A Web-Based Java Framework for Cross-Platform Mobile GIS and Remote Sensing Applications

Ming-Hsiang Tsou, Liang Guo, and Anthony Howser
Department of Geography, San Diego State University, San Diego, California 92182

Abstract: A cross-platform Web-based Java development framework for Mobile Geographic Information Systems (Mobile GIS) and remote sensing (RS) applications is introduced for the notebook computer, Pocket PC, and mobile phone platforms. Using these platforms, Java software technology is examined for its cross-platform utility in the development of various Mobile GIS and map/image display functions. The three case studies developed with Java 2 Standard Edition (J2SE), Java 2 Micro Edition (J2ME), and Mobile Information Device Profile (MIDP) are examined within the context of mobile GIS. Significant challenges in developing cross-platform Mobile GIS applications are also discussed. These obstacles include heterogeneous operating systems, different wireless communications protocols, low-bandwidth network connections, and the general lack of usability.

INTRODUCTION

Compared to Desktop Geographic Information Systems (Desktop GIS), Mobile GIS can provide Geographic Information Services (GIServices) in a more portable platform to facilitate field-based data collection and access (Tsou, 2004a). By connecting to wireless Internet map/image servers and Global Positioning Systems (GPS) data feeds, Mobile GIS devices can display map layers and remotely sensed imagery effectively for various field-based GIS tasks.

Many practical field-based GIS applications and tasks require multiple hardware devices for the query and display of spatially associated data. An example scenario may involve delivery truck drivers who require maps of the optimal/shortest routes between their delivery/pickup points. The drivers may wish to display their maps on Pocket PC devices housed in their vehicles. If the drivers are not in their vehicles, they may want to display the same maps on their mobile phones. Furthermore, the drivers may want to generate hard copies of the same maps from their desktop computers back in their offices. The requirements for this scenario mandate similar user interface functionality and software interoperability with the map data on all three of the computing platforms. Unfortunately, this kind of interoperability has seldom been investigated. A significant obstacle to creating such a cross-platform map/image query-and-display application is the general lack of software tools that can be used to develop applications for multiple computing platforms.

GIScience & Remote Sensing, 2005, **42**, No. 4, p. 278-302.

Fig. 1. Three different Mobile GIS platforms (notebooks, pocket PCs and mobile phones).

Mobile GIS is a relatively new field in geographic information science (GIScience) research. Mobile GIS applications emphasize flexible access to geospatial data and location-based services through mobile computing devices. With the advancement and convergence of GPS, Internet, wireless communication, and mobile computing technologies, Mobile GIS has great potential to play critical roles in the application domains of field data acquisition and validation (Pundt, 2002), and emergency vehicle routing services (Derekenaris et al., 2001).

By exploiting the interoperability characteristics of the Java programming language, Mobile GIS applications for three different platforms are examined and discussed. These platforms include the notebook computer, the Pocket PC, and the mobile phone (Fig. 1). In each of these case studies, functionality and usability of GIS and remote sensing applications is discussed.

CHALLENGES

Presently, Mobile GIS applications are appearing at an increasingly rapid pace in private companies (Crisp, 2003), government agencies, and academic research institutions (Tsou, 2004a). However, there are some major challenges in the development of cross-platform Mobile GIS applications, including:

Non-Standardized User Input. Different Mobile GIS hardware platforms have dissimilar user interfaces and user interaction requirements. Most Pocket PC devices and mobile phones provide a very limited user interface compared to notebook or desktop computers. They are generally limited to such things as touch screens or numeric keypads. Useful and effective user interfaces, which can be utilized on applications across different hardware platforms, are not readily available. Basic mapping

functions with GIS/RS datasets, such as zoom-in, zoom-out, and pan, will require different controls for each hardware platform. A possible solution to this issue includes development of the user interface utilizing the rich cross-platform user interface tools provided in the Java application development framework.

CPU Speed/Power, Memory and Data Storage Capacity. Mobile GIS devices generally have less computing speed/power compared to their Desktop GIS counterparts. Desktop PCs usually have a significantly faster processor that can take advantage of the greater space (large computer case), more power (large power supply constantly plugged into an outlet), and more powerful cooling (high-wattage fans) available to them. These factors allow for a significantly more powerful, faster, and correspondingly hotter computer. On the other hand, Central Processing Units (CPUs) in Mobile GIS devices, which are optimized for portability, may have very little space, may run on very low power, and may have been designed to not run as hot compared to their Desktop GIS hardware counterparts. Currently most Pocket PCs are equipped with a 500 MHz–600 MHz CPU. This is in contrast to Desktop PCs with a 1.5 GHz–3 GHz CPU. Memory and storage is another major limitation of Mobile GIS devices. Most handheld computing devices have very limited memory and storage capacity with up to 4 gigabytes of nonvolatile flash memory. On the other hand, traditional Desktop GIS workstations can have upwards of 200–300 gigabytes of hard disk space. Memory and storage requirements can be significant for large GIS and RS datasets.

Screen Size, Resolution, and Colors. Mobile computing devices generally have smaller screen areas, more limited screen resolution (pixels per unit area), and more limited color depth (number of displayable colors) for GIS/RS data display. The lack of a large screen area can significantly impact the usability of GIS/RS data display applications.

Dissimilar Operating Systems. There are several mainstream operating systems for mobile computing devices. Some include Windows Pocket PC, PalmOS, Symbian, and the Java Phon e Operating System. Adapting customized software for various platforms can be ve ry costly and time consuming. Mobile GIS software designed for the PalmOS mobile computing operating system will not work on a Pocket PC device unless it was developed using an interoperable programming framework such as Java. Currently, most Mobile GIS software packages are platform/operating-system specific.

Table 1 highlights some major differences between the three platforms discussed for Mobile GIS applications. Factors included are user interface, CPU, memory, screen characteristics, and mobile computing platform operating systems.

In the context of mainstream GIS applications, most mobile devices are severely restricted and/or inadequate due to their small screen area, limited memory, and limited computational power. Software application programmers have to overcome these limitations in order to provide a functioning and easy-to-use user interface. In Table 1, some of the limitations might be solved advances in computer engineering technology, such as CPU, battery life, and memory storage. For example, the CPU speed of Pocket PCs has been significantly improved, from 70 MHz in 1999 (Compaq Aero 1530) to 624 MHz in 2005 (HP IPAQ hx4700). Some limitations are inevitable, such as screen size and user interface/input.

Table 1. A Comparison between Three Different Mobile GIS Platforms[a]

Typical Mobile GIS platforms:	Notebook PCs	Pocket PCs, PDAs, hybrid smart phones[b]	Mobile phones
User interactivity/ user interfaces	Regular keyboard, mouse, or pointing devices	Touch screen with stylus pen, limited function buttons	Keypads with limited function buttons
CPU	2.0 GHz (2000 MHz) (Intel Pentium M 760)	600 MHz CPU	200 MHz (some models have multiple CPUs)
Memory and data storage	512 MB RAM and 60 GB hard drive	CF-card with 1–2 GB RAM	5–30 MB RAM
Screen size and resolution	14- or 15-inch screen, 1024 × 768 or 1280 × 800 pixels	3.2- to 4.0-inch screen, 320 × 480 or 640 × 480 pixels	2-inch screen, 176 × 220 pixels
Color depth/display	True color (24 or 32 bits) = 16+ million distinct colors	VGA color (16 bit), 65,536 colors	(16 bit) 65,536 colors or B/W display
Operating system	Windows or MacOS	Pocket PC, Smart Phone OS, or PalmOS	Phone hardware vendor–based, proprietary systems
Battery life	2–5 hours	5–8 hours	5 hours (includes talk time)

[a]Based on 2005 hardware specifications.
[b]PDA/mobile phone combo.

One possible solution to address these limitations is to develop mobile computing device applications for the Java Runtime Environment (JRE). What makes Java an effective candidate for facilitating interoperability between the varying platforms lies at the heart of the Java programming language.

