Platform Independent
Tactical Data Entry Devices

MAJ John Surdu, LTC Ken Alford,
2LT Gene Yu, 2LT William Herrington, and 2LT Charles Maranich

***Index terms*—**Platform Independence, Java, XML, Whiteboard SDK, Palm OS, Windows CE, Bluetooth, Waba

Abstract—As the Army becomes increasingly digitized, the need for technically advanced equipment is imperative. This project investigates the implementation of platform independent software to run on common handheld units. The advantage of a "software first" approach is that it frees the Army from over-reliance on vendors/integrators. The research described in this paper is an attempt at a platform and vendor independent data input device for Field Artillery forward observers.

I. INTRODUCTION

Many of the current digitized systems used by the Army rely on proprietary hardware and software, which often proves to be cumbersome and expensive. While using hardware dependent software often results in better, short-term performance, the Army commits to specific vendors even after some of those technologies have become outdated. This often restricts further innovation and adaptation of the system. The Army is exploring the development of platform independent software to achieve cost savings and greater flexibility.

In particular, the Army is concerned with a replacement for the Handheld Terminal Unit (HTU), which is a portable device that primarily sends requests for indirect fire (i.e., a fire mission) via wireless transmissions from a soldier, on the battlefield, back to the commander's Advanced Field Artillery Tactical Data System (AFATDS) [1].

The HTU is capable of sending a large variety of data back to the AFATDS to aid the commander in "seeing the battlefield," but it is important to note that the principal use of the HTU is to transmit calls for indirect fire support. (A call for fire is the procedure used by a forward observer to request and adjust indirect fire, such as artillery, naval gunfire, and close air support.) The current HTU fielded by the Army is heavy (weighs over eight pounds), difficult to handle, and expensive (approximately $17,000). The software used on the HTU is written in C++ and is not platform independent.

The purpose of this project is to demonstrate the utility of platform-independence software by developing a proof-of-concept replacement for the HTU with platform independent code running on a less expensive, smaller personal digital assistant (PDA), such as a Palm Pilot or Pocket PC.

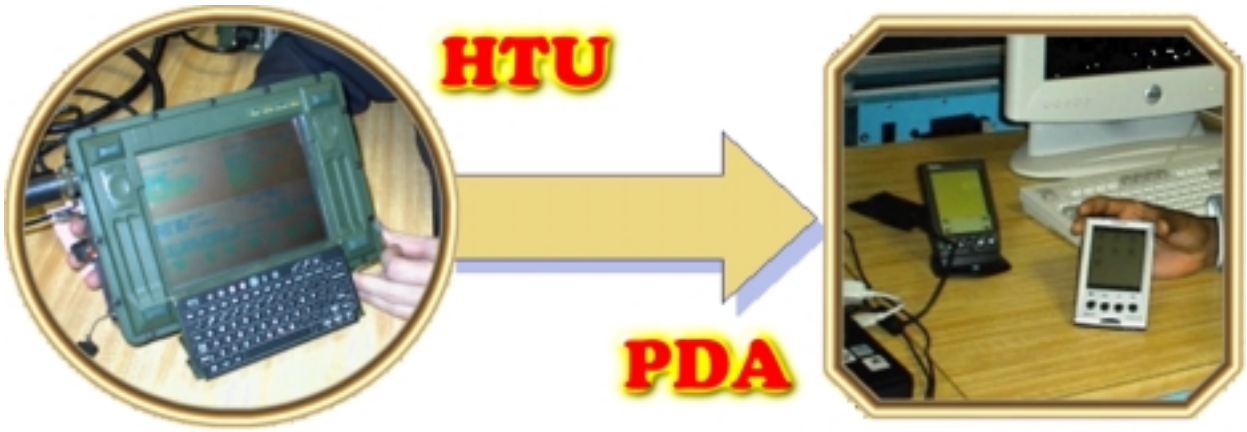

Figure 1. Project Goal

II. REQUIREMENTS

Hardware

The communication link between the PDA and the Single Channel Ground and Airborne Radio System (SINCGARS) must be wireless to permit the user to move freely without taking the heavier SINCGARS radio with him. The combination of the HTU replacement and the communication system should duplicate the call-for-fire functionality of the current HTU, connected by a cable to the SINCGARS radio.

