Proceeding of The National Conference
On Undergraduate Research (NCUR) 2001
University of Kentucky, March 15-17, 2001
Lexington, Kentucky

Developing Platform Independent Tactical Data Entry Devices

Dennis Nishikawa, Jeremy Bukowczyk, and Christopher Rapp
Department of Electrical Engineering and Computer Science
The United States Military Academy
West Point, NY 10996

Faculty Advisor: MAJ John Surdu

Abstract

As the Army ventures into the future and becomes increasingly digitized, a need for more technically advanced equipment is imperative. Several Army systems use proprietary hardware and software solutions that are often heavy and costly, tying the Army to a specific vendor long after new technology has surpassed the capabilities of current systems. Rather than relying on a certain vendor or manufacturer, this project has chosen to pursue the implementation of platform-independent software to run on common handheld units. The research and development portion of this project includes searching for the technology that will best suit the needs of implementing the platform independent software. Candidate technologies include eXtensible Markup Language (XML), Java, and ECMA script, each with unique advantages and disadvantages. The software will run on personal digital assistants (PDAs) running either Windows CE™ or Palm OS™. The advantage of a "software first" approach is that it frees the Army from over-reliance on any particular vendors/integrators. This paper presents the research results of the various candidate technologies, their strengths and weaknesses, and the final design decisions on this software effort.

Keywords: Java. Waba. XML. Whiteboard SDK. PalmOS. WindowsCE. BlueTooth. Platform Independence.

1. Introduction

As the Army ventures into the future, in a world of rapid technological advancements, it must make increased efforts to search for technologies to support a digitized army. Many of the current digitized systems used by the Army rely on proprietary hardware and software, which in many cases have proven to be cumbersome and expensive. Even though this often results in better short-term performance, by relying on hardware dependent software, the Army binds itself to specific vendors even after certain technologies have become outdated. In order to alleviate this burden, the Army has chosen to pursue the research and development of platform-independent software, allowing the upgrade of either the software or hardware as needed.

This paper will present the results from research conducted on candidate technologies of both hardware and software to achieve the goal of implementing platform-independent software. These technologies include portable languages such as Java, Waba, and XML along with the hardware that supports the software such as Palm OS and Windows CE devices. Communication devices will also be included to illustrate advantages of wireless communication through such tools as BlueTooth Wireless Communication devices. Using the technologies mentioned above, proof of concept models were used to demonstrate the functionality and capability of each technology. Finally, this paper will describe the degree in which the intended objective was reached. The conclusion will present the final results of the research and implementation of the product.

2. Software

2.1 Java

When Sun Microsystems presented Java to the world in 1995, their focus of impact was the World Wide Web. Its principle attribute was platform independence [1]. This meant that developers could write code in Java and compile it in such a manner that it would run on any operating system. The user did not need to worry about what Operating System (OS) the user was using. Typically, for platform-specific programming languages, source code must be generated for each platform; with Java, this is not the case. When a user requests a Java program, from a Web server for example, the client receives what is called an applet, the resulting Java code, or executable application. The user needs an interpreter, what has now become more commonly known as a virtual machine or VM, on the machine to execute the applets.

Besides being platform independent, Java is object oriented. The concept behind Object Oriented Programming (OOP) is data encapsulation [2]. When the data and the methods in which to manipulate the data are coupled, an object, defined by classes, is created. Several advantages exist in OOP; for example, it models real world objects. OOP is also very modular in nature. Each object is its own entity and can be coupled and decoupled with other objects easily; furthermore, and perhaps most important, OOP is extensible, meaning when operating environments change, only a few alterations must be made. Extensibility will be revisited in the presentation of incorporating Java with XML.

Over the past five years, Sun has developed subclasses and different implementations of the Java virtual machines to coincide with the current technologies. Java has also taken into consideration the various echelons of programming requirements. For common programming use, Sun provides J2SE or Java 2 Standard Edition. For large corporate software development, Sun offers J2EE or Java 2 Enterprise Edition. And recently, the edition of interest in regards to this paper, Sun developed a more compact version of Java, called J2ME, or Java 2 Micro Edition, for software development on Personal Digital Assistant (PDA).

