

Biographical Information

Bryan G. Hassin
Software Engineer
R7 Solutions

Specific Responsibilities

Joined R7 Solutions in 2001. Responsible for architecting, designing, and implementing mobile and web-based GISes. Also responsible for making key technology decisions for R7 software products as well as client deployments.

Past Experience

Internet Systems Analyst, UUNet Technologies, 1996 – 2000.

Educational Information

B.S. – Electrical Engineering, Rice University
B.S. – Computer Science, Rice University
M.S. – Computer Science, Rice University

Professional Memberships

MOBILE GIS: HOW TO GET THERE FROM HERE

Bryan G. Hassin
R7 Solutions, Inc.
2472 Bolsover St.
Suite 345
Houston, TX 77005

ABSTRACT

Advances in mobile hardware and the increasing sophistication of handheld software have made Mobile GIS a reality. Despite this progress, Mobile GIS presents a number of key technology challenges, including data synchronization, security, and architecture. This paper will cover Mobile GIS application development from start to finish, discuss inherent technical challenges, and examine techniques to overcome them and produce high-quality Mobile GIS applications today.

PAPER

I. Introduction

This paper presents a guide to developing Mobile GIS applications from design to implementation. It begins by defining Mobile GIS, continues with a survey of relevant architectures and development tools, and concludes with an in-depth examination of a real Mobile GIS.

What is Mobile GIS?

What is Mobile GIS? The obvious definition is “GIS on mobile devices.” However, the term Mobile GIS can also encompass stationary devices communicating with mobile devices over a network.

Some example Mobile GIS applications:

- **Work Order Dispatching:** A field engineer in charge of locating and marking water lines synchronizes a mobile device with a corporate network each day. On his device he can see his work orders for the day, map the locations of each work order, add notes, and close out each work order as he finishes. When his mobile device synchronizes again, the notes and status of each work order are transferred to the corporate network and new work orders are downloaded to the device.
- **Asset Inventory:** A city government representative uses a mobile device to gather data about municipal property. At the site of each asset he is able to add information about its apparent condition and to use an attached GPS device to mark its exact location. His device then transfers the data to the city network when he synchronizes.

- Pipeline Inspection: A pipeline engineer inspects pipelines in remote areas. Using a cellular modem, his mobile device is able to map the pipeline and access related technical documents and previous inspection records. He can add his own inspection notes and update them in the main database in real time.

Why Mobile GIS?

Having defined Mobile GIS, the question still remains as to why it should be used. There are many reasons, but some of the most compelling are:

- access to geodata in the field, where it is often needed the most
- ability to *capture* data in the field and in real-time
- ability to append positional information to data capture
- GIS functionality in the field, where again it is often needed the most

Why *Not* Mobile GIS?

If Mobile GIS is so appealing, why isn't it widely deployed? Again there are many reasons:

- Hardware resources on mobile devices have typically been inadequate to run sophisticated GIS applications.
- Those devices with adequate hardware resources have typically used power at such an alarming rate as to render them useless for practical applications in the field.
- Software—and especially GIS software—has not typically been widely available for mobile devices.
- Connectivity in the field has typically been sparse, making it difficult to connect mobile devices to other mobile devices or networks.
- Mobile devices have typically cost a significant amount while delivering poor performance.

Mobile Device Trends

Fortunately, recent advances in mobile technology are beginning to address these shortcomings. Mobile devices today are boasting processing power and memory on par with that found in desktop computers only two years ago. Hardware manufacturers are finding ways to squeeze more screen size onto smaller devices and software providers have created more efficient user interfaces to maximize the benefit of those screens. Wireless connectivity has become much more readily available than it was even just one year ago; many metropolitan areas feature large-scale wireless networks and it is becoming increasingly difficult to find remote areas where cellular or satellite connections are not accessible.

Mobile technology advances, though, are not limited to hardware. Software and operating systems for mobile devices have also grown more sophisticated. In fact, the newest operating system for mobile devices, Microsoft Windows XP Tablet PC Edition, is actually a *full* version of Windows XP. This is revolutionary for the mobile application

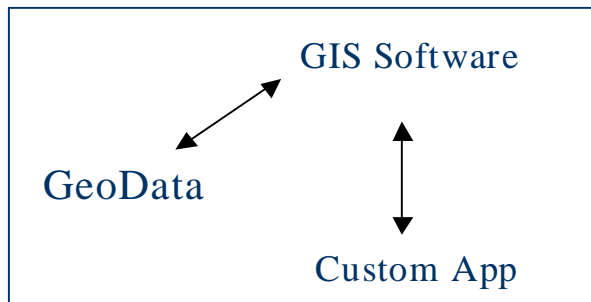
developer, as it allows him to develop applications for the Tablet PC exactly as he would for any desktop computer.