The replacement HTU PDA must meet the following requirements:

a) Must be ruggedized and connect securely to any chosen Palm OS or Windows CE device
b) Must allow for serial connection to any Palm OS or Windows CE device
c) Must allow for the addition of a signal security system
d) Must allow data rates comparable with existing system

The bridge module (Bridge) must meet the following requirements:

a) Must be ruggedized and connect securely to the SINCGARS
b) Must allow for connection to the data port of the SINCGARS
c) Must allow for the addition of a signal security system
d) Must have data rates and format at the SINCGARS consistent with the current system
e) Must contain control features to allow the SINCGARS to transmit data received to the remote module
f) Must allow data rates comparable with existing system

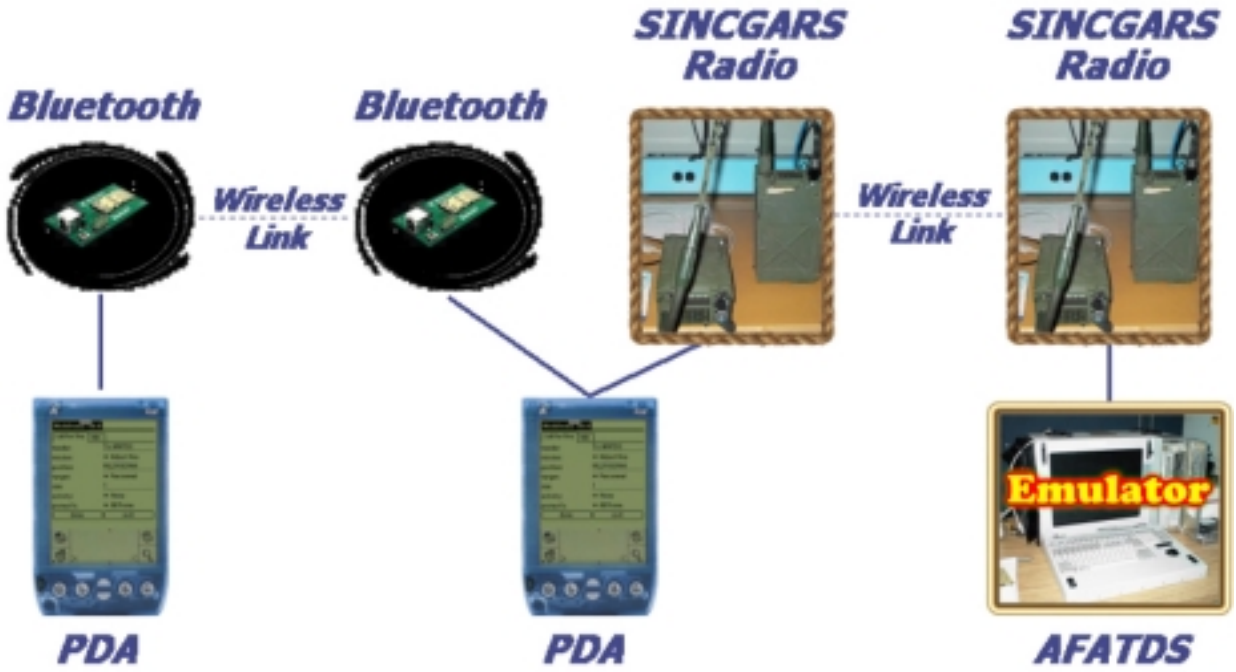

Figure 2. Project Design Concept

Software

The main software requirements for the HTU PDA are:

a) Must be platform independent (i.e., at this point Windows CE and Palm OS)
b) Must be capable of communicating wirelessly to the Bridge machine
c) Must be capable of asynchronous communications
d) Must be able to log and store message history
e) Must allow an artillery forward observer to input all types of calls for fire specified in Army Field Manual 6-30 [2]

The main software requirements for the Bridge are:

a) Must be able to communicate through the serial port to the SINCGARS radio
b) Must be able to communicate wirelessly with the HTU PDA
c) Must store a history of messages

III. ENABLING TECHNOLOGIES

In designing this system, a number of enabling technologies were considered. These included Java, Waba, Kada, Visual Age Java, and XML. As it turns out, despite its relative success on the desktop (through Java and some well accepted scripting languages), platform independence on small devices is elusive. After considering a number of candidate technologies, we chose to implement this prototype using Java 2 Micro Edition (J2ME), Bluetooth, and XML.

Since its original release in 1995, one of Java's major claims has been "write once, run anywhere [3]." The user interface application programmer interface (API) was constructed in such a way that the developer does not have to worry about specific platform issues. This platform independence is gained through the use of platform *specific* virtual machines that interpret the Java byte codes.