Sun provides various virtual machines for the different versions of Java. Java Virtual Machine (JVM) supports all the classes included in the J2SE version. JVM runs on common desktop operating systems such as Windows and Unix based systems. Kilobyte Virtual Machine (KVM) is designed to support the more limited version of Java, J2ME CLDC and the corresponding kAWT classes. Currently, the only PDA operating system on which KVM runs is PalmOS. As for the Window CE device, Sun offers Personal Java (PJava). PJava and KVM are both subsets of Java 2; however, they do not necessarily support all of the same classes. In order to achieve true platform independence, only classes supported by both PJava and J2ME VMs must be implemented. All Java's resources mentioned in this paper are free and can be downloaded from Sun's Java web site [3].

Java supports the intended goal of platform independence; however, there are several other concerns, including serial data communication, that must be considered and criteria that must be met. These issues will be addressed throughout the paper and the final results will be presented in the conclusion.

The use of Java includes several advantages, primarily platform-independence and serial capabilities. Another important factor for choosing this environment was the recent development of WHITEboard SDK for development with the Bluetooth devices. There are several classes, under CLDC and kAWT, written especially for development with handheld units.

On the other hand, using Java does have one drawback. The two virtual machines that essentially allow Java to be platform-independent, KVM and PJava, do not support all of the same classes. However, this issue can be resolved by relying on the intersection of classes supported by both environments. Clearly, the advantages of using Java outweigh the trivial disadvantages thereof.

2.2 Waba

Waba is a programming platform intended specifically for small devices. Waba defines a language, a virtual machine, a class file format and a set of foundation classes. The design of Waba allows developers to use Java development tools to develop Waba programs; however, Waba is not a derivative of Java and has no connection to Sun Microsystems, the owner of the Java brand and related trademarks [3]. In fact, the software to build and run Waba programs is available free on Wabasoft's web site [4]. The software is licensed under the GNU license, just like Linux.

The syntax of the Waba programming language is a strict subset of the syntax of the java language. This allows developers who are familiar with Java to quickly start programming in Waba. The Waba class file and bytecode

format are strict subsets of the class file and bytecode format supported by Java. This allows developers to use Java development tools to write programs for the Waba platform as long as they only use the subset of functionality supported by Waba, much like using J2ME.

The Waba language, virtual machine, and class file format were designed to be optimal for small devices. Features that would use substantial amounts of memory or that were deemed unnecessary for small devices were omitted from the design of the Waba language and platform. Waba has a set of foundation classes designed to be as small as possible while still containing the functionality needed to write fully featured programs for small devices.

Waba comes with a set of "bridge" classes that allow Waba programs to run anywhere Java is available. Waba programs can run as Java applets and applications. Using the bridge classes, a Waba program can run under Windows and UNIX and could appear on a web page as a Java applet. With a native Waba virtual machine, the same program could run on a small device, such as the PalmPilot.

Waba provides several advantages including mobility, functionality, reliability, and portability. Waba was designed for small, usually mobile, devices. Waba virtual machines are available that are under 64K in size (including foundation classes) and that run programs in less than 10K of memory. Waba allows developers to quickly develop versatile programs on almost any platform with development tools that are cheap (free in many cases), familiar, and easy to use. The Waba language is object-oriented and includes language features such as bounds checking and garbage collection that speed development time and make for more robust applications. The Waba foundation classes were designed specifically to encapsulate the functionality required to build applications for small devices. Small devices normally only contain memory and no outside storage device, so if a program corrupts memory, the whole machine may need to be reset. Waba safeguards memory access, like PJava, to prevent these types of failures. Since Waba uses garbage collection, memory leaks are extremely rare compared with programs written in other languages. With Waba, developers can write one program that will run on a PalmPilot device, Windows CE (version 2.2.1. and later) device, and any machine that supports Java (either the JDK 1.02, 1.1, 1.2 or 2.0) [4].

The advantages of Wabasoft's Waba programming language are similar to those of Java's. First, it is interoperable. Much like Java, the code can be ported from desktop's running NT to PDAs running PalmOS or WinCE. Wabasoft's intention for creating Waba was for development with handheld devices, which make it simple to program. Additionally, its syntactical similarity to Java and its serial capabilities make it very attractive.

However, the main concern with Waba is that it is proprietary. The intent of this project was to move away from proprietary vendors in order to produce expandable software. Because of Waba's only recent debut to the world, it is not established or as widely accepted as Java. Finally, its inability to communicate with the Bluetooth API stack disqualifies this programming language as a valid candidate.