The most important mobile device trend is the decreasing cost of hardware—the mobile devices themselves—as well as accessories, such as GPS units. So what was once a cost-prohibitive, underpowered, immature technology is now a real possibility.

II. Developing Mobile GIS Applications: Architectures

In the previous section, it was established that Mobile GIS has advantages for certain applications and that the technology has reached a point where it is feasible to deploy. This paper now examines *how* to build such applications, beginning with architecture.

Architecture 1: Stand-Alone Client



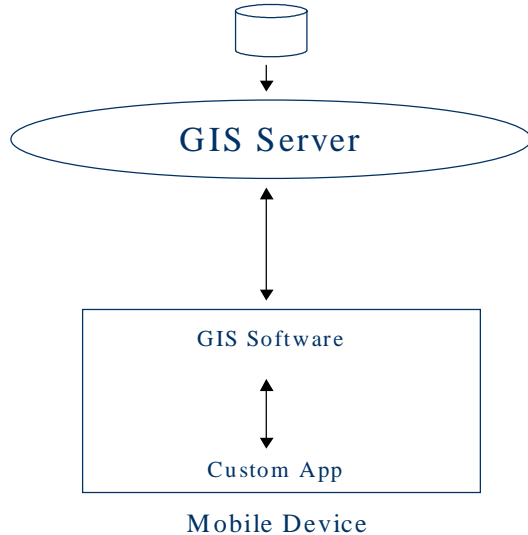
Mobile Device

The simplest Mobile GIS architecture is the Stand-Alone Client. In this architecture the application resides entirely on the mobile device. The device stores geodata, out-of-the-box mobile GIS software to interpret and display that data, and the customized application, which is built on top of the GIS software.

Another approach would be to develop a custom application from scratch and have it interpret and display the geodata directly, eliminating the need for separate GIS software. While it is possible that some applications might benefit from this approach, it is likely that most would suffer from the increased development and testing time such an approach would take.

Regardless of the approach, this architecture has some pretty major limitations. First, the hardware resources of the mobile device restrict the amount of geodata the application can support. Second, this architecture does not allow for communication with any other applications or collaborators using the same application.

Architecture 2: Client-Server

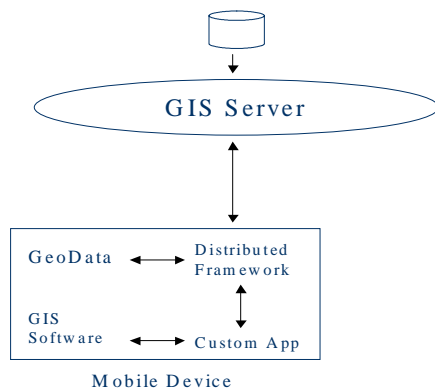


To address these limitations, one might reasonably adopt a client-server architecture. Here the geodata is moved to a separate computer and served to the client by GIS server software. The customized application remains the same and the only change to the GIS software is its configuration, which now looks to the GIS server for geodata.

This has some advantages over the Stand-Alone Client. First, the geodata is constrained now by the virtually limitless resources of an enterprise server. Second, multiple mobile devices running the same application can access the server concurrently, making this a potentially multi-user architecture.

Unfortunately, this architecture also introduces a new variable: communication. What happens when the mobile device is unable to communicate with the GIS server due to range, interference, or another prohibitive factor? Then the application cannot access its geodata and it is useless. Because inconsistent connectivity is an inherent, common problem in mobile applications, this scenario must be considered.

Architecture 3: Distributed Client-Server



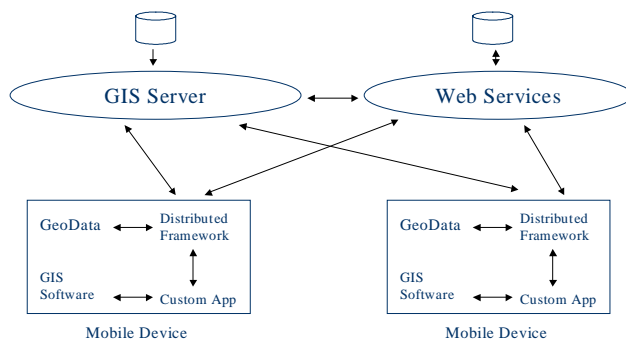
To address the problems of inconsistent connectivity, it is necessary to employ two key distributed systems concepts: persistence and resource management.