Unlike most computer programming languages where raw programming code is compiled for a *specific* CPU and operating system, the Java programming language is compiled for a generic CPU and operating system. This makes the code more portable. When raw Java programming code is compiled, a Java compiler creates standardized Java "bytecode." This bytecode represents a highly optimized set of instructions for a Java Virtual Machine (JVM) (Schildt, 2004). The JVM, a key component of a JRE, is actually software that simulates a generic CPU and operating system, and it interprets Java bytecode in a uniform way (Hamilton, 1996). JVMs are available for most computing platforms and operating systems and they are packaged within a JRE. As long as the varying computer platforms have the same version of the JRE installed, they will read, interpret, and execute the Java bytecode in the *exact* same manner regardless of the operating system, computer hardware platform, etc. Because of the portability of Java bytecode and the availability of Java Runtime Environments (with corresponding JVMs) for all three platform study cases, it was deemed to be an appropriate software programming framework to develop interoperable GIS/RS applications for different hardware platforms.

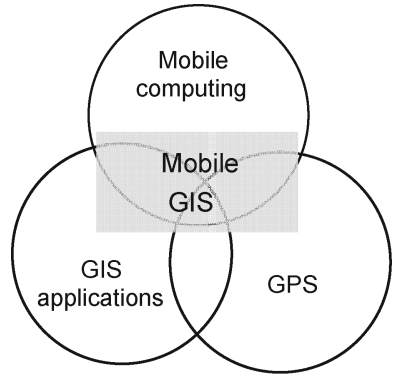

Fig. 2. The components of Mobile GIS technology.

THE DEVELOPMENT OF MOBILE GIS AND JAVA PROGRAMMING

Mobile GIS extends the capabilities of traditional Desktop GIS with a higher level of portability, flexibility, and utility. Mobile GIS are integrated software/hardware frameworks for the access of geospatial data and services through mobile devices via wireline or wireless networks (Tsou, 2004a). Unique features of Mobile GIS include the ability to incorporate Global Positioning System (GPS) receivers and ground truth measurement equipment and functionality alongside the GIS applications. Typical Mobile GIS applications include field data acquisition and validation, onsite real-time incident investigation and analysis, real-time work order management, and real-time service/dispatch requests. In general, Mobile GIS is composed of three distinct technological components: Internet-enabled GIS applications, GPS technology, and handheld mobile computing platforms (Fig. 2).

The platforms of popular mobile computing devices include handheld computers such as the Pocket PC and combination Personal Digital Assistant (PDA)–smart phone variants. The two most common operating systems (OS) for mobile computing devices are the Microsoft Windows Compact Edition (CE) OS and the Palm OS. Generally, Pocket PCs and smart phone variants allow for data storage, indexing, and querying. Typically, data entry is accomplished with a scaled-down keyboard, a pen-like stylus, or both. Mobile computing devices, like their larger notebook computer cousins, are powered by rechargeable batteries.

The adoption of mobile computing technology for personal and commercial use is quite evident in mainstream society. Everyday examples include several variants of digital wireless communications technology. These include cellular phones, Wireless Application Protocol (WAP)–based devices, and Bluetooth-based devices for inter-device communications and networking. This technology can also be used to query, access, and display/execute existing entertainment content such as MPEG Layer-3 (MP3) audio files and digital video files, news alerts, stock market alerts, etc. Like the adoption of email in industry, mobile computing technology has also become a useful and accepted part of organizational infrastructure and practices. The rise in mobile

computing technology utilization is highlighted by estimates showing a 1.3 billion users and a $20 billion market by 2005 (Intergraph, 2002).

Mobile GIS is an integrated software/hardware framework for the access of geospatial data and services through mobile devices via wireline or wireless networks (Tsou, 2004a). There are two major application areas of Mobile GIS: field-based GIS and location-based services (LBS). Field-based GIS focuses on GIS data collection. Specific applications generally include field data validation and updating. These tasks may involve adding or editing map features or attribute tables on an existing GIS dataset. LBS generally focus on business-oriented location-management functions including location search, navigation, and routing, vehicle tracking, etc. (Jagoe, 2002; OGC, 2003a). A major difference between the field-based GIS and LBS application areas involves data editing. Most field-based GIS applications require editing of the original GIS dataset. This includes feature attribute modification. A specific example may entail the survey of out-of-order stoplights by a local government employee. LBS, on the other hand, rarely involve changes to an original GIS dataset. Instead, the original data are used as persistent reference data, features, or "background" map layers for query, navigation, or tracking applications.

Because Mobile GIS applications generally require cross-platform functionality, the Java programming framework has been identified as a feasible and effective tool for Mobile GIS software development. One of the original goals of Java was to provide a software application development solution for a heterogeneous and network-based distributed environment of varying systems (Gosling and McGilton, 1996). Because of Java's powerful cross-platform capabilities, many software developers and organizations launched initiatives to explore the utility of using Java for online and distributed applications (Halfhill, 1997).

Distributed Java applications are generally implemented in two kinds of execution environments: a compile-time environment and a runtime environment (Schildt, 2004). The server-side compile-time environment generally entails the use of Sun Microsystem's Java Development Kit (JDK). The critical components of the JDK include a Java compiler (Javac.exe), a Java interpreter (Java.exe), and several standardized Java class libraries. Application programmers use the Java compiler to automatically link relevant Java class libraries and compile their text-based Java source code into a Java class file represented in Java bytecode format. This Java bytecode file can then be executed and interpreted on any local or networked computing system with a compatible Java Runtime Environment. The Java Virtual Machine, a key component of the runtime environment, actually interprets the bytecode file, links and loads any additional requisite compiled classes, and, in the ideal situation, produces the exact same actions, results, output, etc. on any system. This works because the Java Virtual Machine of the Java Runtime Environment, on the client-side system, takes the intermediate Java bytecode as input and produces machine code for that specific platform/system/device at runtime. This facilitates programming portability because the same Java bytecode file can be used as input to any system. Backers of the Java programming language describe this critical feature with the phrase, "Write Once, Run Anywhere (WORA)" (Douglas, 1996; WORA, 2005). The main caveat is the requirement of a compatible Java Runtime Environment, and associated Java Virtual Machine, on the client-side system. An out-of-date version of the runtime may not contain the proper libraries, syntax, or functionality that is called for in the

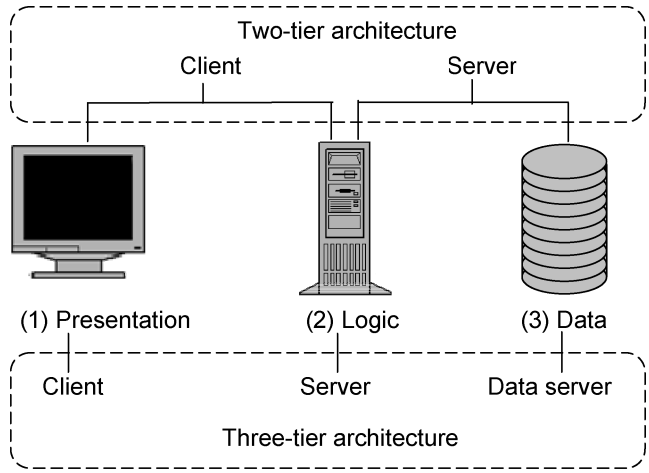

Fig. 3. Client-server system partition.

original programming code by the application developer. Thus, it is critical that the application developer and the end user ensure that the client-side system has an up-to-date runtime environment. To facilitate this, Sun Microsystems and many other software vendors include automatic runtime environment detection and update functionality when executing Java applications over networks and on client-side systems such as Internet Web browsers.

Java also supports execution over a network such as the Internet. The server-side bytecode class file may reside on a Web server and the client-side runtime system may be a Web browser with a Java Virtual Machine, or a mobile or desktop computing device with a Java Runtime Environment. This powerful feature of the Java application framework is called Remote Method Invocation (RMI) and further facilitates software application portability. In this instance, a constant Internet connection and a compatible Java runtime is all that is required to run an application. Backers of the Java programming language describe this feature with the expression, "The network is the computer" (Sun Microsystems, 1998).

In general, there are two different architectures for the deployment of Java-based GIS applications: the two-tier architecture and the three-tier architecture. Both of these architectures can be distinguished by the way the client-server components are partitioned. An application can be divided into three functional components: presentation, logic, and data (Shan and Earle, 1998). Figure 3 depicts the two-tier and the three-tier architectures.