2.3 Alternate Technologies

Other technologies considered in this project include Kada, Pocket Linux, IBM J9, and several versions of Java Virtual Machines. Kada was a short-lived solution in the project. Kada was announced to the world, but not delivered. Kada claimed to be completely platform-independent, in that, the same VM would run under PalmOS and WinCE. Pocket Linux made similar claims; however, Pocket Linux is too immature at this time. IBM J9 is an embedded solution to the Java issue. Its embedded attribute provides easier maintainability and use of the class structure. Another Java environment solution was the Tao VM. Tao is a multi-platform engine that runs Java bytecodes. Its emphasis is on networked connections and multimedia platforms. Although Tao provided functionality to several platforms, it did not for PalmOS.

2.4 Java & XML

XML, the eXtensible Markup Language, is a subset of SGML, Standard Generalized Markup Language, which was implemented and generally used by the Army. The most important aspect of XML is that it is platform independent, textual information [5]. Much as Java source code is portable, XML provides a portable manner in which to store and transmit data. This is accomplished by storing data in plain text format, which is the essence of its interoperability. For example, if an address book is stored in XML format on a local machine running Linux, this information can be viewed on a remote machine running Windows NT. Manipulating data across different platforms becomes very feasible with XML.

In addition to platform independence, XML has many other benefits and advantages. The syntax of XML is much like Hypertext Markup Language (HTML), in that tags surround plain text information. The advantage of

XML over HTML is the availability of user-defined tags. Although XML looks like HTML because they are both markup languages, XML uses tags to represent data, whereas HTML uses tags to format information [6]. World Wide Web Consortium (W3C) provided a limited set of pre-defined tags for XML. Additionally, they allowed users to define their own tags through Data Type Definitions (DTDs). Users can make SML code sharable by defining tags in a Data Type Definition (DTD) file.

Java source code is portable. XML is portable data. By incorporating portable data into portable code, completely portable applications can be developed. Sun Microsystems currently offers several Java APIs for XML parsing and messaging utilities at the Java home page [3].

2.5 Whiteboard SDK

Wireless communication is currently a widely discussed topic that is still in its developmental stages. The use of available wireless hardware is currently only implemented by large communications corporations, such as Ericsson, and select academic institutions for research purposes. Bluetooth Wireless Communication Devices, discussed later in this paper, are not commonly obtainable to the public. Thus, SDKs and APIs are not readily available for development of wireless applications. In recent months Zucotto has created Whiteboard SDK, which is a development kit that provides an Integrated Development Environment (IDE) and an API for Bluetooth development. Whiteboard SDK, Bluetooth edition, is designed to create wireless applications based on Java technology. This is essential to the project because of the platform independence requirement of the code. The ability to use Java to communicate with the Bluetooth devices will allow portability of the code to different hardware devices, including the PalmOS and the WinCE platforms. The current Zucotto product only provides an emulator in a specific environment to test the application. Whereas potentially all source code can only be tested on the emulator, Zucotto will provide an environment to port the code from the emulator to PDAs in the near future. More information on Whiteboard SDK can be found at Zucotto's web site [7].

2.6 PalmOS

When PCs first came on the market, a lot of mainframe and minicomputer companies tried to compete by adapting their existing designs to smaller boxes. They all failed, because the PC was a new form of computing with different requirements. Handhelds are also a new form of computing, and simply adapting old designs will not necessarily make a good handheld. A handheld is used in short bursts to enter or access information, rather than the long sessions a user spend in front of a PC. If a user is using a handheld to search for information in a corporate price book or look up a phone number to make an important call, the user does not want to fumble around for even a second. Palm powered handhelds are designed from the ground up to be as simple and intuitive as possible. Rather than "dumbing down" a PC, Palm started fresh with a new design that focused first on giving users the shortest and easiest path to information. For this reason Palm has succeeded in the handheld market when so many PC companies failed. Products based on the Palm OS are the world leaders in handheld computing, with more than 75% market share worldwide [8], and are the market leader in both the US and Europe. Sales of Palm Powered handhelds more than doubled in each of the last two years.

Palm powered handhelds are designed to be customized and easily expanded. Every Palm Powered handheld includes the basics that all users need, in a system designed to be wearable -- so small and lightweight that users can carry it comfortably in a pocket or purse all day long. It's easy to add extra features that are just right for users -- wireless modems, GPS systems, MP3 players, digital cameras, joysticks, voice recorders, data capture devices, and a lot more. This is a very different philosophy than Windows CE.

The Palm OS is an open standard, licensed to more manufacturers than any other handheld platform. Companies selling or developing Palm Powered handhelds include Handspring, Motorola, IBM, Kyocera, Nokia, Samsung, Sony, Symbol, Franklin Covey, and TRG [9].