- *Persistence* - when the mobile device cannot connect to the server, it persistently tries to reconnect.
- *Resource Management* – when the mobile device cannot connect to the server, it uses a small set of cached data (a subset of that on the server) that resides locally on the mobile device.

When the mobile device is able to connect again, it synchronizes its data with the server. The component that handles the logic for persistence and resource management is called the distributed framework.

This architecture will support most Mobile GIS applications in a robust, reliable way, but it does not allow for extensibility on the back end. What happens when the application needs to support more back-end functionality than that which is available in the GIS server?

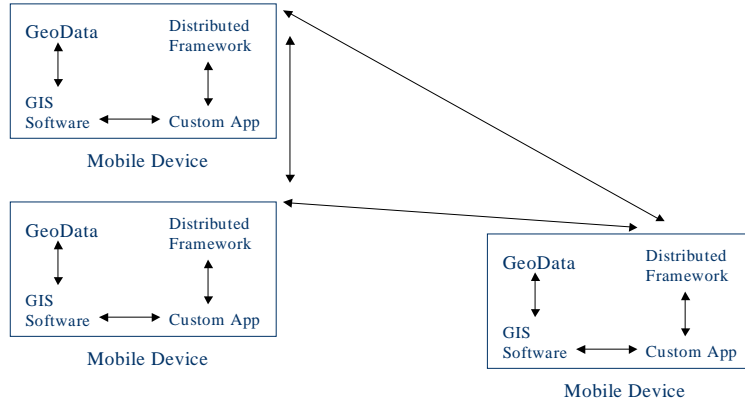
Architecture 4: Services



To provide more back-end extensibility, this architecture views the GIS server as a web service and allows for other web services to be part of the application as well. As long as these web services use the same communications protocol, the mobile device(s) can communicate with all of them. Furthermore, the web services can also communicate between themselves. A natural fit for this common communications protocol is SOAP XML, the industry standard for passing messages between software components.

This architecture supports robust communication between any number of mobile devices and web services but it might not be ideal for some applications, such as those designed for collaboration in remote areas where connectivity to servers unavailable.

Architecture 5: Peer-to-Peer



In this scenario, a peer-to-peer architecture will allow for communication between mobile devices while still addressing the shortcomings of the Stand-Alone Client architecture.

Because a server is no longer available to house geodata, the data must be stored on the devices themselves. But if each mobile device stores 100% of the data, then the architecture is restricted in the same manner as the Stand-Alone Client.

To allow for more data storage throughout the application, each mobile device houses just a subset of the geodata. When Mobile Device A needs data, it relies on its distributed framework's resource management to know if it has that data locally. If it does not, the distributed framework must know how to access that data on Mobile Device B. As this example shows, this architecture necessitates the use of another distributed systems concept: *naming* – unique identifiers for each device and logic to differentiate between them.

But what happens when B is out of range of A? A cannot access the data it needs. To address this problem this architecture must employ the final distributed systems concept covered in this paper: *redundancy* – maintaining data on one or more other devices as an alternate data store. So in this case, the distributed framework on mobile device A knows that the data for which it is looking can also be found on Mobile Device C.

This section has shown that there are many system architectures available to the mobile application developer. A particular application's needs and constraints will determine which architecture is best.

III. Developing Mobile GIS Applications: Development Tools

Having discussed Mobile GIS architectures, this paper now addresses tools for implementing them.

Platform: J2ME

Java 2 Micro Edition (J2ME) is Sun's version of Java targeted at mobile devices. To deploy a J2ME application, the developer must first choose a configuration; the application may then be developed using a profile for that configuration. There are four configuration-profile options:

- Connected Limited Device Configuration (CLDC) – This configuration uses the KVM, an extremely lightweight version of the Java Virtual Machine (JVM).
 - Mobile Information Device Profile (MIDP) – The MIDP is the only profile available for the CLDC and is intended for mobile devices with very limited resources, such as mobile telephones.
- Connected Device Configuration (CDC) – This configuration uses a full JVM, but its profile determines how much functionality is available.
 - Foundation Profile (FP) – As its name indicates, this profile provides just the basic foundation classes.
 - Personal Basis Profile (PFP) – The PFP provides the foundation classes as well as additional graphics functionality.
 - Personal Profile (PP) – This profile is a superset of the PFP and actually provides the entire Advanced Windows Toolkit (AWT) to the developer.