The two-tier design allocates the presentation component to the client and the data component to the server (Peng and Tsou, 2003). The logic component is allocated between the clients and servers, although most of the application portion of the logic component is on the client environments (Sadoski, 2000). Thus, the first tier generally consists of the presentation and logic components and the second tier consists of the data and some logic components. Typical examples of two-tier architectures include client-side Java applications, or Java applets, running on Internet Web browsers (Fig. 4, left).

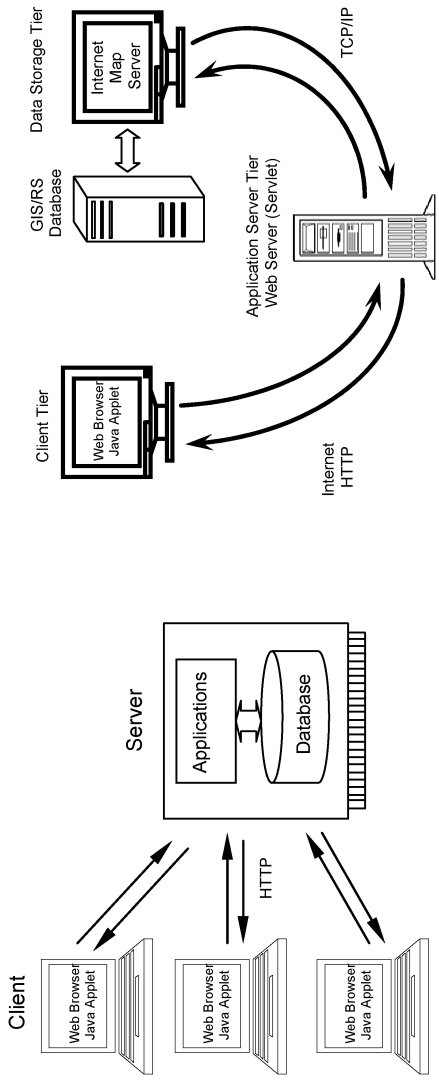

Fig. 4. Client-side application (left) and server-side application (right).

In these examples, Internet users can transparently download applications and access data remotely with a Java applet. Specific applications of Java applets include online GIS analytical tools providing advanced user interfaces, unlimited user interaction, and high-quality image rendering techniques (Andrienko et. al, 1999; Coddington et al., 1999). The applet can run within a Web browser or through a special applet viewer that is included with Java Runtime Environments. With either interface, the presentation tasks and simple data processing occur on the client side—that is, on the user's platform/system/device. On a Web browser, it is embedded and enabled with special HTML-like applet tags. Since direct or indirect communication is established with the server and data, applets allow most of the data processing to take place on the client computer. This can potentially reduce the processing load on the server and support a higher degree of availability to other users (Limp, 2001). Unfortunately, the applet implementation of a two-tier architecture has some drawbacks. If the size of an applet and/or its dataset is too large, computational processing will be slow and may tax the client computing device. The downloading time and functional performance can also be major concerns. These issues may effectively turn the client-side system into an overtaxed fat client, and even paralyze the client machine (Huang and Worboys, 2001). Moreover, the security of the data and files manipulated by the applet on the client may be an issue. This is a consideration for sensitive GIS and RS datasets and applications.

In a three-tier architecture, the presentation component (the first tier) is also handled on the client side, the logic component (the second tier) resides on the server side, and the data component (the third tier) resides on a stand-alone data server (Peng and Tsou, 2003). Examples of three-tier architecture systems typically include Web applications using server-side Java applications, or Java servlets (Fig. 4, right). Much of the functionality of these server-side components involves marshaling data, requests, and responses. Like the Java applet extending the capabilities of a Web browser (or a stand-alone applet viewer) on the client side, the servlet extends the functionality of the server on the server side (Huang and Worboys, 2001). Applets "listen" for and respond to client-side messages or requests; and proceed to gather and process data and execute instructions at the server side according to their programs. This middle-ware tier is used to process user requests, produce maps, and manage server tasks (Tsou et al., 2002). With it, complex data processing and data security can take place away from the client. After execution, the results, or other output, are generated and transferred to the client (Tsou and Buttenfield, 2002). Unlike in a two-tier architecture, nearly all of the computation and data processing is performed on the server side in a three-tier architecture. Because of the lack of any significant processing and execution at the client side, the client is usually referred to as a "thin client." A thin client only manages the presentation component of the three-tier architecture. This generally entails a networked computing device providing a user interface and a means to receive and/or display server-generated output.

Three-tier architectures can be more flexible than two-tier architectures. Three-tier architectures can provide flexible access to an ever-expanding number of datasets, databases, datastores, or Internet map servers available based on user requirements (Fig. 4). Unfortunately, three-tier architectures can suffer from too many client requests for services. This can lead to slow data transfer between clients and servers, and unacceptably slow interface display and interactivity. In the context of Mobile

Table 2. Two-Tier versus Three-Tier Java Architectures

Three-tier architecture	Two-tier architecture
Advantages	
1. Robust server-side Java solution	1. Robust client-side Java solution
2. Easy to develop and deploy	2. Advanced graphical user interface (GUI)
3. Allows complex processing to take place at the server side	3. High-quality image rendering possible
4. Data security is guaranteed	4. Unlimited user interaction with client-side applets and data
5. Adherence to distributed GIS standards and compatible with most modern browsers	5. No need to frequently send and receive messages across the Internet
Disadvantages	
1. Low interactivity level because of a limited display	1. Lack of complex analytical capability at the client side
2. Can hamper processing with too many client requests	2. Difficult to transfer or process large datasets
3. High-bandwidth requirement	3. Difficult to implement
4. Substantial data download delays possible with the ever-increasing size of vector and raster formats	4. Lack of data and software security
	5. No adherence to standards

GIS, this can take the form of painfully slow updates of sever-generated map imagery. Table 2 summarizes some advantages and disadvantages of the two architectures discussed.

CROSS-PLATFORM JAVA FRAMEWORK FOR MOBILE GIS AND REMOTE SENSING APPLICATIONS

Programs developed for the Java programming language should be able to be compiled and executed on any machine. Therefore, it is an ideal tool for cross-platform and distributed systems. These advantageous factors led several major GIS vendors to develop Java-based tools and solutions for serving geospatial data to Web users. These tools include ESRI ArcIMS, Autodesk MapGuide, Intergraph GeoMedia WebMap, and MapInfo MapXtreme. However, very little formal GIS research has been conducted on comparing cross-platform Java applications, and their differences and limitations (Tsou, 2004b).

Currently, there are three main Java framework technologies within the Java 2 Platform: Java 2 Micro Edition (J2ME), Java 2 Standard Edition (J2SE), and Java 2 Enterprise Edition (J2EE). *The Java 2 Standard Edition* is a framework of Java programming tools and libraries in support of general-purpose, personal, and client-side application development. J2SE is the most common edition of the Java 2 framework. Many useful Application Programming Interfaces (APIs) developed for J2SE are frequently employed in Internet GIS and online mapping applications. These libraries include Java Advanced Imaging APIs, Java 2D APIs, and Java 3D APIs.

The Java 2 Enterprise Edition is a Java framework that provides application development tools and libraries to support server-side, institutional, and IT-based application development. Software programmers can use J2EE Enterprise JavaBean class libraries to create re-usable distributed components for different Web Services and Business-to-Business (B2B) applications. Several key technologies of the Enterprise Edition include the integration of Java and Extensible Markup Language (XML), Java Servlets technology, and Enterprise JavaBeans.

The Java 2 Micro Edition is a Java framework that provides application development tools for small computing devices including cellular phones, pagers, smart cards, and Personal Digital Assistants (PDAs)/handheld computers. An example of this technology includes Java Card technology, which enables very small footprint Java programs to execute on smartcard and other handheld devices with very limited Random Access Memory (RAM). J2ME also supports small device "profiles." These are stated minimum collections of Java API libraries for each particular device or application. These specifications minimize program and runtime size while providing enough tools and libraries for embedded application development. An example Java profile includes the Mobile Information Device Profile (MIDP) which specifies a standard set of APIs for wireless communications and mobile devices (Sun Microsystems, 2000).