2.7 Windows CE

Windows CE is the modular, real-time, embedded operating system for small footprint and mobile 32-bit intelligent and connected devices. Windows CE employs Windows' compatibility, advanced application services, support for multiple CPU architectures, built-in networking and communications options to deliver an open foundation for building a wide variety of products. Windows CE powers consumer electronic devices, Web terminals, Internet access appliances, specialized industrial controllers, mobile data acquisition handhelds, and

embedded communication devices. This highly modular platform allows developers to flexibly and reliably build the new generation of small footprint and mobile 32-bit devices that integrate seamlessly with Windows and the Internet [10].

Momentum is building for Windows CE. Windows CE was built from the ground up specifically for the embedded system and appliance market providing a rich, scalable open platform used in a wide variety of embedded systems and products. Today, Windows CE powers products from consumer electronic devices to specialized industrial controllers and embedded communications devices. Windows CE 3.0, the latest version of the operating system, provides a compelling platform to build embedded systems because of the highly modular features and services that developers can use to build 32-bit devices in order to integrate them with Windows and the Internet [11].

2.8 Decision

Platform-independence and serial capabilities were two essential criteria that limited the software choice. Java seems to be the appropriate choice for this project. Java supports both platform-independence and serial communication capabilities, which were tested in proof of concept steps of the design. Additionally, Whiteboard SDK allows the use of J2ME CLDC to control the Bluetooth hardware. Until recently, it seemed as though Waba would also be an alternate choice to provide both platform-independence and serial capabilities. The inability of Waba to communicate with Bluetooth devices made it undesirable. In addition, the development team was predisposed toward the "industry standard" nature of Java.

Palm OS and Windows CE were chosen because they are common PDA OSs. These two operating systems were also used to illustrate platform-independence by running applications written in Java on both devices. Eventually, other operating systems, such as Pocket Linux will be supported as well. Any PDA that can run a PJava/J2ME-compliant JVM should be able to run the code written for this project. Although more research is required, XML seems to be a better method in which to transfer data than the existing VMF Package 11 protocol that the Army currently uses. The VMF protocol is difficult to understand, from a developer's prospective, and XML may be the solution to overcome this issue. XML provides a more enhanced method to store sharable, transferable data. Finally, wireless communication between the PDA and the Army tactical communications devices was selected in order to eliminate the unnecessary hassle of wires and cables.

3. Wireless Technology

Several wireless communication devices were researched in order to determine which device met the criteria and was most appropriate for the solution. The systems that were researched include 916 MHz serial transceivers, IEEE 802.11, IEEE 802.11b, and Bluetooth.

Of the devices researched, Bluetooth seems to be the ideal solution. Bluetooth was designed to facilitate both voice and data communications anywhere in the world using small, low-power, low-cost transceivers while providing a high level of signal security. One drawback to this solution is the fact that Bluetooth technology is currently still in development and not scheduled to be on the open market until later this year. Bluetooth operates in the ISM band with ranges of 10, 25, and 100 meters depending on the class of the device used. The spread-spectrum protocol gives 1600 to 3200 frequency hops per second, depending on the mode of operation, improving signal security. Bluetooth can operate at data rates from 300 bps to 460.8 Kbps, allowing users to set the system to the data rate determine as necessary. Connection to Bluetooth for data transmission is accomplished through either a USB or UART (RS232) port, allowing easy connectivity to any Windows CE or Palm device. Bluetooth allows additional units to be added to the pico net, the Bluetooth equivalent of a Wireless Local Access Network (WLAN), by the master unit [12]. Due to these factors, Bluetooth was determined to be the most appropriate system to use in the development of our wireless communications link.

4. Proof of Concept

4.1 Breakdown

This project consists of three main areas in which to incorporate various software and hardware technologies. In each stage of development, a proof of concept demonstration illustrates the implementation and operability of the various technologies considered. The main emphasis of this product was focused on the serial data communication

portion to pass data from one device to another. The user requires a Graphical User Interface (GUI) in order to input data, which will then be passed serially to another device. Finally, wireless communication devices replaced the cable used to connect various systems together. By breaking the process into smaller parts, faulty software and hardware could be identified much more quickly. Following the completion of all proof of concept models, the three phases will be integrated into a single working prototype.

