDUCTION OF STACK SIZE
Stack size reduction can also lower your programs RAM-requirement. One way to figure out exactly how much stack you need is to fill the entire memory area reserved for the stack with a special data pattern. Then, after the software has been running for a while, preferably under both normal and stressful conditions-use a debugger to examine the modified stack. The part of the stack memory area that still contains the pattern has never been overwritten, so it is safe to reduce the stack area by that amount.

STACK SPACE IN RT-OS
Be especially aware of stack space if using Real-Time operating systems. Most of them create a separate stack for each task and these stacks are used for function calls and interrupts service routines that take place within the context of a task. Determine the amount of stack required in the manner described earlier.

SPECIAL INTERRUPT STACK
Try to reduce the number of tasks or switch to an OS that has a special interrupt stack for execution of all interrupt service routines. This method can significantly reduce the stack size requirement of each task. The size of the heap is limited to the amount of RAM leftover after. [4]

13.3 Adapting code compression to ES

13.3.1 General

The traditional approach to the optimizing problems in ES (Embedded Systems) has been to write the embedded code in assembly language. As the complexity of embedded systems grows, programming in assembly language and optimization by hand are no longer practical or economical.
Programming in a high level language can acquire a code size penalty and one reason to that is that compiler optimization techniques classically have focused on code speed and not code density.

The information in this chapter is originated mainly from two of the most thoroughly works published in the area of code optimization for embedded systems and retargetable code generation:

- "Code Density Optimization for Embedded DSP Processors Using Data Compression Techniques" by Stan Y. Liao, Srinivas Devadas & others - MIT Dept. of EECS
- "Challenges in code generation for embedded processors" by Guido Araujo, Srinivas, Kurt Keutzer, Albert Wang & others - MIT, Cambridge, Synopsys Inc.

The focus is to look at the new objective for code optimization that has entered: "Generation of the densest code obtainable with the highest performance within any reasonable compilation time". The words from the authors are filled with self confidence and resolution. [13, 15]

13.3.2 Retargetable code generation

A retargetable compiler is a compiler not bound to a specific processor; it can easily be changed to generate code for many different processors. Preferably the best development environment (including compiler) should not limit the designer to any particular processor; it should allow the designer to choose the processor best suited to the application. The chosen processor may come from a predefined set or may be specially designed for a particular application. Such a processor is referred to as an Application Specific Instruction Processor (ASIP). As one can understand, it is impractical to develop a new compiler for every candidate architecture and thereby the code optimization and code-generation technique that develops must be retargetable.
Many modern code generation methods use tree pattern matching with dynamic programming, but in the case of irregular special purpose processor architecture their lack of transparency and stability may be problematic. That is because it is difficult to predict the exact code generation result in advance, and the effects of a modification in the code generation rules may be unpredictably wide. [17]

There are 3 various levels of retargetability.
1) Automatically Retargetable
In this level the compiler contains a set of switches that need to be set to specify the target architecture. Essentially, all possible target architectures that the compiler is intended to be used for are already built in. This is typically intended for use with parametric architectures.

2) User Retargetable
The target architecture is specified by the user to the compiler-compiler in some form. The compiler-compiler takes a representation of the instructions in terms

of some primitive operations as input and generates a compiler for the specified architecture.

3) Developer Retargetable
To handle machine specific optimizations that go further than instruction selection is to allow the developer to modify the compiler to target the given architecture. A compiler to be considered retargetable in this level, no new processor dependent optimization capabilities are added to the compiler during retargeting. The developer is using the processor dependent architectural information to tailor the built in optimization algorithms, and sequence them in the most effective order for that architecture. [13,15]

The focus is on Level 3, Developer Retargetable, this is by the authors the only level that can support the optimizations needed for producing high quality code.

To generate the densest and fastest code the compiler must offer a comprehensive suite of optimizations. A number of the common traditional compiler transformations and analyses that are already in use in existing compilers are listed below. No efforts are spent on explaining them. Please refer to: "*Aho. Sethi, Ullman - Compilers: Principles, Techniques, and Tool"s,* if interested.