The performance of similar GIS and remote sensing applications on three Mobile GIS platforms using Java cross-platform development tools is examined. The first test case involves a execution of Java applets on a notebook PC platform running the Windows XP Operating System (OS) through a modern Java-enabled Web browser. The second test case entails a Pocket PC running the Windows CE OS and a third-party Java Virtual Machine (JVM) plug-in called "CrEme," developed by NSIcom Ltd (http://www.nsicom.com). The JVM on the Pocket PC enables Java application execution on Windows CE-based computing devices. Other JVM plug-ins are available for Pocket PCs, Palm OS–based devices, and other mobile computing devices. CrEme was specified for the Mobile GIS application because it is able to provide a full range of Java capabilities including Remote Method Invocation (RMI) for client-server application development; Java Native Methods (JNI) for embedding programming code in other languages within Java; and the Swing 1.1 library for advanced GUI development. The CrEme JVM plug-in was also noted for being very stable and robust. The third test case involves the development of a Java map viewer for the mobile phone platform utilizing the J2ME Wireless Toolkit. J2ME Wireless Toolkit is a set of software tools and libraries for developing wireless Java applications that are based on J2ME's Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP). These are a set of standards that support the small and limited computing environments of small wireless technology such as mobile phones. The applications developed using the toolkit include distributed applications that can be executed on mobile phones and PDAs.

There are similarities and differences between the Java programming frameworks and their corresponding API libraries for the three different platforms addressed. An example of the similarities includes the Java Swing API library for advanced GUI development. J2ME (used in the Pocket PC and mobile phone test cases) and J2SE (used in the notebook PC test case) utilize the same set of Java Swing APIs for GUI development. Since the CPU on a mobile phone or Pocket PC is

significantly less powerful than a CPU on a notebook PC, the same API libraries used in identical applications for both platforms will appear to execute more slowly on the Pocket PC or mobile phone. Further considerations of the Java J2SE and J2ME application development frameworks, as it applies to Mobile GIS application development, include the following.

1. Most Pocket PCs have restricted processing power. The CPU power of a typical Pocket PC is about 200 MHz, as opposed to a notebook PC with a more powerful 2GHz CPU. To overcome this shortfall, it is necessary to choose to implement only the most lightweight and efficient API libraries that best support the required application features. This software engineering task should be done before the development process begins to ensure smaller application footprints and better performance. In the context of the development of GIS or remote sensing data viewers, best practices include utilization of wireless client-server technologies and using Web services to implement complex GIS/RS procedures/functions. A solution developed for the Pocket PC or mobile phone platforms should only entail simple map display functionality on the client side. Finally, consideration should be given to certain aspects of the client application program design. Java Garbage Collection (GC) is a runtime feature that can be programmatically or automatically called to de-allocate references to unused physical memory in the computing device. GC frees up memory for application use. Unfortunately, this process can be taxing on a mobile computing system with minimal CPU resources. Frequent and unnecessary GC calls are computationally expensive, and may render an application unacceptably slow and inefficient for practical use.

2. There are many other significant memory constraints on Pocket PC and mobile phone platforms. Mobile computing devices have very limited storage and runtime memory. This constrains the developer to a small runtime environment and application footprint. Even after choosing lightweight libraries, the developers should carefully select the packaging solutions for application deployment. Part of this entails only including Java libraries and files that are actually implemented in the application. Additional considerations should also be made for the partitioning of the applications across the architecture of the whole system. By diligently allocating components of the application solution between servers and clients, additional memory savings can occur while preserving the functionality of the software.

3. The user display sizes vary greatly between the mobile phone, Pocket PC, and notebook PC platforms (Fig. 1). One of the most challenging aspects of creating cross-platform Mobile GIS solutions is addressing the user interface and data display. Screen real estate on the smaller platforms, as well as duplicating user interface functionality across all three platforms, are significant considerations. Traditional, large, and graphic-intensive GUIs are not possible or practical with the smaller mobile computing platforms. Creation of user-friendly interfaces for Pocket PC and mobile phone application solutions require implementation of such space-saving GUI features/tools as menus, list boxes, combo box dialogs, and screen tabs. The user interface for data browsers and simple map functions need to be carefully considered in application development for GIS/RS applications for mobile computing devices.

In summary, user interface simplification is one of the most important design considerations when developing Mobile GIS solutions on smaller computing devices such as the Pocket PC and mobile phones. With careful selection of lightweight API

libraries and a simplified GUI design, Java programming technology can be an effective tool to build robust and user-friendly GIS application tools for basic GIS/RS data display and analysis functions on mobile computing devices. The following sections describe three different Java implementations of Mobile GIS application solutions for GIS/RS functionality. These prototype solutions have been tested for environmental monitoring and natural resource management applications to demonstrate their potential capabilities.

CASE STUDY 1: J2SE APPLICATIONS ON NOTEBOOK PCS

The first study case entails the development of a Java Web-based imagery analysis application for a notebook PC using the Java 2 Standard Edition (J2SE) application framework. J2SE is the most popular framework for Java development of Web-based GIS and RS applications. The J2SE Java Development Toolkit (JDK) provides robust and well-defined APIs for geospatial data and digital image processing applications. Certain relevant API libraries include the Abstract Windows Toolkit (AWT), the Java 2D, and the Java Advanced Imaging (JAI) libraries. These API libraries provide flexible tools for integrating powerful image manipulation and GIS functionality within a software application solution (Sun Microsystems, 1999). The notebook PC application prototype adopted the newest Java 2D and JAI API libraries to include support and functionality for complex image processing tasks including image warping, image compositing, and image transparency. The Java 2D API library addresses a wide range of graphical, image, and text manipulation applications. The JAI API library focuses on fundamental digital image processing functionality (Rodrigues, 2001).

The JAI API library was used to develop image manipulation functions including semantic zooming, smooth panning, geo-linking viewers, contrast/brightness adjustment, minimum-maximum stretching, feature edge detection, and toggling of individual RGB bands. Custom change detection functionality was also developed for monitoring, detecting, and interpreting changes to natural wildlife habitats over time using RS imagery. Functionality for user-digitized polygons was also developed for the identification and analysis of changes at specific land areas. This feature enables the end user to identify critical areas of change, such as newly burned areas. Change detection is also possible with these tools because they make it possible to compare the shapes and sizes of digitized polygons for different temporal periods.

The test bed implementation was focused on identifying change under rapidly changing habitat conditions including non-native plant invasion, fire and post-fire succession, and recreation and transportation-related impacts to the physical landscape (Stow et al., 2001). All of these functions were accessible using a standard Web browser as an interface to a suite of image processing and GIS tools. One such tool, the Image Swipe Java Applet, is provided as an example in Figure 5.

The Image Swipe Tool was designed for Web-based GIS and RS applications. It was developed for flexible comparative analysis of overlaid geographically registered multi-temporal imagery. It enables the user to view specific portions of the layered top and bottom images at the same time using a vertical or horizontal swipe tool (Fig. 6). The swipe tool slider functions like a curtain and reveals different imagery separated by a thin vertical or horizontal slider bar. A user can manipulate the swipe

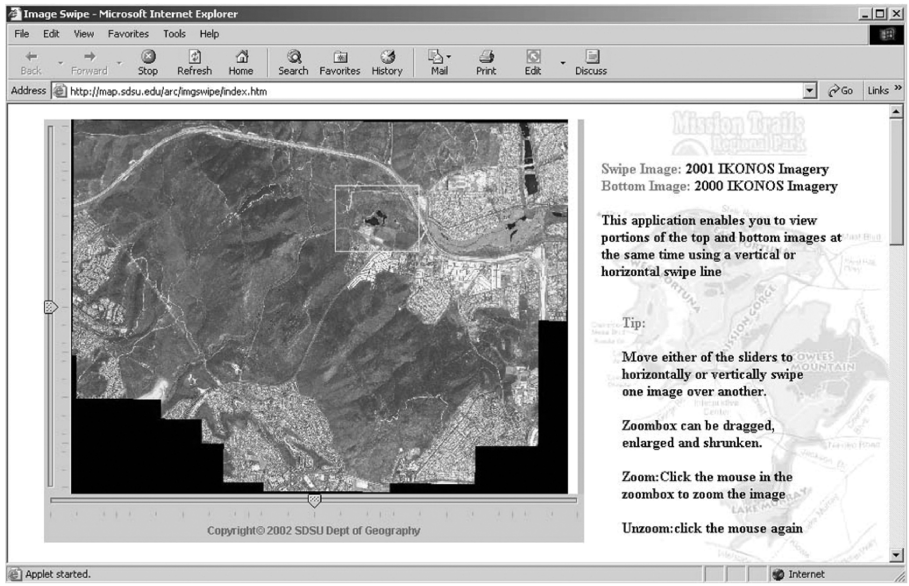

Fig. 5. The Image Swipe Java applet [http://map.sdsu.edu/arc/imgswipe].

to comparatively reveal different portions of the overlaid imagery, thereby aiding in a visual change detection analysis. In the test bed example, users were able to compare land cover changes between 2000 and 2001 using ADAR imagery (Fig. 6). Zoom and pan functionality was also available and implemented with a rectangular yellow zoom box tool.