As platform independence is a requirement of this project, several versions of Java were considered. One such Java version, Java 2 Micro Edition (J2ME) has been evolving over the past year and a half [4-6], but currently it runs on the Connected Limited Device Configuration (CLDC) devices using the Mobile Information Device Profile (MIDP) [6]. Several MIDP devices exist, including Palm Pilots, Handspring Visors, Motorola i85 phones, and other phones and pagers overseas. The version of Java that runs under Windows CE is Personal Java. J2ME CLDC MIDP is very limiting to application programmers, as it was intended for devices even smaller than the Palm Pilot, but it is currently the best hope for full platform independence.

Several commercial vendors offer Java-like languages that promise platform independence. One of these is Waba. Waba has much the same syntax and structure as Java, but Waba is not Java [7]. At the time this project began, Waba was considered untested and unstable for our purposes. Kada is a full implementation of Java 2 Standard Edition (J2SE), the normal Java for desktop computers. Kada has its own compiler that compiles J2SE to Kada byte codes [8]. These byte codes do not run on normal Java virtual machines; they require Kada proprietary virtual machines. For purposes of this project, tying the project to another proprietary vendor did not seem to adhere to the spirit of the project requirements.

The use of eXtensible Markup Language (XML) was considered for this application. XML is a platform independent, industry supported data representation language. 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. Although XML looks like HTML because they are both markup languages, XML uses tags to represent data, whereas HTML uses tags to format information [9]. The World Wide Web Consortium (W3C) provides a limited set of pre-defined tags for XML. Additionally, they allow users to define their own tags through Data Type Definitions (DTDs).

Java is portable source code. XML is portable data. By incorporating portable data into portable code, portable applications can be developed. Sun Microsystems currently offers several Java APIs for XML parsing and messaging utilities at the Java home page [10].

There are essentially two non-proprietary, short-range, wireless communications architectures to consider in this project. The first is IEEE 802.11, and the second is Bluetooth. When this project began, there was little PDA support for either technology. In the spirit of trying new technologies, we chose to use Bluetooth. As there were few Java API's for Bluetooth, this severely hampered development. Zucotto makes Whiteboard SDK, Bluetooth Edition [11] that is a Java implementation of the Bluetooth protocol stack; however, when we began this project, it would only run on the Zucotto device emulator. The more recent version of Whiteboard SDK that does not require their emulator was not released until after the first prototype was built.

Palm OS

When PCs first came on the market, many mainframe and minicomputer companies tried to compete by adapting their existing designs to smaller boxes. They generally failed because the PC was a new form of computing with different requirements. Handheld devices 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 typically spends in front of a PC. . Palm handheld devices are designed to be as simple and intuitive as possible. Information searches and data entry are designed to be short and easy.

Rather than "dumbing down" a PC, Palm started with a new design that focused on giving users the shortest and easiest path to information. Products based on the Palm OS are the world leaders in handheld computing with more than 75% market share worldwide and are the market leaders in both the US and Europe [12]. Sales of Palm powered handheld devices doubled in each of the last two years.

Windows CE

Windows CE is a 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 handheld devices, 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 [13].

IV. DESIGN

Hardware

Our system consists of a wireless connection between a PDA and a PC via Bluetooth and SINCGARS radio. On one end, we have a Palm OS or Windows CE PDA running two pieces of software: a GUI-based call-for-fire program and a communications program that passes output from the call for fire to the Palm's serial port. Data passes to a Bluetooth module connected to the Palm serial port. An independent power module providing a 4.8 V voltage level powers this Bluetooth module. The power module is made up of four 1.2 V rechargeable Nickel Metal Hydride batteries. The data is passed via a 2.4 GHz frequency hop transmission to another Bluetooth module attached to the SINCGARS.

This Bluetooth module is powered by another power module identical to the first. It is controlled by another Palm device through the Palm's serial port. The Palm reads data from the Bluetooth module and then passes it unchanged to the SINCGARS through a serial adapter in the memory

expansion slot. It uses the RS232 communications protocol to communicate with the SINCGARS.

The SINCGARS transmits data to another SINCGARS, which is connected to a PC running an AFATDS emulator program. This program reads data from the SINCGARS through the serial port and displays it in an understandable format in a GUI.