1. Dead code elimination:
2. Global common sub expression elimination:
3. Recurrence rewrites:
4. Re-association:
5. Loop unrolling:
6. Loop invariant code motion:
7. Strength reduction:
8. Vectorization:
9. Instruction Scheduling:

There are 3 challenges that have to be addressed when applying these transformations in a retargetable manner.

1) Many of the above optimizations work best when customized for a specific processor. As an example, the common subexpression elimination, if too aggressive it can create too many candidates for the register allocator. When that occurs, the values of common expressions must be stored in memory. On many machines fetching a value from memory may take more time and space than recomputing simple common expressions.

2) Many of the transformations above must work in concert to improve code. Recurrence rewrite can reduce index register use, accordingly improving register allocation. The relationships among the optimizations will in general depend on the processor.

3) Another important aspect of the transformation is the phase ordering problem. As an example register allocation and instruction scheduling may yield different code quality depending on which is done first and on the specific program. So no fixed ordering of allocation and scheduling is guaranteed to generate the best code for all programs on one processor.

The above, well known optimizations have been developed for traditional general-purpose architectures. However, processors used in embedded systems presents new challenges to optimizing compilers. As an example; offset assignment problems, handling non homogeneous register sets, mode optimization and parallelism issues. In order to generate dense high quality code, new techniques are needed. [15]

13.3.3 Approaches to Code Compression

To reduce the size of executable programs, high performance data compression techniques have been applied. The problem is that the programs to be compressed are treated as sequential data, and before compressed programs can be executed, they are first decompressed and loaded into RAM. It means that savings only are attained on secondary storage. This kind of techniques is not suitable for code size minimization in the context of embedded systems that have size limitations. They are aimed at compression and transmission of sequential data. Random access inside the compressed code is not practical and sometimes totally impossible. Because of the programs non linear control structures (branches and loops), they require random access and it is thereby difficult to execute the program directly from the compressed code.

The authors have built their own compiler on a concept of data compression and they were motivated to use a design that could achieve better results than mere software optimization. The design they used is based on a compression method called an external pointer macro (EPM) model. The compressed data in this model consisted of two important parts:

- A dictionary
- A skeleton

The dictionary contained substrings that occurred frequently in the original data and the skeleton contained symbols from the alphabet of the original data, scattered with pointers to the dictionary. The model was particularly suited for the approach to code minimization because of the simple decoding process and that it could be done in real time.
As a result ordinary sequences of instructions, not only common subexpressions as could be eliminated by an optimizing compiler, are extracted and stored in a dictionary, and occurrences of these instructions are replaced by pointers to the appropriate location in the dictionary. An important characteristic of the approach is that the instruction set of the improved machine is a superset of the original machine; consequently all the programs that could run on the original machine could also run on the enhanced one.

When minimizing the code, two different code size minimization methods were developed.

1. Minimization on parts of the software (a collection of entries)
2. Flexible in its use of the dictionary (large entry in itself)

The first method was a minimization on parts of the software; no hardware modification was necessary. In this method common sequences were extracted to form dictionary entries and were replaced by calls to the dictionary. The suffix relation is generalized for a particular class of blocks called extended blocks and the method applies under this generalization as well.

The second method was more flexible in its use of the dictionary. Unlike the first method, here the dictionary could be seen as a large entry in itself, not a collection of entries. Any substring of the dictionary could be replaced by an appropriate pointer and though some extra hardware was necessary to support this compression model, the total savings in code size were anticipated to overshadow the cost of the extra hardware. This technique could be combined with the first to achieve even superior reduction on code size. [13,15]

13.3.4 Compiler organization - experimental infrastructure

A compiler infrastructure was built were experiments testing various algorithms could be performed (The overall organization of the compiler system can be viewed in figure 21). The experimental environment was interactive which was helpful when writing new algorithms and integrating them into existing systems. It also enhanced the experimenting with algorithms as scripts could be written quickly, experiments that explored the different options and measured the efficacy of different combinations of options.