Portions of the Java programming code used to implement the Java Image Swipe Tool are presented below for illustrative purposes:

```
int swipeWidth; //For horizontal swipe
int swipeHeight; //For vertical swipe

public float horizontalSwipe(float x) {

    swipeWidth = (int)(x * getSize().width);
    repaint();
    return x;
}
public float verticalSwipe(float y){

    swipeHeight = getSize().height - (int)(y * getSize().height);
    repaint();
    return y;
}
public void paint(Graphics g) {

    g.drawImage(bottomImage,0, 0, this); //Draw the bottom image
    //Draw the portion of the top image with horizontal swipe
    g.drawImage(topImage, 0, 0, swipeWidth, imageHeight, 0, 0,
    swipeWidth, imageHeight, this);
```

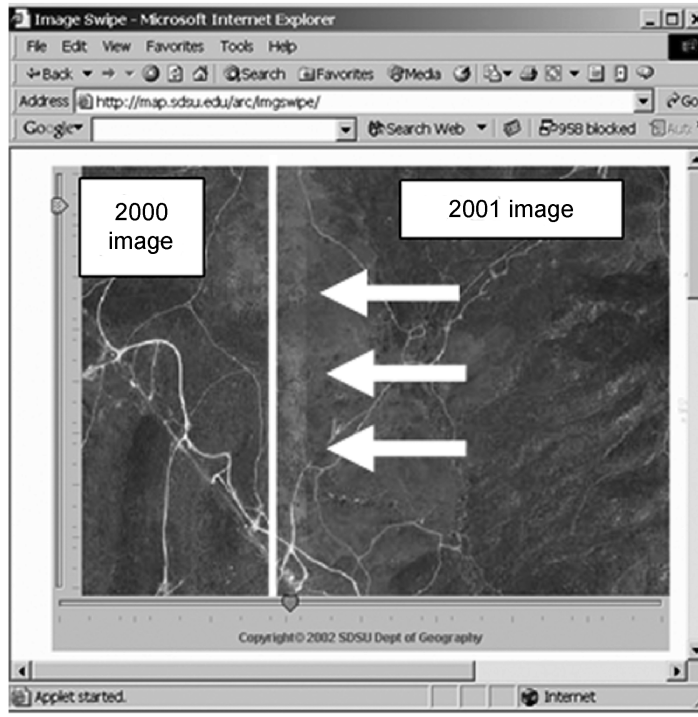

Fig. 6. The Java Imagery Change Detection Function.

```
        //Draw the portion of the top image with vertical swipe g.
        drawImage(topImage, 0, 0, imageWidth, swipeHeight, 0, 0,
        imageWidth, swipeHeight,this);
    }
```

The code for the applet utilizes the *horizontalSwipe* and *verticalSwipe* methods to track the position of the swipe line in order to support client display and user interface functionality. The *swipeWidth* and *swipeHeight* variables store the size of the visual areas covered by each side of the swipe lines in order to fill the user display. The *paint* method facilitates generating and updating the client display based on user input.

In general, Java programming for notebook PC applications is relatively easy and straightforward. The framework provides an effective tool for development of GIS/RS applications. It should also be noted that any notebook computer with an appropriate Java runtime, regardless of its native operating system, should be able to execute this Java Swipe Tool applet.

It should be reiterated that great consideration must be made regarding the version of the Java Runtime Environment available on the client-side mobile computing device. If the appropriate version of the JRE and its corresponding Java Virtual Machine is not available on the client side, the Java applet may not have the appropriate resources to function on the client mobile computing device. To address these issues, hardware and OS-specific JREs or Java Runtime Web browser plug-ins can be downloaded for free at Sun Microsystem's Website (http://java.com/en/). Many newer

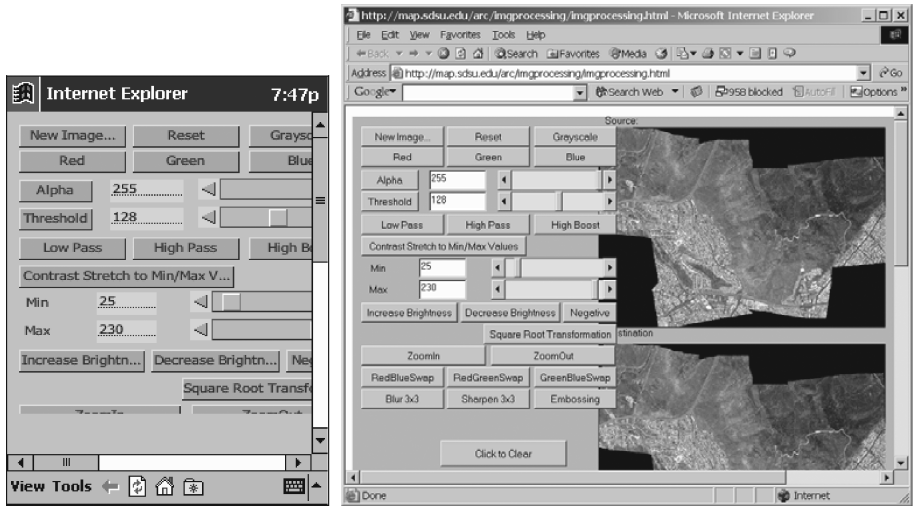

Fig. 7. Crossing-platform Java Web Applet execution: pocket PC (left) and notebook PC (right).

Web browsers also have built in support for the newer versions of the Java Runtime when they are installed on client computing devices. Additionally, many applets and browsers have support for on-demand updates and installation of appropriate Java runtime resources.

CASE STUDY 2: J2ME ON THE POCKET PC PLATFORM USING THE CrEme JVM PLUG-IN

The second study considers the development of a Java Web-based imagery analysis application for the Pocket PC platform utilizing the Java 2 Micro Edition (J2ME) application development framework. Java development of Mobile GIS application tools for the Pocket PC platform and the Windows CE OS is challenging given the fact that native Java support is not available on the system. To address this issue, a compatible Windows CE OS–specific Java runtime is required. One such solution entailed the use of the CrEme third party compact runtime application for the Windows CE OS of the Pocket PC platform. The CrEme application tool was selected for its robust support of many Java API libraries and functionality that are useful for Mobile GIS development. The CrEme tool includes support for Remote Method Invocation (RMI) functionality, and many advanced user interface and image display API libraries. These include the JNI API, the Swing 1.1 API, the Tiny AWT API, and the Truffle graphical toolkits API. A CrEme Web browser plug-in, which functions as an embedded JVM for the Pocket version of Microsoft Internet Explorer, is also included with the application package. With this plug-in, compatible Java applets can be executed through the Pocket Internet Explorer Web browser. Figure 7 comparatively illustrates an identical image analysis applet executing on Web browsers on the Pocket PC and Notebook PC platforms.