Platform: .NET Compact Framework

As the .NET Framework is Microsoft's answer to Sun's Java, the .NET Compact Framework (CF) is Microsoft's answer to Sun's J2ME. It is the .NET Framework for mobile devices. What makes it particularly appealing for the mobile application developer is that a CF application is developed in exactly the same way as a regular .NET desktop application. Once the developer configures his IDE's build properties to deploy to a mobile device instead of to a desktop, his IDE takes care of the rest. Accordingly, developing mobile applications with the CF requires very little specialized experience in mobile development.

Mobile GIS Software: ESRI ArcPad

While it would be inaccurate to state that ESRI's ArcPad is the *only* GIS software for mobile devices, it is quite accurate for ESRI to claim that ArcPad is far and away the industry leader. Currently in version 6.0.2, it allows for direct connections to local geodata or remote connections to ArcIMS, ESRI's streaming map server. Additionally, its interface provides many standard GIS tools out-of-the-box, e.g., pan, zoom, and identify.

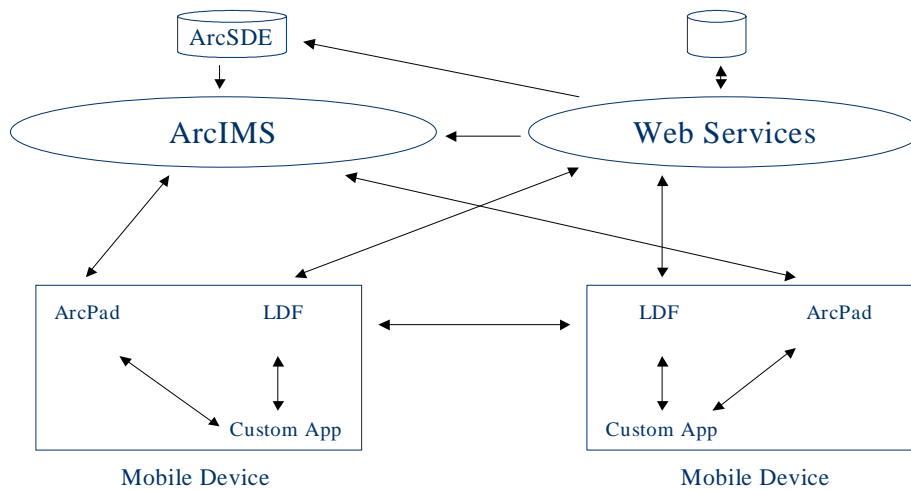
The mobile application developer can customize ArcPad using ArcPad Studio. Studio allows for the creation of custom toolbars and custom forms scripted with VBScript. These can be compiled into an "applet," which is an ArcPad mini-application saved on the mobile device as ArcPadXML. Extensions to ArcPad can also be created using Visual C/C++ for Windows CE devices.

As this section illustrates, there are multiple options available for developing mobile GIS applications. The choices made will depend on the application requirements and skill sets of the developers.

IV. Conclusion

In conclusion this paper examines a real Mobile GIS that encompasses many of the concepts discussed.

Mobile Disaster Response System



This application is designed for personnel responding to an oil spill. Field engineers are equipped with mobile devices with the objective of collaborating in real time as they collect data and devise a response plan. A Mobile Wireless Applications Center (MWAC) houses the servers for this application; it is a van equipped with generators and a high-powered radio antenna that is capable of creating a wireless network several miles in diameter.

ESRI's ArcSDE product stores data about the area and about the oil spill itself, while ArcIMS makes the geodata available to the mobile devices. The mobile devices run ArcPad to present the data to the field personnel with a small customization to intercept user events.

When a user adds a data point, e.g., takes a GPS reading at the perimeter of the oil spill, that coordinate is passed to the distributed framework--R7's Lightweight Distributed Framework (LDF). LDF persistently tries to send that coordinate to a web service in the MWAC. The web service adds the point to ArcSDE's geodata and notifies the mobile devices that they need to refresh their maps with the new data.

LDF also allows for peer-to-peer messaging. This enables users to chat in real-time without having to use another device. What's more, if a mobile device is out of range of

the MWAC but within range of another mobile device, this system uses the other mobile device to relay messages to the server.

Conclusion

This paper has shown that hardware and software trends are making Mobile GIS a reality. After surveying the many options for designing and implementing Mobile GIS applications, it has been illustrated with a real example that robust, reliable Mobile GIS is possible today.

V. References

1. ESRI ArcPad, <http://www.esri.com/software/arcpad/index.html>
2. Microsoft .NET Compact Framework,
<http://msdn.microsoft.com/vstudio/device/compact.aspx>
3. R7 Solutions Mobile Services, http://www.r7solutions.com/services_wireless.htm
4. Sun Java 2 Micro Edition, <http://java.sun.com/j2me/>