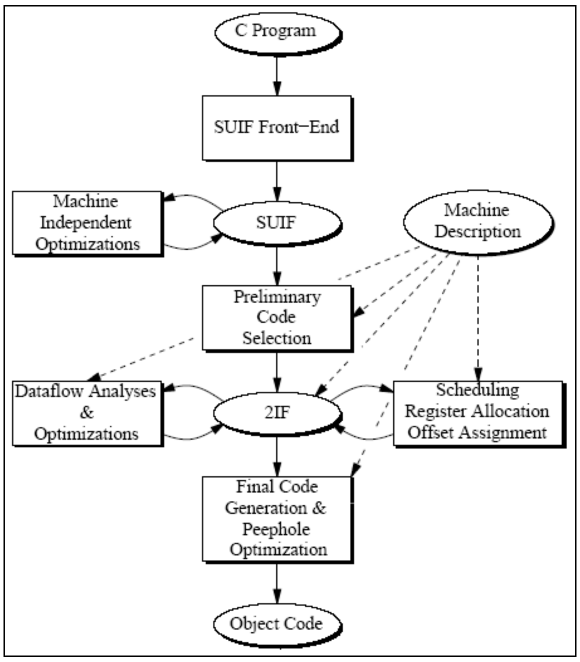

C Program
SUIF Front–End
Machine Independent Optimizations
SUIF
Machine Description
Preliminary Code Selection
Dataflow Analyses & Optimizations
2IF
Scheduling Register Allocation Offset Assignment
Final Code Generation & Peephole Optimization
Object Code

Fig 21: *Compiler organization prototype*

The SUIF Compiler (Stanford University Intermediate Form) was used as the front-end and as the first intermediate form, but SUIF alone was not sufficient for the purpose. The intermediate form (IF) contained too little machine dependent information. Therefore, an intermediate form schema was deigned. The schema itself was machine independent; the contents of the IF, which was an instance of the schema, varied from different machines and could be derived by an appropriate machine description. As for an instance the operators in the second IF (2IF) reflect the actual machine instructions.

A preliminary code selection (PCS) is performed on a program in SUIF to arrive at this second IF. Please refer to figure 21. The PCS engage covering SUIF expression trees with "tree patterns" that correspond to actual machine instructions. As an example some of these instructions may be complex and large instructions. Those high-SUIF constructs that potentially may be implemented using hardware features are retained; others are broken down to "software" implementations. At this level variables and such names are not resolved into actual addresses.

If the second 2IF is to be useful more than just the product of the PCS must exist: The 2IF supports Common sub expression elimination, scheduling under a machine-dependent cost function and various dataflow analyses and their related optimizations. Dataflow analyses usually require a flow graph representation of the program, consisting of basic blocks and flow edges, consequently, there are multiple representations in 2IF and a program can be easily converted from one representation to another. Procedures are represented by a flow graph, from which traces or basic blocks can be selected.

The final phase of the code generation process works like this:

1. Optimized program is retrieved from 2IF
2. Object code is generated.
3. Non computation operators are generated,
4. Operands and results are moved to the appropriate locations
5. Machine modes for the computation operators are set.
6. Finally a peephole optimizer is used to clean the code generated. [15]

13.4 CONCLUSIONS

This study focused on the area of optimization techniques for embedded systems and started with some general optimization techniques: How to increase speed, how to decreasing code size and finally how to reduce memory usage. It showed how important it is to know the code you are about to optimize and the importance of analyzing the incorporated algorithms. Then we looked at compiler requirements for embedded processors, specifically retargetability. Because the typical aim in such processors is to embed the entire program memory on the same die as the processor, the highest possible final code quality were required. High density code required a series of classical compiler optimization techniques, but there were also some new problems introduced in the context of compiling for embedded processors. Code density had never been a prime target of compiler optimization, but it was of critical importance in embedded applications. Finally some techniques for code compression were investigated.
This is the end of the report and I want to thank you, the reader, for taking the time to review the report and for your considerations.