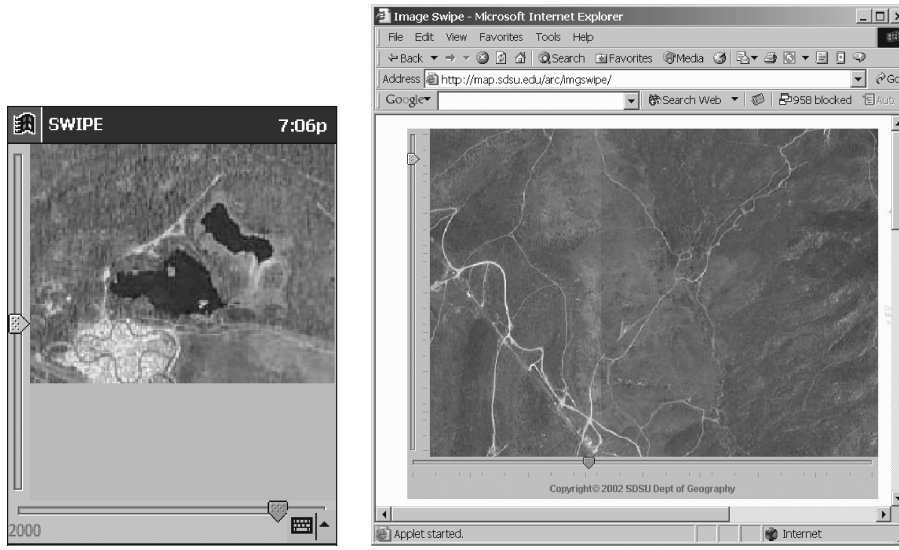

Fig. 8. Crossing-platform Java Web Applet execution: pocket PC with improved GUI (left) and notebook PC (right).

Because of Java's emphasis on cross-platform functionality, Java source code modifications that are required for multiple platform/platform deployment are generally minimal. Java source code originally written in the J2SE framework for the notebook PC was modified to accommodate the display, functionality, and memory limitations of the Pocket PC platform using the J2ME framework. The resulting Java applet for the Pocket PC platform was packaged and deployed as a downloadable Java Archive (JAR) file.

J2ME framework–based applications can integrate newer and existing API classes for development of future Internet-based GIS and RS applications on the smaller platforms. Figure 7 illustrates the differences between the same Mobile GIS image analysis application for the Pocket PC and notebook PC platforms. Typical Pocket PC devices have a significantly smaller screen display area (320 × 240 pixels) compared to standard notebook PC display screens (1024 × 768 pixels). Because of this, usability of the GUI can become an issue as noted in the figure. The Pocket PC version of the applet on the left side of the figure illustrates that the user must rely on the scrollbars to view the entire GUI display area which is readily visible on the notebook PC version of the Mobile GIS application on the right.

Figure 8 illustrates a GUI design that is better suited for user interaction and data display on a Pocket PC platform compared to the previous example. This is noted by the lack of having to use scrollbars to view the GUI and data display in the Pocket PC version of the application. Additionally, the form and functionality of the original notebook PC version appears to be well preserved in its implementation on the Pocket PC platform.

Other programming frameworks exist for the creation of Mobile GIS applications on small platform computing devices. These frameworks include the Palm Conduit

Development Kit (CDK) for Palm OS platforms and Microsoft's .NET Compact Framework for Pocket PC platforms.

The .NET Compact Framework is a major competitor to the J2ME application development framework, and is designed to handle XML Web services messaging. By supporting XML messaging, the Compact Framework can be utilized to develop thin client applications that can employ the many public and private Web services and distributed computing resources currently available. On the surface, this may appear to be an ideal framework for Mobile GIS application development. However, the .NET Compact Framework (CF) is not a true cross-platform solution and is currently limited for development of Windows CE OS–based device applications. J2ME, on the other hand, provides a better solution for the development and deployment of GIS/RS applications on a Mobile GIS system.

CASE STUDY 3: MOBILE PHONE APPLICATION DEVELOPMENT USING THE J2ME WIRELESS TOOLKIT FRAMEWORK

The third case study entails a J2ME wireless GIS data viewer running on a mobile phone platform. Java was utilized to enable GIS data query, display, and manipulation on a mobile phone display. This was demonstrated with GIS data consisting of ESRI Shapefiles.

In order to access GIS data, the data must be stored on a remote Web server. This is because the data are often too large to store within the mobile phone. Instead, the mobile phone is used as a dynamic data viewer that can be used to query, download, display, and manipulate portions of the user-required GIS data. In order to enable this functionality on a memory, processor, and display-constrained computing device such as a mobile phone, an extremely robust and compact application development framework is required. In the case of the wireless GIS data viewer, the J2ME Wireless Toolkit framework was adopted. Using this application framework, a GIS MIDlet was developed for the mobile phone platform. A MIDlet is a compact Java file which conforms to the Mobile Information Device Profile (MIDP) for extremely small computing devices like mobile phones (Riggs et. al, 2001). MIDlets objects extend the *Javax.microedition.MIDlet* API class and contain other special APIs for development over wireless communications and on mobile devices (Sun Microsystems, 2000).

Portions of the Java programming code used to implement the wireless GIS data viewer application are given below for illustrative purposes:

```
import Javax.microedition.midlet.MIDlet;
import Javax.microedition.lcdui.*;
public class GIS extends MIDlet implements CommandListener {
    boolean firstTime = true;
    Viewer view;
    protected void startApp() {
            if(firstTime){
        view = new Viewer();
            firstTime = false;
            }
        display.setCurrent(view);
```

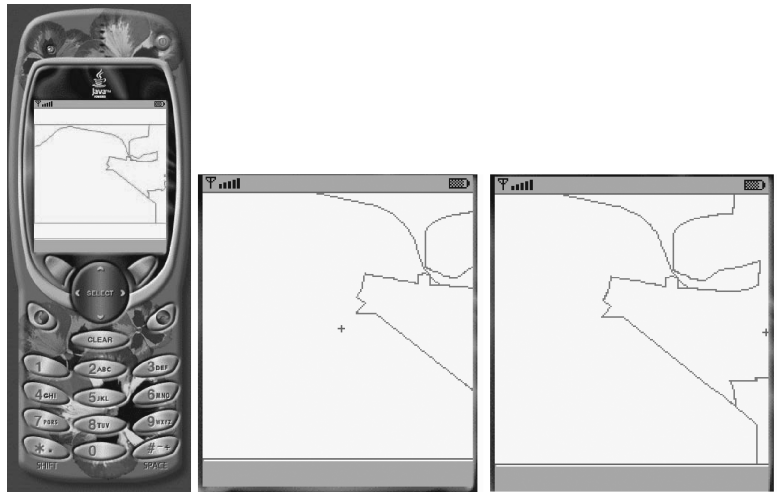

Fig. 9. Demonstration of a map pan function on a mobile phone application test bed.

```
    }
    protected void destroyApp(boolean unconditional) {
        notifyDestroyed();
    }
    protected void pauseApp() {
    }
}
```

The first two lines of the Java code specify the specialized API class libraries for mobile application development. The *Javax.microedition.MIDlet* class library defines three abstract methods that must be defined for any implementation of a MIDlet. The methods include *startApp*, *pauseApp*, and the *destroyApp* methods. When a user starts an application on a MIDP-conforming device, the *startApp* method of the MIDlet application is called by the application. When the user temporarily suspends an application, presumably to handle an incoming mobile phone call, the *pauseApp* method is called. When the user exits the Java application, the *destroyApp* method is called to facilitate necessary housekeeping of the mobile phone's computing system and resources. Figure 9 illustrates the wireless GIS viewer MIDlet application executing on the Java Wireless Toolkit's mobile phone simulator.

A major difference between the Pocket PCs and mobile phone platforms is the appearance and functionality of the user interface. On a standard notebook PC or Pocket PC, users can manipulate mouse cursor devices or use a touch-sensitive display panel to manipulate graphical data such as GIS maps or RS imagery. On a mobile phone, the user interface is far more limited. Input is usually limited to some function keys and a number pad. Due to this constraint, several number keys were employed for user input. This enabled some basic map display and movement functionality on the mobile phone. The number keys were mapped to these functions on the mobile phone:

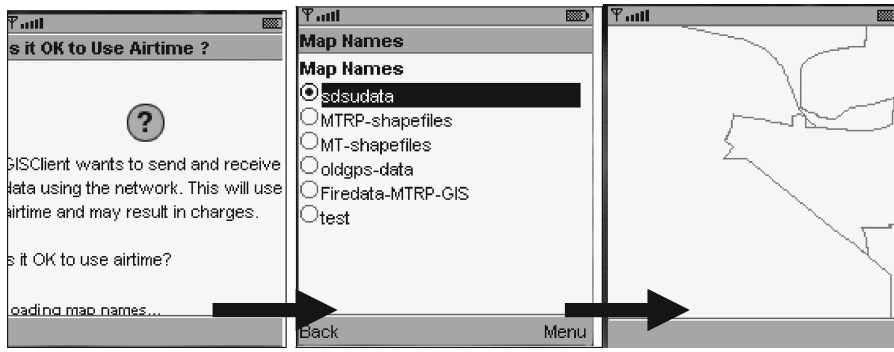

Fig. 10. Demonstration of GIS data download onto a mobile phone using a menu-based user interface.