Software

There are three software components to this project: the HTU PDA (i.e., the replacement for the current HTU), the Bridge module, and the AFATDS emulator. All data sent between these three pieces will be ASCII text, in XML format.

HTU PDA: This module is the soldier's interface into the fire control system. The forward observer will fill in the call for fire and then transmit that message to the AFATDS emulator. The GUI is written in Java 2 Micro Edition, Connected Limited Device Configuration Mobile Information Device Profile (J2ME CLDC MIDP) and can run on any MIDP device, such as MIDP enabled pagers, cell phones, and Palm OS devices.

While CLDC supports several communications protocols, MIDP only supports Hyper Text Transfer Protocol (HTTP) connections. The requirement to use HTTP drove the implementation decision for the Bridge, described below. The requirement for a Bridge also enabled much of the heavy storage and computation to be moved to the Bridge, making the HTU PDA a very "thin" client.

Bridge: Most PDAs do not have two communications ports. While there are Compact Flash serial port adapters for PDAs that have Compact Flash or Personal Computer Memory Card Industry Association (PCMCIA – or just PC Card) interfaces, the J2ME virtual machines only allow a single port to be opened at one time. The Bridge then is currently implemented on a laptop, running J2SE so that the Bridge has access to two ports, a serial port to the SINCGARS radio and a wireless device. The mission of the Bridge initially was merely to forward messages from the AFATDS emulator to the HTU PDA and vice versa. Since the Bridge hardware is a laptop, the architecture module was modified to make the bridge the central data repository and make the HTU PDA client much thinner.

The Bridge software consists of the Tomcat Java server and a Java servlet that implements the logic of the Bridge. The elegance of this solution is that any machine capable of running a Java server can be used as the Bridge if it has wireless connectivity and a serial port. The Tomcat server is not required; any Java server will suffice. The servlet manages the forwarding of messages between the HTU PDA and the AFATDS emulator. As future work, it will also handle messaging with GPS devices, laser rangefinders, and other sensors.

The Bridge will also manage the archiving and tracking of messages so that the HTU PDA needs only maintain an index list of messages, not all the messages themselves. If the forward observer wants to recall an earlier call for fire to repeat that mission, he will search through an

index on the HTU PDA. When he decides to send the mission again, the Bridge will forward the mission from its archive.

The need for a "smart device next to the SINCGARS radio was mandated by the choice of Bluetooth as the wireless technology for this project. At this stage in Bluetooth's evolution, the Bluetooth radio needs a computer to configure it and access its protocol stack. As we later moved to IEEE 802.11b, the need for a Bridge might not be necessary. It is, however, a good design decision, as this will provide a central node for HTU messages, GPS message, and messages from laser rangefinders and other sensors.

AFATDS Emulator: The AFATDS emulator is a simulation of the actual U.S. Army AFATDS. The current HTU and AFATDS exchange messages in Variable Message Format (VMF) 11 protocol. This protocol is cumbersome and difficult to implement.

According to Pete Batcheller, a project manager working on the same problem at Booz, Allen & Hamilton, over fifteen engineers have spent over two person-years translating the code from C++ to Java that translates the data correctly into VMF 11 radio protocol [14]. This proof-of-concept system uses a re-implementation of the messaging in XML. This was done for two reasons: 1) work could begin on the novel concepts of using PDAs in this application rather than on re-implementing VMF and 2) XML appears to be a good replacement for VMF in the long run. Since this prototype is sending data in ASCII text, AFATDS could not understand the data; therefore, an emulator of AFATDS was used to test the functionality of the HTU PDA. Using XML formatted data was platform independent and provided a simple and powerful way to parse the data on the AFATDS emulator. The AFATDS emulator is a GUI that sits on a desktop and receives the replacement HTU's messages (in XML), displays them in a text area, and provides buttons to send back confirmation replies to the HTU.

V. CURRENT STATE

Due to difficulties using Bluetooth in Java, we decided to start over using IEEE 802.11b, which proved to be significantly easier to implement in Java. The client software on the PDA is a multithreaded application, written in J2ME CLDC MIDP. The bridge machine is implemented as a Tomcat [15] Java server, which invokes a J2SE servlet to service requests from the user. The bridge currently runs on a laptop, but it will soon be ported to a ruggedized sub-notebook. The connection to the SINCGARS radio is through the bridge's serial port. The SINCGARS radio is currently being simulated on a desktop platform, but we will soon be using the real radios.