14 References

14.1 Books

1. Boling Douglas, Programming MS Windows CE, 2 ed, 2001, Microsoft Press, ISBN 0-7356-1443-1

2. Burdick Robert, Essential Windows CE Application Programming, 1999 (2002), John Wiley & Sons Inc, ISBN 0-471-32747-6

3. Murray John, Inside Microsoft Windows CE, 1999, Microsoft Press, ISBN 1-57231-854-6

4. Barr Michael, Programming Embedded Systems in C and C++, 1999 O' Reilly & Associates Inc, ISBN 1-5692-354-5

5. Muench Chris, The Windows CE Technology Tutorial, Second Edition, 2000 Addison Wesley, ISBN 0-201-61642-4

6. Grattan Nick, Windows CE 3.0 Application Programming, Prentice Hall PTR, 2001, ISBN 0-13-025592-0

7. Faulkner Xristine, Usability Engineering, Palgrave - Grassroot Series, 2000, ISBN 0-333-77321-7

8. Stephens K; Plew Ronald, Lär dig SQL på 3 veckor, Pagina Förlags AB, 1999, ISBN 91-636-0497-3

9. Pfleeger Lawrence Shari, Software Engineering – Theory and Practice, Prentice Hall, 1999, ISBN 0-13-624842-X

10. Hettihewa Sanjaya, Active Server Pages 2.0, SAMS Publishing Pagina Förlag, 1999, ISBN 91-636-0522-8

14.2 Articles

12. Marwedel Peter; Goossens Gert, Code Generation for Embedded Processors, Kluwer Academic Publishers, Internet, 2004-01-06

13. Liao Y. Stan; Devadas Srinivas; Keutzer Kurt, Code density optimization for embedded DSP processors using data compression techniques, 2003

14. Agesen Ole; Hölzle Urs, Type Feedback vs. Concrete Type Inference: A Comparison of Optimization Techniques for Object-Oriented Languages, 1995

15. Liao Stan; Devadas Srinivas, Code Optimization Techniques for Embedded DSP Microprocessors, MIT Department of EECS, 1998

16. Masselos K., Catthoor F., Goutis C.E., Code size effects of power optimizing code transformations for embedded multimedia applications, VLSI Design Laboratory, Department of Electrical and Computer Engineering, University of Patras, 1996

17. Lassila Eero, Towards optimization code generation by domain sensitive macro expansion, Department of Computing Science and engineering, Digital Systems Laboratory, Finland, Series A, Research Report No: 42, Jan 1997, ISBN 951-22-3439-4

18. Liao Stan; Srinivas Devadas; Steve Tjiang; Albert Wang, Storage Assign-ment to Decrease Code Size, MIT Department of EECS, Cambridge, MA 02139-4307, 1996

14.3 Webpages

19. http://www.microsoft.com/windowsmobile, 2004-02-23, Microsoft Corp.

20. http://www.fujitsu-siemens.com, 2004-02-23, Fujitsu Siemens

21. http://www.locbrowser.com, 2004-02-24, Gil Müller

22. http://hyperphysics.phy-astr.gsu.edu/hbase/gps.html, 2004-02-24, R Nave

23. http://www.webopedia.com, 2004-02-24, Jupitermedia Corp

24. http://www.se.rit.edu/~jrv/research/ar/, 2004-02-24, Jim Vallino

25. http://tycho.usno.navy.mil/gpsinfo.html, 2004-02-26, US Naval Observatory

26. http://www.colorado.edu/geography, 2004-04-12, Peter H. Dana

27. http://www.garmin.com/aboutGPS/, 2004-04-14, Garmin Ltd

28: http://www.agner.org/assem/pentopt.pdf , 2004-10-10, Agner Fog