- *Key 1* was mapped to the zoom-out function. (The zoom-scale was set to change by a factor of 1.5 for each use of this function.)
- *Key 0* was mapped to the zoom-in function. (The zoom-scale was set to change by a factor of 1.5 for each use of this function.)
- *Key 5* was mapped to the pan function.
- *Key 2* was mapped to the move up function.
- *Key 8* was mapped to the move down function.
- *Key 4* was mapped to the move left function.
- *Key 6* was mapped to the move right function.
- *Key #* was mapped to a toggle control for enabling and disabling the move function.

When the move function is enabled, the zoom and pan function will be disabled and a "+" symbol will appear at the center of the image display as a reference point for user manipulation of the feature data that are displayed on the display. When the move function is disabled, the zoom and pan functions will be re-enabled.

Figure 10 illustrates the use of the wireless GIS data viewer for on-demand downloading of GIS data via a menu-driven user interface on the mobile phone over a wireless network. This approach offers an improved solution for data viewing since data do not need to be stored on the client. The data can be queried and retrieved based on user requirements. This can result in substantial savings of client memory. Additionally, this may offer the user a more flexible means to obtain data since the data that are available for viewing may be updated, improved, or supplemented on the server side.

There are issues and considerations to take into account associated with the wireless GIS data viewer. The packaging of GIS data elements such as point, line, and polygon features, for robust transport over a mobile phone wireless network is difficult due to the existence of many different standards and protocols for data

organization and transfer. Next, GIS data can be very large and the transmission may be cut short by the memory and processing constraints of the client mobile phone device, or by the wireless mobile phone network. Finally, end users may incur significant costs associated with transferring data with their wireless communications plans. The additional costs may be in addition to their regular service charges.

CONCLUSION AND FUTURE WORK

Reliable geospatial information management tools with visual and analytical capabilities are in great demand to maximize the use of geospatial data in facilitating decision making processes (Gahegan, 1998; Brown, 1999). With the rapid development of wireless and mobile technology and applications, comprehensive Mobile GIS applications are requested by many GIS and RS users. With the arrival of high-bandwidth wireless mobile networks, these networks have been identified as possible data transport mechanisms for Mobile GIS applications. Challenges still remain for the development of robust and user-friendly Mobile GIS applications.

Heterogeneous operating systems of different mobile devices, including the Windows CE OS, the PalmOS, the Symbian OS, and the Java Phone OS, are additional hurdles to development of robust cross-platform applications. A single Mobile GIS application developed for several differing platforms and operating systems can be extremely expensive and time consuming. Each individual platform and OS may require a nearly complete rewrite of programming code in order to duplicate application functionality and specified requirements. To address this issue, the GIS software industry and mobile computing device vendors need to establish a community-based standards organization for the future development of Mobile GIS applications. Presently, OpenGIS Location Services (OpenLS) specifications by the Open GIS Consortium (OGC, 2003a) offer a good example for setting mobile application standards. In this instance, the organization addresses Location-Based Services (LBS) standards for Mobile GIS and other applications. They have specified the adoption of XML-based Abstract Data Types (ADT) and the GeoMobility Server for cross-platform and cross-device functionality (OGC 2003a, 2003b). Similar standards initiatives need to be pursued for field-based Mobile GIS and mobile RS applications.

Another consideration is the inadequate bandwidth of current wireless communications technology. This limits the feasibility of the development of GIS and RS applications due to their relatively large data sizes. Current wireless technology, in the best-case scenario, can only provide up to 1 Megabit per second of data transfer via an advanced cellular mobile phone network like CDMA or newer 3G systems. Most GIS and RS applications require high-speed network connections since GIS and RS imagery data are routinely 10–500 MB in size. Because of these considerations, and the general lack of supporting application tools and mobile communications infrastructure, it is currently difficult to access available GIS/RS data. Pocket PCs and similar devices are in a more advantageous situation when it comes to bandwidth considerations due to their ability to use high-speed wireless protocols and devices such as IEEE 802.11 (Wi-Fi) and Bluetooth. IEEE 802.11 technology is very robust and well utilized, and can enable wireless data transfer at a rate of 11 megabits per second to 54 megabits per second in the ideal situation. WiMAX is an emerging IEEE 802.16 standard for broadband wireless wide-area network (WWAN) or Metropolitan area

network (MAN) applications. WiMAX can provide a larger coverage of service area than Wi-Fi. Its communication signals can cover a range of 4–6 miles (up to 20 miles for the long-distance setting). To address the wireless communication issue, the development of Mobile GIS applications make it possible to robustly handle erratic signal coverage, with features such as download restart, synchronization, and continuation after line drops. The ideal solution is to provide seamless communication merged from cellular mobile coverage and the Wi-Fi/WiMAX combination.

Another challenge is the usability of Mobile GIS and remote sensing applications. Most Mobile GIS devices are constrained to a platform with a limited visible screen area, memory, computational power, and communications ability. Much research must go into human factors and GUI design. Different platforms may require GUI modifications to best serve its target user needs. Some solutions can include utilization of specialized input systems, including the touch-sensitive display panels on Pocket PCs and Tablet PCs, or voice-recognition technology on Pocket PC and mobile phone devices. Recent innovations in consumer-grade GPS automobile navigation systems may offer hints to solutions for addressing these issues in Mobile GIS applications.

Potential applications of Java framework–based programs for the display and manipulation of GIS and RS data were examined. Visual and analytical functions of applications developed for three different hardware platforms were analyzed and discussed. These application tools examined how users can interactively access, navigate, and explore visual RS and GIS data over mobile computing devices. Although the development of Mobile GIS and RS applications is at its infancy, there is great potential for Java framework-based analytical tools in several application areas including environmental monitoring, natural resources management, emergency management, and homeland security. Mobile GIS data collection and geospatial analysis over Internet connections can result in cost savings, the reduction of manpower requirements for field monitoring tasks, and encourage the sharing of analytical tools.

Other alternative approaches for Mobile GIS applications can be combined with the Java platform. Scalable Vector Graphics (SVG) and mobile web services are two promising future technologies for mobile GIS and remote sensing. SVG is an XML-based, two-dimensional vector graphics media format specified by the W3C in 2001 (version 1.0) and in 2003 (version 1.1) (W3C, 2003). There are three types of SVG probiles: SVG Full, SVG Basic, and SVG Tiny. SVG Full is suitable for desktop or laptop PCs. SVG Basic (smaller than SVG Full) is designed for Pocket PC or PDAs. SVG Tiny is designed for mobile phones. The advantage of mobile SVG (Basic and Tiny) compared to other graphic formats is that it can provide a compact, multimedia-enabled vector display format. SVG images are scalable and dynamic, and can be used within the Java platform. For example, SVG Tiny can become one of the major display formats for J2ME applications.

Mobile web services are an extension of general web services that are built upon XML; Simple Object Access Protocol (SOAP); Universal Description, Discovery, and Integration (UDDI); and Web Services Description Language (WSDL). Mobile web services can combine multiple functions and customizable information provided by web service providers for different mobile applications and users. The advantage of adopting web services for mobile GIS application is that web services can provide flexible combination of multiple web computing techniques with modern enterprise

GIS architecture. The contents of mobile web services include short messaging services (SMS), multimedia messaging services (MMS), and location-based services (LBS). Most mobile web services rely on server-side computing power significantly. For mobile GIS or LBS tasks, web services work like client-side terminals, with more flexible choices of GIS functions provided by remote web servers rather than running GIS functions locally. This is quite different from Java platform applications, which utilize client-side (mobile devices) computing power for major GIS works.