A sample XML implementation of the VMF protocol has been written, along with a schema to help validate the messages during testing. The GUI is rudimentary, but we have an ongoing effort to improve it. (At this point, it must be emphasized that we are not trying to reverse engineer the current HTU; rather, we are looking at new ways to facilitate the call-for-fire process.)

VI. FUTURE WORK

Future Technology

While we have a working prototype that demonstrates the underlying communications infrastructure, much work is needed before the system could be used by soldiers. Some of the improvements include:

- Redesigning the process by which forward observers call for fire and implementing this process in a unique, innovative GUI.
- Completing the implementation of VMF in XML.
- Adding communication with other devices. Soldiers should never have to type coordinates for friendly and enemy locations into the PDA. There are GPS flash card modules that can be inserted into PDA's that may be used in conjunction with a device to send the grid coordinates to the AFATDS from the HTU. A lasing device is a laser beam that reflects off a target and indicates the distance to the target. With the GPS module, the HTU may automatically determine the grid coordinate and the distance to the target. Then, the user could click a button to send an accurate call for fire.
- Adding security. The current prototype is susceptible to the possibility of interception and spoofing. Additionally, there is no user authentication. An enemy could use a captured device to send spurious calls for fire. Also, there are no actual security measures on the HTU itself. In the event that a soldier loses the HTU, we do not want the enemy able to pick it up and call for fire using our fire batteries. It is essential to incorporate some kind of password system and authentication between the HTU and the AFATDS.

VII. CONCLUSIONS

The software and hardware discussed in this paper hold great promise for the Army, but there are still many technological challenges that must be solved. Development of platform independent software on small devices is still difficult, but getting easier. Wireless networking from small, limited devices is just becoming practical in Java. Because of the processor and memory limitations of PDAs, clients must be thin, and intermediaries or servers must take up much of the processing burden.

Support for platform independent software for handheld devices is currently at the "bleeding edge" of technology. While innovations and vendors should appear in the near future, implementing platform independent software for the U.S. Army's tactical entry devices is difficult. Several technologies are promising to help and should aid the soldier tremendously on the digitized battlefield. In the short term, however, a first proof-of-concept system was developed to demonstrate the feasibility and potential benefits of platform independent software. The innovative combination of Java, XML, and wireless technologies has shown potential benefits for the Army.

VIII. REFERENCES

[1] Raytheon Corporation, at http://www.raytheon.com/c3i/c3iproducts/c3i060/c3i060.htm, 20 April 2001.

[2] U.S. Army, *Tactics, Techniques, and Procedures for Observed Fire*. Washington, DC: Headquarters, Department of the Army, 1991.

[3] B. Eckel, *Thinking in Java*. Upper Saddle River, NJ: Prentice Hall, 1998.

[4] J. Knudsen, *Wireless Java: Developing with Java 2, Micro Edition*. New York, NY: Apress: Springer-Verlag New York, 2001.

[5] Y. Feng and J. Zhu, *Wireless Java Programming with J2ME*. Indianapolis, IN: SAMS, 2001.

[6] R. Riggs, A. Taivalsaari, and M. VandenBrink, *Programming Wireless Devices with the Java 2 Platform, Micro Edition*. Boston: Addison-Wesley, 2001.

[7] Wabasoft, "Wabasoft," at http://www.wabasoft.com, 20 April 2001.

[8] Kada Systems, "The Kada VM," at http://www.kadasystems.com/kada_vm.html, 20 April 2001.

[9] Sun Microsystems, "Java Technology and XML," at http://java.sun.com/xml/, 12 February 2001.

[10] Sun Microsystems, "The Source for Java Technology," at http://www.java.sun.com, 12 March 2001.

[11] Zucotto Wireless, "Zucotto Wireless, Inc.," at http://www.zucotto.com/, 24 September 2001.

[12] Palm OS, at http://www.palmos.com/platform/philosophy.html, 12 February 2001.

[13] Microsoft, "Windows CE," at http://www.microsoft.com/Windows/embedded/ce/default.asp, 12 February 2001.

[14] P. Batcheller, 2 February 2001. Joint Combat Information Systems, Booz Allen & Hamilton, Inc. In a series of email messages and phone conversations, Mr. Batcheller described the work his project team had finished and the amount of work required to use the VMF 11 protocol to send data to the AFATDS

[15] The Jakarta Project, "The Jakarta Site," at http://jakarta.apache.org/, 24 September 2001.