4.2 Graphical User Interface

The Graphical User Interface (GUI), the front end of the prototype, must be extremely simple to use. Complete functionality of the current call for fire operation must be implemented. There are seven essential fields in a call for fire request, which are the header, type of mission, position, size, and protection of the enemy, plus type of target in order to choose a round type. The interface below (Figure 1) was produced using the kAWT classes that Sun provides for J2ME CLDC. The classes available for the interface were limited, because PJava and KVM do not support the entire J2ME version of Java. The idea behind the GUI proof of concept was rather simple. The proof of concept GUI illustrates the possibility of creating the necessary interface using the classes provided by Sun for Java. These classes provide the use of buttons, pull down menus, and text fields to enter data. The user input via the GUI will be later tied into the Serial Data Communication aspect of the project.

Figure 1 Graphical user interface screen shot

4.3 Serial Data Communication

As mentioned above, this portion presented the most problems, and therefore, much of the emphasis was focused on the communication issues. In preparation for providing serial communication between the interface and the server, the goal of this section was to pass a string “Hello World” through the serial port in order to demonstrate proof of concept. Given both the PalmOS and the Windows CE devices, the serial port was the common I/O connection. J2ME CLDC provided serial capabilities, however, only to a limited extent. Waba also provided a simple serial class to access the communication port. The environment used to test the communication portion was a PalmOS device and lap top running the PalmOS Emulator (POSE) connected by the serial cradle provided by

Palm. Finally, in order to illustrate Java and Waba's platform independence, a Palm and a Windows CE device were connected running the same code to write the string into one and read from the other.

Once the concept was proven, by passing a string through the serial port, using both Java and Waba, several successive strings could be passed in order to create XML tags for data storage.

4.4 BlueTooth

Prior to obtaining Zucotto's Whiteboard SDK, a pico net was initially established between two PCs using a USB interface to connect the Bluetooth devices (Figure 2a). The devices were tested with the Bluetooth Application and the included Training Tool Kit in order to observe the mechanics of the device.

Upon receiving the Whiteboard SDK, testing of the new Bluetooth devices through serial connectivity could be tested (Figure 2b). Again, the devices were initially tested using the included emulator and MIDlets. After establishing a successful connection between two PCs, sample code was written in J2ME to run on the Whiteboard emulator. This code enabled the two emulators to send messages back and forth. The next step was to integrate the GUI and the serial connectivity from the previous two proof of concepts. As mentioned above, the code still could not be ported onto the PDAs, which was the overall goal of the project. Current technology has not yet offered a solution to this problem; however, Zucotto has claimed to provide such functionality in the very near future.

With the advent of the new environment, allowing code to be ported onto the PDA, a PDA will ideally be able to communicate to a PC via Bluetooth connected serially (Figure 2c). The code for the GUI and the software control mechanism for the Bluetooth device will all be written in J2ME CLDC, making it completely platform-independent.

Concurrently, after communication is established between two PCs via Bluetooth, a server side PDA must be established, which will receive data from a Bluetooth device and also communicate with a PC through a modem. This portion is reflected in Figure 2d. The modem is necessary for later use because digital signals must be converted into an analog signal in order to communicate with the Single Channel Ground and Airborne Radio System (SINCGARS).

Ultimately, the final prototype will consist of a PDA which communicates with an end system, a PC, through a slave Bluetooth device, which communicates with the master Bluetooth device, which then passes data to a SINCARS, via a modem, which communicates with another SINCGARS connected to the afore mentioned end system. This is illustrated in the Figure 2e below.