To summarize, future development of Mobile GIS and RS applications should entail better user interface and human-factor design, and development of more advanced analytical tools and capabilities, including dynamic sketching, geometric calculation, vector overlay, and image feature classification. With its features and capabilities, the Java framework may offer a robust and appropriate cross-platform solution to facilitate the development of the next generation of Mobile GIS and RS applications and tools. Moreover, the cross-platform nature and accessibility of applications developed within the Java framework may broaden the scope and appeal of Mobile GIS, RS, and other geospatial tools and applications for the public at large.

ACKNOWLEDGMENTS

The authors wish to acknowledge that significant portions of the content discussed herein stem from the research of two projects: "A Border Security Decision Support System" (NASA REASoN-0118-0209) and "Integrated Mobile GIS and Wireless Image Web Services for Environmental Monitoring and Management." Additionally, the authors wish to acknowledge and express their appreciation of matching funds received from the National Science Foundation project, "A Scalable Skills Certification Program in GIS" (NSF-ATE DUE 0401990). Additional acknowledgment is extended to John Kaiser, the REASoN program coordinator, and Douglas Stow, the REASoN project Principal Investigator. The authors would also like to recognize Sheth Dhawal and Min Zou, graduate students at San Diego State University, for their Java programming efforts in support of the research.

REFERENCES

Andrienko, G., Andrienko, N., and J. Carter, 1999, "Thematic Mapping in the Internet: Exploring Census Data with Descartes," in *Proceedings of TeleGeo'99 Conference*, Lyon, May 6-7, (1999), Laurini, R. (Ed.), 138-145.

Brown, I. M., 1999, "Developing a Virtual Reality User Interface (VRUI) for Geographic Information Retrieval on the Internet," *Transactions in GIS*, 3(3):207-220.

Coddington, P. D., Hawick, K. A., and H. A. James, 1999, "Web-Based Access to Distributed High-Performance Geographic Information Systems for Decision Support," in *Proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.

Crisp, N., 2003, *Open Location-Based Services: Technical Brief (TB1034A),* Huntsville, AL: Intergraph Corporation, White paper [http://www.intelliwhere.com], accessed on December 16, 2003.

Derekenaris, G., Garofalakis, J., Makris, C., Prentzas, J., Sioutas, S., and A. Tsakalidis, 2001, Integrating GIS, GPS, and GSM Technologies for the Effective Management of Ambulances," *Computer, Environment, and Urban Systems*, 25:267-278.

Douglas, K., 1996, *The Java™ Platform*, Santa Clara, CA: Sun Microsystems, Inc. [http://Java.sun.com/docs/white/platform/Javaplatform.doc1.html], accessed on September 24, 2004.

Gahegan, M., 1998, "Scatterplots and Scenes: Visualization Techniques for Exploratory Spatial Analysis," *Computers, Environment, and Urban Systems*, 21(1):43-56.

Gosling, J. and H. McGilton, 1996, *The Java Language Environment*, Santa Clara, CA: Sun Microsystems, Inc. [http://Java.sun.com/docs/white/langenv], accessed on September 24, 2004.

Halfhill, T. R., 1997, "Today the Web, Tomorrow the World," *Byte*, 22(1): 68-80.

Hamilton, M. A., 1996, "Java and the Shift to Net-Centric Computing," *IEEE Computer*, 29(8):31-39.

Huang, B. and M.F. Worboys, 2001, "Dynamic Modeling and Visualization on the Internet," *Transactions in GIS*, 5(2):131-139.

Intergraph, 2002, *Mobile Resource Management,* Huntsville, AL: Intergraph Corporation, White Paper [http://imgs.intergraph.com/freebies/pdfopener.asp?item=wp&file=WP1020A.pdf], accessed on September 24, 2004.

Jagoe, A., 2002, *Mobile Location Services: The Definitive Guide*, Upper Saddle River, NJ: Prentice Hall.

Limp, F. W., 2001, "User Needs Drive Web Mapping Product Selection," *GEOworld*, February 2001, 8-16.

OGC (Open GIS Consortium), 2003a, *OpenGIS Location Services (OpenLS): Core Services (Part 1-Part5).* (version 0.5.0), Wayland, MA: Open GIS Consortium, Inc., OGC-03-006r1.

OGC (Open GIS Consortium), 2003b, *OpenGIS Location Services (OpenLS): Part 6—Navigation Services,* Wayland, MA: Open GIS Consortium, Inc. (version 0.5.0) OGC-03-007r1.

Peng, Z. R. and M. H. Tsou, 2003, *Internet GIS: Distributed Geographic Information Services for the Internet and Wireless Networks*, New York, NY: John Wiley and Sons, Inc., 710 pp.

Pundt, H., 2002, "Field Data Collection with Mobile GIS: Dependencies between Semantics and Data Quality," *GeoInformatica*, 6(4):363-380.

Riggs, R., Taivalsaari, A., and M. VandenBrink, 2001, *Programming Wireless Devices with the Java 2 Platform, Micro Edition*, Boston, MA: Pearson Education.

Rodrigues, H. L., 2001. *Building Imaging Applications with Java™ Technology: Using AWT Imaging, Java 2D™, and Java™ Advanced Imaging (JAI),* Boston, MA: Addison-Wesley.

Sadoski, D., 2000, *Two Tier Software Architectures. Software Technology Review (STR)* [http://www.sei.cmu.edu/str/descriptions/twotier_body.html], accessed on September 24, 2004.

Schildt, H., 2004. *Java: The Complete Reference, 6/e*, Emeryville, CA: McGraw-Hill Osborne Media.

Shan, Y. P. and R. H. Earle, 1998, *Enterprise Computing with Objects: From Client-Server Environments to the Internet*, Boston, MA: Addison Wesley, 448 pp.

Stow, D., O'Leary, J., Coulter, L., Hope, A., and J. Franklin, 2001, *Application of Digital Imaging Technologies for Monitoring and Managing MSCP/NCCP Reserves,* San Diego, CA: San Diego State University, Department of Geography, Natural Community Conservation Planning Program Report.

Sun Microsystems, 1998, *Java Distributed Computing Technology Further Enabled by Support from IIOP*, Santa Clara, CA: Sun Microsystems, Inc. [http://www.sun.com/smi/Press/sunflash/1998-07/sunflash.980708.2.html], accessed on September 24, 2004.

Sun Microsystems, 1999, *Programming in Java Advanced Imaging*, Santa Clara, CA: Sun Microsystems, Inc., White Paper [http://Java.sun.com/products/Java-media/jai/forDevelopers/jai1_0_1guide-unc/], accessed on September 24, 2004.

Sun Microsystems, 2000, *J2ME Building Blocks for Mobile Device (White Paper on KVM and the CLDC)*, Santa Clara, CA: Sun Microsystems, Inc. [http://Java.sun.com/products/cldc/wp/KVMwp.pdf], accessed on September 24, 2004.

Tsou, M. H., 2004a, "Integrated Mobile GIS and Wireless Internet Map Servers for Environmental Monitoring and Management," *Cartography and Geographic Information Science*, 31(3):153-165 (the (special issue on Mobile Mapping and Geographic Information Systems).

Tsou, M. H., 2004b, "Integrating Web-based GIS and On-line Remote Sensing Facilities for Environmental Monitoring and Management," *Journal of Geographical Systems*, 6(2):155-174 (special issue on Web-Based GIS).

Tsou, M. H. and B. P. Buttenfield, 2002, "A Dynamic Architecture for Distributing Geographic Information Services," *Transactions in GIS*, 6(4):355-381.

Tsou, M. H., Guo, L., Stow, D., and J. Kaiser, 2002, *Web-Based Geospatial Information Services and Analytical Tools For Natural Habitat Conservation and Management. Final Report For NASA Affiliated Research Center.* San Diego, CA: San Diego State University, July 2002.

W3C, 2003, *Scalable Vector Graphics (SVG) 1.1 Specification*, Cambridge, MA: World Wide Web Consortium [http://www.w3.org/TR/SVG/], accessed on August 20, 2005.

WORA, 2005, "Write Once, Run Everywhere," *Wikipedia: The Free Encyclopedia* [http://en.wikipedia.org/wiki/Write_once,_run_anywhere], accessed on August 20, 2005.