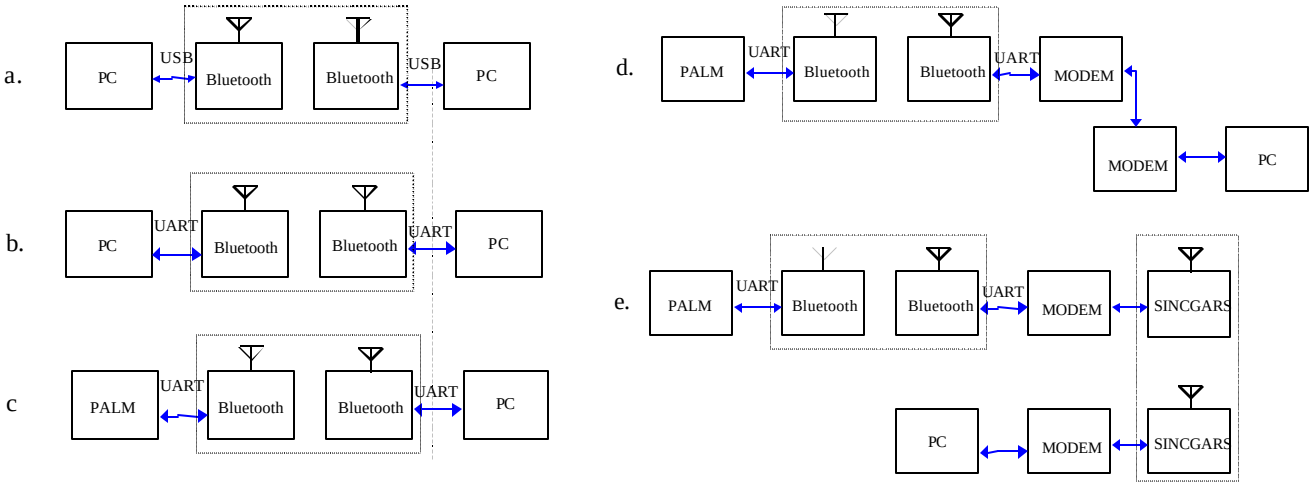

Figure 2 Bluetooth design steps

4.5 Integration of Parts

After all three concepts have been proven, the separate pieces need to be tied together. First, the client must send the correct data to the server depending on what the user chooses from the interface. The devices ought to be connected initially by a cable in order to isolate any ambiguous errors that may arise from either the Bluetooth or

software implementation. The Bluetooth, along with the modem and SINCGARS, can finally be integrated into the prototype once the system runs correctly from end to end (client to server).

The final prototype includes an interface, which is the user's only concern, that sends processed and formatted data to the server. The data is transmitted through two Bluetooth devices, modems and SINCGARS. The server side application stores the information in XML format for later use, access by another file, or view through a browser.

5. Conclusion

Although this project is currently still in development, the end result will be source code written in a platform-independent programming language that can be ported to any machine (capable of running a Java virtual machine). This software will allow users to input tactical data entries, which will be transported to an end system that processes the information. Extensive research has deemed that code written in Java and data transported in XML format will be ideal. The integration of Java and XML technologies provides for completely portable code.

6. Summary

The goal of this project is to produce platform-independent software for tactical data entry devices. Ultimately, Java was chosen as the most appropriate programming language. First, Java provides specific environments, which are ideal to this situation. Sun offers KVM and its corresponding CLDC and kAWT classes for the PalmOS and the Personal Java for the WinCE devices. These classes were designed specifically for development with PDAs. There were several other alternatives to Java, which were created for PDA development. However, Java was one of the only platform-independent languages. Waba shared similar advantages as Java; however, was rejected because of its incapability with Bluetooth. Bluetooth was chosen as the wireless communication piece because it possessed ideal features not supported by its alternatives. The ability to control this device using Java became very integral to this project. Finally, the PalmOS and WinCE devices were chosen because of its common use. They relatively cost efficient handheld devices that support the needs of this project.

Endnotes

(1) Peter Kestenbaum, "What is Java? A 10 minute guide for the uninitiated." On-line. Available from http://www.javaworld.com/javaworld/jw-03-1996/jw-03-10minute.java_p.html. Accessed 12 February 2001.

(2) "Object Oriented Basic Concepts and Advantages." On-line. Available from http://www.mmrg.ecs.soton.ac.uk/publications/archive/melly1995a/html/node3.html. Accessed 12 February 2001.

(3) "The Source for Java Technology." On-line. Available from http://www.java.sun.com. Accessed 12 February 2001.

(4) "Wabasoft." On-line. Available from http://www.wabasoft.com. Accessed 12 February 2001.

(5) Nazmul Idris, "Benefits of using XML." On-line. Available from http://65.1.136.127/developerlife/xmlbenefits/default.htm. Accessed 12 March 2001.

(6) "Java Technology and XML: Portable Code, Portable Data." On-line. Available from http://java.sun.com/xml/. Accessed 12 February 2001.

(7) "Zucotto Wireless." On-line. Available from http://www.Zucotto.com. Accessed 12 march 2001.

(8) (IDC, June, 2000)

(9) "The Philosophy Behind the Palm OS." On-line. Available from http://www.palmos.com/platform/philosophy.html. Accessed 12 February 2001.

(10) "Windows CE." On-line. Available from http://www.microsoft.com/Windows/embedded/ce/default.asp. Accessed 12 February 2001.

(11) "Windows CE Product Information." On-line. Available from http://www.microsoft.com/windows/embedded/ce/guide/default.asp. Accessed 12 February 2001.

(12) Ericsson Microelectronics AB, *ROK 101 007 Bluetooth Module* (Kista-Stockholm: Ericsson Microelectronics AB, 2000).